USING THE STRUCTURAL LOCATION OF TERMS TO IMPROVE THE RESULTS OF TEXT RETRIEVAL BASED APPROACHES TO FEATURE LOCATION

by

BRIAN EDDY

JEFF GRAY, COMMITTEE CHAIR
NICHOLAS KRAFT
JEFFREY CARVER
SUSAN VRBSKY
RANDY SMITH

A DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
in the Department of Computer Science
in the Graduate School of
The University of Alabama

TUSCALOOSA, ALABAMA

2015

ABSTRACT

Software maintenance and evolution make up a considerable portion of the time and effort spent during the life cycle of a software system. During the maintenance and evolution phase, the majority of a developer's time is spent on program comprehension tasks. Feature location (i.e., identifying a starting point for a change), impact analysis (i.e., identifying all source elements involved in a change), and software summarization (i.e., automatically summarizing the responsibilities of a source element) are examples of such tasks. Recent research in these areas has focused on improving each process to ease the burden on developers and decrease the time spent in each task through the use of textual information, dependency graphs, and execution traces. Furthermore, the success of text retrieval in other areas (e.g., traceability) has initiated new studies in automating feature location by the use of text retrieval techniques, such as the vector space model (VSM), latent semantic indexing (LSI), and latent Dirichlet allocation (LDA). Some research has been done to improve LSI and VSM models by combining structural information (i.e., information regarding the creation and use of objects and methods within the code) with the corpus obtained from extracting text from source code. However, little research has focused on improving LDA and more sophisticated topic models (i.e., a statistical model of the abstract topics that occur in a corpus) with structural information. Furthermore, no study has looked at how a developer's knowledge of a software system's structure may be incorporated into text retrieval based feature location for software maintenance tasks.

The research presented in this dissertation makes two main contributions. First, it evaluates a methodology for incorporating structural information into the corpus obtained in the text extraction phase by modifying the weights of terms based on their importance to the individual source elements. Furthermore, this dissertation introduces a novel technique for performing structured text retrieval that allows developers to use their existing knowledge about the structure of a software system. This dissertation is organized into the following parts: a demonstration of the effects of structural weighting schemes on the effectiveness of topic-modeling in feature location, the introduction of a new structured source code retrieval model, a demonstration of the effects of structured queries on the effectiveness of structured source code retrieval for feature location, and additional insights into how and when these approaches should be incorporated.

DEDICATION

To my Dad and my Papa, the men in my life who taught me what it means to live by your heart and what the word "dedication" truly means.

LIST OF ABBREVIATIONS AND SYMBOLS

AST	abstract syntax tree
CAS	content and structure queries
$C(LC, M, P, BC, LV)$	a structural weighting scheme for LDA
$C\{C = 1, S = 1, B = 1\}$	a combination and weighting in structured retrieval
CSB	comments, signature, and body
d_j	the jth document in a corpus
$\vec{d}_j$	a document vector
F	the F statistic for the Friedman or Kruskal-Wallis test
FCA	Formal Concept Analysis
FLT	Feature location technique
$freq_{i,j}$	frequency of ith term in jth document
H_0	the null hypothesis
H_A	the alternative hypothesis
k	a modeling parameter for LSI
K	the number of topics in LDA
k_i	the ith term in a document
ICL	identifiers, comments, and literals

LD	lexicon density
LMPBV	comments, method names, parameters, local variables
$\|L\|$	the cardinality of a lexicon
L_{Co}	the set of comments in a software system
L_{Id}	the set of identifiers in a software system
L_{Li}	the set of literals in a software system
LDA	latent Dirchlet allocation
LSI	latent semantic indexing
$\vec{M}$	term-document association matrix
$maxfreq_{i,j}$	maximum frequency of a term in jth document
M_d	a language model of document d
MRR	mean reciprical rank
$\vec{M}^t$	the transpose of the matrix $\vec{M}$
$\vec{M}\vec{M}^t$	the term-to-term correlation matrix
$\vec{M}^t\vec{M}$	the document-to-document correlation matrix
N	number of documents in corpus
n_i	number of terms in which ith term appears
p	the p statistic for significance testing
$p(A\|B)$	the probability of A given B
$\vec{q}$	a query vector

Q	a query to a language model
$\|Q\|$	the number of overall queries for a system
r	the effect size
r_i	the effectiveness measure for a query
$\hat{R}_{t,d}$	the risk factor for the language model
$\vec{S}$	singular values of $\vec{M}$
SDR	structured document retrieval
SEMERU	Software Engineering Maintenance and Evolution Research Unit
$sim(d_j, q)$	similarity between a document and a query
SVD	singular value decomposition
TR	text retrieval
tf-idf	term frequency - inverse document frequency
$\vec{U}$	eigenvectors from $\vec{M}\vec{M}^t$
UTC	unique term contribution
UTD	unique term density
U_{Src}	set of unique terms in source
VSM	vector space model
$\vec{V}^t$	eigenvectors from $\vec{M}^t\vec{M}$
w	a term weighting
$w_{i,j}$	weighting for ith term in the jth document

XML	extensible markup language
α	Dirichlet hyperparameter for topic proportions
β	Dirichlet hyperparameter for topic multinomials
σ	number of iterations of Gibbs sampling
ψ	the term-topic probability distribution
θ	the topic-document probability distribution
$\cup$	the union of two sets
$\in$	element in set
$\prod_{t \in Q}$	a product over all terms in Q
$\prod_{t \notin Q}$	a product over all terms not in Q
$\triangle$	symmetric difference
$\{t_1, t_2, ..., t_n\}$	a set of terms
$(w_{1,q}, w_{2,q}, ..., w_{n,q})$	weights for each term in a query
Σ	a summation

ACKNOWLEDGMENTS

First, I would like to thank the Department of Education and the GAANN fellowship. Without the support provided by this program, I would not have been able to pursue my degree.

I would like to thank the members of my committee, Dr. Randy Smith, Dr. Jeffrey Carver, and Dr. Susan Vrbsky. Dr. Smith, I appreciate your time and willingness to evaluate my dissertation research. Dr. Carver, I appreciate your software engineering classes, which have made a considerable impact on the way that I view software engineering as a field. Dr. Vrbsky, I appreciate your courses in databases and cloud computing, which have helped me broaden my skills and knowledge.

I would like to thank Dr. Nicholas A. Kraft, who introduced me to the world of academic research, and has been an invaluable resource for me throughout the years. Without you, I would not have been in this excellent program. Your straightforward advice has always been appreciated, and will serve me as I begin my own career.

I would like to thank Dr. Jeff Gray. By working with you, I have learned to see the broader social impact that computer science and software engineering has on the world. You have allowed me to grow as not only a computer scientist, but also as an educator. I have learned to not only imagine what use my knowledge can be to the software engineering community, but the role it can play in the larger world. I will continue to hold onto these ideals throughout my life.

I would like to acknowledge the other graduate students in the department, especially Jeffrey Robinson, Jonathan Corley, and Dustin Heaton. You have been the ones to keep me sane these last few years. I have truly enjoyed the years that we have spent together as graduate students.

To my brothers, sisters, parents, and grandparents. You have been the ones to always stand beside me. To see me through the thick and thin. To acknowledge me when I could not acknowledge myself. The languages of this world are incomplete in that they do not have strong enough words to express how blessed I am to be a member of this family. I know that I would not be where I am today if not for your love and support.

Finally, to my future wife, Adrienne. I am not sure you knew what you signed up for, but despite it all, you have stood beside me, with me. You are the love of my life, the one I am priviledged to share my hopes and my dreams. Know that after the last chapter of this dissertation comes to a close, it will only mean the opening of a brand new chapter in our lives. I am looking forward to what God has in store for us.

CONTENTS

ABSTRACT ii

DEDICATION iv

LIST OF ABBREVIATIONS AND SYMBOLS v

ACKNOWLEDGMENTS ix

LIST OF TABLES xvi

LIST OF FIGURES xxix

1 INTRODUCTION 1

1.1 Software Change 3

1.2 Overview of Text Retrieval 4

1.2.1 The Source Code Lexicon 8

1.2.2 Vector Space Model (VSM) 10

1.2.3 Latent Semantic Indexing (LSI) 12

1.2.4 Latent Dirichlet Allocation (LDA) 14

1.2.5 Language Modeling 15

1.2.6 Structured Document Retrieval 17

1.3 Overview 18

1.3.1 Structural Weighting of LDA 19

1.3.2 Structured Source Code Retrieval 21

1.3.3 Comparing Structured Retrieval with Structural Weighting of LDA 22

1.4 Organization 22

2 RELATED WORK 24

2.1 Text Retrieval Based Feature Location 24

2.2 Combining Additional Information with Text Retrieval 27

2.3 Configuration and Corpus Creation 34

3 PRELIMINARY STUDY ON CONFIGURING LDA 39

3.1 Case Study 40

3.1.1 Definition and Context 40

3.1.2 Overview 40

3.1.3 Subject software systems 42

3.1.4 Benchmarks 43

3.1.5 Effectiveness measure 44

3.1.6 Setting 44

3.1.7 Hypotheses 45

3.1.8 Data Collection and Analysis 50

3.2 Results 53

3.2.1 Part 1: Testing for Interactions among Factors 53

3.2.2 Part 2: Configuring the Query 53

3.2.3 Part 3: Configuring the Text Extractor and K 57

3.2.4 Part 4: Configuring α and β 59

3.2.5 Part 5: Applying the Lessons Learned 61

3.3 Discussion of Results 65

3.3.1 Part 2: Configuring the Query . . . 65
3.3.2 Part 3: Configuring the Text Extractor and K . . . 68
3.3.3 Part 4: Configuring α and β . . . 69
3.4 Threats to Validity . . . 70
3.5 Summary . . . 71
4 STRUCTURAL WEIGHTING OF LDA . . . 72
4.1 Study Design . . . 75
4.1.1 Definition and Context . . . 75
4.1.2 Research Questions . . . 81
4.1.3 Data Collection and Analysis . . . 83
4.1.4 Threats to Validity . . . 85
4.2 Results of Case Study . . . 86
4.2.1 Does structural weighting of comments, leading terms, and local variables affect the accuracy of a LDA-based feature location technique (FLT)? . . . 86
4.2.2 Can a relationship between the contributions of each structural component's lexicon and their weighting factors be found? . . . 118
4.3 Discussion of Results . . . 136
4.3.1 Does structural weighting of comments, leading terms, and local variables affect the accuracy of a LDA-based feature location technique (FLT)? . . . 136
4.3.2 Can a relationship between the contributions of each structural component's lexicon and their weighting factors be found? . . . 141
4.3.3 Recommendations . . . 144
4.4 Using Machine Learning to Find Optimum Configurations . . . 145
4.4.1 Genetic Algorithms . . . 146

4.4.2 Fitness Functions Used in Study . . . 149
4.4.3 The Search Process . . . 150
4.4.4 Searching Eclipse . . . 152
4.5 Summary . . . 159
5 STRUCTURED SOURCE CODE RETRIEVAL . . . 161
5.1 Approach . . . 163
5.1.1 Overview of Indri . . . 163
5.1.2 The Indri Retrieval Model . . . 163
5.1.3 Creating a Structured Corpus . . . 166
5.1.4 Creating Structured Queries . . . 171
5.2 Study Design . . . 174
5.2.1 Definition and Context . . . 174
5.2.2 Research Questions . . . 179
5.2.3 Data Collection and Analysis . . . 181
5.2.4 Threats to Validity . . . 183
5.3 Results of Case Study . . . 185
5.3.1 Does query type affect the accuracy of a structured retrieval-based FLT? . . . 185
5.3.2 Does changing the combination of included fields affect the accuracy of a structured retrieval-based FLT? . . . 225
5.3.3 Does structural weighting affect the accuracy of a structured retrieval-based FLT? . . . 280
5.3.4 How does the best configuration of structural field combination and weighting affect the accuracy of a structured retrieval-based FLT? . . . 350
5.4 Discussion of Results . . . 403

5.4.1 Does query type affect the accuracy of a structured retrieval-based FLT? . . 403

5.4.2 Does changing the combination of included fields affect the accuracy of a structured retrieval-based FLT? 411

5.4.3 Does structural weighting affect the accuracy of a structured retrieval-based FLT? 417

5.4.4 How does the best configuration of structural field combination and weighting affect the accuracy of a structured retrieval-based FLT? 422

5.5 Summary 428

6 LESSONS LEARNED AND FUTURE WORK 429

6.1 What are the benefits and consequences of using structural weighting in LDA? . . . 429

6.2 What are the benefits and consequences of using structured retrieval for feature location? 431

6.3 Future Work 431

6.3.1 Improved Learning Algorithms 431

6.3.2 Empirical Studies 432

6.3.3 Modeling Structural Information 432

6.3.4 Additional Software Tasks 433

7 CONCLUSION 434

7.1 Structural Weighting of LDA 435

7.2 Structured Retrieval 435

7.3 Final Remarks 437

REFERENCES 438

APPENDIX 447

A EXAMPLE QUERIES 447

LIST OF TABLES

3.1 Case study design: Part 1. 41

3.2 Case study design: Parts 2-4. 42

3.3 Case study design: Part 5. 42

3.4 Subject software systems. 43

3.5 Corpora Size Metrics 51

3.5 Corpora Size Metrics 52

3.6 Results of a factorial ANOVA. 54

3.7 The effectiveness measure for 28 configurations ($Text/K$ pairs) of the LDA-based FLT applied to all 618 features. 58

3.8 Key for Section 3.2.4. Each table entry provides an index for an α/β pair. 60

3.9 Case study design: Part 5. 62

3.10 Queries for ArgoUML, feature 4019. 67

3.11 Queries for jEdit, feature 2842444. 67

3.12 Queries for muCommander, feature 311. 67

3.13 Queries for Rhino, feature 352319. 68

4.1 Subject systems 77

4.2 Numbers of methods in the gold sets. 78

4.3 Descriptive Statistics Weighting Leading Comments Alone 88

4.4 MRRs Weighting Leading Comments Alone 90

4.5 Descriptive Statistics Weighting Method Names Alone 92

4.6 MRRs Weighting Method Names Alone 94

4.7 Descriptive Statistics Weighting Parameters Alone 96

4.8 MRRs Weighting Parameters Alone 98

4.9 Descriptive Statistics Weighting Body Comments Alone 100

4.10 MRRs Weighting Body Comments Alone 102

4.11 Descriptive Statistics Weighting Local Variables Alone 104

4.12 MRRs Weighting Local Variables Alone 106

4.13 Top ten configurations and unweighted configuration for each subject system and combined 111

4.13 Top ten configurations and unweighted configuration for each subject system and combined 112

4.13 Top ten configurations and unweighted configuration for each subject system and combined 113

4.14 Friedman Test Results 113

4.15 Main Effects and Interactions for each subject system and combined 117

4.15 Main Effects and Interactions for each subject system and combined 118

4.16 MRRs For Maximizing Weighting On Leading Comments and Method Names . . 118

4.17 Unique Terms, Term Usages, and Present Documents for each lexicon for each subject system 129

4.18 Lexicon Density, Unique Term Density, and Unique Term Contribution for each lexicon for each subject system 130

4.19 Average Method Lexicon Density, Unique Term Density, and Unique Term Contribution for Leading Comments for each subject system 131

4.20 Average Method Lexicon Density, Unique Term Density, and Unique Term Contribution for Method Names for each subject system 132

4.21 Average Method Lexicon Density, Unique Term Density, and Unique Term Contribution for Parameters for each subject system 133

4.22 Average Method Lexicon Density, Unique Term Density, and Unique Term Contribution for Body Comments for each subject system 134

4.23 Average Method Lexicon Density, Unique Term Density, and Unique Term Contribution for Local Variables for each subject system 135

4.24 Query document for JabRef feature 1588028 138

4.25 Eclipse 153

4.26 Terms, Uses, and Document Counts and System-wide Lexicon Density, Unique Term Density, and Unique Term Contribution for Eclipse 154

4.27 Average Method-level Lexicon Density, Unique Term Density, and Unique Term Contribution for each of the lexicons for Eclipse 156

4.28 MRRs Weighting Candidate Lexicons Alone 157

4.29 Number of times the top silhouette coefficient was in the top MRRs 159

5.1 MRRs for the three different query types (Title, Description, Combined) for ArgoUML 186

5.2 MRRs for choosing the best, average, and worst case for each feature for ArgoUML 190

5.3 Percentages for each query type where the best query was found for ArgoUML . . 190

5.4 MRRs for choosing the best, average, and worst case for each feature from all corpora for ArgoUML 192

5.5 Percentages for each corpus where the best query was found from all corpora for ArgoUML 192

5.6 Percentages for each query type where the best query was found from all corpora for ArgoUML 192

5.7 MRRs for the three different query types (Title, Description, Combined) for JabRef 194

5.8 MRRs for choosing the best, average, and worst case for each feature for JabRef . 198

5.9 Percentages for each query type where the best query was found for JabRef 198

5.10 MRRs for choosing the best, average, and worst case for each feature from all corpora for JabRef 200

5.11 Percentages for each corpus where the best query was found from all corpora for JabRef 200

5.12 Percentages for each query type where the best query was found from all corpora for JabRef 200

5.13 MRRs for the three different query types (Title, Description, Combined) for jEdit 202

5.14 MRRs for choosing the best, average, and worst case for each feature for jEdit . . 206

5.15 Percentages for each query type where the best query was found for jEdit 206

5.16 MRRs for choosing the best, average, and worst case for each feature from all corpora for jEdit 208

5.17 Percentages for each corpus where the best query was found from all corpora for jEdit 208

5.18 Percentages for each query type where the best query was found from all corpora for jEdit 208

5.19 MRRs for the three different query types (Title, Description, Combined) for muCommander 210

5.20 MRRs for choosing the best, average, and worst case for each feature for muCommander 214

5.21 Percentages for each query type where the best query was found for muCommander214

5.22 MRRs for choosing the best, average, and worst case for each feature from all corpora for muCommander 216

5.23 Percentages for each corpus where the best query was found from all corpora for muCommander 216

5.24 Percentages for each query type where the best query was found from all corpora for muCommander 216

5.25 MRRs for the three different query types (Title, Description, Combined) for all systems 218

5.26 MRRs for choosing the best, average, and worst case for each feature for all systems 222

5.27 Percentages for each query type where the best query was found for all systems . 222

5.28 MRRs for choosing the best, average, and worst case for each feature from all corpora for all systems 224

5.29 Percentages for each corpus where the best query was found from all corpora for all systems 224

5.30 Percentages for each query type where the best query was found from all corpora for all systems 224

5.31 MRRs and descriptive statistics of the effectiveness measures for the LMPBV corpus and Combined query type for ArgoUML 226

5.32 MRRs and descriptive statistics of the effectiveness measures for the LMPBV corpus and Title query type for ArgoUML 227

5.33 MRRs and descriptive statistics of the effectiveness measures for the CSB corpus and Combined query type for ArgoUML 228

5.34 MRRs and descriptive statistics of the effectiveness measures for the CSB corpus and Title query type for ArgoUML 228

5.35 MRRs and descriptive statistics of the effectiveness measures for the ICL corpus and Combined query type for ArgoUML 229

5.36 MRRs and descriptive statistics of the effectiveness measures for the ICL corpus and Title query type for ArgoUML 229

5.37 MRRs for choosing the best, average, and worst case for each feature for ArgoUML from structural combinations 233

5.38 Percentage of the best queries obtained from each structural combination for ArgoUML 233

5.39 MRRs for choosing the best, average, and worst case for each feature from all corpora and all structural combinations for ArgoUML 235

5.40 Percentages for each corpus where the best query was found from all corpora and all structural combinations for ArgoUML 235

5.41 MRRs and descriptive statistics of the effectiveness measures for the LMPBV corpus and Combined query type for JabRef 237

5.42 MRRs and descriptive statistics of the effectiveness measures for the LMPBV corpus and Title query type for JabRef 238

5.43 MRRs and descriptive statistics of the effectiveness measures for the CSB corpus and Combined query type for JabRef 239

5.44 MRRs and descriptive statistics of the effectiveness measures for the CSB corpus and Title query type for JabRef 239

5.45 MRRs and descriptive statistics of the effectiveness measures for the ICL corpus and Combined query type for JabRef 240

5.46 MRRs and descriptive statistics of the effectiveness measures for the ICL corpus and Title query type for JabRef 240

5.47 MRRs for choosing the best, average, and worst case for each feature for JabRef from structural combinations 244

5.48 Percentage of the best queries obtained from each structural combination for JabRef244

5.49 MRRs for choosing the best, average, and worst case for each feature from all corpora and all structural combinations for JabRef 246

5.50 Percentages for each corpus where the best query was found from all corpora and all structural combinations for JabRef 246

5.51 MRRs and descriptive statistics of the effectiveness measures for the LMPBV corpus and Combined query type for jEdit 248

5.52 MRRs and descriptive statistics of the effectiveness measures for the LMPBV corpus and Title query type for jEdit 249

5.53 MRRs and descriptive statistics of the effectiveness measures for the CSB corpus and Combined query type for jEdit 250

5.54 MRRs and descriptive statistics of the effectiveness measures for the CSB corpus and Title query type for jEdit 250

5.55 MRRs and descriptive statistics of the effectiveness measures for the ICL corpus and Combined query type for jEdit 251

5.56 MRRs and descriptive statistics of the effectiveness measures for the ICL corpus and Title query type for jEdit 251

5.57 MRRs for choosing the best, average, and worst case for each feature for jEdit from structural combinations 255

5.58 Percentage of the best queries obtained from each structural combination for jEdit 255

5.59 MRRs for choosing the best, average, and worst case for each feature from all corpora and all structural combinations for jEdit 257

5.60 Percentages for each corpus where the best query was found from all corpora and all structural combinations for jEdit 257

5.61 MRRs and descriptive statistics of the effectiveness measures for the LMPBV corpus and Combined query type for muCommander 259

5.62 MRRs and descriptive statistics of the effectiveness measures for the LMPBV corpus and Title query type for muCommander 260

5.63 MRRs and descriptive statistics of the effectiveness measures for the CSB corpus and Combined query type for muCommander 261

5.64 MRRs and descriptive statistics of the effectiveness measures for the CSB corpus and Title query type for muCommander 261

5.65 MRRs and descriptive statistics of the effectiveness measures for the ICL corpus and Combined query type for muCommander 262

5.66 MRRs and descriptive statistics of the effectiveness measures for the ICL corpus and Title query type for muCommander 262

5.67 MRRs for choosing the best, average, and worst case for each feature for muCommander from structural combinations 266

5.68 Percentage of the best queries obtained from each structural combination for muCommander 266

5.69 MRRs for choosing the best, average, and worst case for each feature from all corpora and all structural combinations for muCommander 268

5.70 Percentages for each corpus where the best query was found from all corpora and all structural combinations for muCommander 268

5.71 MRRs and descriptive statistics of the effectiveness measures for the LMPBV corpus and Combined query type for all systems 270

5.72 MRRs and descriptive statistics of the effectiveness measures for the LMPBV corpus and Title query type for all systems 271

5.73 MRRs and descriptive statistics of the effectiveness measures for the CSB corpus and Combined query type for all systems 272

5.74 MRRs and descriptive statistics of the effectiveness measures for the CSB corpus and Title query type for all systems 272

5.75 MRRs and descriptive statistics of the effectiveness measures for the ICL corpus and Combined query type for all systems 273

5.76 MRRs and descriptive statistics of the effectiveness measures for the ICL corpus and Title query type for all systems 273

5.77 MRRs for choosing the best, average, and worst case for each feature for all systems from structural combinations 277

5.78 Percentage of the best queries obtained from each structural combination for all systems 277

5.79 MRRs for choosing the best, average, and worst case for each feature from all corpora and all structural combinations for all systems 279

5.80 Percentages for each corpus where the best query was found from all corpora and all structural combinations for all systems 279

5.81 Top 10 configurations for ArgoUML 283

5.82 Top 10 configurations for ArgoUML 285

5.83 Top 10 configurations for ArgoUML 287

5.84 MRRs for choosing the best, average, and worst case for each feature for ArgoUML from structural weighting 291

5.85 The percentage of time that weighting each corpus improved the results for ArgoUML 291

5.86 MRRs for choosing the best, average, and worst case for each feature from all corpora and all structural weighting for ArgoUML 293

5.87 Percentages for each corpus where the best query was found from all corpora and all structural weighting for ArgoUML 293

5.88 Top 10 configurations for JabRef 297

5.89 Top 10 configurations for JabRef 299

5.90 Top 10 configurations for JabRef 301

5.91 MRRs for choosing the best, average, and worst case for each feature for JabRef from structural weighting 305

5.92 The percentage of time that weighting each corpus improved the results for JabRef 305

5.93 MRRs for choosing the best, average, and worst case for each feature from all corpora and all structural weighting for JabRef 307

5.94 Percentages for each corpus where the best query was found from all corpora and all structural weighting for JabRef 307

5.95 Top 10 configurations for jEdit 311

5.96 Top 10 configurations for jEdit 313

5.97 Top 10 configurations for jEdit 315

5.98 MRRs for choosing the best, average, and worst case for each feature for jEdit from structural weighting 319

5.99 The percentage of time that weighting each corpus improved the results for jEdit . 319

5.100 MRRs for choosing the best, average, and worst case for each feature from all corpora and all structural weighting for jEdit 321

5.101 Percentages for each corpus where the best query was found from all corpora and all structural weighting for jEdit 321

5.102 Top 10 configurations for muCommander 325

5.103 Top 10 configurations for muCommander 327

5.104 Top 10 configurations for muCommander 329

5.105 MRRs for choosing the best, average, and worst case for each feature for muCommander from structural weighting . . . 333

5.106 The percentage of time that weighting each corpus improved the results for muCommander . . . 333

5.107 MRRs for choosing the best, average, and worst case for each feature from all corpora and all structural weighting for muCommander . . . 335

5.108 Percentages for each corpus where the best query was found from all corpora and all structural weighting for muCommander . . . 335

5.109 Top 10 configurations for all systems . . . 339

5.110 Top 10 configurations for all systems . . . 341

5.111 Top 10 configurations for all systems . . . 343

5.112 MRRs for choosing the best, average, and worst case for each feature for all systems from structural weighting . . . 347

5.113 The percentage of time that weighting each corpus improved the results for all systems . . . 347

5.114 MRRs for choosing the best, average, and worst case for each feature from all corpora and all structural weighting for all systems . . . 349

5.115 Percentages for each corpus where the best query was found from all corpora and all structural weighting for all systems . . . 349

5.116 Top 10 configurations from all combinations for ArgoUML and the LMPBV corpus 352

5.117 Top 10 configurations from all combinations for ArgoUML and the CSB corpus . 353

5.118 Top 10 configurations from all combinations for ArgoUML and the ICL corpus . 354

5.119 MRRs for choosing the best, average, and worst case for each feature for ArgoUML from all combinations . . . 358

5.120 The percentage of time that weighting each corpus improved the results for ArgoUML . . . 358

5.121 Percentage of the best queries obtained from each structural combination for ArgoUML . . . 358

5.122 MRRs for choosing the best, average, and worst case for each feature from all corpora and all combinations for ArgoUML 360

5.123 Percentages for each corpus where the best query was found from all corpora and all combinations for ArgoUML 360

5.124 Top 10 configurations from all combinations for JabRef and the LMPBV corpus . 363

5.125 Top 10 configurations from all combinations for JabRef and the CSB corpus . . . 364

5.126 Top 10 configurations from all combinations for JabRef and the ICL corpus . . . 365

5.127 MRRs for choosing the best, average, and worst case for each feature for JabRef from all combinations 369

5.128 The percentage of time that weighting each corpus improved the results for JabRef 369

5.129 Percentage of the best queries obtained from each structural combination for JabRef369

5.130 MRRs for choosing the best, average, and worst case for each feature from all corpora and all combinations for JabRef 371

5.131 Percentages for each corpus where the best query was found from all corpora and all combinations for JabRef 371

5.132 Top 10 configurations from all combinations for jEdit and the LMPBV corpus . . 373

5.133 Top 10 configurations from all combinations for jEdit and the CSB corpus 374

5.134 Top 10 configurations from all combinations for jEdit and the ICL corpus 375

5.135 MRRs for choosing the best, average, and worst case for each feature for jEdit from all combinations 379

5.136 The percentage of time that weighting each corpus improved the results for jEdit . 379

5.137 Percentage of the best queries obtained from each structural combination for jEdit 379

5.138 MRRs for choosing the best, average, and worst case for each feature from all corpora and all combinations for jEdit 381

5.139 Percentages for each corpus where the best query was found from all corpora and all combinations for jEdit 381

5.140 Top 10 configurations from all combinations for muCommander and the LMPBV corpus . . . 384

5.141 Top 10 configurations from all combinations for muCommander and the CSB corpus . . . 385

5.142 Top 10 configurations from all combinations for muCommander and the ICL corpus 386

5.143 MRRs for choosing the best, average, and worst case for each feature for muCommander from all combinations . . . 390

5.144 The percentage of time that weighting each corpus improved the results for muCommander . . . 390

5.145 Percentage of the best queries obtained from each structural combination for muCommander . . . 390

5.146 MRRs for choosing the best, average, and worst case for each feature from all corpora and all combinations for muCommander . . . 392

5.147 Percentages for each corpus where the best query was found from all corpora and all combinations for muCommander . . . 392

5.148 Top 10 configurations from all combinations for all systems and the LMPBV corpus 394

5.149 Top 10 configurations from all combinations for all systems and the CSB corpus . 395

5.150 Top 10 configurations from all combinations for all systems and the ICL corpus . 396

5.151 MRRs for choosing the best, average, and worst case for each feature for all systems from all combinations . . . 400

5.152 The percentage of time that weighting each corpus improved the results for all systems . . . 400

5.153 Percentage of the best queries obtained from each structural combination for all systems . . . 400

5.154 MRRs for choosing the best, average, and worst case for each feature from all corpora and all combinations for all systems . . . 402

5.155 Percentages for each corpus where the best query was found from all corpora and all combinations for all systems . . . 402

A.1 Example Title Query for ArgoUML 447

A.2 Example Output Title Queries for the ICL corpus for ArgoUML and Feature 549. The {} are used to indicate that every combination of the contained values are used. 447

LIST OF FIGURES

1.1 Text Retrieval 5

3.1 The effectiveness measure for three configurations (*Title*, *Description*, and *Combined*) of the LDA-based FLT applied to 91 ArgoUML features, 38 JabRef features, 149 jEdit features, 90 muCommander features, 93 Mylyn features, 157 Rhino features, and all 618 features. 55

3.2 The effectiveness measure for 36 configurations (α/β pairs) of the LDA-based FLT applied to all 618 features. 61

3.3 The effectiveness measure for three configurations (*Predicted*, $Heuristic_1$, and $Heuristic_2$) of the LDA-based FLT applied to 91 ArgoUML features, 38 JabRef features, 149 jEdit features, and 157 Rhino features. 64

3.4 Source code for Mylyn method *BugzillaAttachmentHandler.uploadAttachment*. . 69

4.1 The Effectiveness Measures for Weighting Leading Comments Alone using Weighting Factors of 1,2,4, and 8 89

4.2 The Effectiveness Measures for Weighting Method Names Alone using Weighting Factors of 1,2,4, and 8 93

4.3 The Effectiveness Measures for Weighting Parameters Alone using Weighting Factors of 1,2,4, and 8 97

4.4 The Effectiveness Measures for Weighting Body Comments Alone using Weighting Factors of 1,2,4, and 8 101

4.5 The Effectiveness Measures for Weighting Local Variables Alone using Weighting Factors of 1,2,4, and 8 105

4.6 The Top Configurations and unweighted configuration for Each System and All Systems Combined 114

4.6 The Top Configurations and unweighted configuration for Each System and All Systems Combined 115

4.6 The Top Configurations and unweighted configuration for Each System and All Systems Combined . . . 116

4.7 The Search Process - An initial population is selected of most likely candidates, pairs are selected for crossover and then mutation, new population is created and the process repeats . . . 152

4.8 The unweighted configuration, top configuration from the untrained population, and results of 50 iterations of the the genetic algorithm for population sizes of 10, 25, and 50 based on MRR using the MRR and silhouette coefficient fitness functions . . . 158

5.1 Example Indri Model . . . 164

5.2 Document Extraction Process . . . 166

5.3 Different types of structured method documents . . . 168

5.4 Example Method . . . 168

5.5 Example Document . . . 169

5.6 The Effectiveness Measures for the three different query types (Title, Description, Combined) for ArgoUML . . . 186

5.7 Stacked bargraphs representing the distance from the best query to the average (bottom) and the worst(top) for ArgoUML. Graph is ordered by distance from best to worst. . . . 188

5.7 Stacked bargraphs representing the distance from the best query to the average (bottom) and the worst(top) for ArgoUML. Graph is ordered by distance from best to worst. . . . 189

5.8 Stacked bargraphs representing the distance from the best query to the average (bottom) and the worst(top) for ArgoUML. Graph is ordered by distance from best to worst. . . . 191

5.9 The Effectiveness Measures for the three different query types (Title, Description, Combined) for JabRef . . . 194

5.10 Stacked bargraphs representing the distance from the best query to the average (bottom) and the worst(top) for JabRef. Graph is ordered by distance from best to worst. . . . 196

5.10 Stacked bargraphs representing the distance from the best query to the average (bottom) and the worst(top) for JabRef. Graph is ordered by distance from best to worst. . . . 197

5.11 Stacked bargraphs representing the distance from the best query to the average (bottom) and the worst(top) for JabRef. Graph is ordered by distance from best to worst. . . . 199

5.12 The Effectiveness Measures for the three different query types (Title, Description, Combined) for jEdit . . . 202

5.13 Stacked bargraphs representing the distance from the best query to the average (bottom) and the worst(top) for jEdit. Graph is ordered by distance from best to worst. . . . 204

5.13 Stacked bargraphs representing the distance from the best query to the average (bottom) and the worst(top) for jEdit. Graph is ordered by distance from best to worst. . . . 205

5.14 Stacked bargraphs representing the distance from the best query to the average (bottom) and the worst(top) for jEdit. Graph is ordered by distance from best to worst. . . . 207

5.15 The Effectiveness Measures for the three different query types (Title, Description, Combined) for muCommander . . . 210

5.16 Stacked bargraphs representing the distance from the best query to the average (bottom) and the worst(top) for muCommander. Graph is ordered by distance from best to worst. . . . 212

5.16 Stacked bargraphs representing the distance from the best query to the average (bottom) and the worst(top) for muCommander. Graph is ordered by distance from best to worst. . . . 213

5.17 Stacked bargraphs representing the distance from the best query to the average (bottom) and the worst(top) for muCommander. Graph is ordered by distance from best to worst. . . . 215

5.18 The Effectiveness Measures for the three different query types (Title, Description, Combined) for all systems . . . 218

5.19 Stacked bargraphs representing the distance from the best query to the average (bottom) and the worst(top) for all systems. Graph is ordered by distance from best to worst. . . . 220

5.19 Stacked bargraphs representing the distance from the best query to the average (bottom) and the worst(top) for all systems. Graph is ordered by distance from best to worst. 221

5.20 Stacked bargraphs representing the distance from the best query to the average (bottom) and the worst(top) for all systems. Graph is ordered by distance from best to worst. 223

5.21 Stacked bargraphs representing the distance from the best query from structural combinations to the average (bottom) and the worst(top) for ArgoUML. Graph is ordered by distance from best to worst. 231

5.21 Stacked bargraphs representing the distance from the best query from structural combinations to the average (bottom) and the worst(top) for ArgoUML. Graph is ordered by distance from best to worst. 232

5.22 Stacked bargraphs representing the distance from the best query to the average (bottom) and the worst(top) from structural combinations for ArgoUML. Graph is ordered by distance from best to worst. 234

5.23 Stacked bargraphs representing the distance from the best query from structural combinations to the average (bottom) and the worst(top) for JabRef. Graph is ordered by distance from best to worst. 242

5.23 Stacked bargraphs representing the distance from the best query from structural combinations to the average (bottom) and the worst(top) for JabRef. Graph is ordered by distance from best to worst. 243

5.24 Stacked bargraphs representing the distance from the best query to the average (bottom) and the worst(top) from structural combinations for JabRef. Graph is ordered by distance from best to worst. 245

5.25 Stacked bargraphs representing the distance from the best query from structural combinations to the average (bottom) and the worst(top) for jEdit. Graph is ordered by distance from best to worst. 253

5.25 Stacked bargraphs representing the distance from the best query from structural combinations to the average (bottom) and the worst(top) for jEdit. Graph is ordered by distance from best to worst. 254

5.26 Stacked bargraphs representing the distance from the best query to the average (bottom) and the worst(top) from structural combinations for jEdit. Graph is ordered by distance from best to worst. 256

5.27 Stacked bargraphs representing the distance from the best query from structural combinations to the average (bottom) and the worst(top) for muCommander. Graph is ordered by distance from best to worst. 264

5.27 Stacked bargraphs representing the distance from the best query from structural combinations to the average (bottom) and the worst(top) for muCommander. Graph is ordered by distance from best to worst. 265

5.28 Stacked bargraphs representing the distance from the best query to the average (bottom) and the worst(top) from structural combinations for muCommander. Graph is ordered by distance from best to worst. 267

5.29 Stacked bargraphs representing the distance from the best query from structural combinations to the average (bottom) and the worst(top) for all systems. Graph is ordered by distance from best to worst. 275

5.29 Stacked bargraphs representing the distance from the best query from structural combinations to the average (bottom) and the worst(top) for all systems. Graph is ordered by distance from best to worst. 276

5.30 Stacked bargraphs representing the distance from the best query to the average (bottom) and the worst(top) from structural combinations for all systems. Graph is ordered by distance from best to worst. 278

5.31 The top weighting configurations and flat configuration for ArgoUML and the LMPBV corpus. The x axis is the configuration, while the y axis is the effectiveness measure. 282

5.32 The top weighting configurations and flat configuration for ArgoUML and the CSB corpus. The x axis is the configuration, while the y axis is the effectiveness measure. 284

5.33 The top weighting configurations and flat configuration for ArgoUML and the ICL corpus. The x axis is the configuration, while the y axis is the effectiveness measure. 286

5.34 Stacked bargraphs representing the distance from the best query from structural weighting to the average (bottom) and the worst(top) for ArgoUML. Graph is ordered by distance from best to worst. 289

5.34 Stacked bargraphs representing the distance from the best query from structural weighting to the average (bottom) and the worst(top) for ArgoUML. Graph is ordered by distance from best to worst. 290

5.35 Stacked bargraphs representing the distance from the best query to the average (bottom) and the worst(top) from structural weighting for ArgoUML. Graph is ordered by distance from best to worst. 292

5.36 The top weighting configurations and flat configuration for JabRef and the LMPBV corpus. The x axis is the configuration, while the y axis is the effectiveness measure. 296

5.37 The top weighting configurations and flat configuration for JabRef and the CSB corpus. The x axis is the configuration, while the y axis is the effectiveness measure. 298

5.38 The top weighting configurations and flat configuration for JabRef and the ICL corpus. The x axis is the configuration, while the y axis is the effectiveness measure. 300

5.39 Stacked bargraphs representing the distance from the best query from structural weighting to the average (bottom) and the worst(top) for JabRef. Graph is ordered by distance from best to worst. 303

5.39 Stacked bargraphs representing the distance from the best query from structural weighting to the average (bottom) and the worst(top) for JabRef. Graph is ordered by distance from best to worst. 304

5.40 Stacked bargraphs representing the distance from the best query to the average (bottom) and the worst(top) from structural weighting for JabRef. Graph is ordered by distance from best to worst. 306

5.41 The top weighting configurations and flat configuration for jEdit and the LMPBV corpus. The x axis is the configuration, while the y axis is the effectiveness measure. 310

5.42 The top weighting configurations and flat configuration for jEdit and the CSB corpus. The x axis is the configuration, while the y axis is the effectiveness measure. 312

5.43 The top weighting configurations and flat configuration for jEdit and the ICL corpus. The x axis is the configuration, while the y axis is the effectiveness measure. 314

5.44 Stacked bargraphs representing the distance from the best query from structural weighting to the average (bottom) and the worst(top) for jEdit. Graph is ordered by distance from best to worst. 317

5.44 Stacked bargraphs representing the distance from the best query from structural weighting to the average (bottom) and the worst(top) for jEdit. Graph is ordered by distance from best to worst. 318

5.45 Stacked bargraphs representing the distance from the best query to the average (bottom) and the worst(top) from structural weighting for jEdit. Graph is ordered by distance from best to worst. 320

5.46 The top weighting configurations and flat configuration for muCommander and the LMPBV corpus. The x axis is the configuration, while the y axis is the effectiveness measure. 324

5.47 The top weighting configurations and flat configuration for muCommander and the CSB corpus. The x axis is the configuration, while the y axis is the effectiveness measure. 326

5.48 The top weighting configurations and flat configuration for muCommander and the ICL corpus. The x axis is the configuration, while the y axis is the effectiveness measure. 328

5.49 Stacked bargraphs representing the distance from the best query from structural weighting to the average (bottom) and the worst(top) for muCommander. Graph is ordered by distance from best to worst. 331

5.49 Stacked bargraphs representing the distance from the best query from structural weighting to the average (bottom) and the worst(top) for muCommander. Graph is ordered by distance from best to worst. 332

5.50 Stacked bargraphs representing the distance from the best query to the average (bottom) and the worst(top) from structural weighting for muCommander. Graph is ordered by distance from best to worst. 334

5.51 The top weighting configurations and flat configuration for all systems and the LMPBV corpus. The x axis is the configuration, while the y axis is the effectiveness measure. 338

5.52 The top weighting configurations and flat configuration for all systems and the CSB corpus. The x axis is the configuration, while the y axis is the effectiveness measure. 340

5.53 The top weighting configurations and flat configuration for all systems and the ICL corpus. The x axis is the configuration, while the y axis is the effectiveness measure. 342

5.54 Stacked bargraphs representing the distance from the best query from structural weighting to the average (bottom) and the worst(top) for all systems. Graph is ordered by distance from best to worst. 345

5.54 Stacked bargraphs representing the distance from the best query from structural weighting to the average (bottom) and the worst(top) for all systems. Graph is ordered by distance from best to worst. 346

5.55 Stacked bargraphs representing the distance from the best query to the average (bottom) and the worst(top) from structural weighting for all systems. Graph is ordered by distance from best to worst. 348

5.56 The top configurations and flat configuration for ArgoUML and the LMPBV corpus. The x axis is the configuration, while the y axis is the effectiveness measure. 352

5.57 The top configurations and flat configuration for ArgoUML and the CSB corpus. The x axis is the configuration, while the y axis is the effectiveness measure. . . . 353

5.58 The top configurations and flat configuration for ArgoUML and the ICL corpus. The x axis is the configuration, while the y axis is the effectiveness measure. . . . 354

5.59 Stacked bargraphs representing the distance from the best query from all combinations to the average (bottom) and the worst(top) for ArgoUML. Graph is ordered by distance from best to worst. 356

5.59 Stacked bargraphs representing the distance from the best query from all combinations to the average (bottom) and the worst(top) for ArgoUML. Graph is ordered by distance from best to worst. 357

5.60 Stacked bargraphs representing the distance from the best query to the average (bottom) and the worst(top) from all combinations for ArgoUML. Graph is ordered by distance from best to worst. 359

5.61 The top configurations and flat configuration for JabRef and the LMPBV corpus. The x axis is the configuration, while the y axis is the effectiveness measure. . . . 363

5.62 The top configurations and flat configuration for JabRef and the CSB corpus. The x axis is the configuration, while the y axis is the effectiveness measure. 364

5.63 The top configurations and flat configuration for JabRef and the ICL corpus. The x axis is the configuration, while the y axis is the effectiveness measure. 365

5.64 Stacked bargraphs representing the distance from the best query from all combinations to the average (bottom) and the worst(top) for JabRef. Graph is ordered by distance from best to worst. 367

5.64 Stacked bargraphs representing the distance from the best query from all combinations to the average (bottom) and the worst(top) for JabRef. Graph is ordered by distance from best to worst. 368

5.65 Stacked bargraphs representing the distance from the best query to the average (bottom) and the worst(top) from all combinations for JabRef. Graph is ordered by distance from best to worst. 370

5.66 The top configurations and flat configuration for jEdit and the LMPBV corpus. The x axis is the configuration, while the y axis is the effectiveness measure. . . . 373

5.67 The top configurations and flat configuration for jEdit and the CSB corpus. The x axis is the configuration, while the y axis is the effectiveness measure. 374

5.68 The top configurations and flat configuration for jEdit and the ICL corpus. The x axis is the configuration, while the y axis is the effectiveness measure. 375

5.69 Stacked bargraphs representing the distance from the best query from all combinations to the average (bottom) and the worst(top) for jEdit. Graph is ordered by distance from best to worst. 377

5.69 Stacked bargraphs representing the distance from the best query from all combinations to the average (bottom) and the worst(top) for jEdit. Graph is ordered by distance from best to worst. 378

5.70 Stacked bargraphs representing the distance from the best query to the average (bottom) and the worst(top) from all combinations for jEdit. Graph is ordered by distance from best to worst. 380

5.71 The top configurations and flat configuration for muCommander and the LMPBV corpus. The x axis is the configuration, while the y axis is the effectiveness measure. 384

5.72 The top configurations and flat configuration for muCommander and the CSB corpus. The x axis is the configuration, while the y axis is the effectiveness measure. 385

5.73 The top configurations and flat configuration for muCommander and the ICL corpus. The x axis is the configuration, while the y axis is the effectiveness measure. 386

5.74 Stacked bargraphs representing the distance from the best query from all combinations to the average (bottom) and the worst(top) for muCommander. Graph is ordered by distance from best to worst. 388

5.74 Stacked bargraphs representing the distance from the best query from all combinations to the average (bottom) and the worst(top) for muCommander. Graph is ordered by distance from best to worst. 389

5.75 Stacked bargraphs representing the distance from the best query to the average (bottom) and the worst(top) from all combinations for muCommander. Graph is ordered by distance from best to worst. 391

5.76 The top configurations and flat configuration for all systems and the LMPBV corpus. The x axis is the configuration, while the y axis is the effectiveness measure. 394

5.77 The top configurations and flat configuration for all systems and the CSB corpus. The x axis is the configuration, while the y axis is the effectiveness measure. . . . 395

5.78 The top configurations and flat configuration for all systems and the ICL corpus. The x axis is the configuration, while the y axis is the effectiveness measure. . . . 396

5.79 Stacked bargraphs representing the distance from the best query from all combinations to the average (bottom) and the worst(top) for all systems. Graph is ordered by distance from best to worst. 398

5.79 Stacked bargraphs representing the distance from the best query from all combinations to the average (bottom) and the worst(top) for all systems. Graph is ordered by distance from best to worst. 399

5.80 Stacked bargraphs representing the distance from the best query to the average (bottom) and the worst(top) from all combinations for all systems. Graph is ordered by distance from best to worst. 401

5.81 Recommended Corpus Structure 427

Chapter 1

INTRODUCTION

Understanding a software system's implementation is a crucial part of a developer's job. During maintenance and evolution, before any changes can be made, bugs fixed, or features added, the developer must first understand the system's implementation and locate source code elements specific to the current maintenance task. Software maintenance and evolution accounts for 60-80% [Alkhatib, 1992; Boehm, 1981; Erlikh, 2000; Standish, 1984] of the cost and effort during the software life cycle, and during this time, over half [De Lucia, Risi, Rizzi, and Scanniello, 2008; Müller, Jahnke, Smith, Storey, Tilley, and Wong, 2000] of developer effort is spent in trying to understand the software system. Given these numbers and the increasing size and complexity of software systems, the need for tools to aid in program understanding is evident.

Although different tools and techniques have been developed to aid in program comprehension activities, there still exists a gap in knowledge. There are two issues that confound the matter. First, software maintenance is not only an engineering problem, it is also an information management problem. The number of source and textual artifacts related to a software project are spread across a wide array of databases and repositories and can span gigabytes or terabytes in size. This makes finding relevant information to a specific task a nontrivial matter. Second, even within the same domain or company, no two software products are alike. Software implementations and documentation differ. Tools and techniques developed for program comprehension must

be general enough to apply to a wide array of systems while still returning useful results [Marcus and Menzies, 2010].

Many techniques for program comprehension have incoporated text retrieval [Antoniol, Canfora, Casazza, Lucia, and Merlo, 2002; Lu and Kan, 2007; Marcus, Sergeyev, Rajlich, and Maletic, 2004; Maskeri, Sarkar, and Heafield, 2008a]. Text retrieval (TR), also referred to as document retrieval, is the study of information retrieval that focuses on retrieving documents of unstructured text by the use of natural language queries. TR methods differ from pattern matching utilities like *grep* in that they are based on statistical models that find relevant text by calculating the similarity between the user's query and the text. Examples of text retrieval techniques include the vector space model (VSM) [Salton and McGill, 1986], latent semantic indexing (LSI) [Deerwester, Dumais, Furnas, Landauer, and Harshman, 1990], and latent Dirichlet allocation (LDA) [Blei, Ng, and Jordan, 2003].

Previous research into text retrieval techniques has attempted to combine textual information with multiple sources of structural (e.g., dependency graphs) and dynamic information (e.g., execution traces). By combining textual information with structural information and dynamic information, researchers have found an improvement over using textual information alone for some systems. There are two existing problems with the existing research. First, previous research has focused on latent semantic indexing (LSI) and the vector space model (VSM), discussed in more detail later. However, more sophisticated techniques such as latent Dirichlet allocation (LDA) and other topic modeling techniques (e.g., probabilistic LSI [Hofmann, 1999], Pachinko allocation [Li, Blei, and McCallum, 2012; Li and McCallum, 2006; Mimno, Li, and McCallum, 2007], hierarchical Dirichlet processes [Teh, Jordan, Beal, and Blei, 2006]) have been shown to produce useful results in other domains along with program comprehension tasks [Baldi, Lopes, Linstead, and Ba-

jracharya, 2008; Hindle, Godfrey, and Holt, 2009; Lukins, Kraft, and Etzkorn, 2008, 2010]. Topic models differ from other TR techniques in that they attempt to model latent topics (e.g., the topic of *printing* in a software system) in documents. Topic modeling allows for comparisons between documents across topics instead of terms. Second, little research has studied the importance of each structural location (e.g., whether method calls are as important as method names?).

This chapter begins with an explanation of the software change process. Then, an overview of the text retrieval process is presented, along with a discussion of how TR may be applied to the software change process. In particular, the work in this dissertation focuses on feature location. This is the process that a developer performs when they are trying to identify the first source code element that will need to be changed to fix a bug or implement a feature. This problem is the main motivation for my studies.

1.1 Software Change

Software change is the process of adding, removing, or replacing functionality in an existing software system [Rajlich, 2011]. The process can be broken up into six distinct phases: initiation, concept or feature location, impact analysis, actualization, verification, and conclusion.

In initiation, a change request is created. The purpose of the change request may vary but the two main categories of change requests include bug reports and feature requests. In the case of bug reports, a user of the system is identifying an unwanted feature of the software system. In the case of the feature request, the user is asking that new functionality be added to the system. Once a bug report is created, triaging is often performed to determine the priority of the change request and to hopefully identify the developers that are most capable of making the change.

Once a change request is obtained by the developer, concept or feature location is performed. The difference between concept location and feature location is subtle [Rajlich and Wilde,

2002]. In the case of concept location, the developer is attempting to identify domain concepts within the change request and then to identify the elements of the source code that relate to that concept. In the case of feature location, the developer is looking at the specific functionality identified in the change request and then to identify where that functionality is implemented. The purpose of both of these tasks is to identify a starting point for making the initial changes required to complete the change request.

Once a starting point is selected, impact analysis is performed to determine the strategy to perform the change and to measure the effect making the change will have on the entire system. One way of measuring this impact would be to identify the number of classes or methods that would be required to change in order to complete the change request.

The remaining phases are responsible for making the change and adding it to the current version of the software system. Actualization is the process of actually executing the strategy determined in impact analysis and adding the new code to make the change. Verification is the process of ensuring correctness of the change. Finally, the new change is added to the repository and the change request is concluded.

The research described in this dissertation focuses on the problem of feature location. Change requests will be used as queries for TR. The ranked list of results represent the order of likelihood that each method in the system implements the functionality referred to in the change request.

1.2 Overview of Text Retrieval

This section presents an outline of the TR process as it is applied to program comprehension tasks. Most recent work in the use of TR for software maintenance tasks operates on models of source code instead of the source code itself. Techniques involved in this process accept a corpus

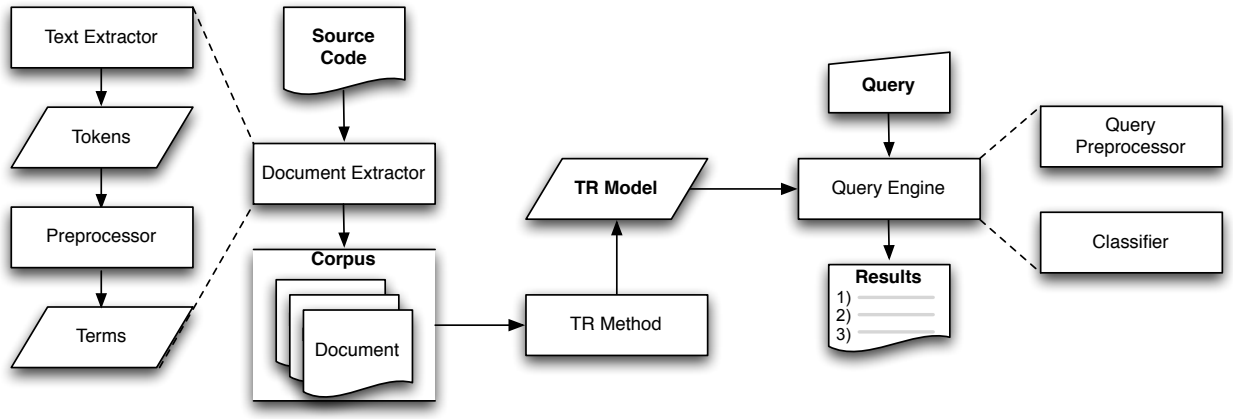


Figure 1.1: Text Retrieval

that represents the source code being analyzed. In TR, a corpus is a collection of documents. The documents that make up a corpus are created from stripping literals, comments, identifiers, or some combination from elements of source code such as files, packages, classes, or methods. The general approach to building a corpus from source code and performing TR is shown in Figure 1.1 and is performed by the following steps:

1. **Document Extraction** The source code is parsed according to a user defined granularity, such as methods or classes. Each method or class constitutes a document in the source code and is comprised of tokens from literals, comments, identifiers, or some combination thereof. These tokens are sent to a preprocessor. Here, preprocessing techniques such as identifier splitting (i.e., splitting identifiers into their component terms), stop word filtering (i.e., removing commonly occurring words from the corpus), short word removal (i.e., removing words that do not pass a length threshold), or stemming (i.e., reducing inflected terms to their base) may be applied. The new sets of terms constitute the documents.

2. **Build the TR model** The corpus is provided as input to the TR method and a new statistical model based on the method is produced.

3. **Formulate the Query** Queries are formed in one of two ways. Either a developer selects a set of words manually, or queries are automatically formulated by the system from other software artifacts. If preprocessing techniques were applied in step one, the same steps will be applied to the query.

4. **Rank Documents** The query engine takes as input the TR model obtained from step 2 and documents are ranked according to a classifier, or similarity measure. The classifier depends on the TR method used. Based on the measure, documents are ranked according to similarity with the query.

Although these steps define the general procedure for TR, steps may be altered or added according to the approach used.

Before discussing the sources of text in source code and giving details about common TR methods, the following offers a brief introduction to the problems of scale and query type. Both problems have an effect on the choices made when identifying the proper model and TR technique to use.

- **Scale** Scale has an important impact on TR [Newby, 2000]. In general TR, there are usually five different scales identified: web search (e.g., Google), enterprise (e.g., corporations), institutional (e.g., universities and colleges), domain-specific (e.g., medicine or law), or personal (e.g., file searching on your personal computer) [Manning, Raghavan, and Schutze, 2008]. Each of these different techniques have various effects on the level of search that can

be done. For instance, when searching the web, a search engine must be capable of handling billions of documents and reasonable results are expected in a short time frame. This requires large networks of computers and algorithms that are highly parallelizable. On the other hand, searching your file system can be accomplished with a single core system.

When searching software, there are often two different scales considered when performing software engineering related tasks. The first focuses on performing searches in a software repository containing multiple software systems. This is the goal of the system Sourcerer [Bajracharya, Ngo, Linstead, Dou, Rigor, Baldi, and Lopes, 2006]. These types of searches require models that are general enough to return good results when trained or built on multiple systems. This may lead to less customization and fine tuning of a particular model and may not be as good for a software developer that is focused on one system. The second focuses on a single system, which requires less computational resources and for models to be trained for a particular system instead of remaining general. The trade-off is that there is less text and information to build the model in the latter approach. The focus of my research is on searching individual software systems for the purposes of feature location.

- **Query Type** Along with scale, the types of queries issued by the developer also have an impact on the software search process. The types of queries used when searching software typically break down into one of two categories. The first type uses natural language fragments as the query. These queries are typically used when the developer is searching using information obtained from a feature request or a bug report. However, it is also possible to use this type of query when comparing source code elements based on the type of language used in both elements' source code. The second type of query uses program semantics or

the source code itself as the query [Reiss, 2009]. In these types of tasks, features about the source code, such as what methods the source code element calls, what libraries or modules it makes use of, or the parse trees of the two elements, are compared. In this case, the text of the source code may be disregarded entirely. Such queries may be used when the developer wishes to identify code clones or find examples of common API usage. The focus of this research is on TR so natural language queries will be used.

The next subsection discusses sources for terms in source code and gives a brief overview of common TR models. As discussed in Chapter 2, each of these models has been used with varying levels of success in previous research studies of automated techniques for feature location tasks.

1.2.1 The Source Code Lexicon

TR techniques are used to index and query documents of text. In the case of source code, steps must be taken to transform programs from source code elements written in a given programming language to documents of natural language text. This is typically done by extracting the natural language tokens from the source code and throwing out the abstract syntax. In this way, terms may be extracted from comments, literals, and identifiers embedded throughout the source code. Preprocessing can then be done to convert tokens of text to terms for our documents, where a document is defined at some particular granularity of source code (e.g., methods, classes, files).

The set of terms that are used by a particular software system is a software lexicon, and the evolution and usage of terms in source code has been a subject for research [Antoniol, Gueheneuc, Merlo, and Tonella, 2007; Fluri, Wursch, and Gall, 2007; Haiduc and Marcus, 2008]. Software lexicons have been studied in the contexts of program comprehension [Abebe, Haiduc, Marcus,

Tonella, and Antoniol, 2009] and quality assessment [Lawrie, Morrell, Feild, and Binkley, 2006]. In a preliminary study [Biggers, Eddy, Kraft, and Etzkorn, 2011] the contributions of each of these sources of terms were investigated. In the study, software lexicons were denoted by $L = \{t_1, t_2, ..., t_n\}$, where $|L| = n$. The focus of the study was on $L_{Src} = L_{Id} \cup L_{Co} \cup L_{Li}$, where L_{Id} is the set of identifier terms, L_{Co} is the set of comment terms, and L_{Li} is the set of (string) literal terms.

A set of 125 software systems composed of a combination of industry-like systems with enforced coding standards, issue tracking systems, and maintenance developers, and open source systems obtained from online software repositories were used to identify the characteristics of each software lexicon. Three density measures were obtained for each lexicon.

The first density measure is *lexicon density*, or the percentage of the terms in the source lexicon that appear in the lexicon (i.e., the amount of the source lexicon that a lexicon contains).

$$LD(L_i) = \frac{|L_i|}{|L_{Src}|}$$

where $L_i \in \{L_{Id}, LCo, L_{Li}\}$.

The second density measure is *unique term density*, or the percentage of the lexicon's terms that are unique to the lexicon (i.e., the amount of unique information that a lexicon contains).

$$UTD(L_i, L_j, L_k) = \frac{L_i - (|L_j \cup L_k)|}{|L_i|}$$

where L_i, L_j, and L_k are distinct lexicons in $\{L_{Id}, L_{Co}, L_{Li}\}$.

The remaining measure is *unique term contribution*. This measure requires the set of terms in the source lexicon that appear in exactly one of $\{L_{Id}, L_{Co}, L_{Li}\}$. This set is computed, U_{Src}, using the symmetric difference operator:

$$U_{Src} = L_{Id} \triangle L_{Co} \triangle L_{Li}$$

Next, for each lexicon the percentage of the terms in U_{Src} that appear in the lexicon (i.e., the amount of U_{Src} that a lexicon contributes) is computed.

$$UTC(L_i, L_j, L_k) = \frac{|L_i - (L_j \cup L_k)|}{|U_{Src}|}$$

where L_i, L_j, and L_k are distinct lexicons in $\{L_{Id}, L_{Co}, L_{Li}\}$.

The results of this study showed that about 75% of the terms in a system's source code lexicon appear in the system's identifier lexicon, whereas only about 40% of the terms in a system's source code lexicon appear in the system's comment lexicon. The density of unique terms in a software system's identifiers remained consistent throughout the 125 systems, however it varied widely for comments and literals. Finally, identifiers are the main contributor of unique terms in a software system; however, literals and comments do contribute a few unique terms.

Although the results of some TR tasks may be improved by omitting one or more of the lexicons, the presence of unique terms in each of the three lexicons makes including these lexicons important for some models. Chapters 4 and 5 break these lexicons down into smaller forms and consider specific structural locations of a method. Furthermore, Chapter 4 will refer back to these measures and consider their relationship to structural weighting.

1.2.2 Vector Space Model (VSM)

One of the simplest models for TR is the vector space model (VSM) [Salton and McGill, 1986]. VSM is a framework for TR that models documents and queries by assigning non-binary weights to index terms. VSM has been applied to multiple software engineering tasks [Pinheiro and Goguen, 1996; Ramesh and Dhar, 1992; Runeson, Alexandersson, and Nyholm, 2007]. More formally, documents are modeled as vectors of weights, where each weight w is positive and non-binary. Let each $w_{i,j}$ be the weight associated with the pair (k_i, dj) where k_i is the ith term in

the vocabulary of the corpus and d_j is the jth document. Then, the query is of the form $\vec{q} = (w_{1,q}, w_{2,q}, ..., w_{n,q})$ and each document is of the form $\vec{d}_j = (w_{1,j}, w_{2,j}, ..., w_{n,j})$. Each model is represented as an n-dimensional vector. This model allows documents to be compared with queries when only a partial match exists.

We can evaluate the degree of similarity between the query (q) and each document (d_j) by finding the correlation between the document and the query. While many measures for correlation exist, the most commonly applied in this field is the cosine of the angle between the two vectors [Antoniol, Canfora, Casazza, and De Lucia, 2000]. In this context, the measure is referred to as *cosine similarity* and is computed by the following formula:

$$sim(d_j, q) = \frac{\sum_{i=1}^{n} w_{i,j} w_{i,q}}{\sqrt{\sum_{i=1}^{n} w_{i,j}^2} \sqrt{\sum_{i=1}^{n} w_{i,q}^2}}$$

Documents are ranked and returned in descending order according to this similarity measure. For the purposes of weighting, many term-weighting techniques exist [Salton and Buckley, 1988], but the most common in the field is term frequency-inverse document frequency, *tf-idf*.

The use of *tf-idf* is based on ideas from clustering that attempt to identify the features of a set that best serve to characterize the data. In the case of TR, the features of a set of documents are the terms. To identify the most important terms in a set of documents, two components are used. The first is the normalized term frequency, *tf*, which identifies the terms most important to a document by identifying the terms that are most frequently used. We can obtain *tf* for a term by dividing the term's frequency with the maximum combined frequency of all terms in the document, i.e., by dividing the number of occurrences of a term by the total number of occurrences of all terms in the document. This only solves part of the problem, however, as terms that appear in a large percentage of the documents are insufficient for differentiating relevant documents from those that

are irrelevant. For this reason a second factor is used, the inverse document frequency, *idf*. This factor helps to identify important features by factoring in the number of documents in which a term appears and giving them a lower weight than more unique terms. The resulting formula for *tf-idf* is:

$$w_{i,j} = \frac{freq_{i,j}}{maxfreq_{i,j}} \times log\frac{N}{n_i}$$

In the given formula, $freq_{i,j}$ is the frequency of the ith term in the jth document, $maxfreq_{i,j}$ is the maximum frequency of all terms inside the document, N is the total number of documents, and n_i is the number of documents in which the ith term appears.

1.2.3 Latent Semantic Indexing (LSI)

VSM suffers from two main issues [Salton and McGill, 1986]. The first is that many unrelated documents might be returned from the answer set. The second issue comes from the model's inability to recognize the relationship between terms. VSM is unable to model synonymy, when two words have the same meaning, or polysemy, when one word has multiple meanings. As a result, the model is unable to find relevant documents when the queries given do not contain any of the indexed terms, and may miss potentially relevant documents when multiple synonyms are used in the corpus. One of the first TR techniques to attempt to handle these problems is latent semantic indexing (LSI) [Deerwester et al., 1990]. Like VSM, LSI has been applied to multiple areas of software engineering, including traceability [De Lucia, Oliveto, and Tortora, 2008; Jiang, Nguyen, Chen, Jaygarl, and Chang, 2008; Lormans and van Deursen, 2005], clone detection [Tairas and Gray, 2009], impact analysis [David, 2008; Poshyvanyk, Marcus, Ferenc, and Gyimóthy, 2009], and feature location [Marcus, Rajlich, Buchta, Petrenko, and Sergeyev, 2005].

LSI attempts to index documents by the concepts described within each document instead

of the terms. To do so, each document and query vector is mapped from the space of weighted terms to a lower dimensional space that is associated with the concepts. The goal is that retrieval in this reduced space is an improvement over the VSM.

To perform the reduction required by LSI, the use of singular value decomposition (SVD) is applied to the VSM. SVD is a form of factorization performed on a matrix of real or complex numbers. The matrix of focus for this technique is the term-document association matrix obtained from VSM. SVD decomposes this matrix, $\vec{M}$, into three components.

$$\vec{M} = \vec{U}\vec{S}\vec{V}^t$$

The three components can be explained as follows:

- $\vec{U}$, is the matrix of eigenvectors from the term-to-term correlation matrix, $\vec{M}\vec{M}^t$
- $\vec{S}$, is an n-square diagonal matrix of singular values, where n is the rank of $\vec{M}$
- $\vec{V}^t$, is the matrix of eigenvectors from the document-to-document correlation matrix, $\vec{M}^t\vec{M}$

By selecting only the k largest singular values and their corresponding columns from U, S, and V^t, a reduced space results that models the concepts of the corpus. The value k must be chosen correctly as to be large enough to fit all the concepts from the corpus, but small enough to filter out all irrelevant details that were present in the non-reduced VSM.

To query this model, a query may be treated as a pseudo-document in the VSM and reduced along with the rest of the corpus. Querying can then take place by using the cosine similarity presented in the earlier section [Kuhn, Ducasse, and Gírba, 2007].

There still exist problems with LSI. While LSI performs better than VSM on synonymy, it still has weaknesses, and the problem of polysemy still remains. These problems are addressed by the next model.

1.2.4 Latent Dirichlet Allocation (LDA)

Latent Dirichlet allocation (LDA) [Blei et al., 2003] is a probabilistic generative model in which term co-occurrences are used to identify the latent (i.e., hidden) topics in a corpus. Each document in a corpus is modeled as a finite mixture over a set of topics, and each topic is modeled as an infinite mixture over a set of topic probabilities. That is, a topic identified by LDA is modeled as a probability distribution indicating the likelihood of a term being assigned to the topic, and a document is modeled as a probability distribution indicating the likelihood that it expresses each topic. LDA is intractable for direct computation [Blei et al., 2003]. Gibbs sampling, a special case of Markov-chain Monte Carlo (MCMC) simulation, can be used to approximate an LDA model by directly estimating the assignment of terms to topics given the observed terms in a corpus [Griffiths and Steyvers, 2004; Heinrich, 2009]. Input parameters to the Gibbs sampler include [Heinrich, 2009]:

- K, the number of topics
- α, the Dirichlet hyperparameter for topic proportions (influences the topic distributions per document)
- β, the Dirichlet hyperparameter for topic multinomials (influences the term distributions per topic)
- σ, the number of sweeps to make over the entire corpus

There are manual and automated methods for choosing K [Griffiths and Steyvers, 2004], though the automated methods may be inadequate when applied to source code [Maskeri, Sarkar, and Heafield, 2008b]. Estimation models for the hyperparameters exist, but common heuristics are α = 50/K and β = 0.01 [Griffiths and Steyvers, 2004]. Output of the Gibbs sampler includes:

- ψ, the term-topic probability distribution
- θ, the topic-document probability distribution

The process illustrated in Figure 1.1 can be instantiated using LDA as the TR method. The document extractor can be configured to produce a corpus in which each document is a vector storing the frequency of each extracted term (i.e., a term frequency vector). The produced LDA model can be queried using similarity analysis of document parameters or predictive document likelihood [Heinrich, 2009]. For example, the similarity between a document and a query can be computed as the conditional probability of the query given the document, in which case a document is relevant to a query if it has a high probability of generating the words in the query [Hofmann, 1999].

1.2.5 Language Modeling

A language model is a statistical model of the distribution of natural language in speech and natural language documents. A statistical language model in text takes the form of a set of probabilities between terms and term phrases and their appearance in a document. Each term and term phrase is assigned a probability and this probability is used to describe the usage of natural language in the document. Statistical language modeling may be applied to TR by predicting the likelihood that a document could produce the terms written in a query [Gao, Nie, Wu, and Cao, 2004].

The probabilities are similar to weighting schemes such as *tf-idf*. Approaches such as VSM

use *tf-idf* as a heuristic for predicting the likelihood of query terms in their document. In VSM, *tf* and *idf* are not computed as part of the model itself. Instead, language modeling approaches do not use the normal approach to *tf-idf*, but incorporate this information directly into the model.

To calculate the likelihood of a document producing the terms in a query, a language model needs to be created for each document. For TR purposes, the general model used is a unigram model (i.e., the model is made up of individual tokens and not phrases). This unigram model is used to calculate the probability of a term being generated given the model [Gao et al., 2004].

$$p(Q|M_d) = \prod_{t \in Q} p(t|M_d) x \prod_{t \notin Q} 1 - p(t|M_d)$$

The first term in this calculation is the probability of generating the terms in the query. The second term is the probability of not generating additional terms. The simplest way to calculate the probability of a term being generated by a language model is to divide the raw term frequency by the total number of tokens in a document. However, there are two main problems with this simplistic approach.

The first problem is that a document is typically only a small subset of a language model. For this reason, it is not clear how well the document reflects the language model in its entirety. Most language models handle this problem by finding the mean probability of a term across all documents where that term is present. The mean probability and the computed probability for the document are then mixed. However, since it cannot be certain that each document where a term is present is part of the same language model, a risk factor is associated with the average [Gao et al., 2004]. This risk factor determines how much emphasis is given to the mean probability in the mixture. The riskier the computed mean probability is considered, the more emphasis there is

placed on the document's computed probability. The new computed probability, where $\hat{R}_{t,d}$ is the risk factor, is:

$$\hat{p}(Q|M_d) = p_{ml}(t,d)^{(1-\hat{R}_{t,d})} x p_{avg}(t)^{\hat{R}_{t,d}}$$

The second problem arises when $p(t|M_d) = 0$. This is the probability when a term does not appear in a document. In this case, using the simplest calculation for the probability of a query would result in $p(Q|M_d) = 0$. For this reason, smoothing algorithms are used to predict the probability of terms not present in a given document. An example of a simple smoothing algorithm would be to use the ratio of all occurrences of the term in the corpus to the total number of terms in the corpus [Gao et al., 2004]. For any document where the term is not present, this ratio is used instead of the 0 probability.

1.2.6 Structured Document Retrieval

The techniques thus far have all been concerned with unstructured natural language documents. However, not all documents are unstructured. For instance, a scientific article may be broken into the title, the abstract, the sections, the paragraphs, and the sentences. Words may appear in multiple locations in the document or in a single location in the document. The approach of performing TR by breaking documents into fragments based on the structure of the document and, either returning the most relevant fragments as a result of a query, or using the fragments to find the most relevant documents, is known as structured document retrieval (SDR) [Lalmas and Baeza-Yates, 2009]. The structure of a document may be either explicitly defined using a mark-up language (e.g., XML) or derived.

Language modeling can be combined with SDR [Ogilvie and Callan, 2002]. In these approaches, each location in the document is represented by its own language model or a set of

child language models from subcomponents. As an example, a scientific article is composed of a title, an abstract, and the body of the paper which is composed of multiple sections. The title, abstract, and each section in the paper are represented by a different language model. The language model for the body of the paper is interpolated from the language models of the sections, while the language model for the article is interpolated from the language models of the body, the title, and the article.

These approaches allow for both structured and unstructured queries. Unstructured queries can be issued on full documents or individual locations. In the case of full documents, the approach described in the previous section on language modeling may be used. To query individual location, the ranking for each component would need to be computed and then some filtering methodology would be required to remove either parent or child components that had lower rankings in the results. The last step is to remove duplications in the results.

For structured queries, new approaches have been defined for querying the language models. Two common types of queries include searching only within a specific location or searching for entire documents where greater weight is given to terms appearing in certain locations over others. In the case of the first type of query, the query would be issued across only those locations. In the case of the second query, the probabilities for each location may be combined (e.g., using a linear combination with the weights being scalar multiple for the probabilities of each location) into a single probability for the entire document.

1.3 Overview

The goal of this dissertation is to investigate how techniques that incorporate structural location (i.e., where in the method the term appears) affect the results of TR as it applies to feature location. The work from this dissertation studied two different techniques. The first combines

structural weighting with LDA as an ad-hoc preprocessing step. The second uses structured document retrieval (SDR) to represent different structural locations as fields and allow for queries to be performed over various combinations of contexts. This dissertation concludes by providing additional insights about the benefits and consequences of each study and when each technique is most appropriate.

1.3.1 Structural Weighting of LDA

TR techniques such as LSI and VSM have been shown to have their results improved when combined with additional structural and dynamic information [Liu, Marcus, Poshyvanyk, and Rajlich, 2007; Poshyvanyk, Gueheneuc, Marcus, Antoniol, and Rajlich, 2007; Revelle, Dit, and Poshyvanyk, 2010; Zhao, Zhang, Liu, Sun, and Yang, 2006]. As a method for incorporating this type of information into LDA, the effects of different structural weighting schemes for LDA were investigated. These weighting schemes take into account the structural location of terms present in source documents. For instance, terms may appear in parameters, method names, variables, and comments within a source document. However, each of these terms may not be uniform in importance.

One criticism of the use of advanced topic modeling approaches on source code is that terms are more sparse than in natural language documents. For instance, the most relevant topic to a method may be the one that describes the method's behavior. However, terms for that topic may be limited to the method name and parameters and it is common for the method name to be limited to the method's signature. In such a case, placing higher importance on the terms in the method name may result in higher probability of a document being associated with the correct topic(s). However, this does not necessarily indicate that other terms should be disregarded.

Emphasizing certain terms (e.g., method names) while deemphasizing others (e.g., method

calls) may lead to a better topic model. An earlier work examined this relationship between method calls and method names [Bassett and Kraft, 2013]. They found that by increasing the weights of method names and lowering the weights of method calls resulted in a higher success rate when retrieving relevant documents. In other words, by placing different weights on terms based on their structural position they observed any change in the effectiveness of LDA on the TR process.

With this in mind, the main questions answered in Chapter 4 are:

1. Does structural weighting of comments, leading terms, and local variables affect the accuracy of a LDA-based feature location technique (FLT)?
   (a) How does structural weighting of the individual stuctural locations affect the accuracy of a LDA-based FLT?
   (b) What are the top configurations for each system and across all systems?
   (c) What are the main effects and interactions between the structural locations?
2. Can a relationship between the contributions of each structural location's lexicon and their weighting factors be found?

In order to perform this study, a proper configuration was required for LDA and its training parameters. For this reason, a preliminary study is presented that was conducted with other researchers at the University of Alabama. The results of this study led to the recommendations for configuring LDA that are described in Chapter 4. This study and the results are discussed in Chapter 3.

1.3.2 Structured Source Code Retrieval

Instead of simply changing the weights of terms and using traditional queries, it is possible to make each structural location in a method document searchable. In this approach, a methodology is introduced for structured source code retrieval based on SDR. To my knowledge, the only technique that uses structured TR on software organizes terms from a source code document into one of four fields and treats each field uniformly [Saha, Lease, Khurshid, and Perry, 2013]. However, this previous study does not take into account additional structure or the possibility of using these structured documents to allow for advanced query languages for the developer.

Very little research has focused on using developer knowledge about a system to improve the results of a query. For instance, a developer may have an understanding of what terms are actually used in the system, what terms relate to method names and class names, and what terms refer to variables or fields. A developer might also have expectations of what context a term is used. Allowing a more robust query system that allows developer input can increase the likelihood of returning relevant results. Furthermore, such a query system would be complementary to existing query refinement and reformulation techniques and provide additional input to such a technique about a term's context.

The first step to studying this problem is to evaluate the effects such a querying system may have on the feature location task. To study this problem, the effects of structured queries were observed on a language modeling approach to SDR, where fields are locations within method documents. The performance of structured and unstructured TR techniques using different query types (i.e., titles, descriptions, and the combination from feature requests), corpora, combinations

of fields, and structural weighting schemes was investigated. Chapter 5 answers the following questions:

1. Does query type affect the accuracy of a structured retrieval-based feature location technique (FLT)?
2. Does changing the combination of included fields affect the accuracy of a structured retrieval-based FLT?
3. Does structural weighting affect the accuracy of a structured retrieval-based FLT?
4. How does the best configuration of structural field combination and weighting affect the accuracy of a structured retrieval-based FLT?

Because this technique allows the developer to change structural weightings and combinations between queries, multiple scenarios are discussed in Chapter 5.

1.3.3 Comparing Structured Retrieval with Structural Weighting of LDA

After performing these studies, experience was gained about the benefits and consequences of using each approach. Chapter 6 seeks to answer the following two questions based on the insights gained throughout this research:

1. What are the benefits and consequences of using structural weighting in LDA?
2. What are the benefits and consequences of using structured retrieval for feature location?

1.4 Organization

The remainder of this dissertation is organized as follows. Chapter 2 discusses related work in the area of feature location. Chapter 3 presents a preliminary study on configuring LDA

for performning feature location tasks. Chapter 4 introduces structural weighting of LDA and how weighting impacts the results. Furthermore, this chapter will present an approach to learning near optimal weighting configurations for systems where an optimal configuration is unknown. Chapter 5 will discuss structured source code retrieval, the design of a structured retrieval system, and the use of content and structure (CAS) queries that may be issued by a developer or recommended by a system. Chapter 6 will provide insights gained from running these studies about when to best use each technique, and discuss areas of future work in both techniques. Chapter 7 will conclude the dissertation.

Chapter 2

RELATED WORK

The research presented in this dissertation covers the use of TR and topic modeling in program comprehension and software maintenance tasks. Previous research has focused on improving TR models in two main ways: by combining the TR model with information gained from the structure of the source code and from running the system and improving the preprocessing stages of the TR process. My research is focused on ways that structural information can be used for improving TR. While additional dynamic information has been shown to improve the results of TR, it requires a running version of the system as well as a set of test cases that effectively cover the maintenance task; either of these may not always be possible, or may be difficult to obtain. Furthermore, my research focuses on the configuration of a TR model and the preprocessing steps. In this chapter, related work is presented in this area.

2.1 Text Retrieval Based Feature Location

One of the earliest works in the use of TR for software maintenance is by Marcus et al. [Marcus et al., 2004], which introduced the use of LSI [Deerwester et al., 1990] for the purposes of concept location. Concept location involves finding all relevant methods to a specific concept throughout the entirety of the source code. They identified relevant methods by computing the similarity measure between the concept and the source code elements. They then used a minimum similarity value as a cutoff for the relevant methods. To evaluate the effectiveness of their approach, they took their relevant set and compared it to a gold set of known relevant source code elements

and computed recall (i.e., the percentage of relevant documents returned) and precision (i.e., the percentage of returned documents that are relevant). The approach ran on NCSA Mosaic and compared to *grep*. LSI was found to be almost as easy to use as *grep* and to give better results than the standard keyword search. This paper showed how TR performs better than keyword searches in software engineering tasks and was one of the first to use TR for a software maintenance task. However, the methodology used only limited parsing techniques and presented only a small case study.

Similar to topic modeling is Formal Concept Analysis [Wille, 2005], which was utilized by Poshyvanyk and Marcus [Poshyvanyk and Marcus, 2007] by combining it with LSI for feature location. The main idea behind FCA is to identify a *concept hierarchy* from a collection of objects and their properties. In FCA, objects are grouped together into concepts when they share values for a set of properties. A sub-concept is a subset of the concept located in the hierarchy above it. Each concept and sub-concept contains two parts, the *extension* and the *intension*. The extension of the concept is the set of objects for the concept while the intension is the complete set of attributes for the concept. The set of all concepts obtained from the FCA comprises a *concept lattice*. In the approach, LSI provides the objects as a collection of documents and the weights of the terms as the attributes. Relevant source elements are those elements that appear within the same concept as the query. The limitation with this approach is that FCA is performed after a query and a ranked list is obtained.

The first use of LDA for the purposes of feature location was performed by Lukins et al. [Lukins et al., 2008, 2010]. In their study, the use of LDA [Blei et al., 2003] was presented as an alternative to LSI and probabilistic LSI for bug localization, i.e., feature location where the

feature is a bug. A possible topic in a software system could contain the terms "buffer," "read," and "scan," where the topic would be a "input."

Lukins et al. studied the effects of LDA over five different case studies on Mozilla, Rhino, and Eclipse. For each study, the rank of the first relevant method or class was used as the evaluation criteria. The purpose of the study was to support the use of LDA for feature location by demonstrating the technique's accuracy, scalability, and sensitivity to source code stability. The goal of case study 1 was to compare LDA TR to LSI TR as standalone techniques on five bugs in Mozilla and three bugs in Eclipse. The results of the first case study showed that LDA performed as good or better than LSI. The second case study examined the accuracy of LDA on all bugs in a given software system. The results from LDA on 35 bugs from Rhino were analyzed. Stability of source code was not shown to affect the accuracy of the technique. Of these results, 77% returned the first relevant method in the top 10 results, and 63% returned in the top five results. At the class level granularity, 54% of the bugs returned the first relevant class as the top results. Case studies 3-5 analyzed 106 bugs over 12 versions of Rhino and 216 bugs in 13 versions of Eclipse. The third case study focused on the accuracy of LDA as a software system scales. It involved the largest analysis discussed in the review. This case study showed that the accuracy of LDA scales well with the software system. The last two remaining case studies focused on the accuracy of LDA compared to the size of the software system and stability, with no significant relationships found. This study introduced LDA as a possible method for TR-based feature location, but it did not take into account any structural information when building the topics nor did it incorporate any additional sources of information.

This section outlines some of the early research into TR-based feature location. The techniques presented in this section use TR as the standalone method for performing feature location

and introduces the basic methodology for performing TR on software. Of particular interest to my research is the study presented by Lukins et al.

2.2 Combining Additional Information with Text Retrieval

Cleary and Exton [Cleary and Exton, 2007; Cleary, Exton, Buckley, and English, 2009] presented an approach aimed at improving the TR process by incorporating additional non-source code software artifacts into the TR model. Their approach used ideas of language modeling and query expansion and was based on the realization that software engineers communicate and record concerns in non-source code artifacts such as bug reports, emails, and external documentation. They created a type of semantic space model for deriving term co-occurrence relationships from a corpus and then find measures of information flow to identify a set of terms that are potentially related to the terms in a query. The use of information flows is based off a process from Song and Bruza [Song and Bruza, 2001]. They use these terms to expand and enhance a query. They compare their approach, which has been referred to as query expansion KL-divergence (QEKLD) [Cleary and Exton, 2007] or cognitive assignment [Cleary et al., 2009], against a classical language modeling approach [Zhai and Lafferty, 2004], a dependency language modeling based approach [Gao et al., 2004], and LSI. They extended the cognitive assignment Eclipse plug-in [Cleary and Exton, 2006] to perform feature location on an open source system called CHIVE. QEKLD was seen to perform equal to or better than the other approaches. This study showed the importance of relevant terms in a query and how proper queries can improve the TR process for feature location. However, the limitation of this approach is that it requires the developers to make available to the technique additional natural language documents outside of the source code.

The first study to combine TR with static analysis for the purposes of feature location was introduced by Zhao et al. [Zhao, Zhang, Liu, Sun, and Yang, 2004; Zhao et al., 2006], which

presented a static, non-interactive approach to feature location (SNIAFL) with the goal of delivering a fully automated non-dynamic feature location technique (FLT). The approach combines TR with Branch Reserving Call Graphs (BRCG), which are call graphs that have the added benefit of providing branch information to the user.

SNIAFL involves four steps. First, the initial set of connections between all features and the source code's functions are acquired. Second, a threshold is determined by finding the greatest difference in similarities between adjacently ranked functions in the initial ranked list of functions. Anything above this threshold is included as one of the feature's initial specific functions. After this step, a BRCG is traversed. Any branch that is found to be irrelevant to the initial set of specific functions is pruned. The next step involves traversing the BRCG again for each feature, when a relevant function is encountered, the system checks if the feature is relevant to any other feature; if not, the relevant function is marked with a 1, otherwise a 0. After this, a pseudo execution trace may be returned to the user for each feature.

An experimental study was performed on an open source software system named DC. Of the 49 functional requirements mentioned in the specifications documentation for the system, the authors chose to use the 21 primitive functions for their study. Each of these 21 primitive functions implemented a single functionality for the system. For each function, SNIAFL was applied to obtain the relevant functions and three groups of data were obtained: the initial and final specific functions of each feature, the relevant functions acquired by the BRCG from the initial specific functions, and the pseudo execution traces constructed by the BRCG. SNIAFL was evaluated on three aspects: the relevant functions, the execution traces, and the specific functions. Precision and recall were used to compare SNIAFL, VSM, and a dynamic approach in obtaining relevant functions. SNIAFL resulted in the highest recall with 99.57% and a precision of 90.97%. When

compared to the execution traces obtained from a purely dynamic approach, SNIAFL was found to be less than effective. On the last criteria, the specific functions, SNIAFL obtained an 85.71% correct ratio of the specific functions, while the correct ratio for the dynamic approach was 95.24% given insufficient test cases and 100% given well-designed test cases. Overall, this study found that SNIAFL worked better than either TR or static analysis alone for obtaining relevant methods and was an acceptable alternative to dynamic analysis when searching for functions specific to only one feature. This study highlights the importance of structural information in TR. However, the results of the case study are limited as the systems used in the study are small (i.e., less than 100 functions).

Hill et al. [Hill, Pollock, and Vijay-Shanker, 2007] presented Dora, which combines TR with structural representations of code similar to Zhao et al. Users formulate queries related to software maintenance tasks and input a seed set of methods. Dora computes the relevance of the query to the methods in the seed set by combining *tf-idf* scores and method features, such as the locations of relevant terms in the method, in a linear regression model. Dora follows call graph edges from the seed set of methods and finds additional relevant methods for the query. Dora then outputs the relevant "neighborhood" (i.e., methods related to the user's task) to the user. Dora was compared to a structural-topology approach called Suade and to two additional lexical and structural techniques, boolean-AND and boolean-OR. Dora was found to be the most successful. Like Zhao, this research shows how structural information can be used to find relevant source code elements. Unlike Zhao however, this research requires an initial set of seed methods that may not be available. Of particular interest to my research is the use of method features. In their approach, they place high emphasis on the appearance of terms in the method name and the number of statements that contain a query term. Unlike my research, their approach does not pay attention

to the importance of other locations and makes use of the simplistic *tf-idf* model. Furthermore, their approach does not take into account the differences between software systems.

Shao and Smith [Shao and Smith, 2009] used an affine transformation to combine LSI with static dependency information extracted from call graphs. Their approach first performs a TR of the source code using LSI and then defines a minimum similarity score threshold on the returned ranked list. Any methods with similarity scores greater than or equal to the threshold are added to a new list and a call graph is created for each method. A hash function is used to create a new list for each method that contains the list of all methods that are one-edge away, that is a list of methods that the method calls directly. A method's similarity score from LSI and the density of the method in the call graphs of all methods (i.e., a measure of how often the method is called by other methods within the threshold) is then combined by the transformation to give the method a new score. After such a score has been obtained for each method in the threshold, a new ranked list is created and reported to the user. A case study compared this approach to LSI on an open source software system, iVistaDesktop, and the approach was found to show some improvement over LSI alone. This paper presented a new way of combining structural information with TR. There are two key limitations to this study, however. First, the study carried out was only on one system. Second, the call graph was only evaluated one-edge away and no empirical study was performed for more than a single edge.

Scanniello and Marcus [Scanniello and Marcus, 2011] used clustering based off of structural dependencies and lexical similarities between methods to improve results over TR alone. In their approach, VSM and cosine similarity are used to index the methods of a software system and then compare them to neighboring methods in a dependency graph. Using these similarities, a new weighted graph is created from the original dependency graph and is used as input to a BorderFlow

algorithm that clusters related methods based on flow. Document ranking is performed on both the methods and the clusters, with the developers being presented with a ranked list of clusters. The ranking of methods is maintained within each of their corresponding clusters. The effectiveness of this technique over VSM is evaluated on four open source systems: ATunes, Art of Illusion, Eclipse, and two versions of jEdit. Effectiveness of the approach was considered by the first relevant method and the use of a Mann-Whitney test between the clustering approach and VSM alone. The authors found that clustering significantly increases the effectiveness of TR over VSM alone. The authors demonstrated how lexical information can be combined with structural dependencies to create logical groupings amongst the elements of a software system. However, in this approach, the relevant cluster must first be found before the relevant method.

Although focused on dynamic analysis, the work by Revelle et al. [Revelle et al., 2010] is related to my research and presents a novel idea to feature location based on data fusion and web mining. Data fusion is the process of combining multiple sources of information in such ways that the combination of the sources produces better results than the sources used individually. The types of data fusion applied in this study include combining LSI, execution traces, and web mining. Web mining is a technique that has been used to analyze the World Wide Web (WWW). Two algorithms are presented, these include Hyperlinked-Induced Topic Search (HITS) [Kleinberg, 1999] and PageRank [Brin and Page, 1998]. HITS identifies hubs and authorities based on whether an element contains links to multiple relevant pages or are pointed to by many hubs. PageRank attempts to score elements based on their relative importance with other elements. In order to compute both of these scores, execution traces are used to build graphs that include methods and calls between the methods.

Experiments were ran on various combinations of LSI, execution traces, and the two web

mining techniques. A study of these data fusions was performed on 45 features of Eclipse and 241 features of Rhino, and were compared against each other, LSI, and other techniques. Web mining as a standalone approach produced results that were comparable to LSI. Of the standalone web mining techniques, HITS was shown to be more effective than PageRank and ranking by authorities performed better than ranking by hubs. When used as a means to filter data for the other techniques, web mining effectively increased all techniques in Eclipse and most of the approaches in Rhino.

Sisman and Kak [Sisman and Kak, 2012] used version histories of a software project to estimate a prior probability distribution for defect proneness of files in a given version of the software project. They computed two different priors: a modification history prior and a defect history prior. The modification history prior is based on the frequency of variation of a file whereas the defect history prior is found by analyzing the version history and change requests for what the authors refer to as bug fixing change sets. With these two priors, the authors apply a temporal decay so that there is more emphasis on files that have been changed recently over files that were changed in the distant past. These priors were used in a TR framework that utilized language modeling and divergence from randomness, two techniques that have been shown to do well with probability priors in the past. The authors conducted a case study on AspectJ [Kiczales, Hilsdale, Hugunin, Kersten, Palm, and Griswold, 2001], an open source language extension to Java. To evaluate their models, the authors calculated the mean average precision (MAP). Their approach showed improvement over existing models including those that used *tf-idf*. This research shows how information in the history of the software system can be used to improve the results of automated software maintenance. More work would need to be performed to determine the best ways for computing the priors and the decay.

Given the complexities of incorporating dynamic analysis and software repository mining in software search, Moreno et al. [Moreno, Treadway, Marcus, and Shen, 2014] implemented a technique which they referred to as Lobster (LOcating Bugs using Stack Traces and tExt Retrieval). Their technique combined the results of a VSM-based TR technique with stack traces extracted from bug reports. Their technique incorporates two key components. The first is a textual similarity measure that is obtained from using any TR model and comparing the model to the query. In the case of their study, the authors chose to use Lucene [Gospodnetic and Hatcher, 2005] which combines VSM with a Boolean model. The second component is obtained by using the code elements given in the stack trace of the bug report. They defined the similarity between a stack trace and a method in the software system as the shortest path distance of any code element in a stack trace to the target code element in the software system through the program dependency graph. These two measures are combined into a single measure, labeled the total similarity, as a linear combination of the two similarity scores. The authors evaluated their study using 17 versions of 14 open source Java systems with varying size and domain. The results of this empirical evaluation showed that incorporating information from stack traces improved the baseline TR technique. The importance of this technique is in the reduced overhead from computing the stack trace similarities versus more intense techniques such as dynamic analysis and software repository mining.

The key importance of this section was to present how TR produces better results when combined with other sources of information. Such information can come from other software repositories, natural language artifacts, dynamic execution traces, or can even be embedded in the source code. My research takes advantage of the last source (i.e., the structure of the source code).

2.3 Configuration and Corpus Creation

The effects of the stemming preprocessing step (i.e., reducing inflected terms to their base) were studied by Hill et al. [Hill, Rao, and Kak, 2012]. The purpose of their study was to determine the comparative effectiveness of different stemmers in the domain of software. The authors performed two studies on software systems in Java analyzing the difference in stemmers on concept location and bug localization. Four traditional TR stemmers were compared: Porter, Snowball, KStem, and Paice. These stemmers were selected based on whether they were heavy or light, morphological or algorithmic, or specialized for the domain of software. In each study, the authors calculated the Mean Average Precision. For the bug localization study, the authors used the iBUGS dataset [Dallmeier and Zimmermann, 2007] for AspectJ. This dataset contained 291 bugs that formed the basis for their experiment. In addition, the authors tested the effectiveness of stemmers on three different types of queries: short descriptions that included titles from a bug description, medium descriptions that included the first comment line in the bug description, and long descriptions that included the entire bug description. The purpose of the different queries was to see if the effectiveness of stemmers was different based on the type of query and to identify those queries for which one stemmer performs the best.

The results of their study showed that in the case of long descriptions, the choice of stemmer had relatively little effect on the task. However, in the case of the short queries, a high variability was observed. They found that the effectiveness of stemmers was based on query type and the specific software engineering task. For instance, while Paice was found to have high performance in concept location, it showed poor performance for bug localization. This study shows the impor-

tance of preprocessing and the effects that variations of terms may have on the text retrieval model. However, this study is limited in that it only focuses on AspectJ.

Similar to the approach of structural weighting discussed in my research, Alhindawi et al. [Alhindawi, Dragan, Collard, and Maletic, 2013] proposed a method for improving the results of TR through method stereotypes, which are terms that describe the abstract role of a method (e.g., get, set, and predicate). The authors presented a tool called StereoCode which automatically identifies a method's stereotype by analyzing static and structural information and then adds a comment before the method defining the method's stereotype. They then carried out TR as described in Chapter 1.

Using LSI as the TR method, the authors conducted a study using two C++ open-source software systems, HippoDraw [Kunz, 2001] and Qt [Manly and Olson, 1999]. The authors performed feature location using LSI with and without the added stereotype information. The authors evaluated their approach using a number of measures, including recall, precision, the rank of the first relevant method, the rank of the last relevant method, and the ranks of all methods. The study demonstrated that the added stereotype information improved the query results for the feature location process with improvements in each of the measures. While this study shows how structural information improves TR by adding the terms directly to the document in the corpus, it only incorporates method stereotypes.

Another weighting scheme was presented by Zamania et al. [Zamania, Lee, Shokripoura, and Anvikb, 2014]. In their study, they focused on improving algebraic models to TR by using only nouns and weighting terms based on when they were used in the repository. The focus of their study looked at improving VSM which has been shown to perform poorly compared to query-likelihood

models and LDA [Binkley, Lawrie, Uehlingera, and Heinzb, 2015]. In addition, their results were focused on the file level granularity as opposed to the method level that is the focus of this work.

Another study that focused on enhancing the corpus was carried out by Saha et al. [Saha et al., 2013]. The authors introduced BLUiR (Bug Localization Using information Retrieval), an automatic bug localization tool that leverages structural information in code to improve the TR process. BLUiR was built on Indri [Strohman, Metzler, Turtle, and Croft, 2005], an open-source information retrieval toolkit. BLUiR takes the source code files of the software system as input and builds the abstract syntax tree (AST) for each source code file using the Eclipse Java Development Tools. Information for each source file is stored as a structured XML document that is sent to Indri for preprocessing. The authors distinguish between two query types (*summary* and *description*) and within the documents four different fields (class, method, variable, comments). They then perform separate searches for each of the eight different combinations and sum the results for a document across all eight searches.

The authors evaluated BLUiR on four open source projects (Eclipse [desRivieres and Wiegand, 2004], AspectJ, SWT [Northover and Wilson, 2004], and ZXing [Scheuermann, Werner, Kessel, Linnhoff-Popien, and Verclas, 2012]) with approximately 3,400 bugs. The authors compared their tool to a previous tool (BugLocator [Zhou, Zhang, and Lo, 2012]) and evaluated each tool on the top n results, mean reciprocal rank, and mean average precision. The results of the study showed that BLUiR outperformed BugLocator. The authors found that including structural information enables more accurate results for bug localization. This study is similar to my structured retrieval approach; however, the authors treat all document fields in the XML document equally and only compute a simple sum between the rankings. The work does not consider the

possibility of different queries for each feature, nor does it take into account differing structures, and weightings.

Panichella et al. [Panichella, Dit, Oliveto, Di Penta, Poshynanyk, and De Lucia, 2013] investigated the configuration of LDA for software engineering tasks. Their approach, which they termed LDA-GA for LDA genetic algorithms, uses genetic algorithms to find the best configuration for LDA in three different software engineering tasks: traceability link recovery, feature location, and software artifact labeling. Their approach is based on an assumption that the quality of results from LDA in a software engineering task is dependent on the quality of clusters in the LDA model. A cluster is a grouping of objects where the objects inside the cluster are closer to each other than they are to objects outside of the cluster. Using LDA, documents should be closer to other documents within their dominant topic than to those outside of it. To rate the quality of a cluster, the authors used two criteria to calculate a coefficient: the closeness of documents inside of a cluster and the distance of separation between clusters. Using this basic assumption, the authors used genetic algorithms to search for the optimal solution or the highest coefficient. The authors carried out a case study using six open source software systems and the three software engineering tasks. They found that their approach significantly improved the results of LDA in these tasks. While this study finds the optimal configuration for LDA, the model must be created multiple times until an optimal solution is reached. The time required to do so may not be acceptable in an actual development environment. My work applies a modified version of their approach to finding the optimal structural weighting in LDA over time.

This section showed the importance of the corpus and proper configuration in the TR process. Studies similar to the structural weighting scheme were discussed and the results of these

studies showed how structural information in the corpus can improve the results of a TR technique.

This chapter presented previous work in the area of TR-based feature location. The following chapters discuss new approaches to this area that expand upon the uses of structural location to TR. The next chapter discusses preliminary work on configuring LDA. This study will lead directly into Chapter 4.

Chapter 3

PRELIMINARY STUDY ON CONFIGURING LDA

Before considering the effects of structural weighting in LDA, this chapter discusses how to properly configure LDA for feature location. As a member of a research team at the University of Alabama, we investigated the affect different configurations can have on the LDA-based FLT.

TR techniques are highly configurable. For example, when using LSI [Deerwester et al., 1990] we must select k, the number of (reduced) dimensions, or when using LDA [Blei et al., 2003] we must select α, β, and K, the two smoothing hyperparameters and the number of topics, respectively.

Which text to extract from the source code is another important configuration decision. In particular, the text extractor has seven possible configurations: (1) identifiers only, (2) comments only, (3) literals only, (4) identifiers and comments, (5) identifiers and literals, (6) comments and literals, and (7) identifiers, comments, and literals. Before we can index the source code, we must choose one of these configurations.

Unfortunately, few studies of TR-based FLTs directly address the decisions that a practitioner or researcher must make when configuring the FLT. The feature location literature contains no empirical evidence that supports the selection of one configuration over another.

We conducted a case study in which we consider the configuration of an LDA-based FLT using 618 features in six open source Java systems. Specifically, we consider five configuration parameters, the first of which is the query. The second configuration parameter that we studied

is the extracted text, and we are aware of no study that considers this parameter. The remaining configuration parameters are the number of topics (K) and the two smoothing hyperparameters (α and β).

3.1 Case Study

This section describes the design of a case study that measured the effects of different configurations on the accuracy of an LDA-based FLT.

3.1.1 Definition and Context

Our primary *goals* are to understand whether five factors interact and to what extent the five factors affect the performance of an LDA-based FLT. The five factors of interest are the query (*Query*), the source code text to extract (*Text*), and the three LDA parameter values (K, α, β). The *quality focus* of the study is on establishing the importance of proper configuration to attain optimal performance from an LDA-based FLT and on informing the configurations of five parameters. The *perspective* of the study is of a software developer performing a change task on a software system and using an LDA-based FLT to identify a starting point (i.e., a method) from which to begin the change. The *context* of the study spans 618 features from six open source Java systems (ArgoUML[1], JabRef[2], jEdit[3], muCommander[4], Mylyn[5], Rhino[6]).

3.1.2 Overview

We consider six factors in the case study, which has five parts. The design of Part 1 is listed in Table 3.1, the designs of Parts 2-4 are listed in Table 3.2, and the design of Part 5 is listed in Table 3.3. The first factor is *Query*, which is a categorical variable. The next factor is *Text*, which

[1] http://argouml.tigris.org
[2] http://jabref.sourceforge.net
[3] http://www.jedit.org
[4] http://www.mucommander.com
[5] http://www.eclipse.org/mylyn/
[6] https://developer.mozilla.org/en-US/docs/Mozilla/Projects/Rhino

is a categorical variable that represents the text extractor configuration. The third factor is K, which is a ratio variable that represents the number of topics. α and β are the fourth and fifth factors, representing the smoothing hyperparameters, and are ratio variables. *System* is the sixth factor and is a categorical variable. We describe its categories in the next section.

Part 1 of the case study focuses on the interactions among five factors (*Query*, *Text*, K, α, β). We use a full factorial design to permit the detection of interaction effects using factorial ANOVA. To ensure the feasibility of this part of the case study, we limit each of the five factors to three possible values (for a total of 243 distinct configurations).

Table 3.1: Case study design: Part 1.

Factor	Values
Query	*Title*, *Description*, *Combined*
Text	*I*, *CL*, *ICL*
K	100, 200, 500
α	0.5, 1.0, 50.0
β	0.01, 0.1, 0.5

We focus on a different factor (or pair of factors) in each of Parts 2-4, controlling *System* throughout. Part 2 focuses on *Query* and controls the other factors. *Query* has three categories (*Title*, *Description*, *Combined*). Part 3 focuses on *Text* and K. *Text* has seven categories (*I*, *C*, *L*, *IC*, *IL*, *CL*, *ICL*), and K has four categories per subject system. Because the value of *Text* may affect the size of the corpus (i.e., the numbers of documents and terms), and because K should be proportional to the size of the corpus, we vary these factors together. Part 4 of the case study focuses on α and β. We vary these factors together, assigning six values to each factor (0.01, 0.1, 0.25, 0.5, 0.75, 1).

Part 5 of the case study applies the lessons learned in Parts 1-4 and compares our pre-

Table 3.2: Case study design: Parts 2-4.

Part	Factor(s) of Interest	Controlled Factors
2	*Query*	*Text* (*ICL*), K (100 or 200), α ($50/K$), β (0.01)
3	*Text*, K	*Query* (*Combined*), α ($50/K$), β (0.01)
4	α, β	*Query* (*Combined*), *Text* (*ICL*), K (100 or 200)

dicted best configurations for four systems (ArgoUML, jEdit, JabRef, and Rhino) to two generic configurations informed by heuristics from the literature.

Table 3.3: Case study design: Part 5.

Configuration	*Query*, *Text*, K, α, β
$Predicted_{ArgoUML}$	*Combined*, *ICL*, 500, 1.0, 0.1
$Predicted_{jEdit}$	*Combined*, *ICL*, 400, 1.0, 0.25
$Predicted_{JabRef}$	*Combined*, *ICL*, 400, 1.0, 0.25
$Predicted_{Rhino}$	*Combined*, *ICL*, 300, 1.0, 0.5
$Heuristic_1$	*Combined*, *ICL*, 200, 0.25, 0.01
$Heuristic_2$	*Combined*, *ICL*, 200, 0.25, 0.1

3.1.3 Subject software systems

We chose the six subjects of our study — ArgoUML, JabRef, jEdit, muCommander, Mylyn, and Rhino. Table 3.4 lists four size metrics for each of the six subject systems: source lines of code (SLOC), comment lines of code (CLOC), Java file count, and method count. The table also lists the number of features that we study for each system. Further, the application domains of the systems are as follows. ArgoUML is a UML modeling tool, and JabRef is a bibliography reference manager. jEdit is a programmer's text editor, and muCommander is a cross-platform file manager. Mylyn is an Eclipse plug-in that provides a task-focused interface for ALM, and Rhino is a JavaScript engine that provides a compiler, an interpreter, and a debugger.

Table 3.4: Subject software systems.

System	Version	SLOC	CLOC	Files	Methods	Features
ArgoUML	0.22	117,649	104,037	1,407	11,348	91
JabRef	2.6b	74,350	25,927	579	5,323	38
jEdit	4.3	98,460	42,589	483	6,550	149
muCommander	0.8.5	76,649	68,367	1,069	8,811	90
Mylyn	1.0.1	99,310	23,503	936	9,067	93
Rhino	1.6R5	45,225	15,451	129	2,565	157
Total		511,643	279,874	4,603	43,665	618

3.1.4 Benchmarks

We studied features that correspond to issues reported via Bugzilla or an equivalent issue tracking system. Most of the issue reports are requests to change an unwanted functionality (i.e., to remove a faulty feature), though some are requests to add a new functionality (i.e., to add a new feature). Two approaches are used to recover the set of methods modified to fix each bug or to add each functionality. The patches submitted to Bugzilla are used to recover the set of methods modified to address each issue, or the diffs stored in Subversion are used to recover the set of methods modified.

We studied 618 features total. Specifically, we considered the following numbers of features for each system: 91 for ArgoUML, 38 for JabRef, 149 for jEdit, 90 for muCommander, 93 for Mylyn, and 157 for Rhino. The selected features represent a subset of the available features for each benchmark. We exclude any available feature for which the bug report's title or description is empty after the application of our preprocessing steps (i.e., splitting, normalizing, filtering, and stemming).

Due to the large number of features that we consider, and to eliminate potential bias, we

automatically formulate three queries for each feature. Specifically, we use as the query the bug report's title, its description, or its title and description combined [Dit, Revelle, Gethers, and Poshyvanyk, 2012]. This query formulation process is conservative, in that it does not rely on developer experience, and unbiased, in that it does not allow us to influence the results.

3.1.5 Effectiveness measure

Though modifying or removing a functionality requires that the developer identify all entities to be changed, the goal of automatic feature location is to identify a single method from which the developer can begin the change [Lukins et al., 2010; Poshyvanyk et al., 2007]. Other methods associated with the feature can then be identified using impact analysis.

Because static FLTs rank all methods in a system, recall and precision are not useful accuracy measures in this context. In particular, in the context of static FLTs, recall is always 1.0, and precision is always $1/n$ (where n is the number of methods). Thus, the rank of the first relevant method is used instead [Lukins et al., 2008, 2010; Poshyvanyk et al., 2007; Revelle et al., 2010]. This measure, which Poshyvanyk et al. [2007] term the "effectiveness measure" for feature location, indicates the number of entities that the developer must examine (if following the ranking) before reaching a method that actually belongs to the feature. That is, the measure quantifies the number of false positives that a developer must examine. Thus, in this study we adopt the effectiveness measure as our accuracy measure.

3.1.6 Setting

We implemented our document extractor in Python v2.6 using ANTLR v3 [Parr and Quong, 1995] and an open source Java 1.5 grammar[7]. We extract documents at the method level of granularity using a term count weighting scheme and consider every method to be distinct. That is, if

[7] `http://antlr.org/grammar/1152141644268/Java.g`

method bar is nested within method foo[8], each method is considered separately, and the text for method bar is not considered to be part of the text for method foo. We associate any comment that is contained in a method with that method. We associate any block comment (or series of line comments) that precedes a method with that method.

`java.lang` class names are filtered before splitting tokens. Tokens are split based on camel case, underscores, and non-letters. After splitting tokens, the original token is retained. We normalize to lower case before filtering English stop words, Java keywords, and terms shorter than three characters. A Porter stemmer[9] is applied to retained terms.

For the second and fourth parts of the first case study, we set $K = 100$ for Rhino and $K = 200$ for the other five systems. Other researchers have set $K = 100$ [Lukins et al., 2008, 2010] for Rhino, and we adopt the value because our task and setting are similar to those of the former studies. We heuristically set $K = 200$ for the other five systems based on their sizes. As our study is concerned primarily with relative performance, not absolute performance, it is not critical that the optimal K is chosen for each system.

R lda v1.2.3 is used to compute and query LDA models. Because R lda implements a CGS algorithm, σ must be set, the number of sweeps to make over the entire corpus. We set $\sigma = 500$, which provides a balance between execution efficiency and model convergence. Our classifier is conditional probability (P).

3.1.7 Hypotheses

This subsection describes the hypotheses for each part of the case study.

[8] In Java, a method may contain a class (which may contain a method).

[9] `http://tartarus.org/~martin/PorterStemmer/python.txt`

3.1.7.1 Hypotheses for Part 1

For the five factors (*Query*, *Text*, K, α, β), all two-way interactions, all three-way interactions, all four-way interactions, and the five-way interaction are tested for statistical significance.

For the 10 two-way interactions, each null hypothesis is of the form:

$H_0 : \mu + \nu$ There is **no interaction** between factors μ and ν.

Further, each alternative hypothesis is of the form:

$H_A : \mu * \nu$ There is **interaction** between factors μ and ν.

For example:

$H_0 : Query + Text$ There is no interaction between factors *Query* and *Text*.

The remaining 9 null hypotheses are analogous. We tested these hypotheses using the effectiveness measure.

If a null hypothesis can be rejected with high confidence ($\alpha = 0.05$), we accept an alternative hypothesis stating that the two factors interact. For example, the alternative hypothesis corresponding to the example null hypothesis is:

$H_A : Query * Text$ There is interaction between factors *Query* and *Text*.

The remaining 9 alternative hypotheses are analogous.

The null and alternative hypotheses for the 10 three-way interactions, the 5 four-way interactions, and the five-way interaction are formulated similarly.

3.1.7.2 Hypotheses for Parts 2-5

This subsection compares a large number of configurations, and no presupposition is made about the direction of the difference between any two configurations. Thus, all of our hypotheses are two-sided. In particular, each null hypothesis is of the form:

$H_0 : \mu = \nu$ Configuration μ **does not significantly affect** the accuracy of the LDA based FLT compared to configuration ν.

Further, each alternative hypothesis is of the form:

$H_A : \mu \neq \nu$ Configuration μ **does significantly affect** the accuracy of the LDA based FLT compared to configuration ν.

The second part of the case study focuses on *Query* and control of the other factors. *Query* is a categorical variable with three categories (*Title*, *Description*, *Combined*), so three null hypotheses were formed to test whether LDA-based feature location produces different results when using different queries. In particular, for each null hypothesis, μ and ν are distinct query types in $\{Title, Description, Combined\}$. For example:

$H_0 : Title = Description$ *Title* does not significantly affect the accuracy of the LDA based FLT compared to *Description*.

The remaining two null hypotheses are analogous. We tested these hypotheses using the effectiveness measure.

If a null hypothesis can be rejected with high confidence ($\alpha = 0.05$), we accept a two-sided alternative hypothesis stating that a query type has an effect on the ranking of the first relevant method compared to another query type. For example, the alternative hypothesis corresponding to the example null hypothesis is:

$H_A : Title \neq Description$ $Title$ does significantly affect the accuracy of the LDA based FLT compared to $Description$.

The remaining two alternative hypotheses are analogous.

The third part of the case study we focuses on *Text* and K and control of the other factors. *Text* is a categorical variable with seven categories (*I*, *C*, *L*, *IC*, *IL*, *CL*, *ICL*), and K is a ratio variable with four values, so 378 null hypotheses were formed to test whether LDA-based feature location produces different results when using different text and different numbers of topics. In particular, for each null hypothesis, μ and ν are distinct pairs in $\{I, C, L, IC, IL, CL, ICL\} \times \{75, 100, 150, 200\}$. For example:

$H_0 : (I, 75) = (C, 100)$ $(I, 75)$ does not significantly affect the accuracy of the LDA-based FLT compared to $(C, 100)$.

The remaining 377 null hypotheses are analogous. We tested these hypotheses using the effectiveness measure.

If a null hypothesis can be rejected with high confidence ($\alpha = 0.05$), we accept a two-sided alternative hypothesis stating that the text/topics pair has an effect on the ranking of the first relevant method compared to another text/topics pair. For example, the alternative hypothesis corresponding to the example null hypothesis is:

$H_A : (I, 75) \neq (C, 100)$ $(I, 75)$ does significantly affect the accuracy of the LDA-based FLT compared to $(C, 100)$.

The remaining 377 alternative hypotheses are analogous.

The fourth part of the case study focuses on α and β and control of the other factors. The hyperparameters α and β are ratio variables with six values each (0.01, 0.1, 0.25, 0.5, 0.75, 1), so 630 null hypotheses were formed to test whether LDA-based feature location produces different

results when using different values of α and β. In particular, for each null hypothesis, μ and ν are distinct pairs in $\{0.01, 0.1, 0.25, 0.5, 0.75, 1\} \times \{0.01, 0.1, 0.25, 0.5, 0.75, 1\}$. For example:

$H_0 : (0.5, 0.01) = (0.25, 0.1)$ $(0.5, 0.01)$ does not significantly affect the accuracy of the LDA-based FLT compared to $(0.25, 0.1)$.

The remaining 629 null hypotheses are analogous. We tested these hypotheses using the effectiveness measure.

If a null hypothesis can be rejected with high confidence ($\alpha = 0.05$), we accept a two-sided alternative hypothesis stating that the α/β pair has an effect on the ranking of the first relevant method compared to another α/β pair. For example, the alternative hypothesis corresponding to the example null hypothesis is:

$H_A : (0.5, 0.01) \neq (0.25, 0.1)$ $(0.5, 0.01)$ does significantly affect the accuracy of the LDA-based FLT compared to $(0.25, 0.1)$.

The remaining 629 alternative hypotheses are analogous.

The fifth part of the case study compares our predicted best configurations for four systems (ArgoUML, jEdit, JabRef, and Rhino) to two generic configurations informed by heuristics from the literature. In particular, for each null hypothesis, μ and ν are distinct pairs in $\{\text{Predicted}_{ArgoUML}, \text{Predicted}_{jEdit}, \text{Predicted}_{JabRef}, \text{Predicted}_{Rhino}\} \times \{\text{Heuristic}_1, \text{Heuristic}_2\}$. For example:

$H_0 : Predicted_{ArgoUML} = Heuristic_1$ $Predicted_{ArgoUML}$ does not significantly affect the accuracy of the LDA-based FLT compared to $Heuristic_1$.

The remaining 7 null hypotheses are analogous.

If a null hypothesis can be rejected with high confidence ($\alpha = 0.05$), we accept a two-sided alternative hypothesis stating that the predicted configuration has an effect on the ranking of the first relevant method compared to the heuristic configuration. For example, the alternative hypothesis corresponding to the example null hypothesis is:

$H_A : Predicted_{ArgoUML} \neq Heuristic_1$ $Predicted_{ArgoUML}$ does significantly affect the accuracy of the LDA-based FLT compared to $Heuristic_1$.

The remaining 7 alternative hypotheses are analogous.

3.1.8 Data Collection and Analysis

We collected two kinds of data for this case study. First, size metrics were collected for the corpora. In particular, a corpus was built using each of the seven text extractor configurations, and we collected three size metrics for each corpus:

Terms the number of unique terms

Uses the total number of term uses (i.e., instances)

Docs the number of non-empty documents

Table 3.5 lists the size metrics. Note that the *Docs* values for some corpora are less than the numbers of methods in the system. For example, Rhino's *L* corpus contains 522 non-empty documents, whereas Rhino contains 2,565 methods. This is because some Rhino methods contain no string literals — documents for such methods are empty.

We also collected the effectiveness measure, which is the primary data of interest in our case study. For each feature and configuration, we collected one effectiveness measure for each query. We then analyzed the data for each system/configuration pair to determine the minimum,

Metric	*I*	*C*	*L*	*IC*	*IL*	*CL*	*ICL*
Terms	11,065	5,866	1,565	12,735	11,549	6,283	13,033
Uses	326,417	144,162	13,357	470,579	339,774	157,519	483,936
Docs	11,348	10,781	2,352	11,348	11,348	10,853	11,348

(a) ArgoUML

Metric	*I*	*C*	*L*	*IC*	*IL*	*CL*	*ICL*
Terms	8,081	3,725	2,411	9,476	9,426	4,969	10,500
Uses	223,638	44,780	23,643	268,418	247,281	68,423	292,061
Docs	5,323	2,151	1,760	5,323	5,323	2,983	5,323

(b) JabRef

Metric	*I*	*C*	*L*	*IC*	*IL*	*CL*	*ICL*
Terms	8,749	4,162	1,714	9,861	9,159	4,788	10,150
Uses	259,082	59,208	13,842	318,290	272,924	73,050	332,132
Docs	6,549	4,311	1,519	6,550	6,549	4,737	6,550

(c) jEdit

Metric	*I*	*C*	*L*	*IC*	*IL*	*CL*	*ICL*
Terms	10,491	4,552	1,624	11,943	11,222	5,289	12,556
Uses	273,507	122,604	7,932	396,111	281,439	130,536	404,043
Docs	8,811	4,393	923	8,811	8,811	4,624	8,811

(d) muCommander

Metric	*I*	*C*	*L*	*IC*	*IL*	*CL*	*ICL*
Terms	11,591	3,210	1,309	12,514	11,988	3,724	12,818
Uses	396,570	33,553	16,810	430,123	413,380	50,363	446,933
Docs	9,067	2,348	1,217	9,067	9,067	3,117	9,067

(e) Mylyn

Table 3.5: Corpora Size Metrics

Metric	*I*	*C*	*L*	*IC*	*IL*	*CL*	*ICL*
Terms	5,448	2,606	1,192	6,521	5,713	3,009	6,705
Uses	128,472	29,688	6,017	158,160	134,489	35,705	164,177
Docs	2,565	1,235	522	2,565	2,565	1,434	2,565

(f) Rhino

Table 3.5: Corpora Size Metrics

maximum, median, and the percentage of features for which the FLT failed, a phenomenon described in the next paragraph. The FLT fails only in the second part of the case study, in which we study different text extractor configurations, and in the fourth part of the case study, in which we study factor interactions.

For each feature/configuration pair, the FLT assigns a rank in the range $[1,n]$, where n is the number of methods, to each document in the corpus. That is, the range of the effectiveness measure is $[1,n]$. However, for some configurations the number of non-empty documents, m, is less than n. In such a case, the rank assigned to each empty document is implementation defined — a valid implementation may assign an empty document (which may represent a method in the gold set) any rank in the range $[m+1,n]$. Thus, in such cases, an effectiveness measure in the range $[m+1,n]$ is meaningless because it is not based on the similarity between the query (i.e., description of the feature) and the document. For a system/configuration pair, if the effectiveness measure for a feature is in the range $[m+1,n]$ then we consider the FLT to have failed and is reported.

In our initial data analysis, we use the Kruskal-Wallis test, a non-parametric test similar to the (parametric) one-way ANOVA test. If a Kruskal-Wallis test reveals a significant effect, a post-hoc test is conducted using pairwise Mann-Whitney tests with Holm correction. Further, if

a Mann-Whitney test reveals a significant difference in accuracy between two configurations, we compute the effect size ($r = Z/\sqrt{N}$, where N is the total number of samples). We use the following (standard) interpretations of the effect size, r: negligible for $|r| < 0.1$, small for $0.1 \leq |r| < 0.3$, medium for $0.3 \leq |r| < 0.5$, and large for $|r| \geq 0.5$.

3.2 Results

This section reports the results for each part of our case study.

3.2.1 Part 1: Testing for Interactions among Factors

Table 3.6 lists the results of a factorial ANOVA applied to the effectiveness measures for the 243 configurations applied to 618 features. We list all main effects but only statistically significant interaction effects. Four of the five factors have significant main effects, which is consistent with the results of Parts 2-4 of the case study. Only α does not have a significant main effect. There are four significant two-way interactions: (1) *Text* and K, (2) K and α, (3) *Text* and β, and (4) α and β. We anticipated the first and fourth interactions and accounted for them in the designs of Parts 3 and 4 of the case study. Further, the second and fourth interactions are inherent in LDA [Blei et al., 2003]. Though there is an interaction between *Text* and β, any adjustment must be to the β parameter. This is because *ICL* is the only value of *Text* for which there is good performance with no failures (see Part 3 of the case study). There is one significant three-way interaction, between K, α, and β. Again, this interaction is inherent in LDA. There are no significant four-way or five-way interactions.

3.2.2 Part 2: Configuring the Query

This subsection presents a statistical analysis of the results and then offers a discussion of the results.

Table 3.6: Results of a factorial ANOVA.

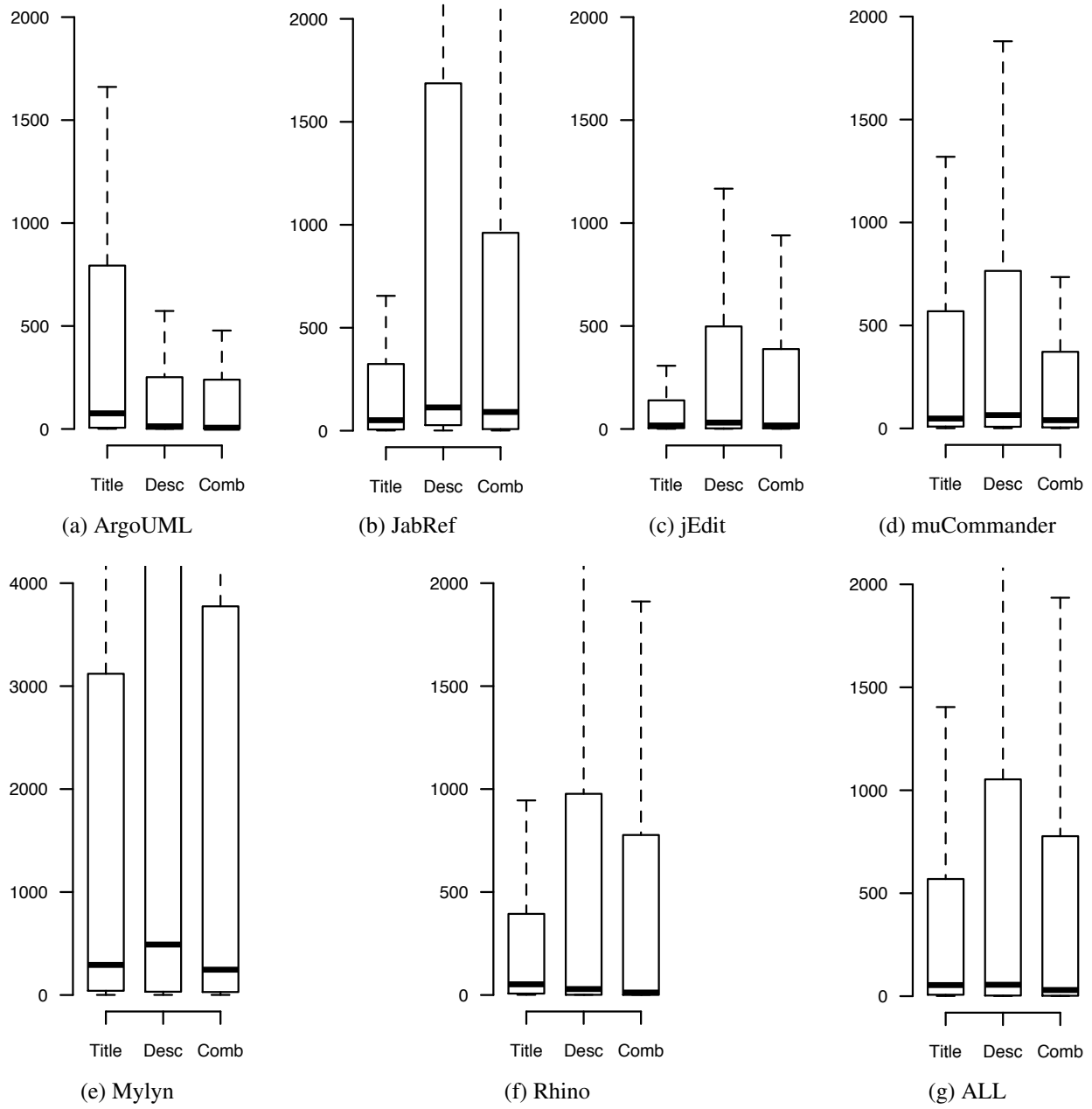
Factor	F value	p value
Query	79.3594	< 0.001
Text	57.6865	< 0.001
K	22.7080	< 0.001
α	2.8419	0.06
β	16.0175	< 0.001
$Text : K$	3.2958	0.01
$K : \alpha$	3.4374	< 0.01
$Text : \beta$	26.8548	< 0.001
$\alpha : \beta$	26.7096	< 0.001
$K : \alpha : \beta$	1.9650	< 0.05

3.2.2.1 Statistical analysis

Figure 3.1 illustrates box plots that represent statistics describing the effectiveness measures for the test data. Recall that a small effectiveness measure is better than a larger one (i.e., rank 1 is better than rank 1,000). Several of the maximum effectiveness measures are beyond the scales of the diagrams. However, we chose the scales to highlight the (small) differences between the medians. Further, we omit outliers for readability.

The box plots show that there is no consistent pattern across all systems, except that for each system there is relatively little difference between the medians of the three configurations. *Description* generally has the worst performance, though for ArgoUML and Rhino, the medians for *Description* are smaller (better) than those of $Title$. Similarly, *Combined* generally has the best performance, though for JabRef the median for *Combined* is larger (worse) than that of $Title$ and for jEdit the median for *Combined* is equal to that of $Title$.

For each system we conducted a Kruskal-Wallis test to determine whether the *Query* factor



(a) ArgoUML (b) JabRef (c) jEdit (d) muCommander

(e) Mylyn (f) Rhino (g) ALL

Figure 3.1: The effectiveness measure for three configurations (*Title*, *Description*, and *Combined*) of the LDA-based FLT applied to 91 ArgoUML features, 38 JabRef features, 149 jEdit features, 90 muCommander features, 93 Mylyn features, 157 Rhino features, and all 618 features.

has a significant effect on the accuracy of the LDA-based FLT. The test revealed a significant effect only for ArgoUML ($\chi^2(2) = 11.74, p < 0.003$), and a post-hoc test for ArgoUML using Mann-Whitney tests with Holm correction showed small differences between *Description* and

Title ($p < 0.008, |r| = 0.20$) and between *Combined* and *Title* ($p < 0.002, |r| = 0.24$). Based on our statistical results, for ArgoUML we can reject the null hypotheses which states that *Description* and *Combined* do not significantly affect the accuracy of the LDA-based FLT compared to *Title*, and we can instead accept the corresponding alternative hypotheses. However, the effect sizes are small in practice.

We also conducted a Kruskal-Wallis test on all 618 features, and the test revealed a significant effect ($\chi^2(2) = 7.67, p < 0.03$). The post-hoc test using Mann-Whitney tests with Holm correction showed the negligible difference between *Combined* and *Title* ($p < 0.006, |r| = 0.08$). Based on our statistical results, when the data from all six subject systems are considered together, we can reject the null hypothesis which states that *Combined* does not significantly affect the accuracy of the LDA based FLT compared to *Title*, and we can instead accept the corresponding alternative hypothesis. However, the effect size is negligible in practice.

Across all systems, *Combined* outperforms *Title*, and though we did not find a statistically significant relationship between them, *Title* generally outperforms *Description*. This is an interesting finding, because it indicates that length alone does not explain the effectiveness of a query. That is, if shorter queries provided the best performance, we would expect *Title* to outperform both *Description* and *Combined*. Similarly, if longer queries provided the best performance, we would expect both *Description* and *Combined* to outperform *Title*. Instead, the results indicate that *Title* and *Description* complement each other, as their combination outperforms either of them in isolation. Based on the results of this part of our study, we recommend using *Combined* (i.e., the combination of the title and the description) when automatically formulating a query string for LDA-based feature location.

3.2.3 Part 3: Configuring the Text Extractor and *K*

This subsection presents a statistical analysis of the results and provides a discussion of the results.

3.2.3.1 Statistical analysis

Table 3.7 lists statistics describing the effectiveness measures for the 28 configurations applied to all 618 features. In particular, for each configuration we list the minimum (best) rank, the maximum (worst) rank, the median rank, and the percentage of times that the configuration failed (see Section 3.1.8). In Table 3.7, all of a configuration's failures were included in the data and were assigned a rank equal to the number of methods in the particular system. For example, a failure for ArgoUML was assigned the rank 11,348 (the number of methods in ArgoUML), whereas a failure for Mylyn was assigned the rank 9,067 (the number of methods in Mylyn). No configuration that includes identifiers fails, whereas configurations that exclude identifiers fail from 5% to 28% of the time. Specifically, the *CL* configurations fail only about 5% of the time (for 30 of the 618 features), whereas the *L* configurations fail about 28% of the time (for 172 of the 618 features).

Table 3.7 highlights a surprising result. Configurations 8, 23 and 24 — (*C*,200), (*CL*,150) and (*CL*,200), respectively — have the lowest median ranks among the 28 configurations, even though these configurations each fail 5% to 8% of the time and are penalized harshly for each failure. The configuration with the lowest median rank and no failures is configuration 28 — (*ICL*,200) — with a median rank of 32.5. We conclude that failures may be an acceptable trade-off for the concomitant gain in accuracy, particularly if the LDA-based FLT is to be combined with another static or dynamic analysis (to form a hybrid technique).

ID	(*Text*,*K*)	Min	Max	Median	% Failed
1	$(I,75)$	1	10,206	70.5	0
2	$(I,100)$	1	10,151	70	0
3	$(I,150)$	1	10,152	57.5	0
4	$(I,200)$	1	10,141	54	0
5	$(C,75)$	1	9,101	58	8
6	$(C,100)$	1	8,948	38	8
7	$(C,150)$	1	10,093	37	8
8	$(C,200)$	1	10,159	31	8
9	$(L,75)$	1	11,348	145.5	28
10	$(L,100)$	1	11,348	122	28
11	$(L,150)$	1	11,348	133	28
12	$(L,200)$	1	11,348	138.5	28
13	$(IC,75)$	1	11,090	57	0
14	$(IC,100)$	1	11,076	48.5	0
15	$(IC,150)$	1	11,154	42.5	0
16	$(IC,200)$	1	10,884	46	0
17	$(IL,75)$	1	10,537	69	0
18	$(IL,100)$	1	10,547	60.5	0
19	$(IL,150)$	1	10,277	43	0
20	$(IL,200)$	1	10,332	52	0
21	$(CL,75)$	1	9,469	49	5
22	$(CL,100)$	1	10,244	34	5
23	$(CL,150)$	1	9,826	31	5
24	$(CL,200)$	1	10,481	29.5	5
25	$(ICL,75)$	1	10,673	48.5	0
26	$(ICL,100)$	1	10,632	51	0
27	$(ICL,150)$	1	10,719	37	0
28	$(ICL,200)$	1	10,772	32.5	0

Table 3.7: The effectiveness measure for 28 configurations (*Text*/*K* pairs) of the LDA-based FLT applied to all 618 features.

We conducted a Kruskal-Wallis test on all 618 features. In the test, all of a configuration's failures were included in the data and were assigned a rank equal to the number of methods in the particular system. Thus, this test skewed the results in favor of the configurations with the fewest failures.

The Kruskal-Wallis test (failures assigned maximum rank) revealed a significant effect ($\chi^2(27) = 215.37, p < 0.001$). Based on the post-hoc test using Mann-Whitney tests with Holm correction, we can reject 89 (of 378) null hypotheses. Effect sizes range from negligible to small. In particular, the largest effect size (between configuration 9 and 24) is $|r| = 0.20$.

3.2.4 Part 4: Configuring α and β

This subsection presents a statistical analysis of the results and provides a discussion of the results.

3.2.4.1 Statistical analysis

Figure 3.2 illustrates box plots that represent statistics describing the effectiveness measures for the 36 configurations applied to all 618 features.

Among each group of six configurations that share the same α value, the configuration with $\beta = 0.01$ (the smallest value for β) performs the worst. That is, configurations 1, 7, 13, 19, 25, and 31 perform worst in their respective groups. Similarly, among each group of six configurations that share the same α value, the configuration with $\beta = 0.1$ (the second smallest value for β) performs either second- or third-worst. That is, configurations 2, 8, 14, 20, 26, and 32 perform second- or third-worst in their respective groups.

For each system, we conducted a Kruskal-Wallis test to determine whether the α and β factors together have a significant effect on the accuracy of the LDA-based FLT. The test revealed significant effects for JabRef ($\chi^2(35) = 54.06, p < 0.03$), jEdit ($\chi^2(35) = 76.13, p < 0.001$), mu-

Table 3.8: Key for Section 3.2.4. Each table entry provides an index for an α/β pair.

1	2	3	4	5	6
(0.01,0.01)	(0.01,0.10)	(0.01,0.25)	(0.01,0.50)	(0.01,0.75)	(0.01,1.00)
7	8	9	10	11	12
(0.10,0.01)	(0.10,0.10)	(0.10,0.25)	(0.10,0.50)	(0.10,0.75)	(0.10,1.00)
13	14	15	16	17	18
(0.25,0.01)	(0.25,0.10)	(0.25,0.25)	(0.25,0.50)	(0.25,0.75)	(0.25,1.00)
19	20	21	22	23	24
(0.50,0.01)	(0.50,0.10)	(0.50,0.25)	(0.50,0.50)	(0.50,0.75)	(0.50,1.00)
25	26	27	28	29	30
(0.75,0.01)	(0.75,0.10)	(0.75,0.25)	(0.75,0.50)	(0.75,0.75)	(0.75,1.00)
31	32	33	34	35	36
(1.00,0.01)	(1.00,0.10)	(1.00,0.25)	(1.00,0.50)	(1.00,0.75)	(1.00,1.00)

Commander ($\chi^2(35) = 67.46, p < 0.001$), and Mylyn ($\chi^2(35) = 128.18, p < 0.001$). For JabRef and muCommander, post-hoc tests using Mann-Whitney tests with Holm correction showed no significant differences between any two configurations (α/β pairs). Based on an analogous post-hoc test for jEdit, we can reject only three of the 630 null hypotheses, and similarly, based on the post-hoc test for Mylyn, we can reject only 30 null hypotheses (or about 5% of the 630 null hypotheses). Of the 30 rejected null hypotheses for Mylyn, all pertain to configurations 1, 7, and 13, which share the common value 0.01 for β.

We also conducted a Kruskal-Wallis test on all 618 features, and the test revealed a significant effect ($\chi^2(35) = 250.21, p < 0.001$). Based on the post-hoc test using Mann-Whitney tests with Holm correction, we can reject 95 (of 630) null hypotheses. However, effect sizes range from negligible to small. In particular, the largest effect size (between configurations 7 and 34) is

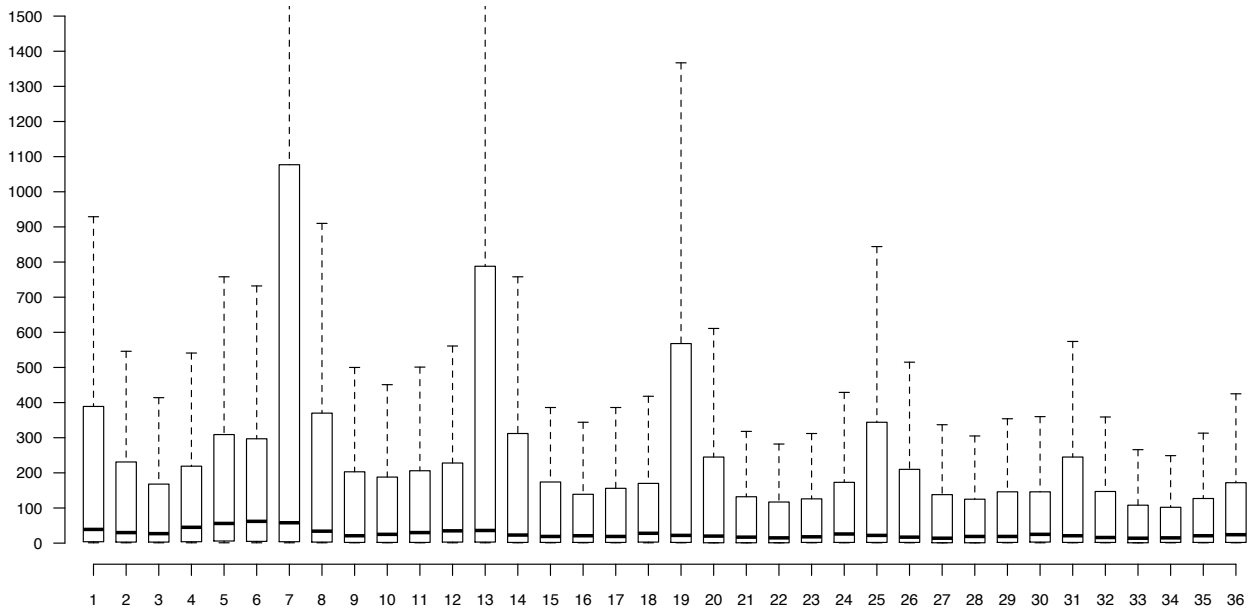


Figure 3.2: The effectiveness measure for 36 configurations (α/β pairs) of the LDA-based FLT applied to all 618 features.

$|r| = 0.21$. The stability of the medians across the 36 configurations demonstrates the small effect sizes.

3.2.5 Part 5: Applying the Lessons Learned

This subsection describes the results of applying the lessons learned and offers recommendations for configuring the LDA-based FLT.

3.2.5.1 Results of Applying the Lessons Learned

In this section we apply the lessons learned from Parts 1–4 of the case study. In particular, the lessons learned are used to predict an improved configuration for each of four subject systems — ArgoUML, JabRef, jEdit, and Rhino— when compared to two generic configurations informed by heuristics from the literature. We chose ArgoUML because it is the largest of our subject systems, Rhino because it is the smallest, and JabRef and jEdit because they are of similar size and have similar source line to comment line ratios.

Table 3.9 repeats the predicted configurations and the heuristic configurations. We first review the heuristic configurations. Both $Heuristic_1$ and $Heuristic_2$ have the value *Combined* for the *Query* parameter. To the best of our knowledge, Lukins et al. [2010] specify the only (manual) query formulation process for TR-based feature location. However, to avoid bias in our study, we automatically formulate queries from issue reports. Thus, we select *Combined*, because it performed the best in Part 2 of our study. Both $Heuristic_1$ and $Heuristic_2$ have the value *ICL* for the *Text* parameter. We select *ICL* because most of the previous studies of text retrieval based feature location use *ICL*. Both $Heuristic_1$ and $Heuristic_2$ have the value 200 for the K parameter. We select 200 topics because our systems are of medium size. We use the heuristic $50/K$ and set α to 0.25 for both $Heuristic_1$ and $Heuristic_2$. Finally, $Heuristic_1$ has the value 0.01 for β, and $Heuristic_2$ has the value 0.1 for β.

Table 3.9: Case study design: Part 5.

Configuration	*Query*, *Text*, K, α, β
$Predicted_{ArgoUML}$	*Combined*, *ICL*, 500, 1.0, 0.1
$Predicted_{jEdit}$	*Combined*, *ICL*, 400, 1.0, 0.25
$Predicted_{JabRef}$	*Combined*, *ICL*, 400, 1.0, 0.25
$Predicted_{Rhino}$	*Combined*, *ICL*, 300, 1.0, 0.5
$Heuristic_1$	*Combined*, *ICL*, 200, 0.25, 0.01
$Heuristic_2$	*Combined*, *ICL*, 200, 0.25, 0.1

All predicted configurations have the value *Combined* for *Query*, because *Combined* performed the best in Part 2 of our study. Further, all predicted configurations have the value *ICL* for *Text*. We select *ICL* because in Part 3 of our study, *ICL* performed the best of the configurations that did not have any failures. Similarly, we select the K values based on the results of Part 3. Specifically, in Part 3 we observed that 200 was too small a value for K when paired with *ICL* for

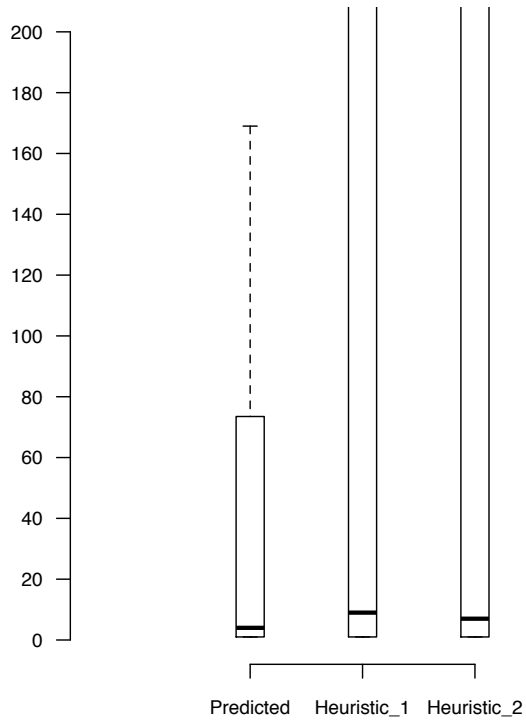
Text. Indeed, we found that performance increased when setting K to 300 instead. Thus, we set $K = 300$ for Rhino, the smallest of our systems. We conjecture that further increasing the value of K for our larger systems can further increase performance. Thus, we set $K = 400$ for jEdit and JabRef, and we set $K = 500$ for ArgoUML, the largest of our systems. We set $\alpha = 1.0$ for all systems, because that value resulted in the best performance in Part 4 of our study. Finally, we set β for each system based on the value of K. We select $\beta = 0.5$ for Rhino ($K = 300$), because that is the β value that resulted in the best performance in Part 4. For the remaining systems, we use a β value that is inversely proportional to the value of K.

Figure 3.3 illustrates box plots that represent statistics describing the effectiveness measures for the test data. The box plots show that our predicted configurations outperform the heuristic configurations in all cases. Moreover, the predicted configurations significantly affect the accuracy of the LDA based FLT compared to the heuristic configurations.

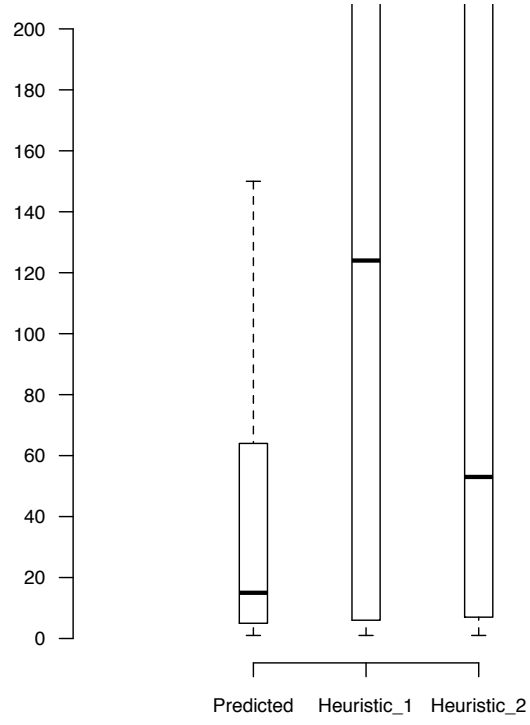
Recommendations

Based on our results, we have a number of recommendations for configuring the LDA based FLT. First, we define system sizes. We consider a small system to be one with less than 100 KLOC, less than 10 K unique terms, and less than 200 K term usages, whereas we consider a small-medium system to be one with between 100 KLOC to 200 KLOC, approximately 10 K unique terms, and between 200 K to 500 K term usages. We consider a medium-large system to be one with greater than 200 KLOC, greater than 10 K unique terms, and approximately 500 K term usages, and finally, we consider a large system to be one with greater than 1,000 KLOC. Based on our results, the following are our recommendations:

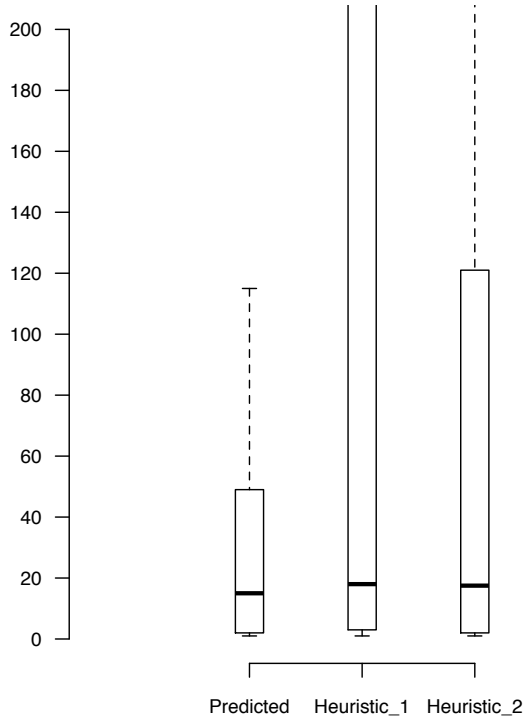
1. Set $K = 300$ for small systems, set $K = 400$ for small-medium systems, and set $K = 500$ for



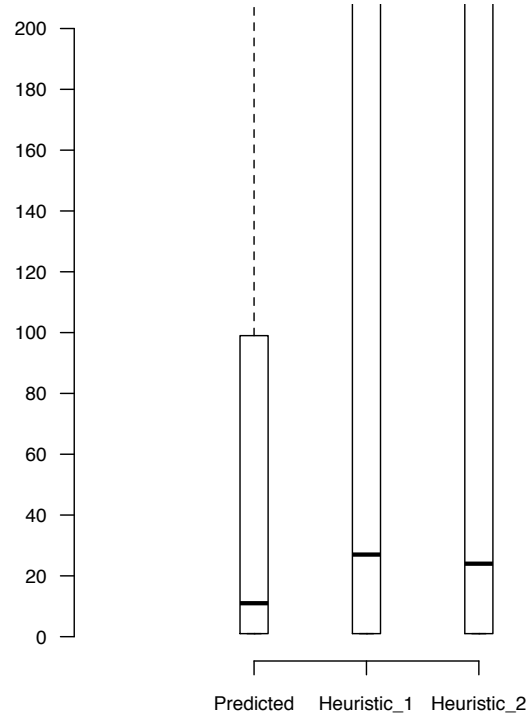
(a) ArgoUML



(b) JabRef



(c) jEdit



(d) Rhino

Figure 3.3: The effectiveness measure for three configurations (*Predicted*, $Heuristic_1$, and $Heuristic_2$) of the LDA-based FLT applied to 91 ArgoUML features, 38 JabRef features, 149 jEdit features, and 157 Rhino features.

medium-large and large systems. Our recommendation regarding large systems is based on prior experience with Eclipse and Mozilla [Lukins et al., 2008, 2010].

2. Set $\alpha = 1.0$ for all systems.

3. Set β for each system based on the value of K. Set $\beta = 0.5$ for small systems, and for larger systems, set β to be inversely proportional to K. For example, set $\beta = 0.25$ for small-medium systems, and set $\beta = 0.1$ for medium-large and large systems.

3.3 Discussion of Results

This section discusses the results of our study.

3.3.1 Part 2: Configuring the Query

Our statistical analysis revealed that the *Query* factor has a significant effect on the accuracy of the LDA-based FLT.

Figure 3.1g shows that the three queries produce similar results overall. However, it does not help us to understand the relative performance of the different queries on the same feature. That is, for a given feature, we cannot determine from the figure whether the three queries provide similar performance. Therefore, we investigated how often all three queries return the same effectiveness measure and found that this happened only 3% of the time (20 of 618 times). We also found that for 17 of those 20 features, all three queries returned 1, the best possible effectiveness measure. We then investigated how often all three queries return effectiveness measures within 10 ranks of each other (17% of the time or 102 of 618 times), within 50 ranks of each other (32% of the time or 198 of 618 times), and within 100 ranks of each other (40% of the time or 248 of 618 times). Note that at least one of the queries performs noticeably worse than the others 83% of the time.

In the following paragraphs, we highlight a number of features and discuss the performance of the LDA-based FLT given different queries for each of these features.

Consider feature 4019[10] for ArgoUML. The three queries are shown in Table 3.10. For each of the three queries, the LDA-based FLT returned 1. In particular, each of the queries returned *ProjectBrowser.loadProject*, a method from the gold set, first in the list of results. Upon inspection of the queries, we note that the *Title* and the *Description* are of similar length and that four of the five words in *Title* also appear in *Description*. Thus, it is probably not surprising that the queries performed similarly.

Next, consider feature 2842444[11] for jEdit. Table 3.11 lists the three queries. For each of the three queries, the LDA-based FLT returned 1. However, each of the queries returned a different method (from the gold set) first in the list of results. Moreover, the three different methods come from two distinct classes — *HyperSearchRequest* and *HyperSearchResults*. Upon inspection of the queries, we note that the *Title* and the *Description* have noticeably different lengths. In particular, *Title* has 7 words, whereas *Description* has 51 words (which include 5 of the 7 words from *Title*). This feature is an example of a finding described in the previous section. In particular, this feature demostrates that length alone does not explain the effectiveness of a query.

Consider feature 311[12] for muCommander. The three queries are shown in Table 3.12. For the *Title* query the LDA-based FLT returned 5, for the *Description* query it returned 3, and for the *Combined* query it returned 1. Next consider feature 352319[13] for Rhino. Table 3.13 lists the three queries. For the *Title* query the LDA based FLT returned 62, for the *Description* query it returned 3, and for the *Combined* query it returned 1. These features are examples of another

[10] http://argouml.tigris.org/issues/show_bug.cgi?id=4019
[11] http://sf.net/tracker/?func=detail&aid=2842444&group_id=588&atid=300588
[12] http://trac.mucommander.com/ticket/311
[13] https://bugzilla.mozilla.org/show_bug.cgi?id=352319

finding that we described in the previous section. In particular, these features demonstrate that *Title* and *Description* can complement each other to provide improved performance as *Combined*. This observation is consistent with the way LDA models documents — unique or rare (within the corpus) word co-occurrences are key to differentiating documents from the corpus.

Table 3.10: Queries for ArgoUML, feature 4019.

Query	Contents
Title	save project dialog rememb load
Description	save project dialog assum save filenam last project save project load
Combined	*Title* + *Description*

Table 3.11: Queries for jEdit, feature 2842444.

Query	Contents
Title	manual stop hypersearch hyper search oper request
Description	maximum result option prompt stop continu hypersearch hyper search nice manual stop button purpos patch stop button ad us default set icon hypersearch hyper search pane highlight multi result button disabl search current activ enabl otherwis stop basic mimick max result mechan except temporari properti indic stop button click handl fine
Combined	*Title* + *Description*

Table 3.12: Queries for muCommander, feature 311.

Query	Contents
Title	free space indic flip
Description	free space indic look compar what normal normal bar amount space drive fill bar fill drive cool bar gradual chang color space drive orang color red color
Combined	*Title* + *Description*

We also investigated how often all three queries returned effectiveness measures greater than 100 and found that this happened 28% of the time (174 of 618 times). The most common

Table 3.13: Queries for Rhino, feature 352319.

Query	Contents
Title	cant restart continu catch block
Description	attempt restart continu captur catch block except thrown function enteractivationfunct enter activ function scriptruntim script runtim java line nativecal nativ call call nativecal nativ call activ invok interpret java line scriptruntim script runtim enteractivationfunct enter activ function frame scope throw frame scope nativewith nativ creat catch block instead nativecal nativ call attach file testcontinu test continu java testcontinu test continu jointli reproduc note wont root caus run itll trigger interpret except handler thatll run npe continu stack wasnt restor properli root caus guess rewrit interpret java line doesnt specifi frame scope instead walk parent chain nativecal nativ call help chang signatur scriptruntim script runtim enteractivationfunct enter activ function accept nativecal nativ call instead gener scriptabl help enforc practic
Combined	*Title* + *Description*

causes of such performance include queries that contain misleading words and gold set methods that are short or contain only words that are common in the corpus. For example, consider method *BugzillaAttachmentHandler.uploadAttachment*, which is listed in Figure 3.4. This method is in the gold set for feature 151257[14] for Mylyn. Indeed, this method is the only member of the gold set for feature 151257. The LDA-based FLT performs poorly for this feature, because *BugzillaAttachmentHandler.uploadAttachment* is relatively short and because its words often co-occur in the corpus.

3.3.2 Part 3: Configuring the Text Extractor and *K*

Our statistical analysis revealed that the *Text* and *K* factors have a significant effect on the accuracy of the LDA based FLT.

The results in Table 3.7 demonstrate that comments and literals play an important role in the performance of the LDA-based FLT. That is, using identifiers only (*I*) for the LDA based

[14] `https://bugs.eclipse.org/bugs/show_bug.cgi?id=151257`

```
@Override
public void uploadAttachment(TaskRepository repository, AbstractTask
task, ITaskAttachment attachment, String comment, IProgressMonitor
monitor) throws CoreException {
    try {
        String bugId = task.getTaskId();
        BugzillaClient client = connector.getClientManager()
.getClient(repository);
        client.postAttachment(bugId, comment, attachment);
    } catch (IOException e) {
        throw new CoreException(new BugzillaStatus(Status.ERROR,
     BugzillaCorePlugin.PLUGIN_ID,
                RepositoryStatus.ERROR_IO, repository.getUrl(), e));
    }
}
```

Figure 3.4: Source code for Mylyn method *BugzillaAttachmentHandler.uploadAttachment*.

FLT would result in reduced performance. Comments often provide a rich set of terms related to the problem domain, and literals often map directly to error messages or other aspects of the user interface that are mentioned in issue reports. On the other hand, identifiers often reflect the solution domain.

The configurations that provide the best performance are those where *Text* is *C*, *CL*, or *ICL* and where *K* is 200. Though $(CL, 200)$ provides the best absolute performance, it is subject to a 5% failure rate. Moreover, because the corpora grow substantially when adding identifiers — that is, when switching from *CL* to *ICL* — we tested whether increasing *K* further (e.g., to 300) would permit *ICL* to provide the best performance. Indeed, when we increased *K* from 200 to 300, *ICL* provided the best performance.

3.3.3 Part 4: Configuring α and β

In the statistical sense, the α and β factors have little influence on the accuracy of the LDA-based FLT. However, like Griffiths and Steyvers [2004], we observe that β has more influence than α. In addition, in the context of our case study, 0.25 and 0.50 are the β values that provide the

best accuracy. Even though the observed effect sizes are relatively small, we find it interesting that the *de* facto standard heuristics from the natural language document clustering community are not optimal in this (source code) context.

3.4 Threats to Validity

Our study has limitations that impact the validity of our findings, as well as our ability to generalize them. We describe some of these limitations and their impacts.

Threats to conclusion validity concern the degree to which conclusions we reach about relationships in our data are reasonable. We made no assumption about the distribution of the effectiveness measures, so we used a non-parametric statistical hypothesis test. Moreover, we used an adjustment method to control the family-wise error rate of our hypothesis tests. The test results are consistent with our observations.

Threats to construct validity concern the adequacy of the study procedure with regard to measurement of the concepts of interest and can arise due to poor measurement design. One such threat relates to our benchmarks. Specifically, errors in our gold sets would affect the correctness of our effectiveness measures. To mitigate this threat, we used previously used gold sets [Dit et al., 2012; Revelle et al., 2010]. The six gold sets were produced by other researchers and made available to the community.

Threats to internal validity include possible defects in our tool chain and possible errors in our execution of the study procedure, the presence of which might affect the accuracy of our results and the conclusions we draw from them. We controlled for these threats by testing our tool chain and by assessing the quality of our data. Because the same tool chain was applied to all subject systems, any errors are systematic and are unlikely to affect our results substantially.

Additional threats to internal validity are due to the preprocessing steps we applied to tex-

tual information extracted from source code. For example, applying a stemmer can cause terms with different meanings to be mapped to the same stem, thus causing overfitting of the data. Applying a more advanced stemmer may mitigate this issue. However, we believe that such a change is unlikely to affect our results substantially. Other decisions that potentially affect our results are the choices to split tokens, to retain the original (unsplit) tokens, and to filter stop words.

Another threat to internal validity pertains to our queries, which were obtained directly from issue report titles and descriptions. It is possible that these titles and descriptions do not accurately reflect the features of interest. However, four of the subject systems in our study are used primarily by software developers, who are likely more able than end users to accurately describe the faulty features. Although we certainly could have obtained better results by tuning the queries [Lukins et al., 2010; Poshyvanyk et al., 2007], our query formulation process prevented us from introducing bias.

Threats to external validity concern the extent to which we can generalize our results. The subjects of our study comprise 618 features in 6 open source Java systems, so we cannot generalize our results to systems implemented in other languages. However, the systems are from different domains and have characteristics which are similar to those of systems developed in industry.

3.5 Summary

This chapter presented preliminary work on configuring a LDA-based FLT. LDA is a highly configurable technique, and consideration must be taken to properly choose the correct α, β, K, as well as the best query and terms from source code. This preliminary work looked at each of these factors and the interactions between them. Based on the results of this study, recommendations were given for different sizes of software systems. The next chapter uses the recommendations from this work for my configurations.

Chapter 4

STRUCTURAL WEIGHTING OF LDA

This chapter considers the problem of structural weighting in LDA. Previous research has shown [Liu et al., 2007; Poshyvanyk et al., 2007; Revelle et al., 2010; Zhao et al., 2006] that combining structural and dynamic information with LSI and VSM will improve the results of these techniques for feature location. Little research has been performed to examine the combination of such sources with topic modeling. Of the two types of information, structural information fits uniquely well with TR techniques as both require static information and can be computed without the need of a running system. For systems where execution is not possible or too time consuming for an individual feature location task, structural information may be already factored into the topic model.

Structural information has been combined with TR techniques [Zhao et al., 2006] to improve results in various ways. Typical techniques involve creating the full call graph from a set of source files. Instead, Bassett and Kraft [Bassett and Kraft, 2013] proposed structural weighting as a lightweight method for improving the traditional LDA model with some of the information found in a call graph. Their study compared the accuracy of an LDA-based FLT when increasing the weighting of terms derived from method names or methods calls. Their hypothesis was that a method's name contains terms that describe its behavior and purpose but that those terms are not necessarily common throughout the method's document. Furthermore, increasing weights of terms in method calls might be considered a lightweight alternative to adding call graph information.

Term weighting is a preprocessing transformation in which weights are applied to terms to adjust their importance within a document or corpus. Several different weighting schemes exist. As an example, *tf-idf* places increased importance on terms that appear frequently in a particular document, but appear infrequently throughout the entire corpus. In the *tf-idf* model, importance is based on (normalized) term counts rather than on the role or meaning of a particular term.

The work by Bassett and Kraft was a first step in producing a general structural weighting scheme for LDA. The importance of different weighting schemes seems sensible in that unlike natural language documents, it is believed that source code elements will only contain a small number of dominant topics. Increasing the weight of important terms that may be less frequent in the source element's document, should give a better topic distribution around those dominant topics for that source element. However, work performed in the original study was preliminary. Their study only investigated method names and method calls, and did not give any insights into how to best define a weighting scheme for a new software system. My research increases the number of weighting configurations under consideration from 16 in the original study to 1,024. My research considers a broader scope of variables that has potential to have a significant effect on the results of a TR-based FLT. Namely, my research focuses on leading comments, method names, parameters, body comments, and local variables. I have carefully identified these term sets (referred to as lexicons from henceforth) based on prior experience and understanding of how each lexicon may play a role in the model. Leading comments often contain information pertaining to the main responsibility of a method, but this responsibility may be less evident from the source element's identifiers. Parameters often describe the initial state of a method and could contain additional information about the method's responsibility that is complementary to the information found in a method name. Body comments (i.e., comments found within a method body) are written

by the developers to provide explanations of particular statements that may not be clear from the source code identifiers alone. It is possible for local variables to reflect information that is similar to the parameters and may be complementary to the other lexicons selected.

My approach examines different weighting configurations based on multiple factors. First, each lexicon is observed to understand how it affects the performance of an LDA-based feature location technique when only the terms present within that lexicon are weighted. Second, I consider the top configurations for each system and look for any main effects caused by weighting the different lexicons. These results are compared to the metrics discussed in Chapter 1, to try to give an explanation of the results. Finally, a technique based on genetic algorithms is presented that allows developers to learn the best configuration for a software system over time. This technique has the benefit of constantly improving the configuration until an optimal solution is discovered. I use a large benchmark to perform my analysis. The benchmark contains over 350 features and bugs from four open source Java systems. For each of the five lexicons previously listed, I use 4 different levels of weighting, resulting in 1024 different weighting schemes for each system. The final section of this chapter will use Eclipse as a large system with an unknown weighting configuration.

The main contributions of this chapter are an expansion of the structural term weighting scheme presented by Bassett and Kraft [Bassett and Kraft, 2013], the results of the empirical study, and the knowledge that is derived from the results.

The remainder of the chapter is organized as follows. Section 4.1 presents my empirical study, Section 4.2 describes the results of this study, and Section 4.4 introduces a basic training technique for weighting configurations and the results of my training technique on Eclipse. Finally, Section 4.5 presents a summary of the contributions and describes future work.

4.1 Study Design

An empirical study was conducted to evaluate the effects of structural term weighting on terms derived from comments, method signatures, and local variables on the accuracy of a LDA-based FLT. This work is a partial replication of the work presented in Bassett and Kraft [Bassett and Kraft, 2013] as well as an expansion. For this reason, I adopt similar notation and scaling factors used in this earlier study. Namely, I use scaling factors that double with each increase up to a maximum of 8. By doubling the weight, the number of configurations requiring testing reduces from 32,000 to a more feasible 1,024 configurations for the study. Second, by doubling each time instead of increasing linearly, changes in the performance should be more evident.

This section describes the design of the empirical study as well as a discussion of the threats to the validity of the results.

4.1.1 Definition and Context

The primary goal of the study is to understand the effects structural weighting schemes for terms derived from comments, the method signature, and local variables have on the accuracy of a LDA-based FLT.

In this study, I computed 1,024 separate topic models for each of four open source Java software systems, resulting in 4,096 separate topic models. Each topic model corresponds to a different structural weighting scheme-software system pair. Over the four software systems, I compared the accuracy of a FLT for 372 features and bugs.

For the weighting schemes, I considered terms derived from leading comments (i.e., those preceding the method), method signature (method names and parameters), body comments (i.e., those in the method body), and local variables (i.e., identifiers only visible in the current method).

My choices are based off of prior research in the area of source code summarization [Eddy, Robinson, Kraft, and Carver, 2013; Haiduc, Aponte, and Marcus, 2010]. In these studies, researchers asked human subjects to rank automatically generated summaries of source code elements and to identify any terms that the subjects found relevant to the element. Subjects identified summaries that contained leading terms (i.e., terms contained in the method signature and leading comments) as those that produced the more descriptive summaries. Their reasoning was that terms from these sources often described the intent and behavior of the element better than other related terms. Therefore, I hypothesize that by raising such terms, there will be an improvement in the performance of the LDA-based technique as the system will be able to more accurately identify the dominant topics belonging to the source document. In addition, I believe that comments in a method's body may help to describe the individual functions in the methods, and that local variables are complementary to parameters and the other lexicons included in the study.

As in Bassett and Kraft [Bassett and Kraft, 2013], I multiplied the number of terms originating from each of these groups by a scaling factor. For comparisons, I used the same scaling factors presented in Bassett and Kraft's paper: 1, 2, 4, and 8. These scaling factors multiply the raw term counts utilized by LDA. I use similar notation to that used by Basset and Kraft. The notation is:

$$C(LC, M, P, BC, LV)$$

where M is the method names scaling factor, LC is the leading comments scaling factor, P is the parameters scaling factor, BC is the body comments scaling factor, and LV is the local variables scaling factor. I allow each scaling factor to vary independently, resulting in a total of 1,024 configurations. Configuration $C(1,1,1,1,1)$ establishes the baseline (uniform term weighting).

4.1.1.1 Subject Software Systems

Four open source Java software systems were used as the subject software systems in the study — ArgoUML[1], JabRef[2], jEdit[3], and muCommander[4]. These systems represent various size software systems and application domains. They are comparable to systems developed in industry, and been used in previous studies that I have worked on. For each of these subject systems, benchmarks are available online.

A description of the subject systems and versions is described in Table 4.1. For each of the subject systems, I give the source lines of code (SLOC), the comment lines of code (CLOC), the number of methods in the Java source code, and the number of features and bugs used in this study.

The domains of the systems are as follows: ArgoUML is a UML modeling tool with support for all UML 1.4 diagrams. JabRef is a cross-platform bibliography reference management tool. jEdit is a configurable and customizable text editor for programmers. muCommander is a cross-platform virtual file browser.

Table 4.1: Subject systems

System	Version	SLOC	CLOC	Methods	Features
ArgoUML	0.22	117,649	104,037	12,542	91
JabRef	2.6	74,350	25,927	5,323	39
jEdit	4.3	98,460	42,589	7,305	150
muCommander	0.8.5	76,649	68,367	8,811	92
Total		367,108	240,920	33,981	372

[1] `http://argouml.tigris.org`
[2] `http://jabref.sourceforge.net`
[3] `http://www.jedit.org`
[4] `http://www.mucommander.com`

4.1.1.2 Benchmarks

I utilized the SEMERU[5] benchmarks for my study. This is a set of benchmarks created by other researchers and used in a number of studies in the feature location community. Each benchmark contains a set of features and a "gold set" [Dit, Guerrouj, Poshyvanyk, and Antoniol, 2011].

Each feature in the benchmarks corresponds to an issue reported via an issue tracking system for the software system. An issue report is either a request to change an unwanted functionality or a request to add a new functionality to the code. For the purposes of this study, I use both types and do not distinguish between the two. Queries are drawn from these issue reports by combining the title with the description [Dit et al., 2011]. These queries reduce bias by mimicking queries that a developer might formulate without intimate knowledge of the system.

A gold set corresponds to the set of methods that were believed to have been changed in order to resolve the issue report. These methods were obtained by using the diffs stored in Subversion and extracting the set of methods that were changed when the feature was resolved. Table 4.2 gives a summary of the number of methods found in each gold set used in this study, as well as descriptive statistics of the methods.

Table 4.2: Numbers of methods in the gold sets.

System	Min	Median	Mean	Max	StdDev	Total
ArgoUML	1	3	7.70	72	13.11	701
JabRef	1	4	7.18	33	8.78	280
jEdit	1	3	4.99	41	5.58	748
muCommander	1	3	7.80	104	16.06	718

[5] `http://www.cs.wm.edu/semeru/data/benchmarks`

Of the available benchmarks in SEMERU, I used 372 features and four open source Java systems.

4.1.1.3 Effectiveness Measure

The goal of automatic feature location is to identify a single method from which a developer can begin a change [Poshyvanyk et al., 2007]. Because static FLTs rank all methods in a system, recall and precision are not useful accuracy measures in this context. Furthermore, the objective of feature location differs from a traditional search because the task is only concerned with identifying an initial starting point. Other methods are identified by using the starting point to predict changes during impact analysis. In the context of static FLTs, recall is always 1.0, and precision is always $1/n$ (where n is the number of methods). Thus, the rank of the first relevant method is used instead [Liu et al., 2007; Lukins et al., 2008, 2010; Poshyvanyk et al., 2007; Revelle et al., 2010]. This measure, which [Poshyvanyk et al., 2007] term the "effectiveness measure" for feature location, indicates the number of entities that the developer must examine (if following the ranking) before reaching a method that actually belongs to the feature. That is, the measure quantifies the number of false positives that a developer must examine.

The effectiveness measure gives a way to interpret the effectiveness of a weighting configuration on a single feature. To evaluate the effectiveness of a weighting configuration for the FLT across all queries for that subject system, I use the mean reciprical rank. The mean reciprocal rank (MRR) of the FLT is given by the average of the reciprocal of the effectiveness measure given some sample set of queries [Voorhees, 1999]:

$$\text{MRR} = \frac{1}{|Q|}\sum_{i=1}^{|Q|}\frac{1}{r_i} \tag{4.1}$$

where Q is the set of queries and r_i is the effectiveness measure for some query Q_i. The more effective the technique, the higher the MRR should be. When determining the top configurations for each system, I use the MRR, however, because MRR does not give any information to the spread of the results, I also show the spread of the effectiveness measures.

4.1.1.4 Setting

To build my corpus, I use a text extractor and preprocessor implemented in Java 7 using an open source Java 1.5 grammar and ANTLR v3. The text extractor and preprocessor has five steps. The first step extracts documents from methods and treats inner methods as distinct methods. The text of the inner method (e.g., a method inside an anonymous class) will only be attributed to that method, and not the containing method. This step outputs each document as a separate XML file for further processing. The next step rewrites the separate XML files into a single XML file with comments preceding methods being folded into the text of that method. This single XML file is then reread as a corpus for preprocessing. During the preprocessing stage, terms in the corpus are split, normalized, stemmed, and all stop words are removed. The final step converts the corpus into a format that can be read by a text retrieval tool (in this case Mallet [6]). In this final step, terms are duplicated according to weights provided by the user and corresponding to the current structural term weighting configuration under study. I use the term weighting configurations described at the beginning of this section.

As in Bassett and Kraft, the preprocessor implements the steps described in Figure 1.1 in Chapter 1. Identifiers are filtered from `java.lang` before splitting tokens on camel case, underscores, and non-letters. After splitting, the original token is retained [Marcus et al., 2004]. The terms are then normalized to lowercase and then filtered by an English stop word list [Fox,

[6] `http://mallet.cs.umass.edu/`

1992], Java keyword list, and term length (with any terms less than three character being removed). A Porter stemmer[7] is then applied.

I use Mallet to generate the LDA model from the formatted corpus. I use the same modeling parameters proposed by Biggers et al. [Biggers, Bocovich, Capshaw, Eddy, Etzkorn, and Kraft, 2012]. For ArgoUML, I use $K = 500$, $\alpha = 1.0$, and $\beta = 0.1$. For the other three subject systems, I use $K = 400$, $\alpha = 1.0$, and $\beta = 0.25$. In addition to these parameters required by LDA, Mallet also requires a number of iterations for Gibbs Sampling [Casella and George, 1992] (used to approximate the LDA model). Again, as in the previous study, I use the default value of 1,000. I use Hellinger distance [Nikulin, 2001] to rank the results of a query comparing the topic distributions of the model to the inferred distribution from the query.

4.1.2 Research Questions

The focus of the case study is to address the following questions:

1. Does structural weighting of comments, leading terms, and local variables affect the accuracy of a LDA-based feature location technique (FLT)?
   (a) How does structural weighting of the individual stuctural lexicons affect the accuracy of a LDA-based FLT?
   (b) What are the top configurations for each system and across all systems?
   (c) What are the main effects and interactions between the structural lexicons?
2. Can a relationship between the contributions of each structural component's lexicon and their weighting factors be found?

[7] `http://tartarus.org/~martin/PorterStemmer/`

Parts (a), (b), and (c), are all questions that help to answer *Research Question 1* which addresses whether different weighting configurations have an effect on the accuracy of an LDA-based FLT. In order to answer these question, I first consider how weighting the structural lexicons independently affects a LDA-based FLT.

For *Research Question 1(b)* I conducted a Friedman test (discussed in Section 4.1.3) to determine whether a statistically significant effect existed between the top configurations and the unweighted configuration. If so, I form 55 hypothesis tests corresponding to each pair of weighting configurations in the top ten configurations. The independent variables for these tests are the weighting configurations with scaling factors (1, 2, 4, 8) and the dependent variables are the effectiveness measures.

For each hypothesis test, I do not presuppose the directionality of the difference between two configurations. Therefore, each hypothesis test is two-tailed. For each configuration pair, a null hypothesis is formulated to evaluate whether there is a significant difference when one of the two configurations is used over the other. If, after testing the null hypothesis, the null hypothesis can be rejected with a high confidence ($\alpha = 0.05$), an alternative hypothesis is accepted. Accepting the alternative hypothesis corresponds to there being a significant difference between the two structural weighting scheme configurations. These hypotheses are similar to the hypotheses used in the previous study.

An example null hypothesis:

$H_0 : C(4,2,4,4,4) = C(2,4,4,4,4)$

Configuration $C(4,2,4,4,4)$ **does not significantly affect** the accuracy of the LDA-based FLT compared to configuration $C(2,4,4,4,4)$.

The corresponding alternative hypothesis:

$H_A : C(4,2,4,4,4) \neq C(2,4,4,4,4)$

Configuration $C(4,2,4,4,4)$ **does significantly affect** the accuracy of the LDA-based FLT compared to configuration $C(2,4,4,4,4)$.

The remaining 54 null and alternative hypotheses are analogous.

4.1.3 Data Collection and Analysis

For the results of the FLT, I followed the same data collection process followed by Bassett and Kraft [Bassett and Kraft, 2013].

For each subject software system, the 1,024 topic models corresponding to the 1,024 different weighting scheme configurations were built. All queries were then ran for that system against each of the trained topic models. I collected the effectiveness measure corresponding to each query for each configuration. This resulted in a list of effectiveness measures for each query and configuration pair. This list of effectiveness measures was used as the basis for my analysis.

From this list of effectiveness measure, I computed descriptive statistics for each project and configuration pair. The descriptive statistics computed include the minimum rank found, the first quartile, the median, the third quartile, and the maximum rank found. Together, these descriptive statistics describe the spread of the effectiveness measures for that configuration on that system. I include the descriptive statistics and boxplots displaying the spread of this information for the independently weighted lexicons, as well as for the top ten configurations for each system and for all systems overall.

In addition to the descriptive statistics, I also compute the MRRs for each system and configuration pair using the effectiveness measures on that configuration for the given system. The

numbers obtained ranged from 0 to 1 with higher values indicating better results. These numbers are compared directly to discover trends in the data. The configurations with the highest MRRs form the top ten best configurations for each system. This was also performed for all systems combined.

To test my hypotheses in *Research Question 1(b)*, I first conducted a Friedman test to determine whether a statistically significant effect exists in the data. A Friedman test is a non-parametric statistical test that detects significant differences in data. It is the non-parametric analog to the one-way repeated measures ANOVA and is used when distributions of data cannot be assumed to be normal. The Friedman test was performed across the effectiveness measures. I performed a Friedman test for each individual system, as well as the top ten configurations for all four systems combined. If in any system I found a statistical difference, I performed post-hoc Wilcoxon signed-rank tests (the non-parametric analog to the t-test) with Holm p-value correction to identify which pairs of configurations had a statistically significant difference.

In addition to the quantitative comparisons for *Research Question 1(b)*, I also evaluate the results qualitatively and attempt to give some insight into possible reasons for the results.

As an extension to the qualitative analysis of *Research Question 1(b)*, in *Research Question 2* I calculate multiple system wide statistics and metrics for each lexicon. The statistics I gather include the number of terms in each lexicon system-wide, the number of uses for each term in the lexicon system-wide, and the number of documents that terms from the lexicon appears in. I also gather system-wide lexicon density, unique term density, and unique term contribution for each lexicon. Because the system wide values include calculations for documents where a lexicon does not appear, I also computed summary statistics for each of these values at the method level. Summary statistics include the minimum, median, maximum, mean, and standard deviation. Each

of these values collected give some insight into various ways that the different lexicons compose the entire system's lexicon. They are evaluated qualitatively to determine whether there may be a link between these values and the scaling factors applied in the top configurations.

4.1.4 Threats to Validity

The study has limitations that may affect the validity of my findings. This section describes some of the limitations as well as my attempts to mitigate them.

Threats to conclusion validity concern how reasonable the conclusions reached about the relationships in the data are. As in the previous study, in order to mitigate these I used non-parametric statistical tests and did not make any assumptions about the distributions of my effectiveness measures. In addition, I used Holm correction to account for errors in my p-values.

Threats to construct validity concern how well the measurements used in the study describe the concept being studied. Possible threats to construct validity include how well the effectiveness measure measures the feature location process and whether my benchmarks are accurate. I used established measurements and benchmarks that were used in previous research [Dit et al., 2011, 2012; Revelle et al., 2010] and made publicly available online.

Threats to internal validity include possible errors in executing my study procedure or defects in my tool chain. To mitigate these problems, I tested my tool chain and assessed the quality of my results at each step in the procedure. Because the same tool chain was applied to all subject systems, any errors are systematic and should not affect my results substantially.

As with the study by Bassett and Kraft, the queries are another threat to internal validity if they do not describe the features of interest. ArgoUML and jEdit are tools developed for programmers. Therefore, it is likely that those who wrote the issue reports were more likely to accurately describe the issue than other users.

Threats to external validity concern the extent to which I can generalize my results. All four of my subject systems are written in the Java programming language, therefore I am unable to generalize to different languages. However, the four subject systems represent a wide range of sizes and domains. Furthermore, the selected subject systems are similar to systems found in industry. Therefore, my results should be similar for other systems written in Java.

In the next section, I describe the results of my study using an LDA-based FLT for the four benchmarks. Due to the large number of configurations, only a subset of the complete results are included in my dissertation.

4.2 Results of Case Study

This section will discuss the results of the study by question.

4.2.1 Does structural weighting of comments, leading terms, and local variables affect the accuracy of a LDA-based feature location technique (FLT)?

This question is broken down into three parts. I look at the effects of weighting a lexicon by itself, and then in conjunction with the other lexicons. I give the top configurations and report any main effects and interactions that are found as the result of a factorial ANOVA.

4.2.1.1 How does structural weighting of the individual structural lexicons affect the accuracy of a LDA-based FLT?

For each of the structural components, I give the overall descriptive statistics of all effectiveness measures for each of the four software systems and all software systems combined, as well as the MRRs for each individual software system. The descriptive statistics give a high-level view of the configuration across the systems while the MRRs give a system-specific performance measure. I also use boxplots to give a visual representation of the data presented in the descriptive statistics. For clarity, the boxplots do not show outliers.

Leading Comments

Most comments are not written in the underlying programming language of the system. Instead, they are one of the few components of the software system that are expressed in the natural language of the developer. Leading comments in particular are believed to be particularly relevant when explaining the responsibilities and role of a method. Therefore, a software system with good leading comments should result in better performance when given heavier weighting in a FLT. I address how raising the weights on leading comments affects each of the subject software systems.

Table 4.3 gives the descriptive statistics for raising leading comments alone. Figure 4.1 gives the boxplots for the same information with outliers removed. The whiskers of the boxplots represent 1.5 times the interquartile range (IRQ). For each of the subject systems, we see that the minimum value for a search is 1. This means that for at least one of the queries issued on the subject system, the relevant method was returned as the first method in the ranked list. This is the optimal ranking for a feature location task. In ArgoUML, we can see that for the 1Q, median, and 3Q, the values go lower as the weight on the leading comments increases. We can see from the boxplot, that this results in a gradual decrease in the spread of the rankings as the weighting is increased. However, in the maximum value, we do not see the gradual decrease as in the other values. There is an overall decrease from the unweighted configuration to the maximum weighting, but in the case of a weighting factor of 4 we see an increase over the unweighted configuration. This result is an outlier and can be the result of slight differences in the training of LDA on a specific method's topic model. For JabRef, the 1Q, and median go lower as the weighting on the leading comments increases, however the 3Q increases in the maximum weighting from the previous weighting factor as does the maximum value. The median represents the point in which half of the queries are below

Config	min	1Q	median	3Q	max
C(1,1,1,1,1)	1	24	181	1093	10231
C(2,1,1,1,1)	1	22	148	904	9863
C(4,1,1,1,1)	1	16	132	718	10521
C(8,1,1,1,1)	1	14	72	532	8758

(a) ArgoUML

Config	min	1Q	median	3Q	max
C(1,1,1,1,1)	1	49	124	515	3940
C(2,1,1,1,1)	1	22	123	399	3861
C(4,1,1,1,1)	1	18	113	363	3679
C(8,1,1,1,1)	1	18	83	401	4390

(b) JabRef

Config	min	1Q	median	3Q	max
C(1,1,1,1,1)	1	10	49	240	6134
C(2,1,1,1,1)	1	7	31	231	6211
C(4,1,1,1,1)	1	5	22	163	5720
C(8,1,1,1,1)	1	6	29	126	4915

(c) jEdit

Config	min	1Q	median	3Q	max
C(1,1,1,1,1)	1	19	119	355	7354
C(2,1,1,1,1)	1	21	93	323	6821
C(4,1,1,1,1)	1	17	81	289	7429
C(8,1,1,1,1)	1	13	70	253	6391

(d) muCommander

Config	min	1Q	median	3Q	max
C(1,1,1,1,1)	1	13	101	616	10231
C(2,1,1,1,1)	1	12	91	559	9863
C(4,1,1,1,1)	1	7	77	456	10521
C(8,1,1,1,1)	1	7	70	427	8758

(e) All Systems

Table 4.3: Descriptive Statistics Weighting Leading Comments Alone

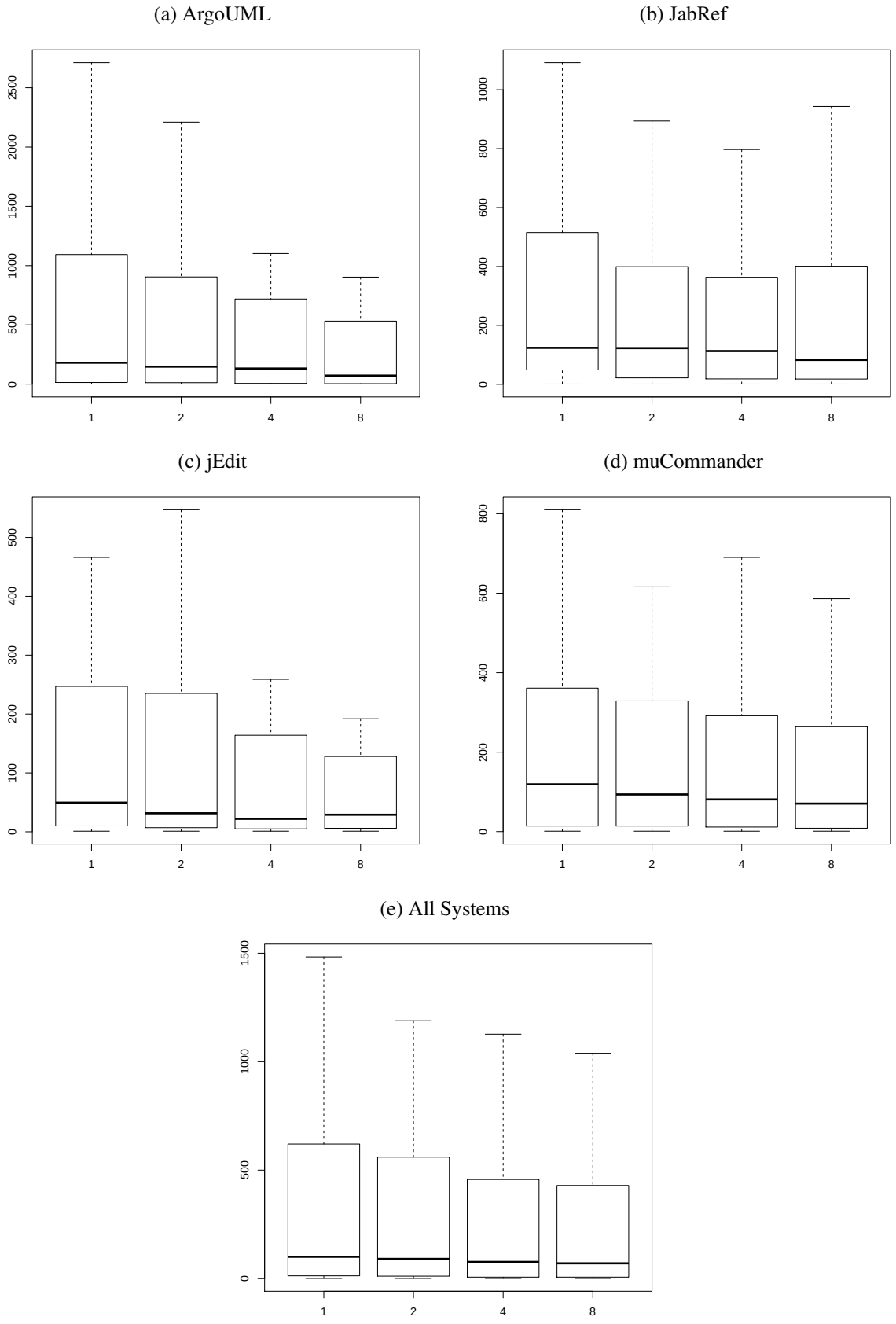


Figure 4.1: The Effectiveness Measures for Weighting Leading Comments Alone using Weighting Factors of 1,2,4, and 8

System	C(1,1,1,1,1)	C(2,1,1,1,1)	C(4,1,1,1,1)	C(8,1,1,1,1)
ArgoUML	0.0989	0.1064	0.1218	0.1410
JabRef	0.0838	0.1038	0.1101	0.1176
jEdit	0.1427	0.1513	0.1573	0.1648
muCommander	0.1027	0.1170	0.1353	0.1254

Table 4.4: MRRs Weighting Leading Comments Alone

and half are above. These results show that half of the effectiveness measures have improved with the weighting factor. Considering the results, it is not clear whether the next half are from all effectiveness measures increasing or from a small number of outliers. I calculated the percentage of effectiveness measures that improved from weighting leading comments at 4. From a weighting factor of 4 to 8, 64% of all effectiveness measures improved, 15% remained the same as they were already at 1, and 21% decreased. This results in 79% (i.e., 31 of the 39 effectiveness measures) increasing; this increases to 92% (i.e., 36 of the 39 effectiveness measures) over the unweighted configuration. In jEdit, with the exception of the 3Q and the max for a weighting factor of 2, we can see that the 1Q, median, and 3Q, go lower as the weight on the leading comment increases. The boxplots for jEdit show a continuous decrease in the spread of the effectiveness measures. The only exception to this is for a weighting factor of 2. However, we can observe a large decrease in the spread of effectiveness measures between the unweighted configuration and the maximum weighting factor. muCommander shows a decrease in each measure except for the maximum and upper 1.5 IRQ for a weighting factor of 4. From each of the individual systems, we see a general trend to a decrease in the spread of effectiveness measures as the weighting factor increases. This is supported by the collected results for all systems. It is important to note that for the collected

statistics over all systems, the maximums are set by the largest system (ArgoUML) as each of the highest effectiveness measures come from this system.

Table 4.4 reports the MRRs for each of the subject systems as the weighting factor increases. While the spread of effectiveness measures gives an idea of the percentage of variability of the effectiveness measures, the MRR allows a configuration to be rated based off of central tendency. A higher MRR indicates better effectiveness measures. Three of the software systems show an improvement in the MRR for each increase in the weighting factor. In muCommander, however, there is an observed drop in the MRR when going from a weighting factor of 4 to 8. This is due to 44% of the effectiveness measures showing a decrease in effectiveness, while 14% remain the same and only 42% improve in the change of the weighting factor. Therefore, while the spread of the effectiveness measures improves, the MRR decreases. However, it is important to note that for all systems, using a weighting factor of 8 improves over the unweighted configuration for all systems.

Method Names

I observe the configurations where the method name is raised alone. The method name was one of the two structural components considered by Bassett and Kraft. In their study, they found that the performance of a FLT continued to rise as the weighting on a method name rose. I repeated this study.

Table 4.5 presents the descriptive statistics for raising the weighting of method names alone, while Figure 4.2 presents the boxplots with outliers removed. Looking at ArgoUML, decreases can be observed in the values for the 1Q, 3Q, and maximum as the weighting factor increases. However, while the median trends to an overall decrease, there is more fluctuation as the weighting factor

Config	min	1Q	median	3Q	max
C(1,1,1,1,1)	1	24	181	1093	10231
C(1,2,1,1,1)	1	24	202	1054	9641
C(1,4,1,1,1)	1	20	114	794	9259
C(1,8,1,1,1)	1	15	125	606	8972

(a) ArgoUML

Config	min	1Q	median	3Q	max
C(1,1,1,1,1)	1	49	124	515	3940
C(1,2,1,1,1)	1	31	147	486	3769
C(1,4,1,1,1)	1	14	93	298	3661
C(1,8,1,1,1)	1	5	64	306	3721

(b) JabRef

Config	min	1Q	median	3Q	max
C(1,1,1,1,1)	1	10	49	240	6134
C(1,2,1,1,1)	1	7	41	186	5911
C(1,4,1,1,1)	1	7	31	173	6036
C(1,8,1,1,1)	1	5	31	162	5792

(c) jEdit

Config	min	1Q	median	3Q	max
C(1,1,1,1,1)	1	19	119	355	7354
C(1,2,1,1,1)	1	17	81	299	6991
C(1,4,1,1,1)	1	7	61	233	6439
C(1,8,1,1,1)	1	5	48	194	6113

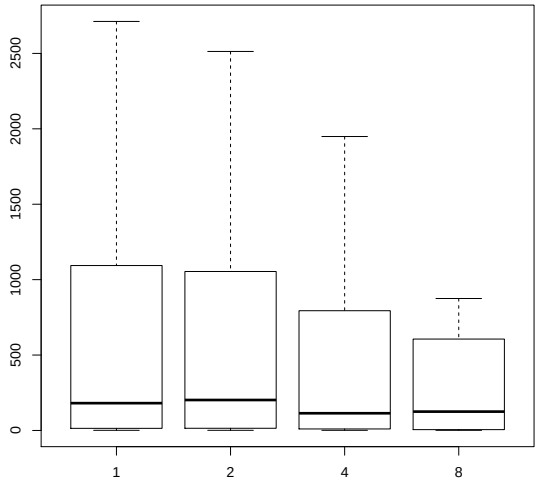
(d) muCommander

Config	min	1Q	median	3Q	max
C(1,1,1,1,1)	1	13	101	616	10231
C(1,2,1,1,1)	1	12	92	512	9641
C(1,4,1,1,1)	1	8	74	438	9441
C(1,8,1,1,1)	1	5	64	331	8972

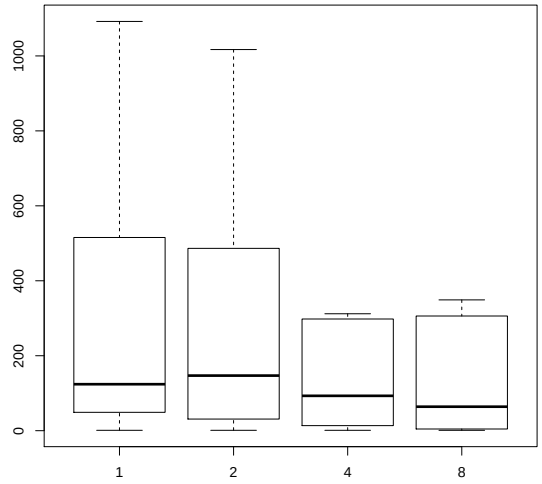
(e) All Systems

Table 4.5: Descriptive Statistics Weighting Method Names Alone

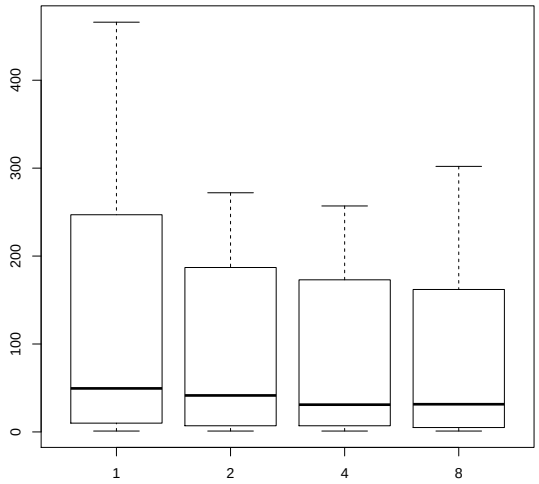
(a) ArgoUML



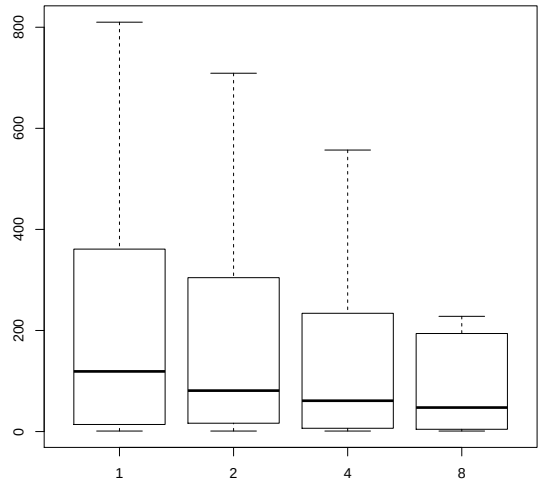
(b) JabRef



(c) jEdit



(d) muCommander



(e) All Systems

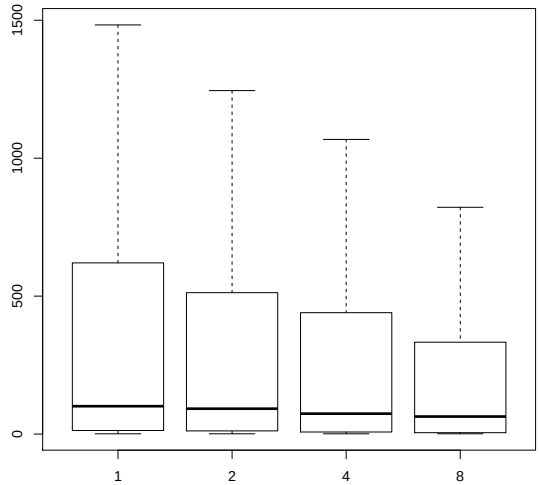


Figure 4.2: The Effectiveness Measures for Weighting Method Names Alone using Weighting Factors of 1,2,4, and 8

System	C(1,1,1,1,1)	C(1,2,1,1,1)	C(1,4,1,1,1)	C(1,8,1,1,1)
ArgoUML	0.0989	0.1119	0.1259	0.1223
JabRef	0.0838	0.1171	0.1225	0.1328
jEdit	0.1427	0.1523	0.1630	0.1628
muCommander	0.1027	0.1124	0.1258	0.1364

Table 4.6: MRRs Weighting Method Names Alone

increases. Considering the boxplots, an overall decrease can be obverved in the spread as the weighting factor increases. For JabRef, there is an overall trend for the spread to decrease as the weighting factor increases. The 1Q continues to decrease as the weighting factor increases, however, the median spikes at weighting factor 2, the 3Q spikes at weighting factor 8, and there is an observed fluctuation that tends to decrease for the max. There is a sudden decrease in the upper 1.5 IRQ at weighting factor 4.. This sudden jump may be attributable to the doubling of the weighting factors. Instead of using a linear progression of weighting factors, I double for each increase to show larger changes in the spread. In jEdit we see a consistent decrease in the 1Q, median, and 3Q as we increase the weighting factor. However, for the 1.5 IRQ and the max value, there is an increase. In muCommander, the spreads of the effectiveness measures continue to decrease as the weighting factors decrease. Similar to leading comments, for method names we can also see a general trend to a decrease in the spread of the effectiveness measures for each of the four subject systems. Again, in the collected results for all systems, we see the spreads decrease as the weighting factor increases.

Table 4.6 shows the MRRs for each of the four subject systems. Similar to the previous study, we see an increase in performance with the heaviest weighting for method names for each of the four subject systems. However, while the previous study found a slight decrease in performance

for ArgoUML at weighting factor 8, we found an increase over the unweighted configuration. However, there is a decrease in the MRR between a weighting factor of 4 and a weighting factor of 8. This decrease in the MRR is contrary to the decrease in the spread of the effectiveness measures. The explanation comes from a few exceptional effectiveness measures that showed a large increase from weighting factor 4 at weighting factor 8. The largest increase in the MRR can be observed for JabRef and muCommander. Similarly to ArgoUML, there is a decrease in performance for jEdit at weighting factor 8 from weighting factor 4.

Parameters

Parameters are expected to give complementary information to the method names. As method names are meant to describe the primary responsibility of a method, descriptive parameter names and their types are expected to describe information that is needed to accomplish the primary responsibility. I observe the configurations where the weights of parameter types and parameter names are raised alone.

Table 4.7 gives the descriptive statistics for raising the weighting of parameters alone. Figure 4.3 gives the boxplots with outliers removed. Looking at ArgoUML, we see that the median continues to increase as the weighting factor increases. For each other value, there are fluctuations but a trend to increase. For each value, giving parameters a weighting factor of 8 increases over the unweighted configuration. The smallest spread is seen at a weighting factor of 4. However, for this weighting factor the 1Q and median are higher than the previous two weighting factors. For JabRef, we see a similar situation to that of ArgoUML. However, where for ArgoUML the medians constantly increased with each new weighting factor, in JabRef there is fluctuation. JabRef also shows a slightly smaller spread in weighting factor 8 than the unweighted configuration. This can

Config	min	1Q	median	3Q	max
C(1,1,1,1,1)	1	24	181	1093	10231
C(1,1,2,1,1)	1	33	220	1156	10559
C(1,1,4,1,1)	1	30	293	905	10044
C(1,1,8,1,1)	1	29	334	1273	11354

(a) ArgoUML

Config	min	1Q	median	3Q	max
C(1,1,1,1,1)	1	49	124	515	3940
C(1,1,2,1,1)	1	44	212	567	4232
C(1,1,4,1,1)	1	51	134	405	4014
C(1,1,8,1,1)	1	42	164	506	3826

(b) JabRef

Config	min	1Q	median	3Q	max
C(1,1,1,1,1)	1	10	49	240	6134
C(1,1,2,1,1)	1	9	62	226	5609
C(1,1,4,1,1)	1	11	56	290	6817
C(1,1,8,1,1)	1	9	80	351	6648

(c) jEdit

Config	min	1Q	median	3Q	max
C(1,1,1,1,1)	1	19	119	355	7354
C(1,1,2,1,1)	1	26	139	388	7458
C(1,1,4,1,1)	1	20	126	489	7920
C(1,1,8,1,1)	1	25	127	435	7433

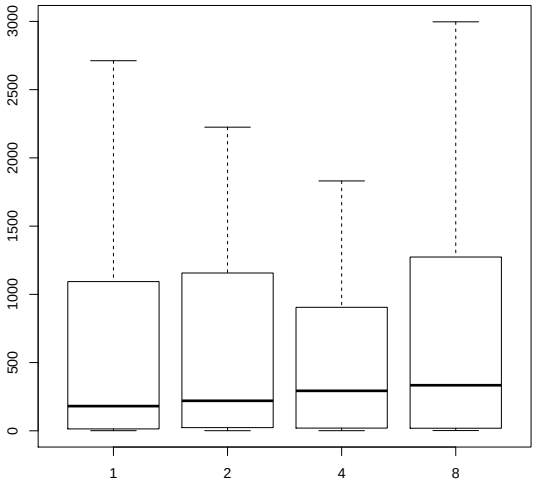
(d) muCommander

Config	min	1Q	median	3Q	max
C(1,1,1,1,1)	1	13	101	616	10231
C(1,1,2,1,1)	1	17	115	628	10559
C(1,1,4,1,1)	1	17	129	633	10044
C(1,1,8,1,1)	1	17	154	826	11354

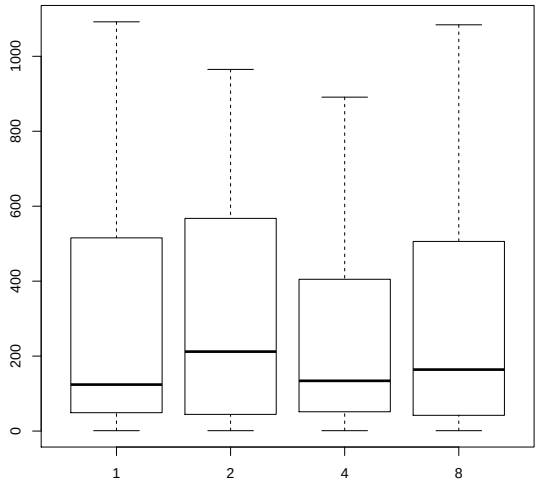
(e) All Systems

Table 4.7: Descriptive Statistics Weighting Parameters Alone

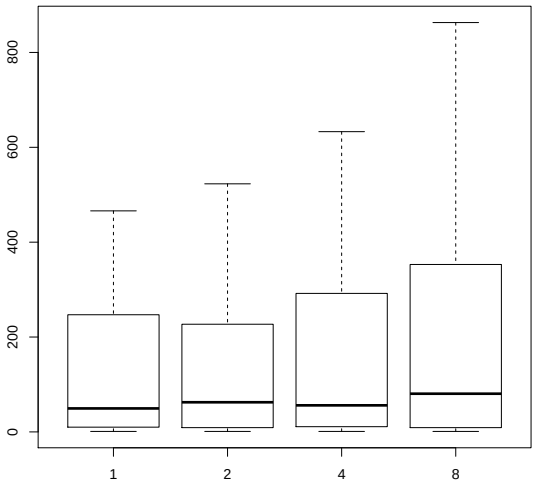
(a) ArgoUML



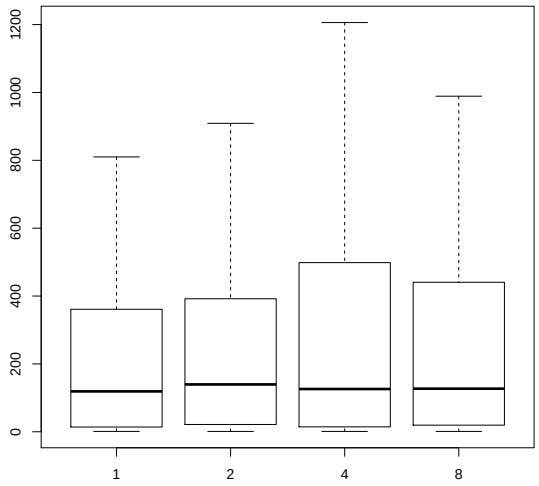
(b) JabRef



(c) jEdit



(d) muCommander



(e) All Systems

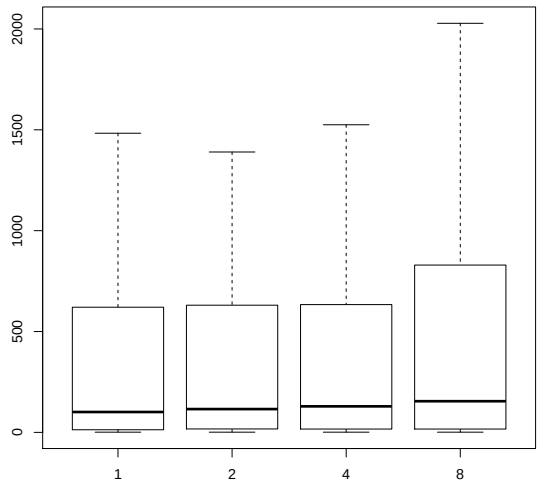


Figure 4.3: The Effectiveness Measures for Weighting Parameters Alone using Weighting Factors of 1,2,4, and 8

System	C(1,1,1,1,1)	C(1,1,2,1,1)	C(1,1,4,1,1)	C(1,1,8,1,1)
ArgoUML	0.0989	0.0767	0.0970	0.0558
JabRef	0.0838	0.0903	0.0940	0.0850
jEdit	0.1427	0.1509	0.1365	0.1384
muCommander	0.1027	0.0889	0.0940	0.0878

Table 4.8: MRRs Weighting Parameters Alone

be seen by looking at the 1Q, 3Q, and max. A weighting factor of 8 still results in a higher median value than the unweighted configuration. In the boxplot of jEdit, a gradual increase can be seen in the spread. There are some minor fluctuations for each of the values as the weighting factor increases, but the trend is for the spread to increase. As the weighting factor for jEdit increases, so does the upper 1.5 IRQ. Comparing the unweighted configuration to weighting factor 8 shows that for all values except for 1Q there has been an increase. For muCommander, the greatest spread can be observed at a weighting factor of 4. This configuration has a higher 3Q and max effectiveness measure, but a lower 1Q and median when compared to weighting factors 2 and 8. When looking at all systems, there is a typical trend for the spreads to increase as the weighting factor increases. For all systems, each of the values in the spread increase with each factor increase.

Looking at the MRRs for parameters in Table 4.8, we see that for three of the four subject systems, the MRR for a weighting factor of 8 is lower than the unweighted configuration. This is for muCommander which also has the highest MRR at a weighting factor of 4. For ArgoUML and muCommander, the highest MRR is found in the unweighted configuration. For jEdit, a weighting factor of 2 results in a slightly higher MRR over the unweighted configuration, but both the weighting factor of 4 and of 8 show a decrease when compared to the unweighted configuration.

This suggests that weighting parameters alone is not very effective and can result in a decrease in performance for some systems.

Body Comments

Similar to leading comments, body comments are expressed in the natural language of the developer. The difference between the two is sometimes thin because body comments can serve the same purpose as leading comments. Body comments can also serve the purpose of describing a difficult part of code or highlighting parts of code that need improvement (e.g., "TODO" in comment fields). It is also possible for body comments to have information pertaining to the specific bug that is being searched for.

Table 4.9 shows the descriptive statistics for raising the weighting of body comments alone. Figure 4.4 shows the boxplots with outliers removed. Unlike leading comments, method names, and parameters, weighting body comments does not show a trending increase or decrease in three of the four subject systems. In ArgoUML, comparing the unweighted to having a weighting factor of 8 shows an increase in the 1Q and the median, but a decrease in the 3Q, the max effectiveness measure, and the upper 1.5 IRQ. When looking at JabRef, we see similar levels of fluctuation. However, when comparing the unweighted configuration to having a weighting factor of 8, each of the values decrease from the unweighted configuration. Because a consistent decreasing trend is not observed, it is not apparent whether this decrease indicates that increasing the weighting on body comments is having a positive impact on the spread of effectiveness measures for this system. Interestingly, in jEdit we see a spike in the spread from the unweighted configuration to the configuration with a weighting factor of 2. After this spike, there is a decline in the spread for each subsequent weighting factor. By weighting factor 8, each of the values has decreased below

Config	min	1Q	median	3Q	max
C(1,1,1,1,1)	1	24	181	1093	10231
C(1,1,1,2,1)	1	26	199	1019	9981
C(1,1,1,4,1)	1	31	280	1093	9761
C(1,1,1,8,1)	1	29	197	1043	9956

(a) ArgoUML

Config	min	1Q	median	3Q	max
C(1,1,1,1,1)	1	49	124	515	3940
C(1,1,1,2,1)	1	52	137	522	3578
C(1,1,1,4,1)	1	58	143	485	3870
C(1,1,1,8,1)	1	48	120	475	3806

(b) JabRef

Config	min	1Q	median	3Q	max
C(1,1,1,1,1)	1	10	49	240	6134
C(1,1,1,2,1)	1	11	40	291	6033
C(1,1,1,4,1)	1	7	37	258	5984
C(1,1,1,8,1)	1	9	37	228	5724

(c) jEdit

Config	min	1Q	median	3Q	max
C(1,1,1,1,1)	1	19	119	355	7354
C(1,1,1,2,1)	1	25	90	298	7229
C(1,1,1,4,1)	1	23	68	293	7111
C(1,1,1,8,1)	1	20	75	337	6834

(d) muCommander

Config	min	1Q	median	3Q	max
C(1,1,1,1,1)	1	13	101	616	10231
C(1,1,1,2,1)	1	17	98	618	9981
C(1,1,1,4,1)	1	17	102	621	9761
C(1,1,1,8,1)	1	15	89	658	9956

(e) All Systems

Table 4.9: Descriptive Statistics Weighting Body Comments Alone

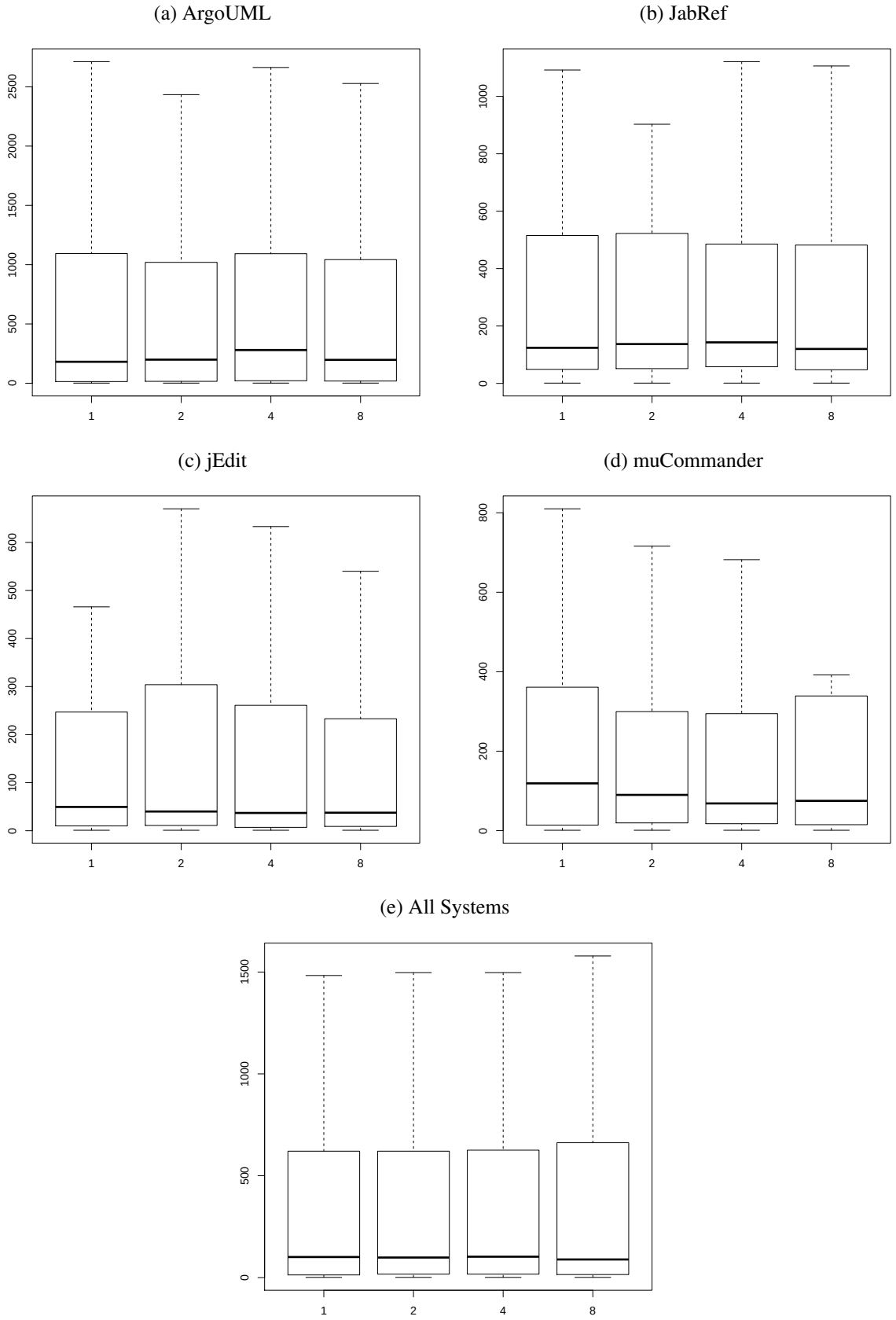


Figure 4.4: The Effectiveness Measures for Weighting Body Comments Alone using Weighting Factors of 1,2,4, and 8

System	C(1,1,1,1,1)	C(1,1,1,2,1)	C(1,1,1,4,1)	C(1,1,1,8,1)
ArgoUML	0.0989	0.1141	0.0850	0.1080
JabRef	0.0838	0.0971	0.0869	0.0936
jEdit	0.1427	0.1395	0.1386	0.1415
muCommander	0.1027	0.0962	0.1184	0.1086

Table 4.10: MRRs Weighting Body Comments Alone

the unweighted configuration with the exception of the upper 1.5 IRQ. In muCommander, we see a general trend for the spreads to decrease, with a weighting factor of 8 having all but the 1Q lower than the unweighted configuration. At a weighting factor of 8, we see an interesting drop in the upper 1.5 IRQ. Important to note, however, is that for each value other than the upper 1.5 IRQ, there has been an increase from a weighting factor of 4 where there was a decrease from weighting factors 2 to 4. Overall for all systems, it appears that weighting body comments may not make a substantial difference in the spread of effectiveness measures. In fact, for all systems combined we see that comparing the unweighted configuration to a weighting factor of 8 actually results in a worse measure for the 1Q, 3Q, and upper 1.5 IRQ.

Table 4.10 shows the MRRs for the four subject systems when weighting body comments alone. For three of the four systems, we see a slight increase in the MRR for that system. However, for each of these three systems, the highest MRR is seen at either a weighting factor of 2 or a weighting factor of 4. This may indicate that body comments can be of some limited benefit to the topic models, but placing too great of an emphasis will limit the knowledge gained from terms in other lexicons including important terms that may be contained in the leading comments or method names. For jEdit, there was a decrease in the MRR from the unweighted configuration. The lowest MRR for jEdit is seen at a weighting factor of 4, and the highest MRR is seen for the unweighted

configuration. Overall, the changes to the MRR are small and seem to only change slightly whether body comments are weighted or not.

Local Variables

Local variables are identifiers that are found only within the scope of a method. Local variables store data that is required for a method to perform their main operation. Due to local variables being tied specifically to the method where they are created, they may contain information that helps to understand the main responsibility of the method.

Table 4.11 gives the descriptive statistics for raising the weighting of body comments alone. Figure 4.5 gives the boxplots with outliers removed. In ArgoUML, there are fluctuations in the spread as the weighting factor increases. When comparing the unweighted configuration to having a weighting factor of 8, there is an increase in each of the values except for the upper 1.5 IRQ. In JabRef, the spread decreases for when a weighting factor of 2 is given. However, for each subsequent increase in the weighting factor there is an increase in the spread. The greatest spread is seen when local variables are given a weighting factor of 8. Both jEdit and muCommander show similar situations as the weighting factor is increased. There are slight fluctuations as the weighting factor increases. However, the trend is for the spread to increase as the weighting factor increases. For both systems, the spread is the greatest when a weighting factor of 8 is given. Overall for the four subject systems, weighting local variables is seen to have a negative effect on the spread of the effectiveness measures.

Table 4.12 shows that for each of the four subject systems, giving local variables a weighting factor of 8 decreases the performance over the unweighted configuration. In ArgoUML, there is a continuous decline in the MRR for each increase in the weighting factor resulting in a weighting factor of 8 having the lowest MRR. This is the same for both jEdit and for muCommander.

Config	min	1Q	median	3Q	max
C(1,1,1,1,1)	1	24	181	1093	10231
C(1,1,1,1,2)	1	25	266	1141	10040
C(1,1,1,1,4)	1	24	300	1127	10481
C(1,1,1,1,8)	1	26	345	1210	10681

(a) ArgoUML

Config	min	1Q	median	3Q	max
C(1,1,1,1,1)	1	49	124	515	3940
C(1,1,1,1,2)	1	43	127	415	3720
C(1,1,1,1,4)	1	69	226	484	3867
C(1,1,1,1,8)	1	57	253	605	4101

(b) JabRef

Config	min	1Q	median	3Q	max
C(1,1,1,1,1)	1	10	49	240	6134
C(1,1,1,1,2)	1	8	71	336	6492
C(1,1,1,1,4)	1	9	100	285	6339
C(1,1,1,1,8)	1	9	111	395	6192

(c) jEdit

Config	min	1Q	median	3Q	max
C(1,1,1,1,1)	1	19	119	355	7354
C(1,1,1,1,2)	1	18	136	487	7792
C(1,1,1,1,4)	1	22	161	442	7281
C(1,1,1,1,8)	1	24	218	512	8041

(d) muCommander

Config	min	1Q	median	3Q	max
C(1,1,1,1,1)	1	13	101	616	10231
C(1,1,1,1,2)	1	13	138	682	10040
C(1,1,1,1,4)	1	17	184	777	10481
C(1,1,1,1,8)	1	16	221	785	10681

(e) All Systems

Table 4.11: Descriptive Statistics Weighting Local Variables Alone

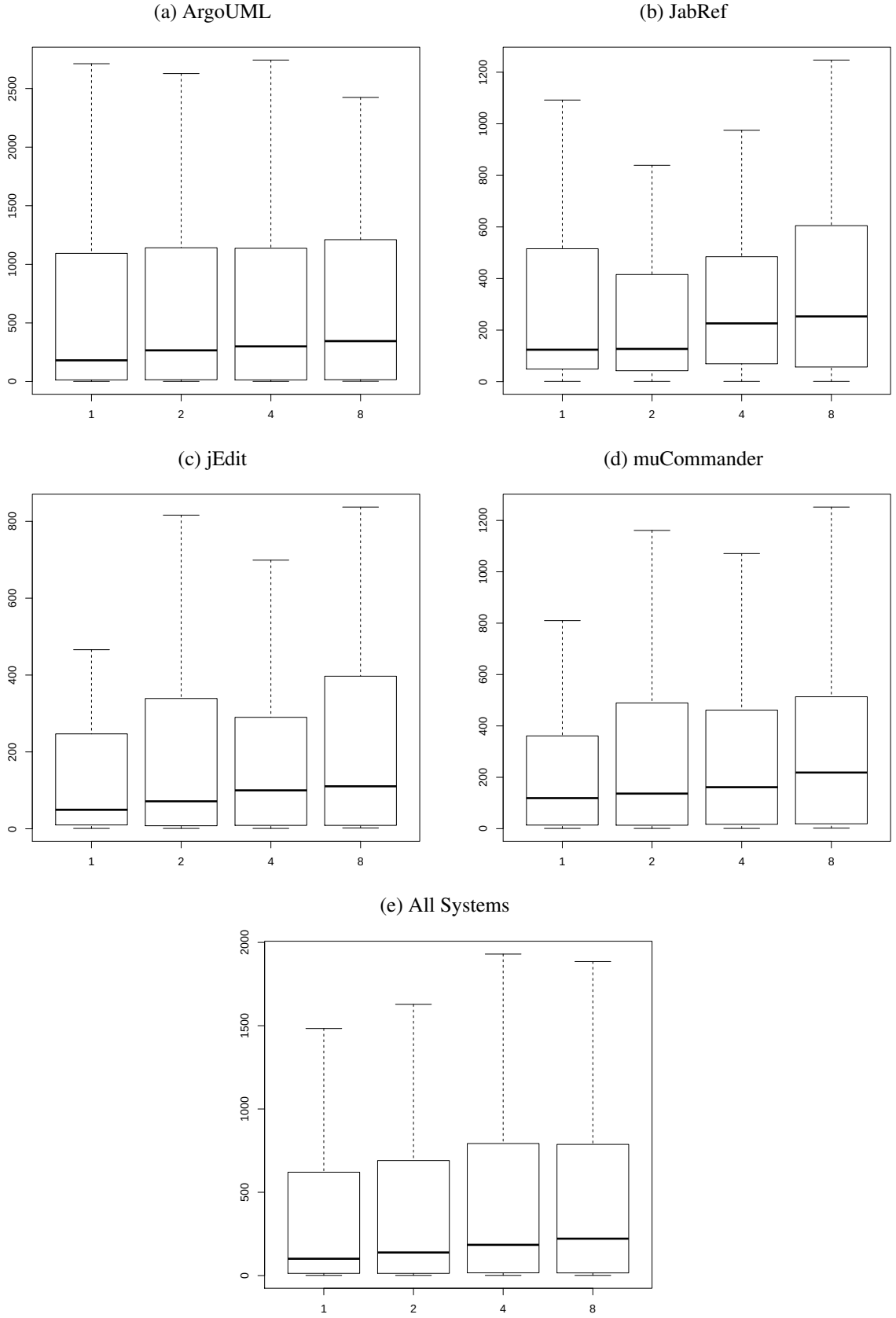


Figure 4.5: The Effectiveness Measures for Weighting Local Variables Alone using Weighting Factors of 1,2,4, and 8

System	C(1,1,1,1,1)	C(1,1,1,1,2)	C(1,1,1,1,4)	C(1,1,1,1,8)
ArgoUML	0.0989	0.0979	0.0848	0.0548
JabRef	0.0838	0.0995	0.0895	0.0772
jEdit	0.1427	0.1397	0.1215	0.0972
muCommander	0.1027	0.0935	0.0920	0.0667

Table 4.12: MRRs Weighting Local Variables Alone

However, the situation is different for JabRef, where the highest MRR is given for a weighting factor of 2. After this, there is a decline in the MRR from a weighting factor of 2 to a weighting factor of 4, and then again when the weighting factor is increased from 4 to 8. Overall, weighting local variables seems to have a negative impact on the performance of the technique when given high weighting factors.

4.2.1.2 What are the top configurations for each system and across all systems?

The previous question looked at how weighting the individual components by themselves might help to improve or decrease the performance of an LDA-based FLT. Weighting a single lexicon over others can give some insight into how important each lexicon is in relation to the others. In the results of the previous question, leading comments and method names generally produced an increase in performance, while parameters and local variables tended to produce a decrease in performance, and weighting body comments tended to show little change overall.

While looking at how weighting each lexicon independently can be useful, it does not give any insights into the best weighting configuration for a system. In this question, I address the best configurations for each subject system. Due to the large number of configurations involved in this experiment, for the purposes of feasibility in discussion, I limit this question to the top ten

configurations for each subject system according to MRR. I include the unweighted configurations for comparison.

Table 4.13a gives the top configurations for ArgoUML. In these results, we see that each of the top configurations has the maximum weighting factor of 8 for the leading comments. This follows from the previous question which showed that ArgoUML had the highest MRR when giving high weights to leading comments. The same can be said for method names. When looking at the top configurations, method names were given the highest weighting factor for seven of the ten configurations. Parameters, however, were only weighted in three of the top ten configurations, and was never given the highest weighting factor for any of the configurations. Weighting body comments is shown to have some effect. However, it is not consistent across the top ten configurations what weighting factor should be given to body comments. Only one of the top ten configurations left body comments as unweighted. The most common weighting factor given to body comments is a weighting factor of 8, and this is for four of the top ten configurations. This might indicate that body comments contain some relevant information that should be accounted. The most common weighting for local variables in ArgoUML was unweighted. Three other configurations gave local variables a weighting factor of 2, and two gave local variables a weighting factor of 4. Only one configuration of ten gave local variables the top weighting factor of 8. These results are consistent with the previous question.

Table 4.13b illustrates the top ten configurations for JabRef. When looking at the leading comments, there is a split in the number of configurations that use a weighting factor of 8 and those that use a weighting factor of 4. High weighting factors for JabRef give the best results. However, there is more variation on the weighting factor for JabRef than for ArgoUML. This may indicate a difference in how leading comments are used between the two systems. Interestingly,

there may be more importance on method names in JabRef than for ArgoUML. For each of the ten configurations, the highest weighting factor of 8 is used for method names. For three of the top ten configurations, there is a weighting factor of 8 given to parameters. However, each other configuration uses a weighting factor of 1 or 2. In most configurations, using a lower weighting factor for parameters leads to a higher MRR. In three of the top ten configurations, body comments are left unweighted. For five of the ten configurations, a weighting factor of 4 or 8 is used for body comments. Similar to ArgoUML, there may be some information in body comments that should be considered when choosing a weighting configuration. Nine of the ten configurations use the unweighted configuration or a weighting factor of 2.

Table 4.13c shows the top ten configurations for jEdit. As in ArgoUML, jEdit used a weighting factor of 8 for leading comments in each of the top configurations. For method names, there was an even split between using a weighting factor of 4 and a weighting factor of 8. The most common weighting factor for parameters was 4 for JabRef; this is in contrast to the results of weighting the parameters alone which showed a drop in performance when using a weighting factor of 4 or 8 for parameters. jEdit was the only subject system that showed a lower MRR for a weighting factor of 8 for body comments than the unweighted configuration. However, when considering the top configurations, a weighting factor of 4 or 8 is seen to be used in half of the top configurations. It is interesting to note that only two of the configurations used the unweighted configuration. This is less than the number of configurations in either JabRef or muCommander, despite both systems having showed an increase over the unweighted configuration when using a weighting configuration for body comments. In nine of the ten configurations, the unweighted configuration or a weighting factor of 2 are used for local variables.

The top ten configurations for muCommander are given in Table 4.13d. Similar to JabRef,

leading comments were given either a weighting factor of 4 or 8 in each of the top configurations and method names were given a weighting factor of 8 in all top configurations. Half of the top configurations for muCommander use a weighting configuration of 4 or 8. Only two of the configurations used the unweighted configuration for parameters. This contrasts to the results for weighting parameters alone, which showed that the unweighted configuration had the highest MRR. The most common weighting factor for body comments in muCommander is to use the unweighted configuration. Only three of the ten weighting configurations are body comments given a weighting factor of 4 or 8. This is the lowest number for any of the subject systems. The weighting factors for local variables are also slightly different in muCommander with a weighting factor of 4 or 8 being used in five of the ten configurations. However, the unweighted configuration is used for the top three configurations.

The top configurations for all systems combined are shown in Table 4.13e. For each of the ten configurations, leading comments and method names are given a weighting factor of 8. This would seem to indicate that across all systems, leading comments and method names have a positive impact on the performance of the LDA based FLT. For each of the other three lexicons, the increase in performance is not as high. For parameters, nine of ten configurations used a weighting factor of 1 or 2, and only one configuration used a weighting factor of 4. In the case of body comments, seven of the ten configurations used a weighting factor of 1 or 2. Two of the remaining configurations used a weighting factor of 4 and one used a weighting factor of 8. The most common weighting factor was a weighting factor of 2. Local variables had a weighting factor of 1 or 2 for seven of the ten configurations, but the most common weighting factor was unweighted. Of the three configurations that weighted local variables higher, each of them used a weighting factor of 4. No weighting configuration used a weighting factor of 8 for local variables. However, it is

important to note that for the top configuration, each of the lexicons was given a weighting factor of at least 2, with body comments receiving a weighting factor of 4. This may indicate that overall each of the lexicons provide some information that is important to the topic model.

Figure 4.6 shows the boxplots and spreads for the top ten configurations and the unweighted configuration for each of the subject systems and for all systems combined. From the boxplots, it can be seen that each of the top configurations produces a greatly reduced spread when compared with the unweighted configuration. ArgoUML shows the most substantial changes between the unweighted configuration and the top configurations. For each configuration of this system, the upper 1.5 IRQ has been reduced to be lower than the 3Q of the unweighted configuration. When looking at all the systems combined, we see a similar situation where each of the ten top configurations have a substantially reduced spread when compared to the unweighted configuration.

A Friedman test for each of the four systems was conducted to determine if this reduction meant changing the structural term weighting scheme has a significant effect on the results of the LDA-based FLT for a system. The Friedman test is the non-parametric analog of the repeated measures ANOVA. In addition, I conducted a Friedman test for the combined data sets of all four systems over all 372 query results. The results of the tests are shown in Table 4.14. To maintain feasibility of this study, I limited this analysis to only the top ten configurations. This reduces the number of post-hoc tests required for a system from over 500,000 to a more manageable 55.

I conducted post-hoc Wilcoxon signed-rank tests with Holm correction to find configuration pairs with statistically significant differences ($p < 0.01$). From the results, it was found that all top configurations were significantly different from the unweighted configuration. However, none of the top configurations were found to be significantly different.

System	Config	MRR
ArgoUML	$C(8,8,1,8,4)$	0.1559
	$C(8,8,1,2,2)$	0.1559
	$C(8,8,4,8,2)$	0.1556
	$C(8,4,1,4,1)$	0.1556
	$C(8,8,1,1,1)$	0.1546
	$C(8,8,2,8,1)$	0.1546
	$C(8,8,4,4,4)$	0.1546
	$C(8,4,1,8,8)$	0.1545
	$C(8,8,1,2,1)$	0.1545
	$C(8,4,1,4,2)$	0.1495
	$C(1,1,1,1,1)$	0.0989

(a) ArgoUML

System	Config	MRR
JabRef	$C(8,8,8,8,2)$	0.1657
	$C(4,8,1,1,2)$	0.1656
	$C(8,8,8,8,1)$	0.1656
	$C(4,8,2,4,1)$	0.1649
	$C(8,8,1,4,1)$	0.1649
	$C(8,8,2,4,2)$	0.1646
	$C(4,8,1,2,2)$	0.1597
	$C(4,8,8,2,2)$	0.1596
	$C(8,8,2,1,1)$	0.1590
	$C(4,8,2,1,4)$	0.1586
	$C(1,1,1,1,1)$	0.0838

(b) JabRef

Table 4.13: Top ten configurations and unweighted configuration for each subject system and combined

System	Config	MRR
jEdit	$C(8,4,4,1,2)$	0.1869
	$C(8,4,4,4,2)$	0.1859
	$C(8,8,4,2,1)$	0.1859
	$C(8,4,2,2,1)$	0.1848
	$C(8,8,2,1,2)$	0.1840
	$C(8,8,2,8,2)$	0.1828
	$C(8,8,4,8,1)$	0.1827
	$C(8,8,4,2,2)$	0.1798
	$C(8,4,1,8,2)$	0.1788
	$C(8,4,1,4,4)$	0.1767
	$C(1,1,1,1,1)$	0.1427

(c) jEdit

System	Config	MRR
muCom	$C(8,8,1,2,1)$	0.1669
	$C(4,8,2,2,1)$	0.1659
	$C(4,8,8,1,1)$	0.1659
	$C(8,8,4,4,2)$	0.1558
	$C(8,8,4,8,4)$	0.1550
	$C(4,8,2,1,4)$	0.1549
	$C(8,8,1,8,8)$	0.1549
	$C(8,8,8,1,1)$	0.1548
	$C(4,8,4,1,2)$	0.1548
	$C(8,8,2,2,4)$	0.1547
	$C(1,1,1,1,1)$	0.1027

(d) muCommander

Table 4.13: Top ten configurations and unweighted configuration for each subject system and combined

System	Config	MRR
All	$C(8,8,2,4,2)$	0.1664
	$C(8,8,2,1,2)$	0.1658
	$C(8,8,2,2,4)$	0.1658
	$C(8,8,4,1,1)$	0.1647
	$C(8,8,2,4,1)$	0.1646
	$C(8,8,1,2,2)$	0.1644
	$C(8,8,2,1,1)$	0.1634
	$C(8,8,1,8,4)$	0.1633
	$C(8,8,1,2,4)$	0.1633
	$C(8,8,1,2,1)$	0.1629
	$C(1,1,1,1,1)$	0.1159

(e) All Systems

Table 4.13: Top ten configurations and unweighted configuration for each subject system and combined

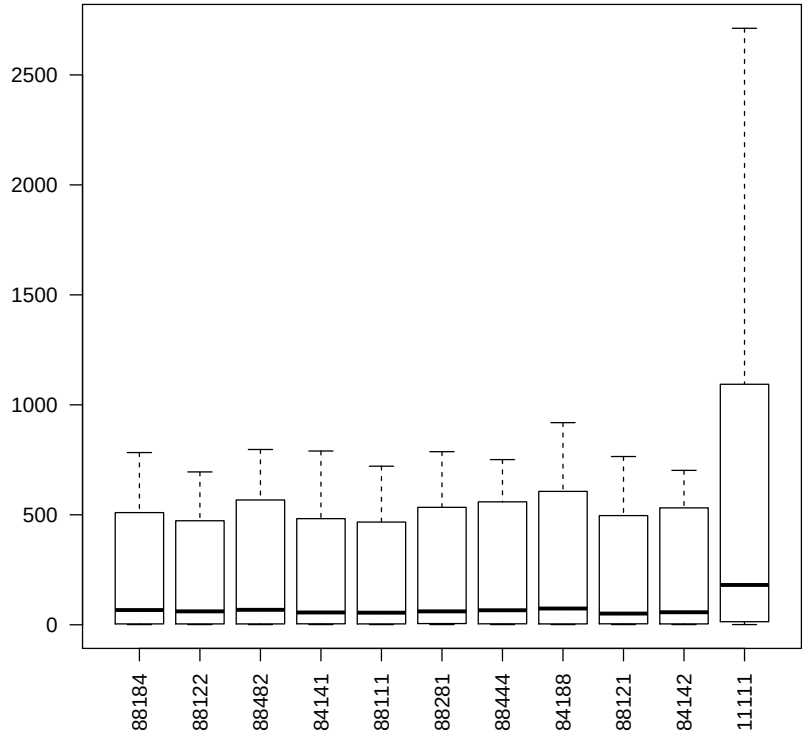
System	χ^2	df	p-value
ArgoUML	128.792	10	< 0.01
JabRef	89.538	10	< 0.01
jEdit	102.321	10	< 0.01
muCommander	107.306	10	< 0.01
All	270.162	10	< 0.01

Table 4.14: Friedman Test Results

4.2.1.3 What are the main effects and interactions between the structural lexicons?

To answer this question, I performed a factorial analysis. Table 4.15 lists the results of a factorial ANOVA applied to the effectiveness measures for the 1,024 configurations for each system and for all 372 features combined. Only the statistically significant main effects and interaction effects are listed. This analysis is conducted to find all main effects, two-way effects, three-way effects, and four-way effects. For each of the four subject systems, there were significant main

(a) ArgoUML



(b) JabRef

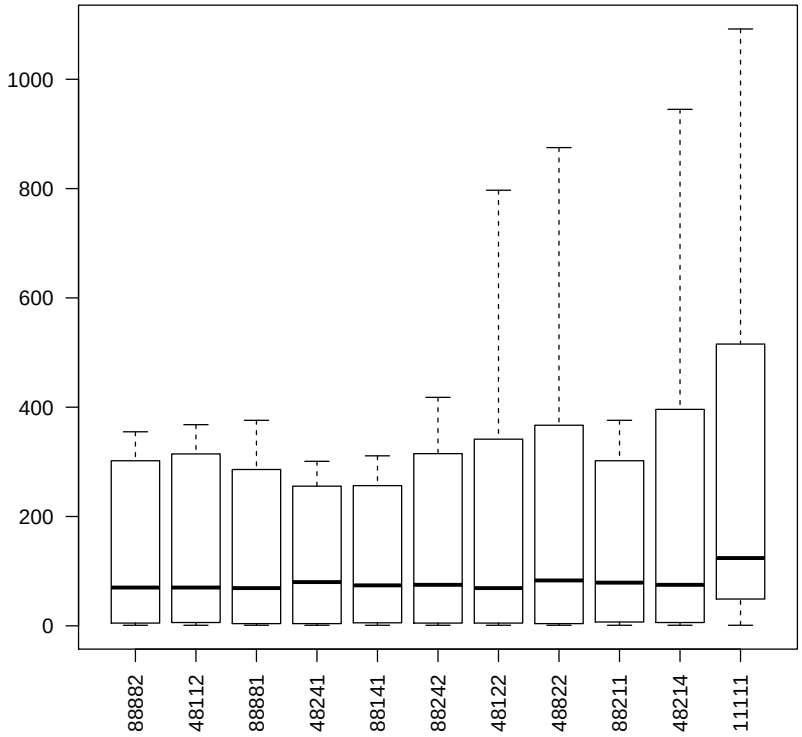
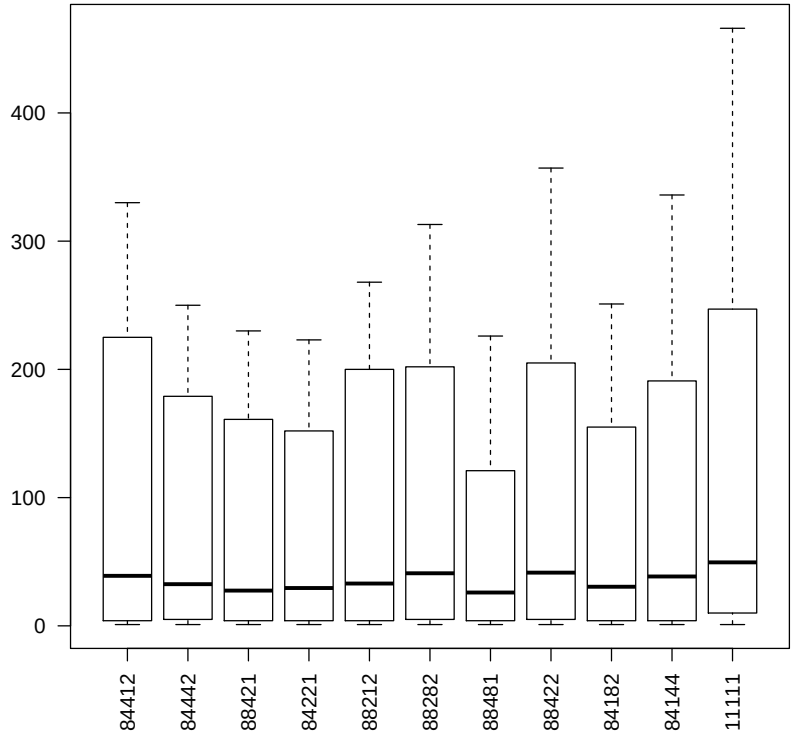


Figure 4.6: The Top Configurations and unweighted configuration for Each System and All Systems Combined

(c) jEdit



(d) muCommander

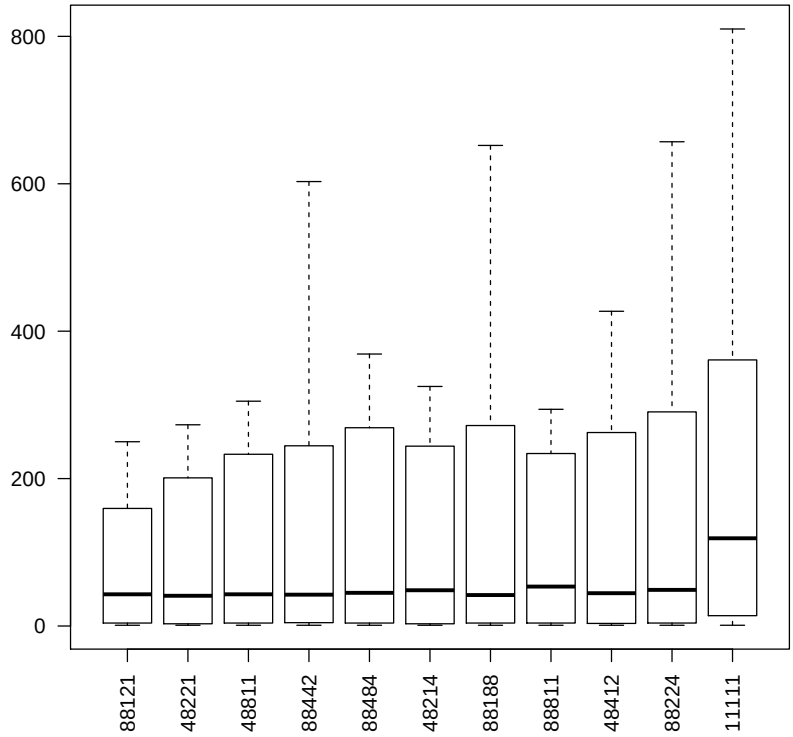


Figure 4.6: The Top Configurations and unweighted configuration for Each System and All Systems Combined

(e) All Systems

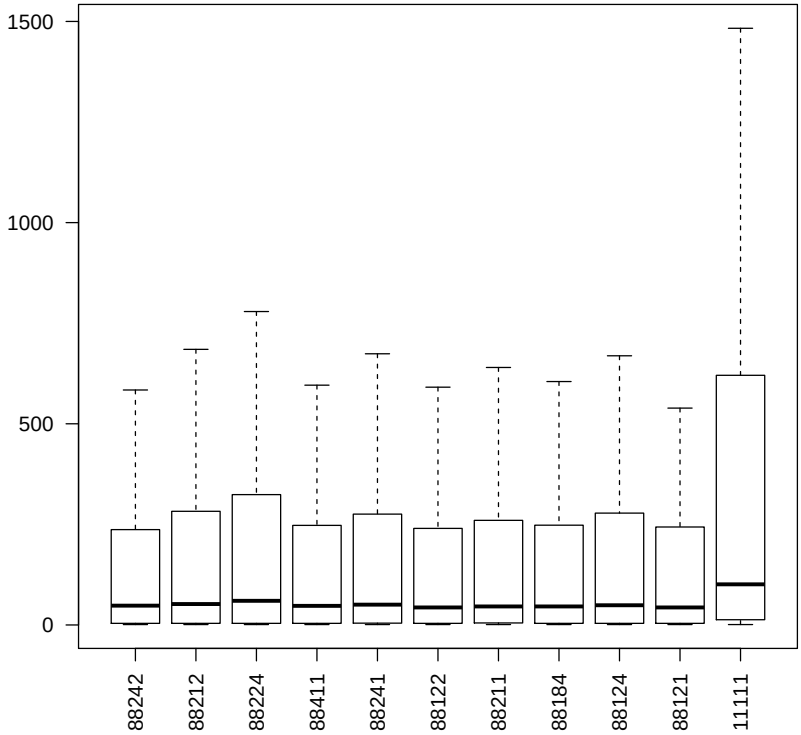


Figure 4.6: The Top Configurations and unweighted configuration for Each System and All Systems Combined

effects from the leading comments, method names, and local variables. This is supported by the results of *Research Question 1(a)* and *Research Question 1(b)*. Parameters were a significant main effect for ArgoUML. Body comments were not shown to have a statistically significant effect on any of the systems. For all systems combined, a significant main effect was seen from all but the body comments. For three of the four subject systems, there was an interaction between leading comments and method names. This interaction was also seen for all systems combined. For JabRef, there were no significant interactions found. For jEdit, there was an additional significant interaction found between the leading comments and the local variables. There were no significant three-way, four-way, or five-way interactions found.

Due to the interactions and effects found from leading comments and method names, I looked at the configurations that gave the maximum weighting factor to both lexicons. The MRRs

System	Variable	Df	F value	Pr(>F)
ArgoUML	LC	3	223.36	< 2.2e-16
	MN	3	37.49	< 2.2e-16
	P	3	6.27	< 0.01
	LV	3	6.09	< 0.01
	LC:MN	9	2.55	< 0.01

(a) ArgoUML

System	Variable	Df	F value	Pr(>F)
JabRef	LC	3	11.96	< 0.01
	MN	3	80.77	< 2.2e-16
	LV	3	14.89	< 0.01

(b) JabRef

System	Variable	Df	F value	Pr(>F)
jEdit	LC	3	154.64	< 2.2e-16
	MN	3	37.59	< 2.2e-16
	LV	3	9.94	< 0.01
	LC:MN	9	6.01	< 0.01
	LC:LV	9	4.57	< 0.01

(c) jEdit

System	Variable	Df	F value	Pr(>F)
muCommander	LC	3	33.55	< 0.01
	MN	3	181.78	< 2.2e-16
	LV	3	15.68	< 0.01
	LC:MN	9	3.16	< 0.01

(d) muCommander

Table 4.15: Main Effects and Interactions for each subject system and combined

System	Variable	Df	F value	Pr(>F)
All	LC	3	382.79	< 2.2e-16
	MN	3	219.37	< 2.2e-16
	P	3	8.86	< 0.01
	LV	3	34.09	< 2.2e-16
	LC:MN	9	10.32	< 0.01

(e) All Systems

Table 4.15: Main Effects and Interactions for each subject system and combined

System	C(8,8,1,1,1)	p-value
ArgoUML	0.1846	< 0.01
JabRef	0.1596	< 0.01
jEdit	0.1812	< 0.01
muCommander	0.1706	< 0.01

Table 4.16: MRRs For Maximizing Weighting On Leading Comments and Method Names

for these configurations are shown in Table 4.16 along with the results of a Wilcoxon signed-rank test (no directionality assumed) between the configuration and the unweighted configuration for each of these systems. The results show a significant difference between these configurations and the unweighted configuration. This shows that for each of the subject systems, using a weighting factor of 8 on leading comments and method names can lead to a statistically significant improvement over the unweighted configuration.

4.2.2 Can a relationship between the contributions of each structural component's lexicon and their weighting factors be found?

The purpose of this question is to understand qualitatively how each of the lexicons might contribute to the overall topic model. For the purposes of answering this question, I collected several measures from the terms in the source code.

Table 4.17 lists multiple counts collected by looking over the corpus for each system. Each of these counts may defined as follows:

- **Terms** - The number of terms without counting duplicates
- **Uses** - The number of terms with duplicates (i.e., the number of instances of the terms)
- **Docs** - The number of documents with terms from that lexicon (e.g., the number of methods with leading comments)

Each of these values give important information about the composition and prevalence of the lexicons in the systems.

In addition to these counts, I calculated the system-wide lexicon density (LD), unique term density (UTD), and unique term contribution (UTC) for each of the systems. Since these values give a high-level look at how each lexicon fits into the system, and do not take into account the fact that all methods do not contain a lexicon or help to understand how each lexicon contributes to the methods for which it is present, I also calculated summary statistics for the three values at the method level. The method level statistics are calculated for the LD, UD, and UTC for which they are present. It is important to note when considering the UTC that these lexicons do not make up the total of all terms in the corpus. There are other lexicons that were not considered in this study.

ArgoUML

For ArgoUML, the two most common lexicons that appear in the docs of the system are the leading comments and the method names. For leading comments, the ratio of terms to uses is approximately 1:24. This indicates that the terms in the leading comments are often reused throughout the system. In Table 4.18a, the system-wide density measures can be observed for

ArgoUML. We see that leading comments contain 25% of the terms in the system. While the number of unique terms in the leading comments lexicon is low, the lexicon contributes over 22% of the unique terms in the system. Table 4.19 contains the method-level density measures for leading comments. For methods where leading comments are present, they contain 41% of the terms in the method. Interesting to note is that at the method level UTD is 50%. This indicates that there are terms that are common in many of the leading comments throughout the system. An example of such a term is "constructor," which is used to indicate the method that represents the constructor for a class. For a method, leading comments contribute 50% of the unique terms in the method. The prevalence of leading comments in the system and their contribution to the unique terms may be an indicator of why increasing the weighting factors for leading comments improves the results of the FLT.

Method names are in the largest number of docs with a ratio of terms to uses for method names is lower with a ratio of 1:6. Method names have a lower LD and UTC than leading comments. However, they still contribute the second most unique terms at 9% and contain the second most terms in the systems at 7%. These lower values may be indicative of the reduced size of method names which normally only consist of a small number of terms for each method. When looking at the method level densities in Table 4.20, it is observed that they contain 15% of the terms in the method and contribute 11% of the unique terms. These values vary widely, however, with standard deviations of 14% and 19%, respectively. This means that a method name may contribute between 0% to 30% of the unique terms in the method, and contain between 0.5% and 28% of the terms. Looking at the UTD, approximately 27% of the terms in a method name are unique to that lexicon and this can vary from 0% to 58%. In the cases that method names contain only a small

LD, but have a higher UTD or UTC, it could be beneficial to raise the weighting factor and this may be the cause of the importance of the method names in the top configurations.

Parameters have a lower usage, but still appear in 55% of all docs with a ratio of 1:13 for terms to their uses. Parameters contain 3% of the terms in the system, and contribute less that 1% of the unique terms in the system. This is not a fair value, however, as parameters are typically used in the body of the method lowering their uniqueness. This has been accounted for in the method level densities by ignoring the parameters' usage. The method level densities are available in Table 4.21. We see that when parameters are present they contain 8% of the terms in the method. Despite the removal of a parameter's usage, they only have a UTD of 6% and contribute only 2% of the unique terms in the system. This means that for methods where parameters are present, other lexicons contribute the same terms that appear in the parameters. This low composition and contribution of unique terms may be the reason why parameters are not valued highly in the top configurations.

Body comments appear the least out of the five lexicons. Body comments only appear in 13% of all docs in the system with a ratio of 1:7. However, despite being in less than a third of the number of docs of parameters, body comments contribute 4% of the unique terms in the system. They also contain the second highest UTD of any lexicon system wide. Table 4.22 shows the method-level densities for body comments. In this table, we can see that for methods where body comments are present, there is a UTD of 56% and a UTC of 12%. Both of these values are higher than the values for method names. However, body comments only contain 8% of the terms in the methods where they appear. The low number of docs for body comments may be a contributing factor as to why body comments do not have a greater effect on the top configurations.

Local variables appear the second least in the total number of docs for the system at 29%

and a ratio of 1:5 between terms and their uses. Local variables contain less than 3% of the terms in the system and contribute the least amount of unique terms out of all lexicons. However, local variables contain the highest composition of unique terms in their lexicon. Looking at the method level densities in Table 4.23 we see that local variables contribute the least amount of unique terms out of all lexicons at less than 1%. They also contain the least amount of terms in the method. This low UTC and overall contribution may be an indicator of why raising the weighting factors for local variables results in poor performance.

JabRef

JabRef shows a lower percentage of docs containing leading comments. For the system, only 28% of the docs contain leading comments with a ratio of 1:10 between terms and their usages. This drop in percentage and in ratio is due to the change in usage between the two systems of leading comments. In ArgoUML, leading comments were used for most methods as quick descriptors of the purpose of the method within the class. Leading comments in ArgoUML identified methods as constructors, helpers, accessors, and so forth in addition to other details. In JabRef, leading comments are used more sparingly and may be clarifications for certain methods or tasks. Looking at the system-wide metrics show that leading comments contain 9% of the terms in the system. This is higher than method names which appear in more docs. Furthermore, while only having a UTD of 2%, leading comments contribute 11% of the unique terms in the system, higher than any other lexicon. The method-level densities show that leading terms have a UTD of 47% and contain 31% of the method's terms and contributes 38% of the unique terms in the methods. While these values are lower than they are for ArgoUML, these are still high values when compared to the other lexicons. The drop in the percentage of docs and the method level densities may

explain why some of the top configurations for JabRef give leading comments a weighting factor of 4 instead of 8.

Method names are present in the most docs of all lexicons. There is a ratio of 1:5 for terms to their uses. While being present in the largest number of docs, method names only contain 5% of the terms in the system. This is due to a low number of terms in each doc for a method name. Of the terms in the method names across the system, over 2.5% are unique to that lexicon. This is the highest percentage of all lexicons in the study. In addition to this, method names contribute over 8% of the unique terms in the system. This is the second most, falling behind leading comments. Looking at the method-level densities, method names are composed of 55% unique terms. This is an increase over ArgoUML. In addition, method names contain on average 21% of the terms in the method and contribute 24% of the unique terms to the method. Each of these values are higher than ArgoUML, and may be a reason why JabRef used a weighting factor of 8 in all top configurations.

Similar to ArgoUML, parameters in JabRef appear in 55% of all docs, and have a close ratio of 1:11. Parameters in JabRef have a slightly higher LD at 4% of the terms of the system. Again, looking at the system-wide density measures for UTD and UTC are not beneficial. Instead, we look at the method level densities. The method level densities for JabRef are higher than the values for ArgoUML. In JabRef, 28% of the parameters are composed of unique terms. Furthermore, parameters contain 17% of the method's terms and contribute over 11% of the unique terms in the methods. For both ArgoUML and JabRef, there is a large standard deviation. However, the increases in the means for JabRef over ArgoUML may help to understand why in the top configuration of JabRef the parameters were given a weighting factor of 8.

Body comments appear in 23% of the docs for JabRef with a term to use ratio of 1:7, the same ratio as ArgoUML, but with 10% more docs. Body comments contain 5% of JabRef's terms

while being composed of 2% unique terms and contributing 7% of the unique terms in the system. Body comments are the third highest contributor of unique terms among the lexicons, and contain the second highest percentage of terms, behind the leading comments. Looking at the method level measures, body comments are composed of 47% unique terms. Body comments contain only 13% of the method's terms for where they appear and contribute only 16% of the unique terms in the methods for which they appear. Each of these values are lower than the values for leading comments and method names which are shown to have a greater impact on the results of the FLT.

Local variables appear in 35% of the docs for JabRef and have a term to use ratio of 1:4, the lowest ratio for any of the lexicons. For JabRef, local variables contain the least amount of the terms for the system amongst the lexicons at 3%. Similar to ArgoUML, local variables contribute the least amount of unique terms to the system. These low values are seen in the method level as well, when local variables appear in a method they contribute less than 1% of the unique terms in the method. This consistently low value of unique terms for each system may be due to a low number of local variables to other terms in the method or it could be due to how local variables are used in general. If a local variable stores the return value of a method call, terms in the local variable may be the same as terms in the method call. Or it could be due to comments that explain the purpose of the local variables sharing terms. However, this sharing of information between local variables may be an indicator of why raising the weighting factor of local variables reduces the performance of the FLT for most configurations.

jEdit

As can be seen in Table 4.17c, leading comments have a higher prevalence in jEdit with 61% of the docs containing terms from the lexicon. As was seen with ArgoUML, the leading com-

ments in jEdit have a high ratio of terms to uses at 1:13 indicating some reuse of terms. However, Table 4.18 still shows that amongst the lexicons, leading comments contribute the highest percentage of unique terms at 16%. Leading comments also contain the highest percentage of terms amongst the lexicons at 15%. Looking at leading comments we see that they are composed of only 1% unique terms. This indicates that while terms may be reused between the leading comments, they provide information that is not available in the other lexicons. Looking at the method level densities, for methods with leading comments, leading comments provide 40% of the unique terms while containing 31% of the method's terms and being composed of 42% unique terms.

In jEdit, 99% of the methods contain method names and there is a ratio of 1:4 between terms and uses. The small percentage of methods that do not have names for a system are anonymous methods that are contained within other methods. A low term to use ratio indicates that there is little repetition between terms in the method lexicon. Terms that are repeated are often terms such as 'get' or 'set' that are used to identify getter and setter methods in the system. The system-wide densities show that method names contribute the second most unique terms for the system, behind leading comments. Method names contain 5% of the terms in the system and over 1% of the terms in the method names lexicon are unique. At the method level, method names in jEdit only contribute 14% of the unique terms in the method. This is the second lowest of any system, with ArgoUML being the lowest. The method names also have the second lowest composition of unique terms at 33% and contains the lowest percentage of the method's terms at 16%, again ArgoUML has the lowest percentages. This may be due to sharing of terms between the method names and the leading comments as both ArgoUML and jEdit have the highest number of docs with leading comments. These lower values may explain why for both systems, there are top configurations that do not give method names a weighting factor of 8.

Parameters appear in 60% of the docs for jEdit with a term to use ratio of 1:11. Considering the system-wide densities, parameters contain the third most terms in the system with over 4%. Considering methods with parameters and removing usages shows that lexicons contribute 10% of the unique terms in the system. This is higher than ArgoUML and muCommander, however lower than JabRef. jEdit also contains the second highest composition of unique terms at 18%.

Body comments in jEdit have the lowest number of docs amongst the lexicons. Only 13% of the docs in the system contain body comments. The ratio of terms to uses is 1:6. It is not surprising to find that body comments contain the smallest percentage of terms for JabRef at 3%. The contribution of unique terms is the third highest at 6% and body comments have the highest composition of unique terms at 2%. For methods where body comments are present, they contribute 14% of the unique terms in the method. This is the second highest of any of the lexicons, behind leading comments. Furthermore, 47% of the body comments are unique terms. Body comments have a lower percentage of the method's terms at only 10%. This low value and the low presence of body comments overall may be one of the reasons that body comments do not have a significant impact on the results of the FLT.

Of the docs in the system, approximately 35% of them contain local variables with a ratio of term to use of 1:5. This is the same percentage of docs as JabRef and a similar ratio. Local variables contain 3% of the terms in the systems and are composed of and contributes the lowest percentages of uniques terms with both percentages being less than 1%. The values are not better at the method level, as they contain only 5% of the and contribute less than 1% of the unique terms to the method. Local variables are only composed of 2% of unique terms. These low values for the uniqueness of local variables may be an indicator for why nine of the top ten configurations use the unweighted configuration or a weighting factor of 2 for local variables.

muCommander

muCommander contains leading comments in 44% of the docs for the system with a high term to use ratio of 1:25. As in all four subject systems, leading comments had the second highest term count of any of the lexicons, falling behind method names. System-wide densities in Table 4.17d show that leading comments in muCommander make up 25% of the terms in the system. This is the second highest lexicon density, falling behind ArgoUML. However, despite this, leading comments also contribute the second lowest percentage of unique terms of the four subject systems, falling behind jEdit at 15% and are composed of less than 1% unique terms. This indicates high reuse of the terms in the leading comments with other lexicons. Looking at the method level densities, when leading comments appear in a method, they have the lowest composition of unique terms of any of the four subject systems. Leading comments in muCommander also have the highest percentages of terms in the lexicon and of unique terms contributed for these methods with 49% and 58%, respectively. Leading comments are typically longer in relation to the methods in muCommander and therefore give more unique terms while also having a large amount of overlap with the other lexicons.

Not surprisingly, method names are present in over 99% of the docs in muCommander with a term to use ratio of 1:5. An important note is that for half of the four subject systems, the term to use ratio for method names is the lowest while, in the other half, method names have the second lowest ratio. This indicates that terms are not often reused throughout the system. This is important to consider when understanding that method names are believed to often describe the main responsibility of a method. This means that only a small number of methods would contain the same method terms, whereas for leading comments with a higher ratio have terms that are often reused between methods. Method names are composed of the highest percentage of unique

terms at over 2% with the second highest percentages for both lexicon density and unique term contribution at 6% and 10%, respectively. For method level densities, an average lexicon density of 20%, an average unique term density of 43%, and a unique term contribution of 21%, give muCommander the second highest densities among the four subject systems. The only system with higher density measures is JabRef, which is the other of the two systems that has a weighting factor of 8 for all top configurations for the FLT.

Parameters appear in 50% of the docs for muCommander with a ratio of 1:12 for terms to uses. Considering the system-wide density measures, parameters contain 5% of the terms for the subject system. This is the second lowest value for any of the lexicons. According to the method level densities, parameters contribute only 9% of the unique terms in the system. This is the second lowest of any of the four subject systems. When parameters appear in methods, they consist of 16% of the terms used by the method and have are composed of 18% unique terms on average. This unique term contribution is the second lowest of the four subject systems, while the lexicon density is the second highest.

Body comments are present in 17% of the docs for muCommander with a ratio of 1:11 for terms to uses. This is the second highest percentage of docs that contain body comments of the four subject systems. Body comments have roughly the same percentage of terms as the method names in muCommander at 6%, but contribute less than half of the unique terms compared to method names. At the method level, body comments contribute 15% of the terms in the method, and only 17% of the unique terms.

Local variables have the second lowest number of terms in the lexicon, but are the lexicon with the lowest number of term uses and appear in the lowest number of docs. The ratio of terms

System	Count	LC	MN	P	BC	LV
ArgoUML	Terms	5,850	6,152	1,506	1,777	3,221
	Uses	145,248	37,731	19,259	12,494	16,873
	Docs	11,309	12,529	7,019	1,652	3,666

(a) ArgoUML

System	Count	LC	MN	P	BC	LV
JabRef	Terms	2,848	3,259	1,216	2,409	2,136
	Uses	28,165	15,108	13,131	16,917	9,568
	Docs	1,516	5,296	2,945	1,179	1,873

(b) JabRef

System	Count	LC	MN	P	BC	LV
jEdit	Terms	4,089	4,749	1,581	1,923	2,497
	Uses	57,096	20,601	17,769	11,034	12,238
	Docs	4,489	7,248	4,416	971	2,549

(c) jEdit

System	Count	LC	MN	P	BC	LV
muCommander	Terms	4,110	4,822	1,629	2,182	2,075
	Uses	105,403	25,822	20,251	25,217	9,038
	Docs	3,917	8,760	4,406	1,540	1,834

(d) muCommander

Table 4.17: Unique Terms, Term Usages, and Present Documents for each lexicon for each subject system

System	Measure	LC	MN	P	BC	LV
ArgoUML	LD	0.2519	0.0654	0.0334	0.0216	0.0292
	UTD	0.0074	0.0109	0.0009	0.0166	0.0292
	UTC	0.2252	0.0858	0.0039	0.0430	0.0024

(a) ArgoUML

System	Measure	LC	MN	P	BC	LV
JabRef	LD	0.0935	0.0501	0.0435	0.0561	0.0317
	UTD	0.0194	0.0274	0.0025	0.0206	0.0031
	UTC	0.1168	0.0886	0.0070	0.0747	0.0064

(b) JabRef

System	Measure	LC	MN	P	BC	LV
jEdit	LD	0.1521	0.0549	0.0476	0.0294	0.0326
	UTD	0.0122	0.0155	0.0025	0.0228	0.0009
	UTC	0.1651	0.0755	0.0108	0.0592	0.0028

(c) jEdit

System	Measure	LC	MN	P	BC	LV
muCommander	LD	0.2504	0.0613	0.0481	0.0599	0.0214
	UTD	0.0079	0.0225	0.0014	0.0097	0.0003
	UTC	0.1560	0.1080	0.0053	0.0453	0.0005

(d) muCommander

Table 4.18: Lexicon Density, Unique Term Density, and Unique Term Contribution for each lexicon for each subject system

System	Measure	min	median	max	mean	Std.dev.
ArgoUML	LD	0.0022	0.4167	0.9798	0.4142	0.2195
	UTD	0.0000	0.5000	1.0000	0.5036	0.2077
	UTC	0.0000	0.5000	1.0000	0.4998	0.2803

(a) ArgoUML

System	Measure	min	median	max	mean	Std.dev.
JabRef	LD	0.0044	0.2667	0.9118	0.3115	0.2234
	UTD	0.0000	0.4706	1.0000	0.4778	0.2151
	UTC	0.0000	0.3333	1.0000	0.3852	0.2745

(b) JabRef

System	Measure	min	median	max	mean	Std.dev.
jEdit	LD	0.0020	0.2857	1.0000	0.3186	0.2240
	UTD	0.0000	0.4000	1.0000	0.4289	0.2153
	UTC	0.0000	0.3333	1.0000	0.4071	0.3086

(c) jEdit

System	Measure	min	median	max	mean	Std.dev.
muCommander	LD	0.0051	0.5333	1.0000	0.4976	0.2472
	UTD	0.0000	0.3636	1.0000	0.3886	0.1724
	UTC	0.0000	0.6061	1.0000	0.5788	0.2956

(d) muCommander

Table 4.19: Average Method Lexicon Density, Unique Term Density, and Unique Term Contribution for Leading Comments for each subject system

System	Measure	min	median	max	mean	Std.dev.
ArgoUML	LD	0.0008	0.1111	1.0000	0.1464	0.1410
	UTD	0.0000	0.2000	1.0000	0.2652	0.3161
	UTC	0.0000	0.0143	1.0000	0.1126	0.1942

(a) ArgoUML

System	Measure	min	median	max	mean	Std.dev.
JabRef	LD	0.0010	0.1250	1.0000	0.2104	0.2245
	UTD	0.0000	0.5000	1.0000	0.5518	0.3369
	UTC	0.0000	0.1176	1.0000	0.2422	0.2952

(b) JabRef

System	Measure	min	median	max	mean	Std.dev.
jEdit	LD	0.0007	0.1071	1.0000	0.1660	0.1806
	UTD	0.0000	0.2500	1.0000	0.3330	0.3533
	UTC	0.0000	0.0435	1.0000	0.1396	0.2336

(c) jEdit

System	Measure	min	median	max	mean	Std.dev.
muCommander	LD	0.0003	0.1250	1.0000	0.1986	0.2144
	UTD	0.0000	0.4000	1.0000	0.4355	0.3283
	UTC	0.0000	0.1053	1.0000	0.2152	0.2748

(d) muCommander

Table 4.20: Average Method Lexicon Density, Unique Term Density, and Unique Term Contribution for Method Names for each subject system

System	Measure	min	median	max	mean	Std.dev.
ArgoUML	LD	0.0005	0.0625	0.6667	0.0816	0.0706
	UTD	0.0000	0.0000	1.0000	0.0657	0.1837
	UTC	0.0000	0.0000	1.0000	0.0246	0.0781

(a) ArgoUML

System	Measure	min	median	max	mean	Std.dev.
JabRef	LD	0.0014	0.1190	1.0000	0.1701	0.1521
	UTD	0.0000	0.2000	1.0000	0.2844	0.3106
	UTC	0.0000	0.0385	1.0000	0.1178	0.1736

(b) JabRef

System	Measure	min	median	max	mean	Std.dev.
jEdit	LD	0.0007	0.0909	1.0000	0.1346	0.1306
	UTD	0.0000	0.0000	1.0000	0.1786	0.2789
	UTC	0.0000	0.0000	1.0000	0.0993	0.2037

(c) jEdit

System	Measure	min	median	max	mean	Std.dev.
muCommander	LD	0.0007	0.1011	0.8889	0.1587	0.1504
	UTD	0.0000	0.0000	1.0000	0.1783	0.2622
	UTC	0.0000	0.0000	1.0000	0.0922	0.1523

(d) muCommander

Table 4.21: Average Method Lexicon Density, Unique Term Density, and Unique Term Contribution for Parameters for each subject system

System	Measure	min	median	max	mean	Std.dev.
ArgoUML	LD	0.0027	0.0682	0.8457	0.0871	0.0798
	UTD	0.0000	0.5000	1.0000	0.5694	0.3099
	UTC	0.0000	0.1034	0.8182	0.1275	0.1151

(a) ArgoUML

System	Measure	min	median	max	mean	Std.dev.
JabRef	LD	0.0028	0.0840	0.9574	0.1307	0.1346
	UTD	0.0000	0.4444	1.0000	0.4792	0.2834
	UTC	0.0000	0.1111	1.0000	0.1604	0.1570

(b) JabRef

System	Measure	min	median	max	mean	Std.dev.
jEdit	LD	0.0017	0.0645	0.9670	0.0958	0.0990
	UTD	0.0000	0.4444	1.0000	0.4799	0.3010
	UTC	0.0000	0.0930	1.0000	0.1400	0.1437

(c) jEdit

System	Measure	min	median	max	mean	Std.dev.
muCommander	LD	0.0069	0.1297	0.8182	0.1501	0.1083
	UTD	0.0000	0.3333	1.0000	0.3591	0.2449
	UTC	0.0000	0.1333	1.0000	0.1691	0.1463

(d) muCommander

Table 4.22: Average Method Lexicon Density, Unique Term Density, and Unique Term Contribution for Body Comments for each subject system

System	Measure	min	median	max	mean	Std.dev.
ArgoUML	LD	0.0008	0.0465	0.2093	0.0559	0.0378
	UTD	0.0000	0.0000	1.0000	0.0340	0.1317
	UTC	0.0000	0.0000	0.1429	0.0072	0.0281

(a) ArgoUML

System	Measure	min	median	max	mean	Std.dev.
JabRef	LD	0.0017	0.0417	0.2222	0.0479	0.0325
	UTD	0.0000	0.0000	1.0000	0.0310	0.1479
	UTC	0.0000	0.0000	0.2000	0.0039	0.0175

(b) JabRef

System	Measure	min	median	max	mean	Std.dev.
jEdit	LD	0.0027	0.0432	0.2222	0.0500	0.0324
	UTD	0.0000	0.0000	1.0000	0.0191	0.1247
	UTC	0.0000	0.0000	0.2143	0.0022	0.0137

(c) jEdit

System	Measure	min	median	max	mean	Std.dev.
muCommander	LD	0.0021	0.0364	0.2222	0.0464	0.0344
	UTD	0.0000	0.0000	1.0000	0.0035	0.0466
	UTC	0.0000	0.0000	0.1538	0.0004	0.0051

(d) muCommander

Table 4.23: Average Method Lexicon Density, Unique Term Density, and Unique Term Contribution for Local Variables for each subject system

to term uses in the lowest with a ratio of 1:4. Looking at the system-wide metrics, local variables have the lowest unique term density and the lowest unique term contribution at less than 1% for both measures. From the method level measures, it can be seen that for muCommander, local variables have the lowest lexicon density, unique term density, and unique term contribution, out of all four of the subject systems.

4.3 Discussion of Results

This section discusses the case study and provides additional information to explain the results.

4.3.1 Does structural weighting of comments, leading terms, and local variables affect the accuracy of a LDA-based feature location technique (FLT)?

This question was broken down into three parts to examine how weighting affects the accuracy of a LDA-based FLT. I looked at weighting comments, leading terms, and local variables individually. I looked at the top configurations for each system and across all systems overall. Finally, I looked at the results of a factorial ANOVA to identify main effects from each component and to identify significant interactions.

Looking over each of the individual components it can be seen that the highest weights of leading comments and method names tend to produce better results, while parameters and local variables produce worse results at higher weights and body comments show more variation between systems. This is not unexpected. Leading comments often describe the purpose and usage of a method. For systems in industry and open source that adhere to commenting standards, increases in the weights of these terms should produce better results. Method names usually define the responsibility of the method, however, will typically only appear once. Increasing the weight of the method name allows it to have more impact on the results despite being infrequent. The

performance improvements for method names in this study were similar to those found by Bassett and Kraft. Parameters are often secondary to the main responsibility of the method, and unlike terms in the method name that might refer to the main topic of the method, terms from the parameters are likely to increase the weight of secondary topics. Local variables are sometimes similar in this regard. The variations in body comments may be due to a low usage of body comments in documents when compared to the other lexicons.

There are three ways that changing the structural weighting scheme typically works to affect the accuracy of the feature location technique: directly, indirectly, and a combination of the two. All three ways deal with changing the topic distributions of the methods in the model.

The accuracy of the FLT is directly affected by the structural weighting scheme when the component being scaled is present in the methods of the gold sets. In this case, the ranks of the methods in the gold sets were directly increased due to new topics distributions being observed from the documents. As an example, consider feature 1588028 from JabRef. This query mentions needing to fix the "DOI" url when exporting to an HTML table. It mentions prefixing the current url with the domain of a website to make it an absolute link. The query document for this feature is given in Table 4.3.1. The method to return for this query is the format method of the DOICheck class. Looking at this method reveals that a majority of the text in this document is from the identifier "fieldtext." This identifier and the split terms contained within do not appear in the query document for this feature. Terms that are important to this query that appear in this method document are "doi," "prefix," "http," "org," and "dx." However, the majority of these terms only appear once in a block comment included in the body of the method. More importantly, “fieldtext” is not prevalent in the same topics as the other terms. This means that for the given feature, the important topics do not line up between the topic distribution for the query and the topic distribution for the

Query Document
export html tabl html tabl wabstract doi url relat link instead absolut link url prefix http dx doi org resource layout tablerefs tablerefsbib layout line resource layout tablerefsabsbib tablerefsabsbib layout line export html tabl doi url

Table 4.24: Query document for JabRef feature 1588028

method. Querying against the unweighted model returns this method as the 74th result. Increasing the weighting of the body comments creates a new topic distribution with increased emphasis on the relevant terms. Therefore when querying against the model with structural weighting scheme $C(1,1,1,8,1)$, this method is returned as the 27th result.

In contrast to the direct effects of the structural weighting scheme on the accuracy of the feature location technique, it is also possible to indirectly affect the accuracy. This is done when the methods returned as the first relevant methods do not contain the structural components influenced by the weighting scheme (e.g., a method without any leading comments or local variables). This happens for two reasons. First, the methods that do contain the structural components should have their topic distributions better estimated in the model. When a query is then given that is unrelated to the method, it will be less likely for that method to be returned as a false positive. The second way is that the term-topic distribution should have a better approximation in the weighted scheme. This leads to better topic distributions for all methods whether they contain the weighted structural components or not. It is also possible that the rankings are affected by a combination of the two effects.

The results of the statistical analysis showed a significant effect from structural weighting

in each of the four subject systems between the unweighted scheme and the top configurations. Although there is not a single weighting configuration that is best for all systems, there are some things that can be noticed in the results. The top configuration for each system included a scaling factor of 8 for leading comments, a scaling factor of 8 for method names on three of the four systems, and a scaling factor of 8 for body comments on two of the four systems. There was a difference in the scaling factors for parameters and local variables. Both ArgoUML and muCommander used a scaling factor of 1 for parameters while JabRef used a scaling factor of 8 and jEdit used a scaling factor of 4. The highest weighting factor for local variables was a 4 for ArgoUML. All other systems gave local variables a scaling factor of 1 or 2. For all systems overall, the top configuration maintained a weighting factor of 8 for both leading comments and method names, a weighting factor of 4 for body comments, and a weighting factor of 2 for both parameters and local variables.

Comparing the top configurations to the results of weighting lexicons individually does not show an exact relationship. When looking at the individual weights for ArgoUML, using the best MRR for each individual component leads to a prediction of the top configuration being $C(8,4,1,2,1)$. However, the top configuration is actually $C(8,8,1,8,4)$. There are a few reasons for this. First, as terms associated with a topic increase for a method, that topic becomes more dominant in the method. As select topics become more dominant, secondary topics lose their impact on the final results. This can be detrimental when multiple methods share the same dominant topic with the query, but different secondary topics. An example of this could be two documents that both have "computer science" as a dominant topic, but where one has a secondary topic of "robotics" and the other has a secondary topic of "graphics". By increasing the weight on "computer science" too heavily, the topics of "robotics" and "graphics" can be lost. If a query then comes

in with topics of "computer science" and "graphics", the right document may not be returned. Second, lexicons may interact with one another and share complementary information. If the weight on one lexicon is increased, the weight on the other lexicon should also increase.

To test whether this second case existed, I performed a factorial ANOVA to identify significant main effects and interactions. I found an interaction between leading comments and method names on three of the four subject systems, with the only subject system that did not have an interaction between these two lexicons being JabRef. This is perhaps due to the fact that JabRef had the lowest percentage of documents with leading comments and therefore did not have enough information to find an interaction. This interaction makes sense. Both leading comments and method names are used to define the main responsibility of the method document. While information and terms between the two lexicons can overlap to some extent, both lexicons tend to provide information that is not present in the other lexicon. This can be seen by looking at the unique term contributions of the two lexicons for each of the four subject systems. This interaction was found for all systems as well. Surprisingly, for jEdit, another interaction was found. This interaction was between leading comments and local variables. It is not clear why this interaction is present in jEdit. However, in some methods, there is overlap between the terms in the leading comments and the terms in the local variables. No other interactions were identified.

For each of the four subject systems, there were main effects from the leading comments, the method names, and the local variables. This is not surprising as increasing the weights of leading comments and method names resulted in the best increase in MRR for the four subject systems, while increasing the weight of local variables resulted in the lowest MRRs for each of the four subject systems. Parameters also showed a main effect for ArgoUML. In ArgoUML, increasing the weight of parameters led to a poor MRR. For the other three subject systems, increasing the

weight of parameters did not show a significant change on the results of the FLT for the software systems. However, when looking at the results for all systems, parameters once again showed a main effect. Body comments did not show as a main effect for any of the four subject systems. This could be due to the fact that body comments were found in the lowest percentage of documents for each of the four subject systems. In a system where the developers make widespread use of body comments, the results could be different.

4.3.2 Can a relationship between the contributions of each structural component's lexicon and their weighting factors be found?

For each of the four subject systems, I looked at the system-wide lexicon density, unique term density, and unique term contribution for each of the five lexicons. I also looked at the method level statistics for these values for methods where at least one term in the method document came from the lexicon.

Leading comments had the second highest percentage of docs in two of the four subject systems. In the case of JabRef, leading comments were behind method names, parameters, and local variables. For muCommander, leading comments were behind method names and parameters. Despite these lower percentages of docs, leading comments had the highest system-wide lexicon densities and unique term contributions of all of the lexicons. These values were the highest for the mean method values as well. For both ArgoUML and jEdit, where leading comments had the second highest percentage, each of the top ten configurations had the highest weighting factor of 8 for leading comments. In the case of both JabRef and muCommander, some of the top configurations used a weighting factor of 4 instead of 8 for leading comments. For all four of the subject systems, using one of the two highest scaling factors was done for all top configurations.

However, in the two systems where leading comments were present in the highest percentage of documents, only weighting factors of 8 were used.

Not surprisingly, method names were present in the highest percentage of documents for all four of the subject systems. The only documents that do not contain method names are documents that refer to anonymous methods in Java. Method names also have the second highest system-wide and mean method level lexicon density and unique term contribution for each of the four subject systems. Looking at the MRRs for the four subject systems, each of the four subject systems have higher MRRs at a weighting factor of 8 versus a weighting factor of 4. However, both ArgoUML and jEdit actually show a decrease in the MRR from a weighting factor or 4 to a weighting factor of 8. This is reflected in the top configurations for each subject system which shows a weighting factor of 8 for all top configurations of both JabRef and muCommander while some of the top configurations for both ArgoUML and jEdit use a weighting factor of 4. In fact, for the top configuration of jEdit a weighting factor of 4 is used. Interestingly, when looking at the mean method level values, ArgoUML and jEdit have the lowest mean lexicon densities, unique term densities, and unique term contributions.

Parameters had the third highest percentage of documents for two of the four subject systems and the second highest percentage for two of the four subject systems. Parameters have the second lowest system-wide and mean method level unique term contribution among the lexicons. Lexicon density was the third highest in two of the four subject systems. The highest weighting factor of 8 is only used in a small number of top configurations among the subject systems. In general, parameters are given a lower weighting factor for the top configurations. However, for JabRef a weighting factor of 8 was used in the top configuration. Furthermore, a weighting factor of 8 is used for three of the top configurations for JabRef. When looking at the method level val-

ues, it can be seen that parameters have the highest mean lexicon density, unique term density, and unique term contribution values of the four subject systems.

Body comments have the lowest percentage of documents for each of the four subject systems. However, despite having the lowest percentage of documents in each of the four subject systems, body comments had the third highest lexicon density in two of the four subject systems and the third highest system-wide and mean method-level unique term contribution in all of the four subject systems. The MRRs for raising body comments alone showed an increase in three of the four subject systems for a weighting factor of 8 over the unweighted configuration. For three of the four subject systems, a weighting factor of 4 or 8 is used for at least half of the top configurations. The exception to this is muCommander which still uses a weighting factor of 4 or 8 for three of the top configurations. muCommander has the lowest system-wide unique term density and unique term contribution of all the four subject systems.

Local variables have the second lowest percentage of documents for three of the four subject systems and the third highest percentage for JabRef. Local variables have the lowest system-wide and mean method-level unique term contributions for each of the four subject systems. Lexicon density is the second lowest for two of the four subject systems and third highest for the remaining two. When looking at the MRRs of weighting local variables alone, each of the subject systems showed a decrease when comparing a weighting factor of 8 to the unweighted configuration. The top configurations also used low weighting factors for local variables. The best weighting factors for local variables were in ArgoUML, which used a weighting factor of 4 for the top configuration, and in muCommander, which used a weighting factor of 4 or 8 for four of the top configurations. ArgoUML has the highest mean method-level unique term contribution amongst

the four subject systems and muCommander has the highest system-wide unique term contribution for local variables.

When comparing the density measures to the results, a few interesting things can be said. First, for a lexicon with high weighting factors, higher system-wide and mean method level unique term contributions are desired. Having low values for this measure seems to indicate that increasing the weight of that lexicon will result in poorer results. While more studies would be needed to prove this, there is some reasoning behind this claim. Lexicons that have low unique term contributions share their terms with other lexicons in the document and across the system. When these values are raised, the existing information that already exists from multiple sources is being raised. This is unlikely to result in better topic distributions. However, when a lexicon is the only source of certain terms in the documents, raising that lexicon can produce better topic distributions assuming that the terms are relevant to the task and not noise. Second, when looking at whether a lexicon will have a main effect on the results, there seem to be two contributing parts. The first is in the prevalence of the lexicon indicated by the percentage of present documents and in the lexicon density. The greater the presence of the lexicon, the bigger the chance that the lexicon will affect the results. Second, is the unique term contribution or the amount of new or shared information contained in the lexicon.

4.3.3 Recommendations

The results of this study can give some early recommendations for determining when to raise the weight of a lexicon. To simplify the process, I recommend using weighting factors of 4 or 8 for any lexicon that is selected and leaving all other lexicons as unweighted. The process for selecting which lexicons to raise can be broken down into two steps:

1. Compute the percentage of documents for which each lexicon is present. In addition, compute the system-wide and average method-level densities. Identify lexicons with a document percentage around 30%, a lexicon density around 0.05, a system-wide unique term contribution of at least 0.08, average method-level lexicon density and unique term contribution of at least 0.10-0.15 each, and average unique term density of at least 0.25.

2. If available, use a testing set of features to calculate the MRRs of the unweighted configuration of an individual lexicon and then that lexicon individually weighted with a factor of 8.

Both of these steps serve to cut down on the number of tests required to determine whether to weigh a lexicon in the configuration of the system. The first step cuts down on the number of lexicons needed to be checked. The second step helps to identify whether raising the weight of the identified lexicon will increase noise or useful information for the FLT. Using the steps as given, I studied other lexicons for JabRef and identified the literals lexicon. Weighting this lexicon along with the leading comments and the method names lexicon resulted in a MRR of 0.1912, which is greater than any MRR found in this study thus far.

4.4 Using Machine Learning to Find Optimum Configurations

While general recommendations are given in the previous section for determining an appropriate configuration for a given system, there is no guarantee that the guidelines will produce the optimum configuration for a system. To find the optimum configuration requires searching all configurations at all levels of weighting. Furthermore, the weighting configurations chosen in the previous study double the weighting factors with each step. This was to show greater variations between values, but does not ensure that the best configuration is selected between the minimum

and maximum weighting configurations. However, this problem is not an easy one. Even if the search is limited to the lexicons selected in the case study, there are five lexicons with weighting factors ranging from 1 to 8. This results in a total of $8^5 = 32,768$ different topic models to check. For JabRef, the smallest system in the study, the time to train a topic model on a reasonable computer can range from 5 to 10 minutes. This is between one hundred to two hundred days to search. This is for a system of less than 75k lines of code. For larger systems, the time becomes worse.

If the best configuration is required, then a better search process will be required to return results in a reasonable manner of time. This section presents a search process based on genetic algorithms that searches through topic models for an unknown system by using information learned during the search process to identify the most likely candidates. The search process was performed using two different fitness functions. The first uses the MRR as in the case study. The second uses the silhouette function [Rousseeuw, 1986] (a measure of the quality of clusters) in a process similar to the process introduced by Panichella et al. [Panichella et al., 2013] for finding the best configuration of parameters of LDA on an unknown system. The first fitness function requires training data such as a list of features or bugs with their corresponding gold sets to be computed. The second does not require training data, but may result in a sub-optimal topic model being found.

The following discusses genetic algorithms, introduces the fitness functions, and then shows the results of the search process. The purpose of this example is not to find the optimal solution, but instead to show that a search process can be used to find increasingly optimal solutions. Therefore, the search concludes after a certain point has been reached.

4.4.1 Genetic Algorithms

Genetic algorithms [Bäck, 1996] are evolutionary algorithms that attempt to mimic the process of natural selection. Genetic algorithms are heuristic search techniques. A population of

candidate solutions is created and then a search heuristic is used to evolve the existing candidates. An iteration of a genetic algorithm is referred to as a generation, and with each generation the best solutions are selected and crossed-over to produce a new candidate population. Crossing-over takes parts of two solutions and trades those parts between the two solutions to create two new candidates. After the cross-over, it is possible for mutations to occur which randomly changes part of a solution. This process is repeated until a new population is created with the same number of candidates as the previous generation. Sometimes, a genetic algorithm will incorporate elitism, which is the option of choosing a small number of the most fit candidates to survive between generations.

Genetic algorithms require both a genetic representation of a solution and a fitness function to evaluate a solution. The genetic representation depicts each solution as a chromosome. A standard representation of a candidate solution is an array of bits or values. Each candidate typically has the same number of values in the array. The fitness function assigns a fitness value to each solution that represents how well the solution solves the given problem. Given an appropriate fitness function, a higher fitness value indicates a better solution.

The algorithm is first initialized with an initial population, which may be randomly selected or include solutions that are known to be likely to produce an optimal solution. A random population will cover the entire range of possible solutions, but a more focused set of solutions will focus on leading to an optimal solution faster.

After an initial population is set, a fitness function computes the fitness value for each of the chromosomes in the population. Pairs of chromosomes are then selected to be the parents of new chromosomes for the next generation. There are two common ways for choosing parents. The first is roulette selection which gives chromosomes with a higher fitness value a proportionately higher

chance of being selected as a parent. The other way is to use rank selection which orders each of the chromosomes by fitness value and then uses the ranks to assign probabilities of each chromosome being selected. Rank selection avoids any chromosome from having a substantially higher chance of being selected over any other chromosome, but this also leads to slower convergence.

After chromosomes are selected, crossover occurs, which is the step where sub-parts of each parent are swapped between each other. There are four common ways for crossover to occur. The first is single point crossover where a single point is selected in the offspring. The information leading up to that point is copied from the beginning of the first parent while the information after that point is copied from the ending of the second parent. In two point crossover, two points are selected and the beginning and end of the offspring come from the first parent while the middle comes from the second parent. In uniform crossover, bits and pieces are copied from each parent. Finally, in arithmetic crossover an arithmetic operation such as a vector product is performed to result in the offspring. It is possible that after crossover occurs a mutation will follow. Possible ways to perform mutation include adding values to random places in the chromosome, swapping the order of the values, inverting the values in the chromosome, or selecting new values from a range of values at random points in the chromosome. Elitism ensures that a small number of chromosomes can remain unchanged between generations. This means that these chromosomes will not go through crossover or mutation.

These steps repeat until a new population is created and then the process begins again. The genetic algorithm continues until a termination condition is reached. Possible termination conditions include a set number of iterations, finding the maximum fitness value, exhausting the search space, or finding a solution with an acceptable margin of error. In many cases, determining the maximum fitness value is as difficult as finding the optimum solution in the search space.

4.4.2 Fitness Functions Used in Study

For the purposes of this study, the search process was performed using two different fitness functions. The reason for this is to compare the silhouette function, which does not require additional training data, to the search process using the actual MRR. The MRR was defined earlier in this chapter. For the purposes of the search process, the MRR is computed for each candidate in the population and used as the fitness value for that candidate.

Panichella et al. [Panichella et al., 2013] use the silhouette function as a measure to evaluate how well a given LDA configuration clusters related documents around their dominant topics. In their paper, they make the assumption that documents should be clustered around dominant topics in topic spaces. In this case, a dominant topic is considered to be a topic that has a maximum probability in the topic-document probability distribution. The silhouette coefficient [Rousseeuw, 1986] was introduced as a way to measure the quality of clusters. The idea behind the silhouette coefficient is that objects inside of the cluster should be more similar to each other than to objects outside of the cluster. The more similarity to objects inside of the cluster and dissimilarity to objects outside of the cluster, the better. To compute the silhouette coefficient, for all objects i, the following must be computed:

- $a(i)$ - the average dissimilarity of i to all objects in cluster A, where A is the cluster for which i is present
- $d(i,C)$ - the average dissimilarity of i to a cluster $C \neq A$, this value must be computed for each possible C
- $b(i)$ - the minimum of all $d(i,C)$

The silhouette coefficient, *s(i)*, can then be computed as:

$$\frac{a(i)-b(i)}{max\{a(i),b(i)\}}$$

In the case of this study, each object *i* refers to a topic distribution that has been placed into a cluster based on its dominating topic. To calculate the dissimilarity, we use the Hellinger distance between the topic distributions. The number of calculations for finding *b(i)* can be simplified by using centroids, which can be calculated as the mean of all topic distributions in the cluster. This allows for the Hellinger distance to be found between the document and the centroid of each C, instead of between *i* and all documents not in A. The fitness value used is the mean silhouette for all documents in the corpus.

There is an important distinction to make between a dominant topic for a method that is determined by the algorithm and a relevant topic that helps handle a feature location task. The algorithm will determine the dominant topic by the number of assignments of terms to each individual topic. Relevant topics are the dominant topics of the queries. If weighting a topic produces better clustering for dominant topics that are not relevant topics to the feature location task, then an improvement in the ranks and the MRR will not be observed. For this reason, I compare the genetic algorithm technique using the silhouette function to the results using the MRR.

4.4.3 The Search Process

The specific process used in this section can be described in the following way:

1. An initial population is created based on the recommendations from the case study. This is done by choosing the lexicons with the highest likelihood to produce better results and giving them higher weighting factors, while giving other lexicons lower weighting factors. Each chromosome is represented as an array of five values corresponding to the five weighting

factors for the lexicons. The search process was executed multiple times with population sizes of 10, 25, and 50.

2. The fitness function was computed for each of the chromosomes in the initial population. The same search process was used for both the silhouette coefficient and for the MRR.

3. I chose to use roulette selection. This should lead to faster convergence. I also choose to use a two point convergence which maintains the best two chromosomes across generations. This ensures that if someone wants to perform a feature location task during the search, they have the best topic model found up to that point.

4. Uniform crossover was performed. This means that multiple points can be selected for crossover and ensures that the same weighting factors will not be consistently swapped between parents.

5. Uniform mutation was performed by selecting random points in the offspring and exchanging the values at those points with other values in the range from 1 to 8.

6. When using this search process in an actual development environment, the search should continue until the entire search space is exhausted or in the unlikely event that an MRR of 1 is identified. However, for feasibility and since the purpose of this study is only to demonstrate the search process, the number of iterations for this study was limited to 50.

Figure 4.7 shows this process. Genetic algorithms have a problem of getting stuck on local optima, so in addition to these steps I also incorporate an additional step if the average fitness value has failed to show acceptable change within five iterations. The additional step is to select half of the non-elite solutions and replace them with a random set of solutions from the remaining search

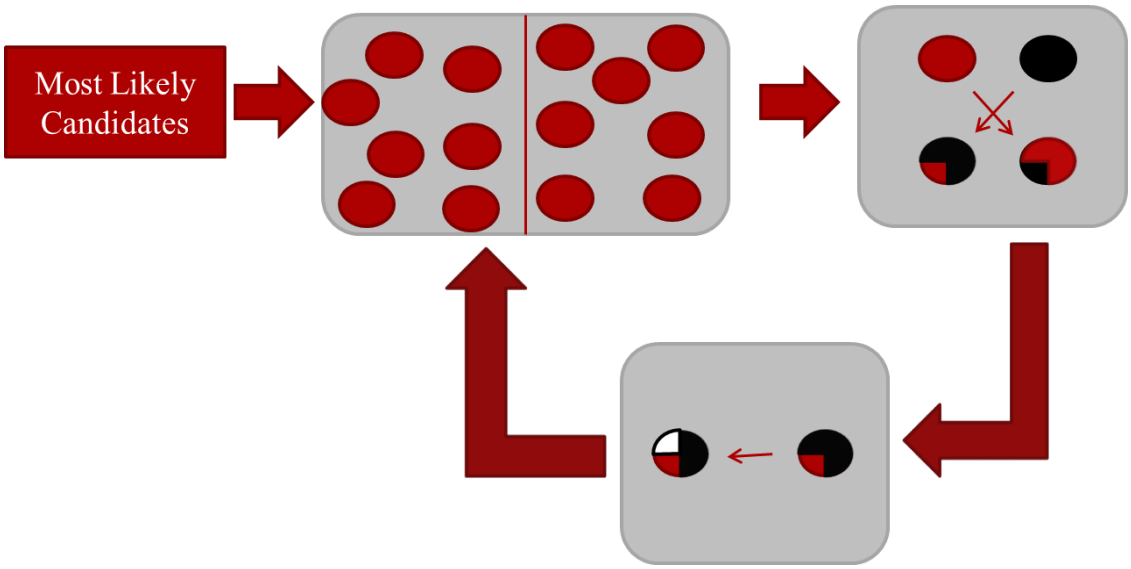


Figure 4.7: The Search Process - An initial population is selected of most likely candidates, pairs are selected for crossover and then mutation, new population is created and the process repeats

space. This results in a soft reset of the search process and allows for the search to continue away from a local optima.

4.4.4 Searching Eclipse

An experiment was conducted to evaluate whether the genetic algorithm would be able to identify better structural weightings over time. The study was performed as both a proof of concept and as an evaluation of the two possible fitness functions. I use the same notation as presented in the previous study. The primary goal of the study is to show that genetic algorithms can be used to identify better weighting schemes when a proper fitness function is applied. The exception occurs when the initial population started with an optimum weighting configuration. The advantage of this process over other search techniques is that a genetic algorithm incorporates past experience and learned information to target the most likely candidates first. In practice, this should lead to improvements in less steps than other search techniques.

I chose to use Eclipse[8] for this experiment. Eclipse is included in the SEMERU benchmark that was used in the previous study. Eclipse is an integrated development environment for Java,

[8] `https://eclipse.org/`

C\C++, and PHP. A description of the system and the version used can be found in Table 4.25. This system was omitted from the previous study for feasibility purposes. The size of the system is substantially larger than the subject systems used in the previous study, and therefore requires far more time to train in LDA. Searching through 32,768 different weighting configurations for this system (using possible weights of 1 through 8 for each of the five lexicons) is infeasible. For this reason, an improved search methodology is required to identify near optimum weighting configurations in as few steps as possible.

System	Version	SLOC	CLOC	Methods	Features
Eclipse	0.3	1,255,149	704,092	126,744	45

Table 4.25: Eclipse

I use the same setting as was used in the previous study, with population sizes of 10, 25, and 50 candidates, and an ending condition of 50 iterations for the genetic algorithm. The search focuses on different weighting schemes of the same five lexicons that were used in the previous study. The focus of the experiment is to address the following two questions:

1. Does the genetic algorithm identify weighting configurations with higher MRRs after 50 iterations?

2. How well does the silhouette coefficient perform in identifying better weighting configurations when compared to the MRR?

The goal of *Research Question 1* is to identify a weighting configuration with a higher MRR in a small number of iterations. The number of iterations was limited to 50 for this experiment due to feasibility for the experiment. Because the actual difference is not important and it is assumed

Count	LC	MN	P	BC	LV
Terms	19,623	42,167	13,315	6,594	18,883
Uses	1,114,457	330,357	341,757	179,292	187,071
Docs	63,308	114,751	73,131	26,920	73,131

(a) counts

Measure	LC	MN	P	BC	LV
LD	0.1955	0.0579	0.0599	0.0314	0.0328
UTD	0.0028	0.0132	0.0048	0.0053	0.0002
UTC	0.0769	0.1098	0.0407	0.0237	0.0008

(b) system-wide measures

Table 4.26: Terms, Uses, and Document Counts and System-wide Lexicon Density, Unique Term Density, and Unique Term Contribution for Eclipse

that the algorithm can continue to run for a longer period of time finding better results over time, the significance of the difference is not computed. For*Research Question 2*, when performing the genetic algorithm using the silhouette coefficient, I also calculated the MRR. In addition to looking at whether the approach leads to a higher MRR after the 50 iterations, I also looked at how often the silhouette coefficient selects the same two elite candidates as if following the approach using the MRR.

The first step is to identify the initial starting population for the experiment. To do this, I compute the system counts and the system-wide and method-level lexicon density, unique term density, and unique term contribution for each of the five lexicons. The results of these computations are shown in Tables 4.26 and 4.27. Based on the recommendations obtained from the previous study, candidates for a high weight include leading comments and method names. Parameters meet all criteria except for the system-wide UTC. For this reason, I chose to also test parameters. The results of weighting each of the three lexicons alone with weighting factors of 1, 2, 4, and 8, are

shown in Table 4.28. The MRRs for Eclipse are significantly lower than any of the other four subject systems. However, these results are higher than other know pure text models [Wang, Lo, and Lawall, 2014] and can be explained due to significantly higher number of methods in Eclipse. Queries that have a poor result will lower the results farther than a poor result in the other four subject systems. The results of weighting the three lexicons independently indicate that a high weight should be appropriate for all three.

To identify the initial population, I compute all weighting configurations where leading comments, method names, and parameters are given a weighting factor between 6 and 8, and body comments and local variables are given weighting factors between 1 and 3. Then, I randomly select 49 configurations from the set. I use the configuration $C(8,8,8,1,1)$ as the final configuration in each of the three populations. Finally, I compute the MRR for each of the 50 configurations and sort them in descending order based on the computed MRR. I use the top 10, 25, and 50 for each of the initial populations.

The results of the experiment are shown in Figure 4.8. Along with the final results obtained from the three population sizes, I include the highest MRR found from the initial population and the MRR of the unweighted configuration. The unweighted LDA configuration results in an MRR of 0.0026 for Eclipse, the highest MRR from the initial population is 0.0046. Looking at the results of using the MRR as the fitness function, the MRR of the weighting configuration identified in the final result gets larger as the population size searched increases. This makes sense as increasing the population size increases the space that is searched. For a population size of 10 and 50 iterations, a total of 50 configurations are searched. However, while there is a slight increase in the final MRR of the weighting configuration identified in the search for all three population sizes, for the

Measure	min	median	max	mean	Std.dev.
LD	0.0009	0.4000	1.0000	0.4168	0.2511
UD	0.0000	0.4390	1.0000	0.4652	0.2159
UTC	0.0000	0.4857	1.0000	0.4873	0.2937

(a) leading comments

Measure	min	median	max	mean	Std.dev.
LD	0.0001	0.0980	1.0000	0.1579	0.1824
UD	0.0000	0.3333	1.0000	0.3896	0.3458
UTC	0.0000	0.0690	1.0000	0.1525	0.2261

(b) method names

Measure	min	median	max	mean	Std.dev.
LD	0.0003	0.1062	1.0000	0.1570	0.1553
UD	0.0000	0.0714	1.0000	0.2245	0.2884
UTC	0.0000	0.0156	1.0000	0.1110	0.1817

(c) parameters

Measure	min	median	max	mean	Std.dev.
LD	0.0016	0.0526	1.0000	0.0804	0.0886
UD	0.0000	0.5000	1.0000	0.5869	0.3322
UTC	0.0000	0.0698	1.0000	0.1111	0.1210

(d) body comments

Measure	min	median	max	mean	Std.dev.
LD	0.0002	0.0455	0.2564	0.0520	0.0324
UD	0.0000	0.0000	1.0000	0.0061	0.0593
UTC	0.0000	0.0000	0.3333	0.0009	0.0080

(e) local variables

Table 4.27: Average Method-level Lexicon Density, Unique Term Density, and Unique Term Contribution for each of the lexicons for Eclipse

Lexicon	1	2	4	8
Leading Comments	0.0026	0.0031	0.0035	0.0039
Method Names	0.0026	0.0030	0.0037	0.0042
Parameters	0.0026	0.0029	0.0030	0.0029

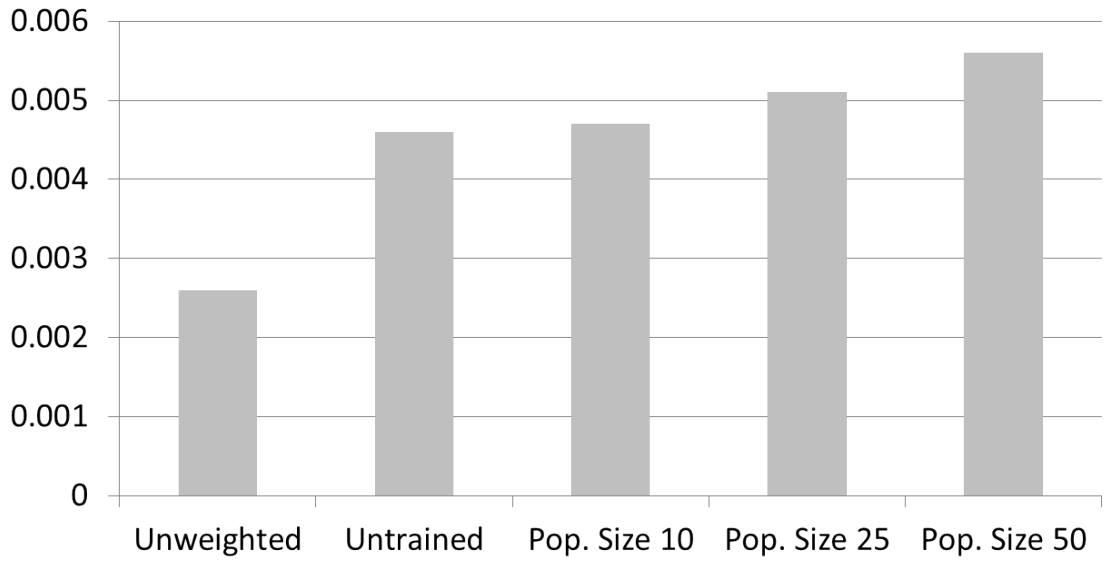
Table 4.28: MRRs Weighting Candidate Lexicons Alone

silhouette coefficient using a population size of 50 did not help identify the configuration with the higher MRR faster than the population size of 25 or 10. There are a few possible reasons for this.

The first possibility is due to the randomness of the candidates selected as pairs in each generation. Given that the paths are non-deterministic and each run of the genetic algorithm will result in a different path, the result could be due to the selection of a poor path. The second possibility may be due to the fact that the silhouette coefficient indicates how well the trained LDA model clusters documents around dominant topics and not relevant topics. This means that the choices in the elite candidates may not be the candidates with the highest MRRs. The candidates identified when searching with a population of size 10 or size 25 may have been identified when searching with a population size of 50. However, they may not have been the candidates with the highest silhouette coefficient in the generation and therefore not carried over to the next generation.

To determine how well the silhouette coefficient selected the candidates with the highest MRR, I computed this alongside the silhouette coefficient at each generation. The number of times the highest silhouette coefficient also had the highest MRR, fell in the top 3 of all MRRs, fell in the top 5, and fells in the top 10 is shown in Table 4.29. It can be observed from this table that the silhouette coefficient does not always identify the weighted configuration with the highest MRR. In addition, the larger population size also resulted in more occurrences where the top silhouette coefficient did not line up with the top MRR. It is important to realize, however, that this does not

(a) MRR



(b) Silhouette

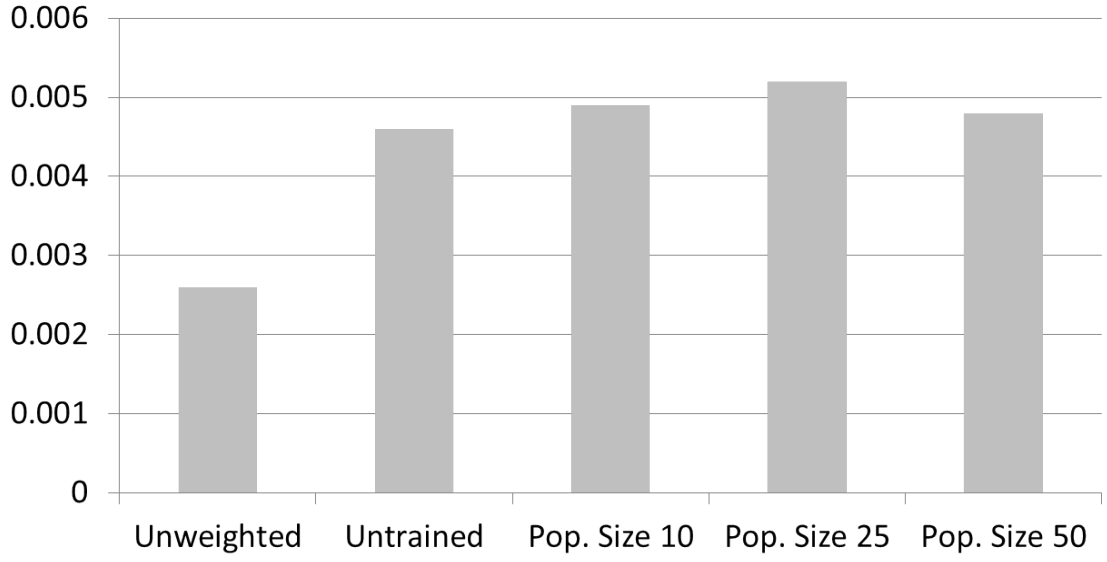


Figure 4.8: The unweighted configuration, top configuration from the untrained population, and results of 50 iterations of the the genetic algorithm for population sizes of 10, 25, and 50 based on MRR using the MRR and silhouette coefficient fitness functions

Population	Top 1	Top 3	Top 5	Top 10
10	44	47	49	50
25	40	42	46	50
50	33	39	43	49

Table 4.29: Number of times the top silhouette coefficient was in the top MRRs

indicate that a smaller population size will result in a better search. If both searches ran until the end, aside from possible noise when training the LDA models, all searches should identify the same final weighting configuration.

Although the results of the search using the silhouette coefficient did not always identify the best weighting configuration, each search did identify after 50 iterations a weighting configuration that was higher than the starting configuration. Furthermore, the top silhouette coefficient was in the top 10 of MRRs the majority of the time. A good strategy may be to use the silhouette coefficient as the fitness function until sufficient training data is available.

4.5 Summary

In this chapter, I looked at weighting configurations of LDA for four subject software systems and showed that weighting lexicons identified by their structure increases the performance of an LDA-based FLT. I also identified characteristics of the lexicons that showed the best increases in performance when weighted highly and made recommendations for identifying which lexicons should receive higher weights. While the recommendations may not identify the configuration with the highest performance, I outlined a search process that can identify better weighting configurations over time.

The limitations of this approach are in the need to identify the lexicons that should receive higher weighting factors and the time required to identify the best weighting configuration. In the

next chapter, I present a way to perform retrieval using language modeling, but still allows the developer to weight terms appearing in certain lexicons. This approach removes the overhead of training different LDA models.

Chapter 5

STRUCTURED SOURCE CODE RETRIEVAL

In the previous chapter, I presented an approach to LDA for source code that used structural weighting to improve the performance of a FLT. This research was based on previous research conducted in configuring LDA for feature location, and on a previous study conducted on using structural weighting for LDA. This approach showed a significant improvement over unweighted LDA when using the proper weighting configuration. The weakness of this approach is that it requires multiple models to be trained in order to properly identify the best weighting configuration for a new software system. In this chapter, I present a new approach that uses structured retrieval and advanced queries to improve the performance of a FLT without requiring the formation of a new index.

The approach presented in this chapter combines language models with SDR. Structured retrieval differs from the previous LDA approach and other common techniques in TR-based feature location in that it does not treat documents as bags of words, and instead considers documents as structured templates that contain contents in the form of the terms. Documents extracted from source code can be structured using AST information. Consider a document representing a method. Distinct sections can represent the signature and body. Within the signature, fields can represent the method name, and a subsection can represent the parameter list, within which fields can represent the parameter types and names. Similarly, sections and fields can represent parts of the method body, including statements and their constituent expressions. Note that a given term may appear

in only the signature (e.g., a parameter type) or in both the signature and body (e.g., a parameter name). Unlike a traditional TR model, in which the structure of a method is not considered, a structured retrieval model allows a developer to issue a query that includes terms and sections/fields of interest.

Context is key in source code search tasks. Identifiers play many roles in source code, and an identifier can convey different information in different contexts. For example, suppose the identifier *test* appears in the name of one method and in a method call in the body of another method. In the former case, *test* may relate to the main responsibility of the method, whereas in the latter case, *test* may relate to one step in a larger responsibility. Using content and structure to search a corpus and retrieving either the most relevant section(s) or document(s) based on content relevance and structural similarity is known as structured document retrieval (SDR) [Lalmas and Baeza-Yates, 2009]. Unlike traditional TR models, SDR models support powerful query languages in which a user may specify several constraints, including the scope of the query and the weight or probability assigned to each term or structural entity.

In this chapter I present:

- A methodology for feature location using CAS queries
- Motivating examples for and an empirical study of the use of CAS queries for feature location

The focus of the empirical study is on how different aspects of the CAS queries affect the performance of the SDR-based FLT. This gives insights into the aspects that are most important, and an understanding about the possible performance gains that can be achieved using the approach. In future studies described in Chapter 6, I plan to use the results of this study as a

base for understanding and improving the way developers will be able to use the technique by recommending queries and providing additional information to developers through improved user interfaces.

5.1 Approach

For my study, I use the Indri Retrieval Model which combines language modeling with an inference network to create a structured retrieval model. In this section I describe my approach, including the Indri retrieval model on which it is based.

5.1.1 Overview of Indri

Indri[1] is a search engine developed as part of the Lemur project between the University of Massachusetts and Carnegie Mellon University. The search engine supports structured query documents written in the Indri Query language and provides the user of the system with a flexible model for defining fields and other attributes of a document in the corpus. The system uses a combination of language modeling and inference networks as the search engine's retrieval model. To study the effects of CAS queries, I used Indri as a structured retrieval search engine and wrapped it in a system for indexing and querying a software system.

5.1.2 The Indri Retrieval Model

Before Indri can index a corpus and build a model, the user needs to provide the search engine with a parameters file. This parameters file contains both fields and metadata about the corpus being indexed. For the purposes of my research, I make direct use of the fields that can be specified in the parameters. Fields are extents of the textual content of a document (e.g. a heading or body tag in an HTML document). Indexed field names are available for use in an Indri query language query. In a query, the user may specify the fields they wish to search. By specifying fields

[1] `http://www.lemurproject.org/indri/`

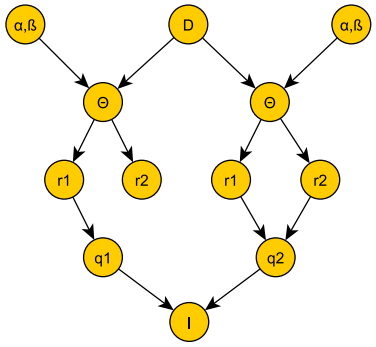


Figure 5.1: Example Indri Model

and using an appropriate file format, users may develop their own structured formats for indexing in the Indri system.

The Indri Retrieval Model combines inference networks for TR [Turtle and Croft, 1990] with language models. An inference network is a method of defining a joint probability distribution over a collection of random variables. It is represented as a directed acyclic graph where each node in the graph corresponds to random variables. Edges in an inference network represent conditional dependencies where nodes that have no connection are independent. With each node is an associated probability function that takes as input the variable from the node's parent variables and gives as output the random variable represented by the node. Documents and language models are represented as nodes in the inference network.

An example of an Indri model is shown in Figure 5.1 The Indri Retrieval Model contains the following nodes (nodes are discussed from parents down to children):

- **Document node (D)** - a document is represented as a multiset of binary vectors where each vector in the multiset corresponds to a position in the document and each binary value represents the presence or absence of a possible feature (e.g., a term, phrase, entity) at that position in the document.

- **Smoothing parameter node** - represent hyperparameters that are used in the smoothing of the language models. By default, Indri uses Dirichlet smoothing and therefore the hyperparameters in the default case represent the hyperparameters α and β.

- **Model node (θ)** - for each feature in the document (e.g., a context) there is an associated model node that contains smoothed multiple-Bernoulli distributions that represent a language model for all text under that field (i.e., a model node representing a method signature would contain all text from the method signature of that document). This allows for structured queries that combine evidence from multiple representations of a document.

- **Representation concept node(r)** - represents binary random variables related to the features in the document representation (e.g., whether a specific term appears in a document). Each representation concept node is a child of a model node and the same representation concept node may appear as a child of multiple model nodes in the network.

- **Belief node (q)** - based on a structured query, these nodes are added dynamically to the network and are binary random variables corresponding to a conditional probability table derived from the query. They allow for combining of beliefs in a query and for weighting of queries. The various belief operators will be described later in this section.

- **Information need node (I)** - belief node that combines all evidence within the network into a single probability. This probability is used for ranking the documents and is the same as P(I |D, alpha, beta)

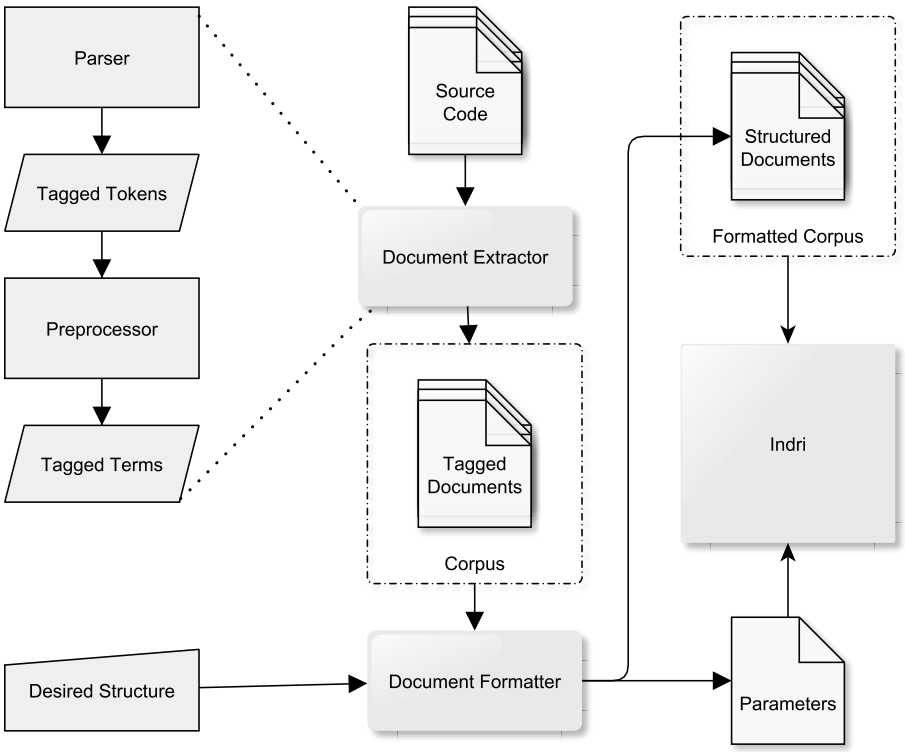


Figure 5.2: Document Extraction Process

5.1.3 Creating a Structured Corpus

I developed a tool to extract structured documents from methods in source code. This is shown in Figure 5.2. The tool takes as input a set of source code files written in the Java programming language and parses the files using ANTLR[2] (the tool is extensible to other languages or to make the current output of the tool more expressive for different documents, all that is needed is a new grammar). Tokens are extracted from the document and tagged with their source entity (e.g., method name, comment). The tool maintains order of tokens and folds leading comments into their respective methods by associating comments immediately preceding a method with that method. These tokens are then fed to a preprocessor that performs splitting, case normalization,

[2] http://www.antlr.org/

stop word removal, and stemming. When performing splitting both the split and original tokens are maintained, however this can be configured to change. The resulting terms are then grouped into documents and placed into a corpus. The next step is the document formatter that reads the corpus and is given as input a structure for the new method documents. The formatter then outputs a corpus of structured documents in the specified format and an Indri parameters file.

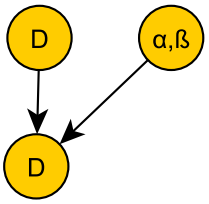
While the objective is to provide the flexibility needed by the developer, there are important considerations to take into account during creation of a structured corpus.

The traditional approach to TR techniques on feature location is to treat method documents as flat files of text with no structure. Indri is able to perform this operation using an inference network with a single language model node, shown in Figure 5.3a.

However, terms in source code may be categorized due to implicit structural information. For example, a developer reading code is able to recognize variables, methods, classes, literals, class fields, comments, etc., due to the usage and relationship of the terms to the code. Furthermore, a developer is able to recognize whether a term is a part of a declaration or use. This information can be extracted automatically by tools to build structured documents from source code.

The structure in a source code document is implicit and is understood in different ways. Therefore it is possible to create different structural representations of a method document. As an example, terms in source code can be broken down into three broad categories: identifiers, literals, and comments. Each of these three categories reflect the different ways that terms can be used. Identifiers for instance define the state and behaviors of the system. Literals are terms that are found in input and output operations. Terms in comments are important to the developer as they give additional information that is not present in the program's instructions. For each structural lexicon in the document, a new model node is created. An inference network with these

(a) Flat File Representation



(b) Identifiers, Comments, and Literals

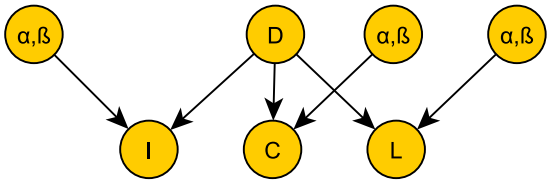


Figure 5.3: Different types of structured method documents

```
/*computes the area of a triangle using heron's formula*/
public static double areaTriangle(Point point1, Point point2, Point point3)
{
    //compute the lengths of the sides
    double side1 = computeDistance(point1, point2);
    double side2 = computeDistance(point2, point3);
    double side2 = computeDistance(point3, point1);

    //compute half of the perimeter
    double p = (side1 + side2 + side3) / 2;

    return Math.sqrt(p * (p - side1) * (p - side2) * (p - side3));
}
```

Figure 5.4: Example Method

three categories would require language model nodes for each of the three lexicons. I show such a network in Figure 5.3b. While this type of document is simple, it allows for quick and easy searches over all of a method's identifiers, comments, and literals. An example of a useful query would be one where the developer believes an error message should print with a certain class of words. This type of model for the structured document reduces the complexity for querying and requires less storage for the index. However, it is not as robust or as flexible as it could be.

```
<method_doc>
  <method_comment>computes</method_comment>
  <method_comment>the</method_comment>
  <method_comment>area</method_comment>
    ...
  <signature>
    <method_name>areaTriangle</method_name>
    <parameter_type>Point</parameter_type>
    <parameter_name>point1</parameter_name>
      ...
  </signature>
  <body>
    <line_comment>compute</line_comment>
    <line_comment>the</line_comment>
    ...
    <local_var_name>side1</local_var_name>
    <method_call>computeDistance</method_call>
    <primary_name_ref>point1</primary_name_ref>
    <primary_name_ref>point2</primary_name_ref>
    <local_var_name>side2</local_var_name>
    <method_call>computeDistance</method_call>
    <primary_name_ref>point2</primary_name_ref>
    <primary_name_ref>point3</primary_name_ref>
    ...
  <body>
</method_doc>
```

Figure 5.5: Example Document

As another approach, Figure 5.4 gives an example of a Java method that may be present in the source code of a system. Inspecting this method, we can identify several categories for terms. The first line of the method is a leading comment related to the method. In the method signature we can identify the method name, parameter types, and parameter names. In the body of the method, we can identify method calls, local variables, line comments, and references. An example of a structured document with this information is given in Figure 5.5 using XML formatting. The root of the document is divided into three main categories, the method comment, the method signature, and the body. In this case, method comments precede methods and are the same as our leading

comments from the previous chapter. Method comments are believed to be used for defining the responsibilities of a method and therefore provide important terms for searching. The method signature can be broken down further into method name, parameter names, and parameter types. I show a nesting structure that allows for multiple types of queries at different levels of granularity. The goal is to allow for more flexible queries that will lead to an increase in accuracy. Similarly, I show nested terms appearing in the body.

This document allows for increased flexibility in the search queries. Any of the structural lexicons can now be queried and with the proper parameters file, indexed by the search engine. However, while this improves the flexibility for certain tasks, it decreases the flexibility for others. For instance, if the developer wishes to query the system over all variables, the query needs to be explicit on which fields to search. This applies to comments and literals as well. Furthermore, it is possible that a finer granularity may not lead to any improvement in relevant results obtained from a structured query. With each new structural lexicon added, I increase the number of language models for a document and these language models are increasingly smaller in size. Language models are an approximation of the usage of terms in a particular context. Therefore, the more information available for the language model, the better that approximation can be. Defining too many lexicons increases the overall complexity of the system while the benefits for doing so should start to see diminishing returns.

Furthermore, the complexity continues to increase not only in the modeling of lexicons, but also in the network. Every field specified in a parameters file given to Indri becomes indexable, meaning that every field creates a new sub-document node under the original method document.

The choice of structure used for a structured method document should be based on the

system being indexed, what is needed to complete the tasks, and the resources available for the process.

5.1.4 Creating Structured Queries

Current queries used in feature location take the general approach of utilizing the description or title from a bug or issue report as a query. This approach is due to the need in research to make the most general queries possible without making assumptions about the developer. While these queries fit the purposes of such studies, very little research has focused on using developer knowledge about a system to improve the results of a query. For instance, a developer may have an understanding of what terms are actually used in the system, what terms relate to method names and class names, and what terms refer to variables or fields. A developer might also have expectations of what context a term is used in. Allowing a more robust query system that allows developer input can increase the likelihood of returning relevant results. Furthermore, such a query system would be complementary to existing query refinement and reformulation techniques and provide additional input to such a technique about a term's context. I wish to leverage a developer's knowledge during the software search process. For this reason, a proper query language should allow for multiple options and be highly configurable. The Indri query language allows for a wide variety of queries. The simplest queries in Indri take the form:

#combine(“side” “computes” Point)

The # signifies a query, *combine* means to search for the terms together in a document, while the query is provided in the parentheses. I wrote a translator for a simplified version of the Indri Query Language. In the simplified version, the aforementioned query is written as:

“side” “computes” Point

For such a query, each term in the query is given equal weighting and the query is issued across all fields. This query does not make use of the structural document, but instead uses the document as a bag of words where each term is given an equal weight. The quotation marks around a term indicate a search for the unstemmed/unnormalized version of the term. Lack of quotations searches for the stemmed/normalized terms. These queries represent content only queries. These queries try to match over all terms in a query instead of based on fields and each field is of equal importance. Developer knowledge is not incorporated into this phase of the retrieval process.

Queries are not limited to these basic types however. It is possible to search for queries based on fields:

[signature](Point point1 areaTriangle)

[method_name](areaTriangle)

[signature]([method_name](area triangle) [parameter_name](point1))

In these queries, the fields appear in the square brackets while the query appears in the parentheses. Each of these queries makes use of the field to increase the likelihood that a particular method will be found. The examples above all search for particular terms in a method signature. The first of these queries looks for the terms over all terms in the signature (i.e., the method name, parameter type, and parameter name). The second query is more specific and searches directly for all methods with the name "areaTriangle." This query is useful if the developer already knows the terms to be a part of the name of methods in the program. The final query is a nested query. It searches for terms in the method signature but also searches for "areaTriangle" in the method name and "point1" in the terms for parameters. Each of these queries returns increasingly specific results and is based on what the developer knows about the source code.

In addition to the basic and structural queries, Indri allows for terms and queries to be weighted. Weighting queries opens up another avenue for increasing accuracy. In the previous chapter, we saw how structural weighting could increase search results. If the developer has certain beliefs about the usage or location of terms, they may choose to vary the weights of terms in queries or the weights of nested queries. The basic structure of a weighted query is:

weight(2.0 [signature](area) 1.0 [body](area))

In the example query, the developer has given greater weight to documents with area appearing in the method signature versus the method body. Perhaps, the developer knows the method they want computes an area, so they believe that area is likely to be in the method name. They want to see methods with area in the method name before other possible choices. By weighting the query, they are more likely to retrieve the results they want. While developers may manually weight their queries, we also wish to identify queries that may return higher results for developers that are less familiar with the system or suggest queries that may produce more relevant results to the developer. There are multiple ways to suggest queries in this case. Two possible approaches include issuing the same query across each field in a source document but varying the weight of each field according to some heuristic or suggesting fields in which a term is most likely to be found. Future research is required to refine these approaches.

Queries form belief nodes that have conditional probability tables associated with them. This allows for multiple types of queries and for queries to be performed efficiently. Weighting of structural lexicons is performed using the weight belief operator. The function for computing the weight is given in the following expression, where n is the number of parent nodes, each with a belief b_i and weight w_i:

$$b_{weight} = \prod_{i=1}^{n} b_i^{w_i/W}$$

5.2 Study Design

I conducted an emprical study to determine the effects of structured retrieval on a FLT. I study the effects of different query types derived from bug reports, the effects of including different structural lexicons in a query, and the effects of different weightings on the structural lexicons. I use three different types of structured queries during my study. The first is derived from the terms in the comments, signature, and body of the method. The second corpus is derived from terms in the indentifiers, comments, and literals. Finally, I look at a finer grained corpus that is composed of terms derived from the leading comments, method names, parameters, body comments, and local variables. The last is composed of children lexicons from the previous. All other terms in this corpus fall into a structural lexicon called the body. The purpose of this last corpus type is to look at a finer granularity of structural lexicons without making the study infeasible. Across the entire study, I look at 4,608 different queries for each feature of four different subject software systems.

In this section, I describe the design of the empirical study as well as present threats to the validity of the results. Since I have already discussed the subject systems, the benchmarks, and the effectiveness measures in the previous chapter, I will not include them in the study description.

5.2.1 Definition and Context

The goal of this study is to understand how the different lexicons (i.e., the query, corpus, combinations of lexicons, and structural weighting) of the structured retrieval system can effect the performance of a FLT. It is not the goal of the study to determine the optimum configuration for every feature for every software system, as this would require knowing the best search strings to use.

I computed 4,608 different queries for each of the four open source Java systems that were presented in the previous chapter. I used the same software systems to make it easier to understand the differences between the two techniques as will be discussed in Chapter 6. For each of the four software systems, I used the same 372 features and bugs that were used in the previous chapter.

I compared three different corpora. The first corpus breaks a method into comments, the signature, and the body (CSB). Comments include both the leading comments that come before the method and the comments that are found within the body. The method signature includes all terms that are found within the header of a method. Finally, the body contains all terms that are found within the definition of the method, including the local variables, literals, method calls, etc. The second corpus that I used includes the identifiers, literals, and comments (ICL). Identifiers are taken from both the method signature and from the body of the method. Finally literals are string literals that as you would find in print statements or error messages. The final corpus is the analogous corpus to the corpus used in the previous chapter. This corpus is composed of the leading comments, the method names, the parameters, the local variables, and the body comments (LMPBV). To avoid the problem that would arise from missing terms if we did not include other fields, I grouped all other terms in the method into an other field. This other field was included in all queries. Since the importance of the study is in looking at the relative results and not the absolute results. This field should not have an impact on our results.

I used the same weighting factors as were used in the previous study. This is for both the reasons listed in the previous chapter and in order to make comparisons between the two techniques easier. However, these weighting factors differ in function compared to the previous study. The weighting factors used in LDA multiply the raw term counts utilized during the LDA training process. However, weighting factors in the structured retrieval model give a relative importance

to the belief of each sub-query. In this study, this translates to giving a relative importance to the results of searching for terms in each structural field. As an example, if a weighting factor of 2 was given to method names while a weighting factor of 1 was given to leading comments, then it would be twice as important that the terms appear in the method name versus the leading comments.

Since there are multiple combinations of structural lexicons and because each structural lexicon can receive a different weighting in the query, I adopt a modified notation to the notation presented in the previous chapter.

The notation for the comments, signature, and body corpus is:

$$C\{C = 1, S = 1, B = 1\}$$

where C is the comments, S is the signature, and B is the body. Each number represents the weighting factor for that structural lexicon. The remaining notations for the last two corpora are analagous.

For the identifiers, comments, and literals corpus, the notation is:

$$C\{I = 1, C = 1, L = 1\}$$

And for the final corpus, the notation is:

$$C\{LC = 1, M = 1, P = 1, BC = 1, LV = 1\}$$

In addition to these three corpora, I also computed the results of each feature using the flat corpus with no fields. This allows for me to compare the structured retrieval approach to the traditional language modeling approach.

5.2.1.1 Setting

There are three distinct parts to building the language model for the study. The first uses a text extractor and preprocessor implemented in Java 7 using an open source Java 1.5 grammar

and ANTLR v3. The tool has four steps. The first step extracts documents from methods and treats inner methods as distinct methods. The text of the inner method (e.g., a method inside an anonymous class) will only be attributed to that method, and not the containing method. This step outputs each document as a separate XML file for further processing. The next step rewrites the separate XML files into a single XML file with comments preceding methods being folded into the text of that method. This single XML file is then reread as a corpus for preprocessing. During the preprocessing stage, terms in the corpus are split, normalized, stemmed, and all stop words are removed. The second part of the corpus construction takes this XML file and a second XML file as input. The second XML contains a mapping of all parts of the first XML file to the structured fields. It then converts the XML files to a set of documents with the proper terms included in the appropriate fields. In addition, it produces a parameters file that is used by Indri for indexing. The third and final part is performed by Indri which takes in the outputted documents in the corpus and the parameters file and creates an index. For this study, I use the default α and β values for Indri that are used to smooth the language model.

As in the previous study, the preprocessor implements the steps described in Figure 1.1 in Chapter 1.2. Identifiers are filtered from `java.lang` before splitting tokens on camel case, underscores, and non-letters. After splitting, the original token is retained [Marcus et al., 2004]. The terms are then normalized to lowercase and then filtered by an English stop word list [Fox, 1992], Java keyword list, and term length (with any terms less than three character being removed). A Porter stemmer[3] is then applied.

I use a similar preprocessor for the queries. Because Indri requires a parameters file for queries that is written in their language, I use a parser that takes queries written in the simplified

[3] `http://tartarus.org/~martin/PorterStemmer/`

language presented in Section 5.1.4, applies the same preprocessing steps that were applied to the method document, and then outputs a parameters file to the query language used by Indri. It then runs the query against the indexed system.

5.2.1.2 Creating Queries

For the purposes of this study, I created queries by using the title, description, and the combination of the two from the feature requests in the benchmark. There are two possible ways that a developer may use CAS queries, the first is to write the queries with their own search strings while defining the fields and the weighting to be used. The second possibility is for queries to be automatically generated while the developer indicates what fields to search and the proper weighting to use. Future studies will look into both possibilities. However, by looking at the three types of queries in this study, the goal is to understand how different types of information and query lengths impact the results. To form the queries, the following process was used:

1. Represent each combination of fields as a binary string where a 1 indicates that a field is present in the query (i.e., for the binary string 101 for the ICL corpus, the first 1 means include identifiers and the second 1 means include literals).

2. Use depth-first search to identify every possible weighting configuration for a given combination.

3. For every field and weighting factor in the combination, include the title, description, or the combination as the search string.

4. Repeat this process for all combinations for a given corpus.

An example of the title queries for ArgoUML can be found in Appendix A.

5.2.2 Research Questions

The focus of the case study is to address the following questions:

1. Does query type affect the accuracy of a structured retrieval-based FLT?

2. Does changing the combination of included fields affect the accuracy of a structured retrieval-based FLT?

3. Does structural weighting affect the accuracy of a structured retrieval-based FLT?

4. How does the best configuration of structural field combination and weighting affect the accuracy of a structured retrieval-based FLT?

For each of the above research questions, I conducted a series of Friedman tests to determine whether a statistically significant effect existed between the different configurations for each question. If so, I form hypothesis tests corresponding to each pair of configurations involved in the test. The independent variables for these tests are the weighting configurations with scaling factors (1, 2, 4, 8) and the different combinations of structural lexicons for each of the corpora, while the dependent variables are the effectiveness measures.

For each hypothesis test, I do not presuppose the directionality of the difference between two configurations. Therefore, each hypothesis test is two-tailed. For each configuration pair, I formulate a null hypothesis to evaluate whether there is a significant difference when one of the two configurations is used over the other. If, after testing the null hypothesis, I find I can reject it with a high confidence ($\alpha = 0.05$), I accept an alternative hypothesis. Accepting the alternative hypothesis corresponds to there being a significant difference between the two structural weighting scheme configurations.

An example null hypothesis:

$H_0 : C\{I = 4, C = 2, L = 4\} = C\{I = 2, C = 4, L = 4\}$

Configuration $C\{I = 4, C = 2, L = 4\}$ **does not significantly affect** the accuracy of the structured retrieval-based FLT compared to configuration $C\{I = 2, C = 4, L = 4\}$.

The corresponding alternative hypothesis:

$H_A : C\{I = 4, C = 2, L = 4\} \neq C\{I = 2, C = 4, L = 4\}$

Configuration $C\{I = 4, C = 2, L = 4\}$ **does significantly affect** the accuracy of the structured retrieval-based FLT compared to configuration $C\{I = 2, C = 4, L = 4\}$.

The remaining null and alternative hypotheses are analogous.

Since queries are independent of one another, it is possible to choose a query for each feature that is either the best for that feature, the worst for that feature, or somewhere in the middle. For this reason, I also perform the same analysis comparing the best, middle, and average for each corpus. If I find that a statistically significant effect existed, I formulate a set of hypotheses to test between each case.

An example null hypthesis of this type:

$H_0 : Best\{ICL\_QUERY\} = Average\{ICL\_QUERY\}$

$Best\{ICL\_QUERY\}$ **does not significantly affect** the accuracy of the structured retrieval-based FLT compared to $Average\{ICL\_QUERY\}$.

The corresponding alternative hypothesis:

$H_A : Best\{ICL\_QUERY\} \neq Average\{ICL\_QUERY\}$

$Best\{ICL\_QUERY\}$ **does significantly affect** the accuracy of the structured retrieval-based FLT compared to $Average\{ICL\_QUERY\}$.

Finally, it is also possible that I do not have to choose between different corpus configurations and can select the overall best configuration for each query. In that case, I only wish to know how the best, average, and worst configurations compare.

An example null hypothesis of this type:

$H_0 : Best\{QUERY\} = Average\{QUERY\}$

$Best\{QUERY\}$ **does not significantly affect** the accuracy of the structured retrieval-based FLT compared to $Average\{QUERY\}$.

The corresponding alternative hypothesis:

$H_A : Best\{QUERY\} \neq Average\{QUERY\}$

$Best\{QUERY\}$ **does significantly affect** the accuracy of the structured retrieval-based FLT compared to $Average\{QUERY\}$.

5.2.3 Data Collection and Analysis

For each of the questions, I followed a slightly different data collection process.

The first step that I followed was to run each of the systems with the flat corpus with no fields. This represents the traditional query-likelihood model used with language modeling. This became my baseline for understanding the effectiveness of the structured retrieval approach.

For *Research Question 1*, I used the feature title, feature description, and the combination of the two as my query types. I ran all queries for each system for each corpus of CSB, ICL, and LMPBV with all three query types. I collected the effectiveness measures corresponding to each query for each query configuration and each corpus. This resulted in a list of effectiveness measures for each query and configuration pair that I used as the basis for my analysis on this problem.

I created boxplots for each system, corpus, and query type. I also include this for all systems overall. Afterwards, I identify the best, average, and worst case from all query types for each corpus and from all corpora. I use a bar graph to show the differences between these values for each feature. In addition, I calculate the percentage of times that each query type is better for each corpus and the percentage of time each corpus has the best rank for a feature. Finally, I calculate the MRRs for all of the above configurations as well as the best, average, and worse cases for each corpus and across all corpora.

To determine whether there are any significant effects, I conduct Friedman tests with wilcoxon posthoc tests and holm correction. I perform this analysis to compare each of the query types in each of the corpora, and to compare the best, average, and worst for each feature for each corpus, and to compare the best, average, and worst, along with the flat corpus.

For both *Research Question 2* and *Research Question 3*, I keep both of the combined and title query types for comparison. Because description makes up a large portion of the combined queries, I do not include it for comparison in these questions. I plot the best, average, and worst cases for each feature when combining the combined and title query types for each corpus and across all corpora. I then perform significance testing to determine whether there is a difference between the best, worst, and average across both query types for each corpus and for all corpora.

In *Research Question 2* I look at each of the different structural combinations for each of the three corpora. For CSB and ICL, there are 7 different configurations each. For LMPBV, there is a total of 31 different configurations. Due to the number of configurations, I do not show the boxplots, but instead report descriptive statistics along with the MRRs. The descriptive statistics computed include the minimum rank, first quartile, median, third quartile, and maximum rank. Together these descriptive statistics describe the spread of the effectiveness measures for each corpus and query type for that system.

I look at the top weighting configurations for *Research Question 3* and produce the boxplots and MRRs for each of these configurations.

The remainder of the analysis for *Research Question 2* and *Research Question 3* is analogous to *Research Question 1*.

For feasiblity, I only look at the title query type for *Research Question 4*. This reduces the number of combined queries required by 2,151 for each system. Since combined queries take significantly longer than title queries, this results in a significant difference in the time required (by 1.5 months). The analysis for this problem is similar to the analysis performed for *Research Question 3*.

5.2.4 Threats to Validity

The study has limitations that may affect the validity of my findings. In this section, I describe some of the limitations as well as my attempts to mitigate them.

Threats to conclusion validity concern how reasonable the conclusions we reach about the relationships in our data are. As in the previous chapter, in order to mitigate these I used non-parametric statistical tests and did not make any assumptions about the distributions of my effectiveness measures. In addition, I used Holm correction to account for errors in my p-values.

Threats to construct validity concern how well the measurements used in the study describe the concept being studied. Possible threats to construct validity include how well the effectiveness measure measures the feature location process and whether my benchmarks are accurate. I used previously established measurements and benchmarks that were used in previous research [Dit et al., 2011, 2012; Revelle et al., 2010] and made publicly available online.

Threats to internal validity include possible errors in executing my study procedure or defects in my tool chain. To mitigate these problems, I thoroughly tested my tool chain and assessed the quality of my results at each step in the procedure. Because The same tool chain was applied to all subject systems, any errors are systematic and should not affect my results substantially.

As with the previous chapter, the queries are another threat to internal validity if they do not describe the features of interest. ArgoUML and jEdit are tools developed for programmers. Therefore it is likely that those who wrote the issue reports were more likely to accurately describe the issue than other users.

Threats to external validity concern the extent to which I can generalize my results. All four of my subject systems are written in the Java programming language, therefore I am unable to generalize to different languages. However, The four subject systems represent a wide range of sizes and domains. Furthermore, the selected subject systems are similar to systems found in industry. Therefore, my results should be similar for other systems written in Java.

Another threat to external validity is in the way developers may actually use the queries. Since I obtain queries from feature requests, and not from the developers, it is possible that the queries may be written differently. However, the purpose of this study is not to find the optimum queries or to instruct the developer on the best way to write the query. Additional studies are required for this. The purpose of the study is to understand how the different aspects of the struc-

tured retrieval process can effect the results of a FLT. For this purpose, the feature request contains information about the feature of interest written by users and developers of the system.

In the next section, I describe the results of my study. Due to the large number of configurations and queries, only a subset of the complete results are included in this dissertation.

5.3 Results of Case Study

In this section, I present the results of the study by question.

5.3.1 Does query type affect the accuracy of a structured retrieval-based FLT?

The first question that I wished to address was how different queries can affect the accuracy of the FLT. For this question, I used three different types of queries obtained from the feature request of each of the studied features. The ultimate goal of this technique is to allow developers to use their own knowledge of the system to formulate the best queries possible. However, the goal of this question is to study how different aspects of a query may affect the accuracy of the technique, and the hope of using queries obtained from the feature request is to eliminate bias during comparison while still giving insights into query effects.

5.3.1.1 ArgoUML

In Figure 5.6, we see the boxplots for the different corpora under study. For comparison, I also included the results of performing the FLT on the flat corpus. For both CSB and ICL, the Title queries had slightly lower spreads versus the Description and Combined queries. While the spread for Title in the LMPBV corpus is slightly larger than the other two, the 3Q and 1Q value are both reduced when compared to the other corpora. In contrast, for the flat corpus, the Title spread is larger than the other corpora, but has a lower 1Q measure.

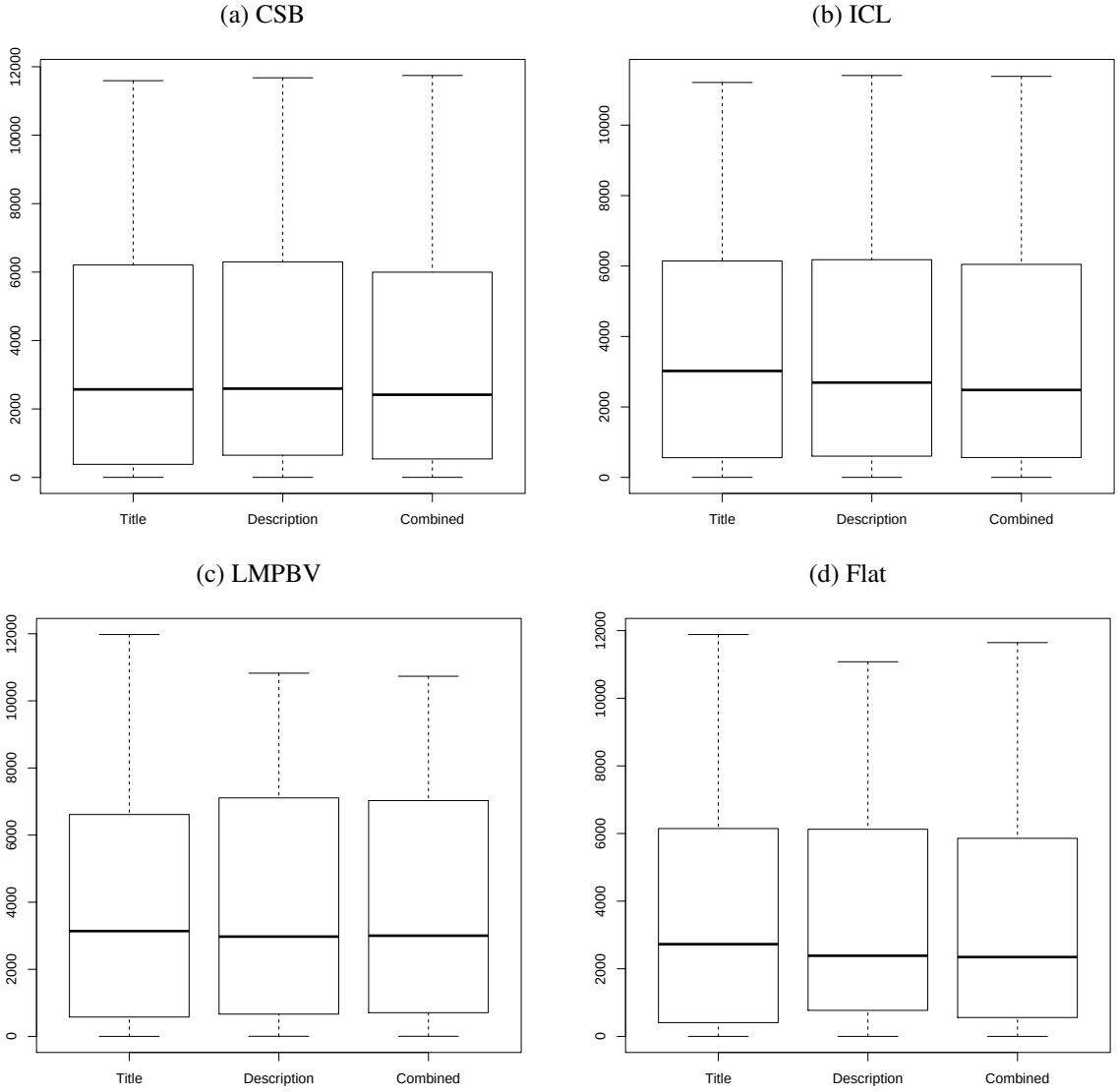


Figure 5.6: The Effectiveness Measures for the three different query types (Title, Description, Combined) for ArgoUML

Query	CSB	ICL	LMPBV	Flat
Title	0.0688	0.0387	0.0098	0.0617
Description	0.0275	0.0286	0.0074	0.0155
Combined	0.0374	0.0283	0.0072	0.0262

Table 5.1: MRRs for the three different query types (Title, Description, Combined) for ArgoUML

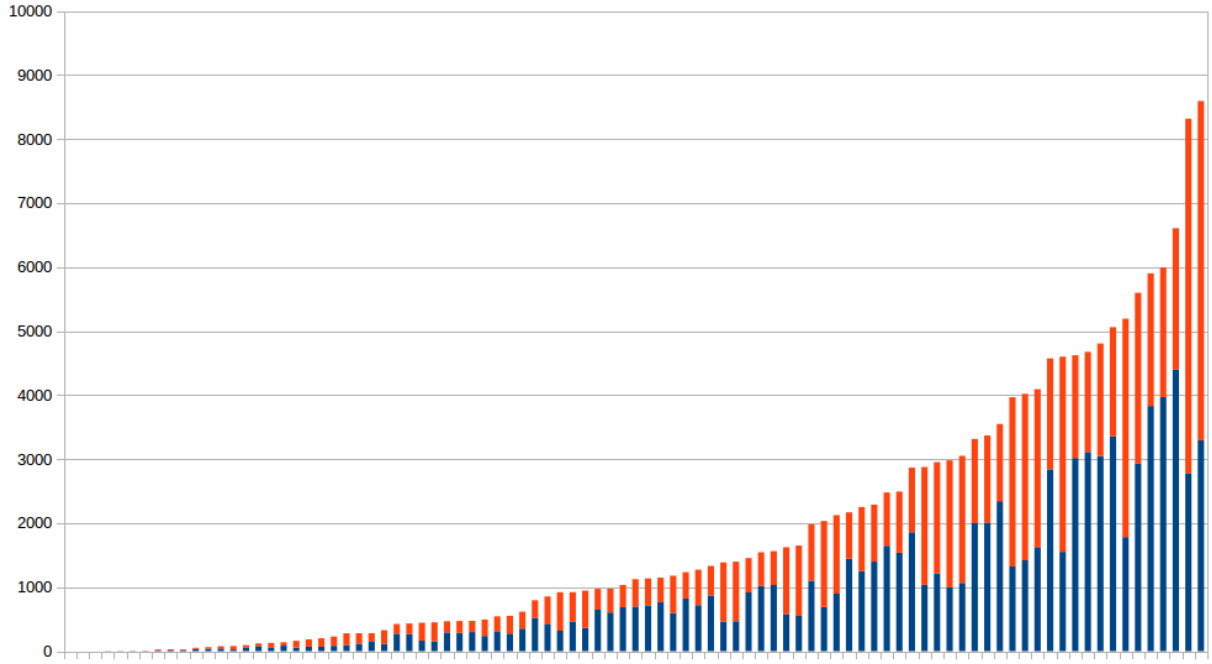
We can see the MRRs for the four different corpora and the three different query types for ArgoUML in Table 5.1. Of the three different query types, the highest MRRs for each of the four corpora come from the Title queries with the highest MRR being for the CSB corpus with the Title queries. For the remaining three corpora, Flat has the second highest MRR and LMPBV has the lowest MRRs for all four corpora. For both the ICL and the LMPBV corpora, the Description queries have a slightly higher MRR for both corpora.

I ran a Friedman test with a wilcoxon post-hoc analysis to determine whether there were any significant differences between the three different query types for each of the four corpora. The results of the analysis found that there is significant difference between each of the three query types for the CSB corpus. For ICL, there does not exist a significant difference between Description and Combined or Title and Combined, but there does exist a significant difference between Title and Description. For LMPBV, there exists no significant differences between the three query types. Finally, for the flat corpus, there exists a significant difference between the Title and the Descriptions query types.

While a single query type may be used for each feature, this does not have to be the case. In this approach, each feature can have a query that is uniquely tailored to identify the relevant source code elements for that feature. For this reason, I looked at what would happen if I chose a query that was the best for each feature, the worst for each feature, or near the average. In Figure 5.7, I show the differences between the different cases for each feature in each of the four different corpora.

Of the four corpora, CSB has the smallest differences between the best queries and the worst. The largest difference between the best and worst query is greater than 10,000. The maximum difference between the average and the best query is just over 6,971. The greatest mean

(a) CSB



(b) ICL

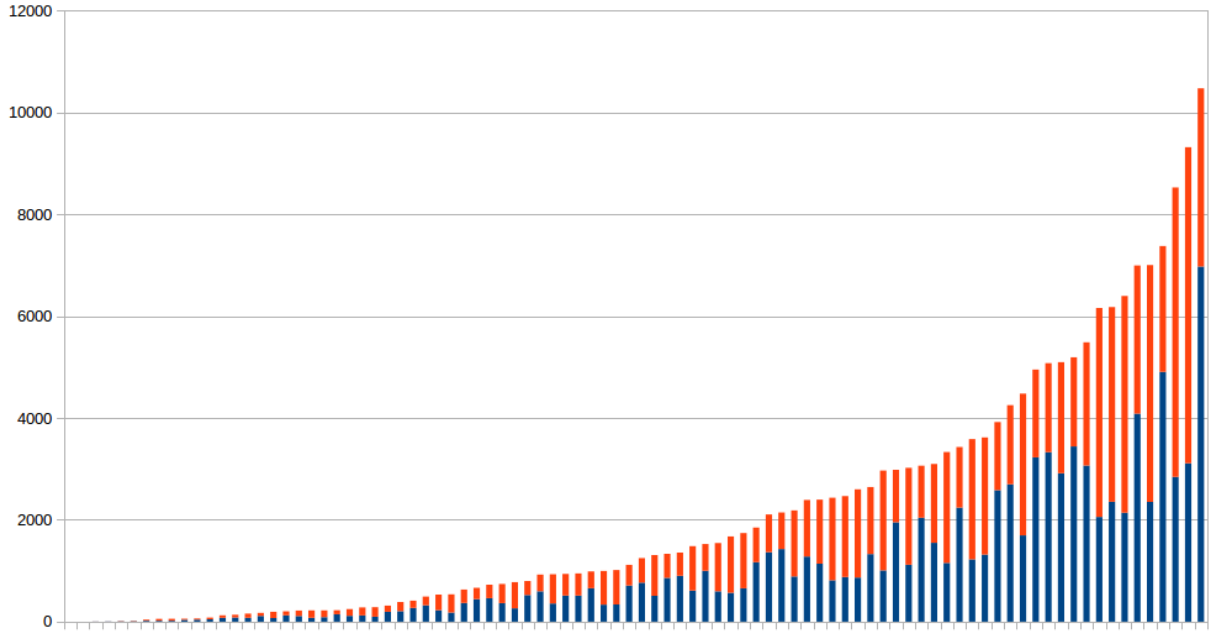
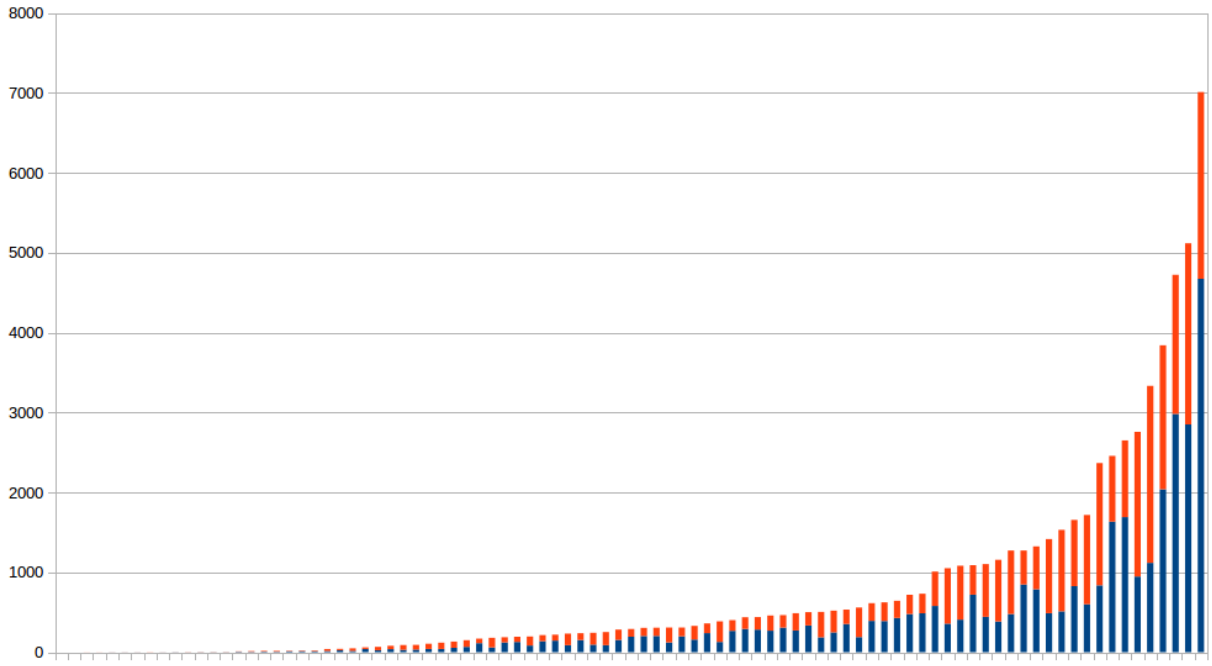


Figure 5.7: Stacked bargraphs representing the distance from the best query to the average (bottom) and the worst(top) for ArgoUML. Graph is ordered by distance from best to worst.

(c) LPMBV



(d) Flat

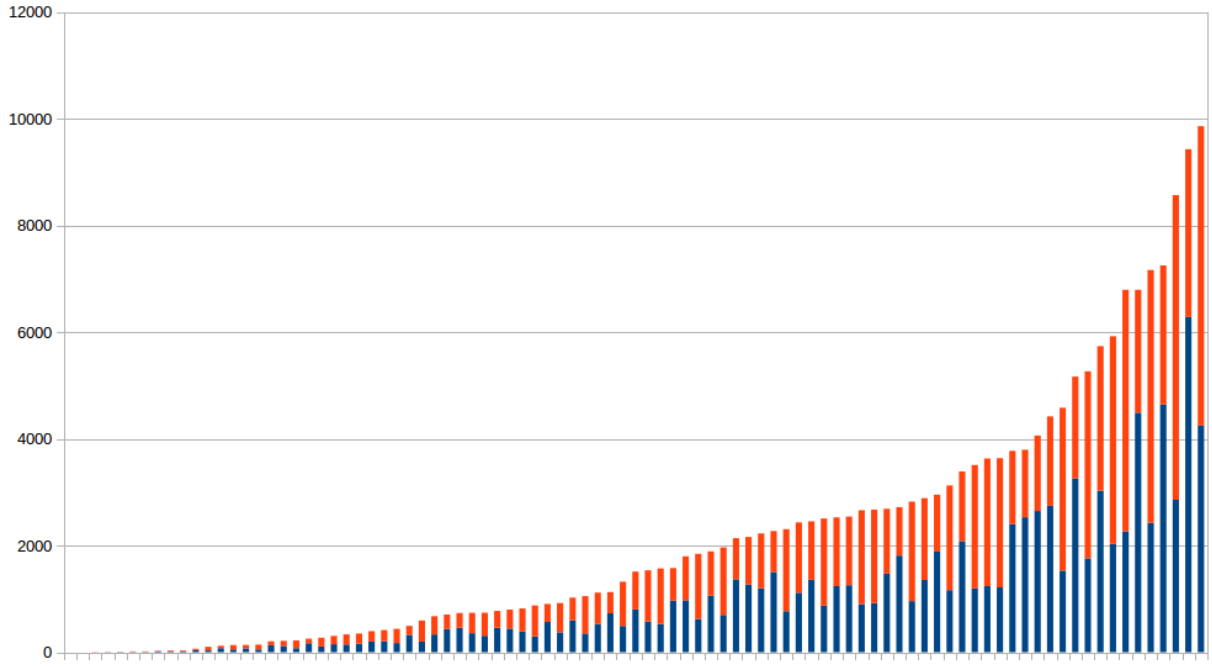


Figure 5.7: Stacked bargraphs representing the distance from the best query to the average (bottom) and the worst(top) for ArgoUML. Graph is ordered by distance from best to worst.

Query	CSB	ICL	LMPBV	Flat
Best	0.0707	0.0416	0.0113	0.0728
Average	0.0304	0.0269	0.0068	0.0091
Worst	0.0258	0.0253	0.0057	0.0038

Table 5.2: MRRs for choosing the best, average, and worst case for each feature for ArgoUML

Query	CSB	ICL	LMPBV	Flat
Title	64	57	57	55
Description	15	21	35	22
Combined	21	22	8	23

Table 5.3: Percentages for each query type where the best query was found for ArgoUML

distance between the best query and the worst is 2,113, while the greatest mean distance between the average query and the best query is 1,047. The smallest means come from LMPBV which has a mean distance between the worst query and the best query of 724 and the mean distance between the average query and the best query is 381.

I also calculated new MRRs based on the best, average, and worst case queries. The results can be found in Table 5.2. For each of the four corpora, choosing the best query for each feature results in a higher MRR than any of the single query types. This means that there is not a single query type where all features perform better. The largest improvement for MRR is for the flat corpus, while the smallest improvement is for LMPBV.

To understand whether a single query type dominated when selecting the best queries, I calculated the percentages of each query type where the best query was selected. I did this for all four corpora and the results can be found in Table 5.5. From this table it is also observed that for

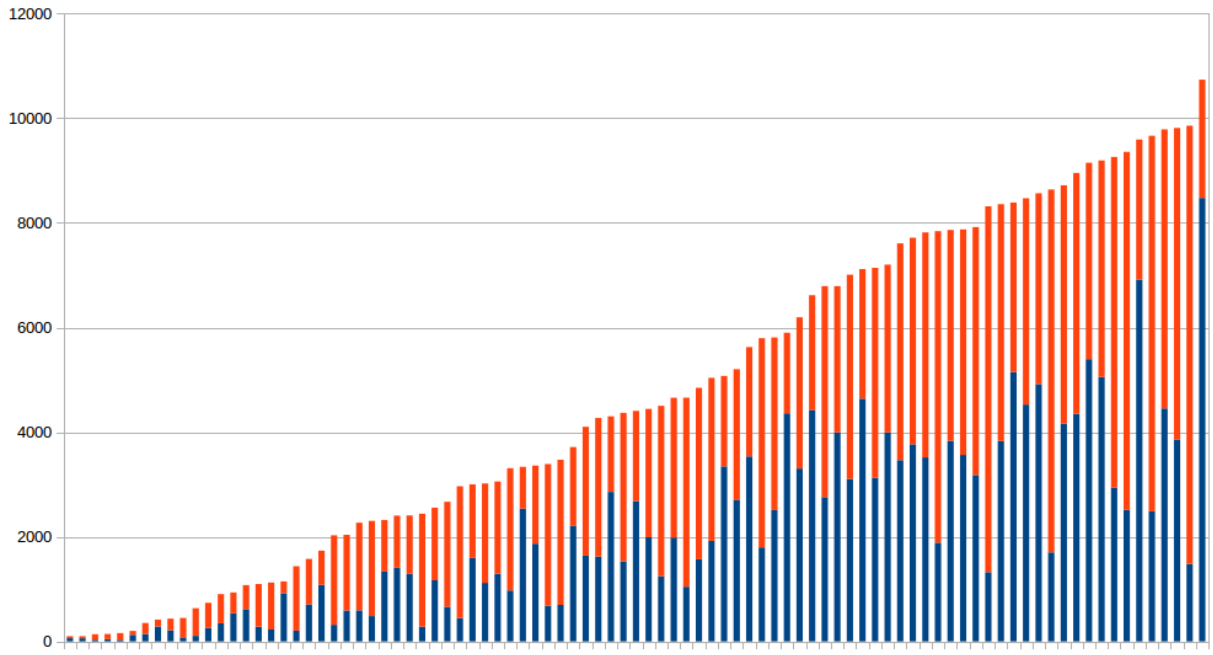


Figure 5.8: Stacked bargraphs representing the distance from the best query to the average (bottom) and the worst(top) for ArgoUML. Graph is ordered by distance from best to worst.

each of the four corpora, the Title query type has the highest percentage of preferred queries. For all but one of the corpora, the description has the lowest percentage of preferred queries.

I performed a Friedman test with a wilcoxon post-hoc for each of the four corpora to determine whether there was any significant difference between the best, average, and worst cases. For each of the four corpora there was a significant difference found between each of the three cases.

While it increases the complexity of the model, it is possible to combine each of the three different corpus types and to use them interchangeably. For this reason, I calculated the best, average, and worst case when choosing the query using any of the corpora. The differences between the values can be seen in Figure 5.8.

In the case of allowing for all corpora, the greatest distance between the best query and the

	Best	Average	Worst
MRR	0.0888	0.0020	0.0007

Table 5.4: MRRs for choosing the best, average, and worst case for each feature from all corpora for ArgoUML

	CSB	ICL	LMPBV	Flat
Percentage	19	18	12	43

Table 5.5: Percentages for each corpus where the best query was found from all corpora for ArgoUML

	Title	Description	Combined
Percentage	59	16	25

Table 5.6: Percentages for each query type where the best query was found from all corpora for ArgoUML

worst query is 10,738. The mean distance between the best query and the worst query is 4,706, while the mean distance between the best and the average is 2,114.

I computed the MRR for the best, average, and worst case of each feature. The results of these calculations can be found in Table 5.4. Again, the MRR increases over the top MRRs for each corpus indicating that there is not a single corpus that has the best query for every feature. I show the percentage of best queries that come from each corpus in Table 5.5. CSB and ICL have roughly the same percentage of best queries around 18%. The corpus with the largest percentage of best queries is the flat corpus at 43%, while LMPBV had the smallest percentage at 12%. I also calculated the percentage of best queries that came from each query type. Similar to previous results, the Title query type has the highest percentage of best queries at 59%. The lowest percentage comes from the Description query type at 16%.

Finally, I performed a Friedman test with a wilcoxon post-hoc for the best, average, and worst case of each feature across all corpora. The results of the analysis showed that there is a significant difference between each case.

5.3.1.2 JabRef

The boxplots for JabRef can be found in Figure 5.9. For three of the four corpora, there is not a consitently smaller spread for any of the three query types when compared to the others. For CSB, the smallest spread is for Title while the largest spread is for Description. For both ICL and LMPBV, Title has the lowest 1Q measure, however for LMPBV, Title also had the largest spread. The smallest spread for ICL came from Description, while the largest spread came from Combined. In the case of LMPBV, Description had the smallest spread but it did not have the lowest 1Q or 3Q measure. For the flat corpus, Combined had the smallest spread while Title had the largest spread, however Title had a lower 1Q value than Description.

I calculated the MRRs for each query type for all four corpora and recorded their values in Table 5.7. For each of the four corpora, the MRR for the Title query type is significantly higher than the MRR values for the Description and the Combined query types. The highest MRR is found for the flat corpus with the Title query type while the lowest MRR is also found for the flat corpus but with the Description query type. For the Description query type, CSB slightly outperformed the other three corpora. While for the Combined query type, the flat corpus had the highest MRR.

I ran a Friedman test with a wilcoxon post-hoc analysis to determine whether there were any significant differences between the three different query types for each of the four corpora. The results of the analysis found that there were no significant differences between any of the three query types for any of the four corpora.

Despite the Friedman test not finding any significant differences between the three query

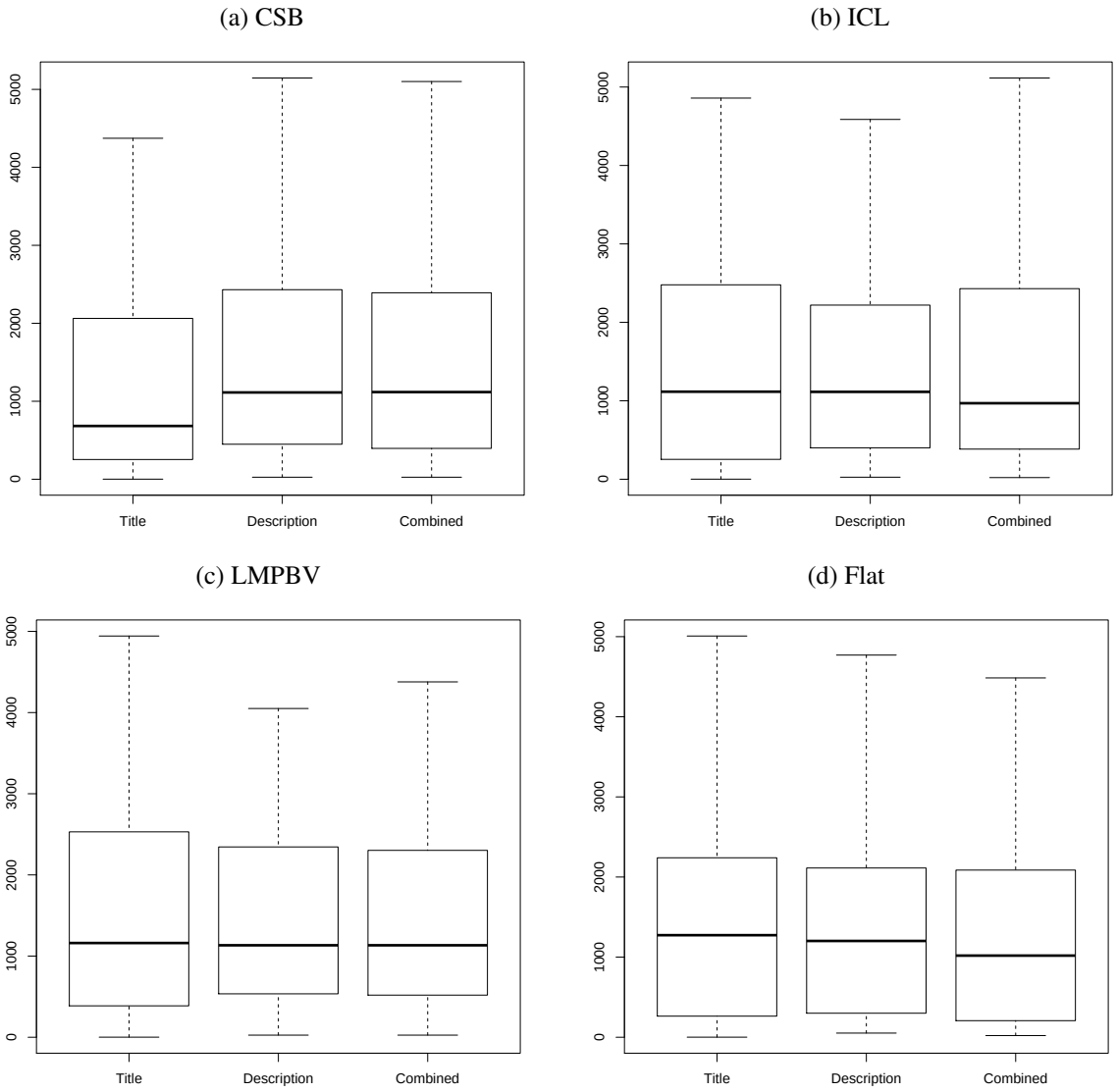


Figure 5.9: The Effectiveness Measures for the three different query types (Title, Description, Combined) for JabRef

Query	CSB	ICL	LMPBV	Flat
Title	0.0342	0.0316	0.0429	0.0832
Description	0.0038	0.0036	0.0032	0.0027
Combined	0.0040	0.0040	0.0032	0.0046

Table 5.7: MRRs for the three different query types (Title, Description, Combined) for JabRef

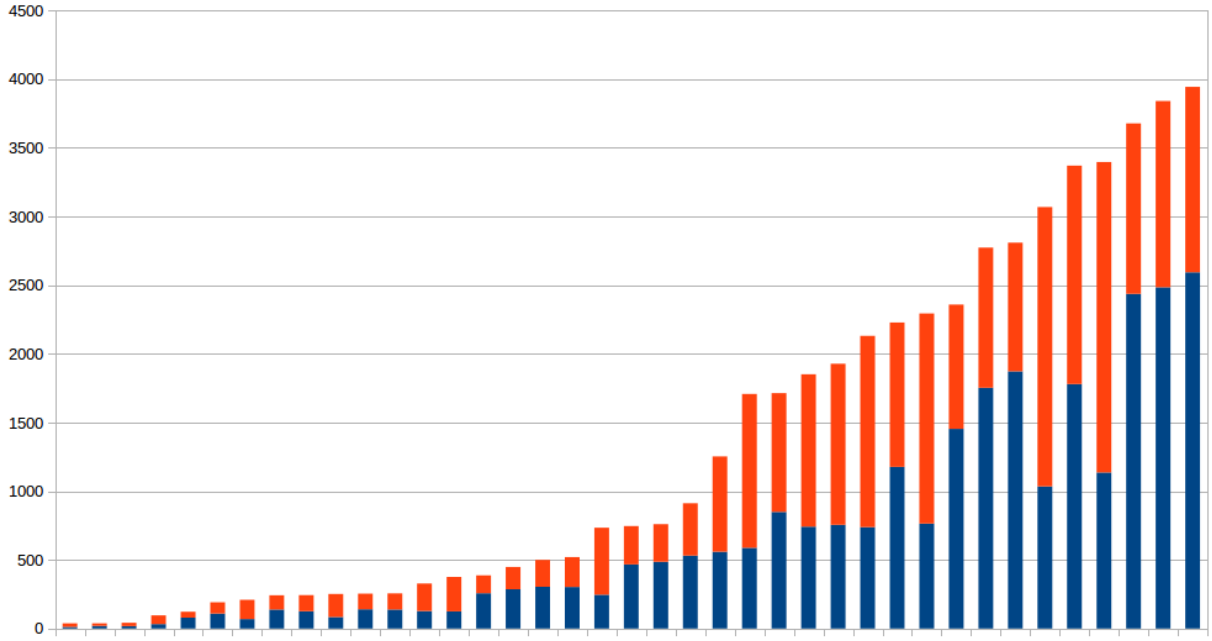
types, improvements may still be found if we choose the best result for each feature from the three query types. In Figure 5.10, I show the differences between the best, average, and worst cases for each feature for the four different corpora.

The greatest distance between the best and worst query can be found in the flat corpus at 4,334, slightly higher than the maximum for ICL at 4,106. However, the greatest mean distance between the best query and the worst query can be found in the CSB corpus at 1,334. The greatest distance between the best query and the average query is 2,710 and it is found in the flat corpus, while the greatest mean distance between the best query and the average query is once again found in the CSB corpus. The smallest mean distance between the best query and the worst query is 684 and it is found in the LMPBV corpus. LMPBV also has the smallest mean distance between the best query and the average query at 337. This corpus also has the smallest distance between the best and the worst query at 0. For three of the four corpora, the mean distance between the best and the average query is smaller than the distance between the average and the worst query.

The MRRs for the best, average, and worst queries for each of the four corpora in JabRef can be found in Table 5.8. As could be predicted from the results of the previous Friedman test, the differences between the MRRs for the best queries and the MRRs of the best query type for each corpus are small. The largest difference is found in ICL at 0.02. The smallest change is found for LMPBV at less than 0.0001. This result is not surprising after comparing the MRR to the mean difference between the best and the worst queries for each feature. Also of note for this system, the greatest MRR is found for the flat corpus.

I calculated the percentages of times that the best query was selected from each of the three query types for each of the four corpora. The results of these calculations can be found in Table 5.11. For three of the four corpora, the Title query type had the best query over 50% of the time.

(a) CSB



(b) ICL

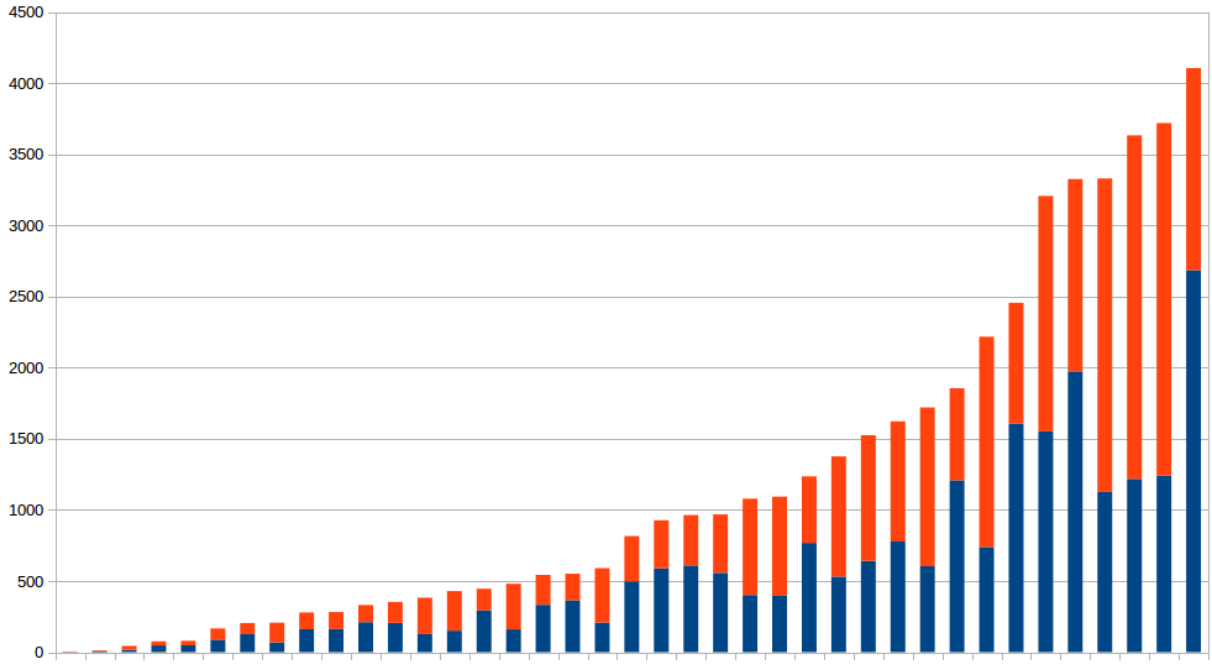
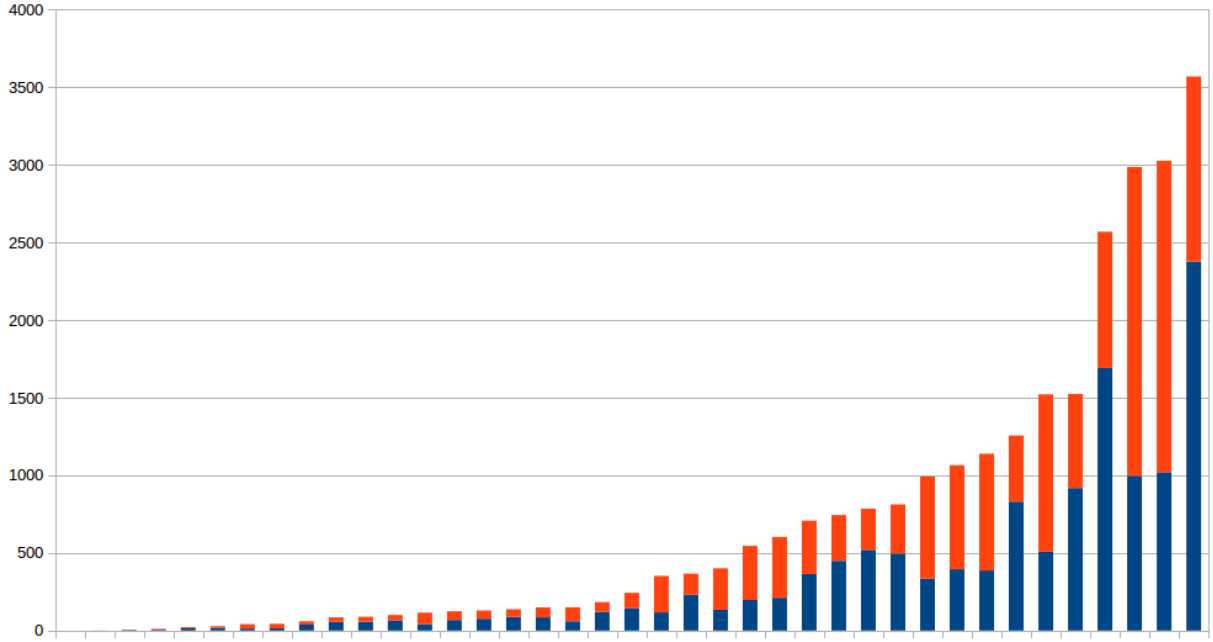


Figure 5.10: Stacked bargraphs representing the distance from the best query to the average (bottom) and the worst(top) for JabRef. Graph is ordered by distance from best to worst.

(c) LPMBV



(d) Flat

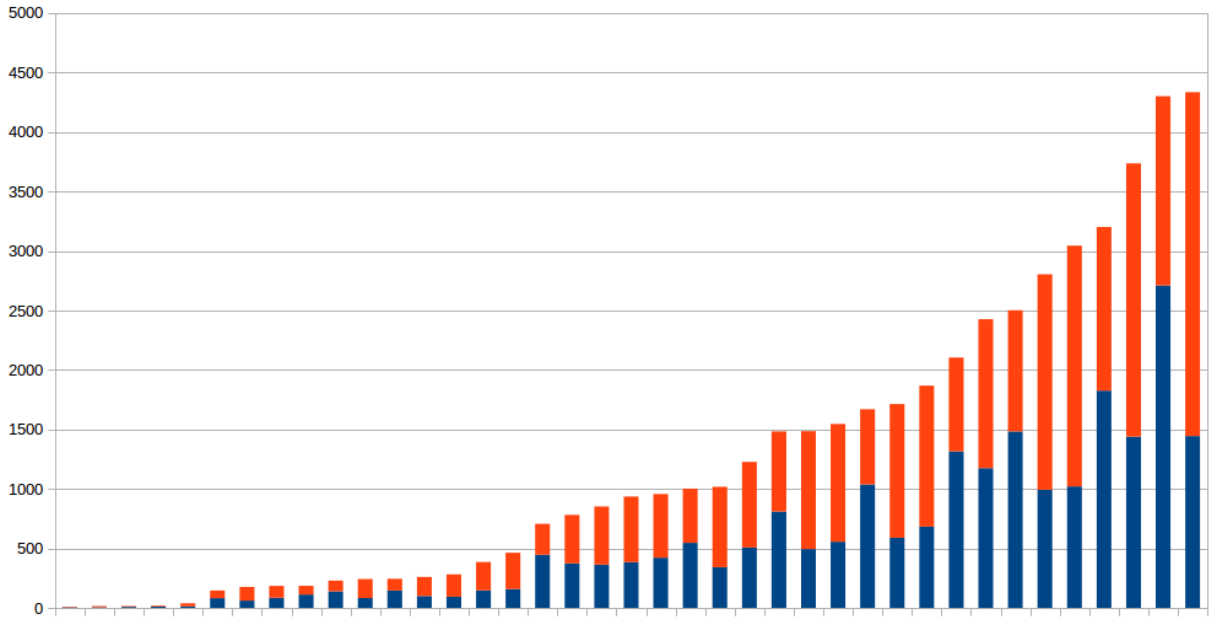


Figure 5.10: Stacked bargraphs representing the distance from the best query to the average (bottom) and the worst(top) for JabRef. Graph is ordered by distance from best to worst.

Query	CSB	ICL	LMPBV	Flat
Best	0.0359	0.0336	0.0430	0.0843
Average	0.0031	0.0029	0.0036	0.0024
Worst	0.0020	0.0018	0.0030	0.0016

Table 5.8: MRRs for choosing the best, average, and worst case for each feature for JabRef

Query	CSB	ICL	LMPBV	Flat
Title	56	51	56	38
Description	36	31	38	38
Combined	8	18	6	24

Table 5.9: Percentages for each query type where the best query was found for JabRef

The highest percentage for Title was 56% and was found for both CSB and LMPBV. In contrast, Combined was selected far less with less than 25% for each corpus and only 8% for CSB and 6% for LMPBV. The flat corpus had a more even distribution between the three corpora with both Title and Description having the best query 38% of the time and Combined having its highest percentage of the four corpora at 24%.

I ran a Friedman test with a wilcoxon post-hoc analysis to determine whether there was any significant difference between the best, average, and worst queries for each feature. The results of my analysis found that for three of the four corpora, there was significant differences between each of the three cases. The exception was found for the LMPBV corpus which found that there was a significant difference between the best and worst queries, however there was no significant difference found between the best and the average queries.

Despite finding only small differences between the MRRs of the best queries and the MRRs

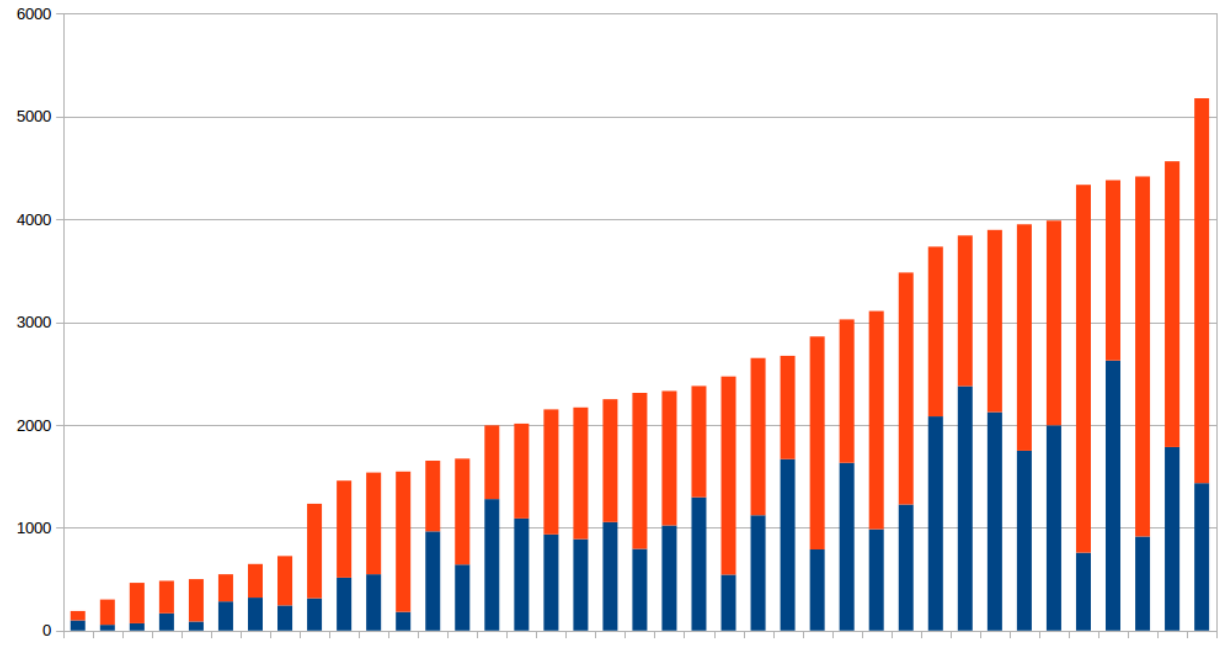


Figure 5.11: Stacked bargraphs representing the distance from the best query to the average (bottom) and the worst(top) for JabRef. Graph is ordered by distance from best to worst.

of the best query type for each corpus, there still exists the possibility that each of the four corpora perform significantly different. In this case, selecting the best queries from each corpus could lead to a significantly higher difference in the MRRs and the effectiveness measures. The distances between the best, average, and worst queries from all of the corpora can be found in Figure 5.11.

The greatest distance between the best and the worst query for all corpora is 5,167, much higher than any of the distances found for the four corpora individually. The smallest distance between the best and the worst query is 190 and the smallest distance between the best and the average query is 99. The greatest distance between the best and the average query is 2,376. The mean distance between the best and the worst queries is 2,386 which is over 1,000 more than the mean distance found when looking at the four corpora individually. Furthermore, the mean distance between the best and the average query is found to be 991 which is higher than the mean

	Best	Average	Worst
MRR	0.1288	0.0020	0.0007

Table 5.10: MRRs for choosing the best, average, and worst case for each feature from all corpora for JabRef

	CSB	ICL	LMPBV	Flat
Percentage	19	11	14	47

Table 5.11: Percentages for each corpus where the best query was found from all corpora for JabRef

	Title	Description	Combined
Percentage	57	20	23

Table 5.12: Percentages for each query type where the best query was found from all corpora for JabRef

best and worst queries, there should be a more substantial change in the MRR.

I show the the computed MRRs for the best, average, and worst case of each feature in Table 5.10. Unlike the previous MRRs choosing the best query across all query types and corpora does show a substantial change in the MRR value. Table 5.11 shows the percentage of times the best query was selected from each corpus. The flat corpus contained the highest percentage of the best queries selected. ICL contained the lowest percentage of the best queries at only 11%. The percentage of times each query type resulted in the best query is found in Table 5.12. This table shows that Title had the highest percentage of queries that were selected as the best. Unlike when looking at the corpora individually, when looking across all corpora, the Combined query type has a higher percentage than the Description query type.

Finally, I performed a Friedman test with a wilcoxon post-hoc for the best, average, and

worst case of each feature across all corpora. The results of the analysis showed that there is a significant difference between each case.

5.3.1.3 jEdit

The boxplots for jEdit can be found in Figure 5.12. For each of the four corpora, the spread for Title is lower than the other corpora with the exception of the upper 1.5 IRQ for LMPBV. The same consistency is shown for the largest spread of the four corpora. The largest spread can be found for the Description query type in each of the four corpora, with the exception of the upper 1.5 IRQ for each of the four corpora. While the median is higher for the Description query type for all four corpora, the medians between Description and Combined are close for both ICL and LMPBV.

I calculated the MRRs for each query type for all four corpora and recorded their values in Table 5.13. For each of the four corpora, the MRR for the Title query type is significantly higher than the MRR values for the Description and the Combined query types. The highest MRR is found for the flat corpus with the Title query type while the lowest MRR is found for the LMPBV corpus with a tie between the Description and the Combined corpora. For three of the four corpora, the MRR for the Description query type is lower than the MRR for the Combined query type. For each query type, the flat corpus has a higher MRR than the corresponding query type for the other three corpora.

I ran a Friedman test with a wilcoxon post-hoc analysis to determine whether there were any significant differences between the three different query types for each of the four corpora. The results of the analysis found that there were significant differences in each corpus between each query type, with the exception of the Title and Combined query types for the flat configuration.

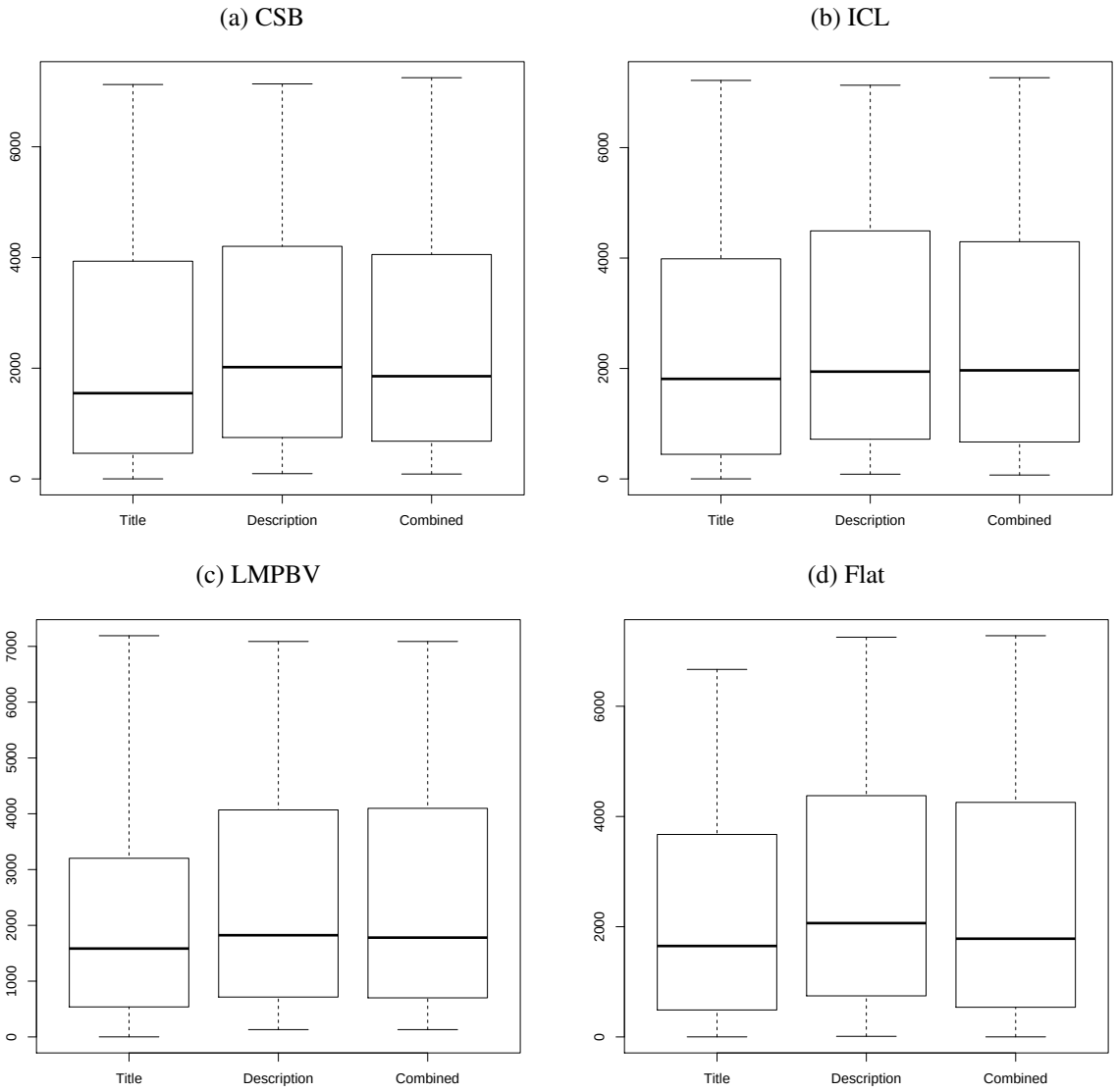


Figure 5.12: The Effectiveness Measures for the three different query types (Title, Description, Combined) for jEdit

Query	CSB	ICL	LMPBV	Flat
Title	0.0398	0.0403	0.0153	0.0579
Description	0.0012	0.0013	0.0010	0.0029
Combined	0.0013	0.0014	0.0010	0.0097

Table 5.13: MRRs for the three different query types (Title, Description, Combined) for jEdit

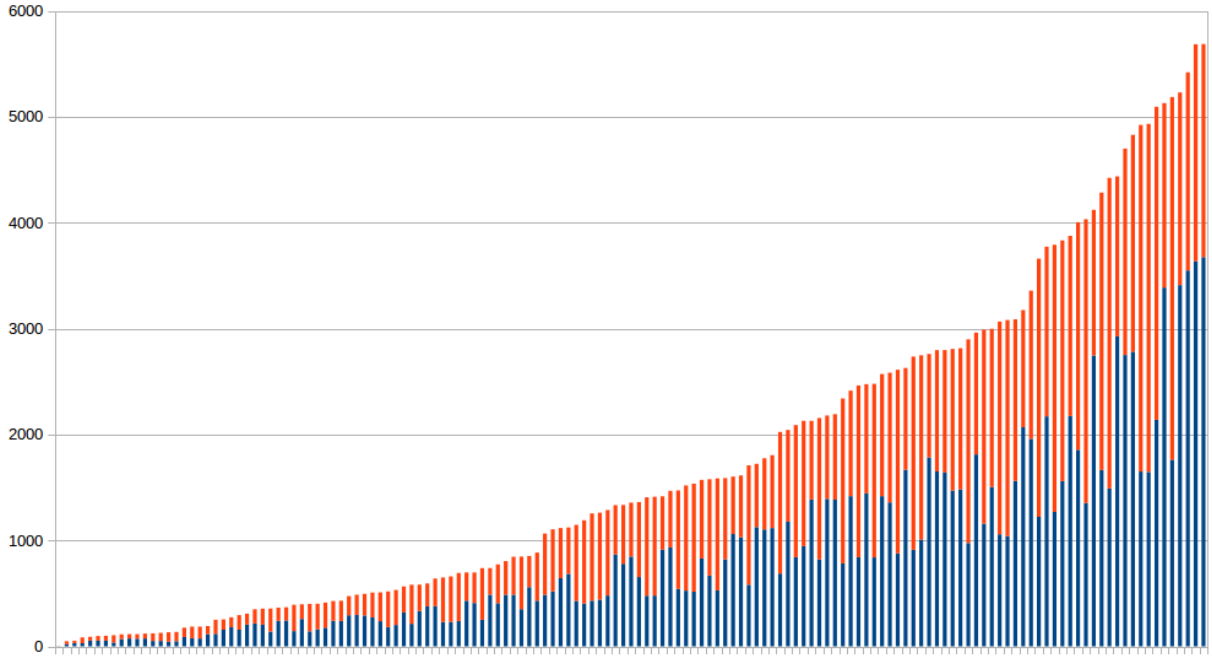
For this corpus, the upper 1.5 IRQ and the 3Q measures are lower for the two query types, however the median, lower 1.5 IRQ, and 1Q measures are similar between the two query types.

I show the differences between the best, average, and worst cases in Figure 5.13. The greatest distance between the best and worst query can be found in the ICL corpus at 6,447. The greatest mean distance between the best query and the worst query can be found in the CSB corpus at 1,734. The greatest distance between the best query and the average query is 4,018 and it is found in the ICL corpus, while the greatest mean distance between the best query and the average query is once again found in the CSB corpus at 862. The smallest mean distance between the best query and the worst query is 674 and it is found in the LMPBV corpus. The LMPBV corpus also has the smallest mean distance between the best query and the average query at 389. This corpus also has the smallest distance between the best and the worst query at 0. For three of the four corpora, the mean distance between the best and the average query is larger than the distance between the average and the worst query.

The MRRs for the best, average, and worst queries for each of the four corpora in jEdit can be found in Table 5.14. The differences between the MRRs for the best queries and the MRRs of the best query type for each corpus are small for each of the four corpora. For CSB, ICL, and LMPBV, the change in MRR is less than .0002. The largest difference is found in the flat corpus and is .0010. The smallest change is found for LMPBV at less than 0.0001. The greatest MRR between the four corpora is once again found for the flat corpus.

I calculated the percentages of times that the best query was selected from each of the three query types for each of the four corpora. The results of these calculations can be found in Table 5.17. For each of the four corpora, the Title query was the best query type over 50% of the time. The highest percentage for Title was 71% and was found for the LMPBV corpus. For each of the

(a) CSB



(b) ICL

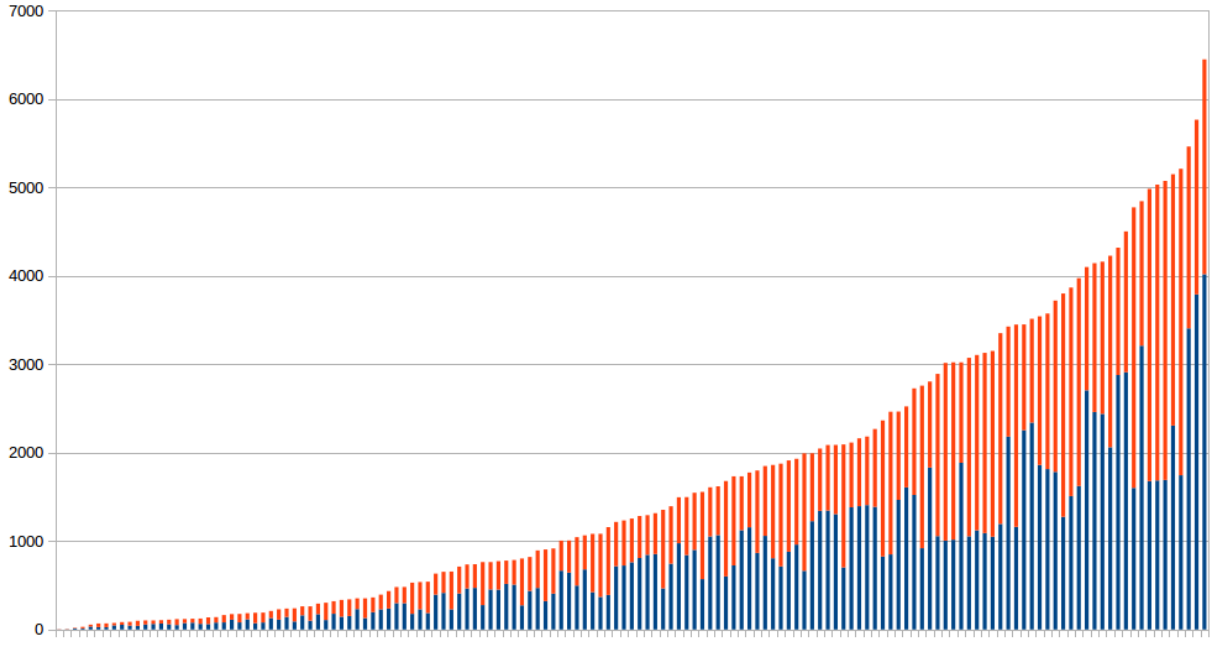
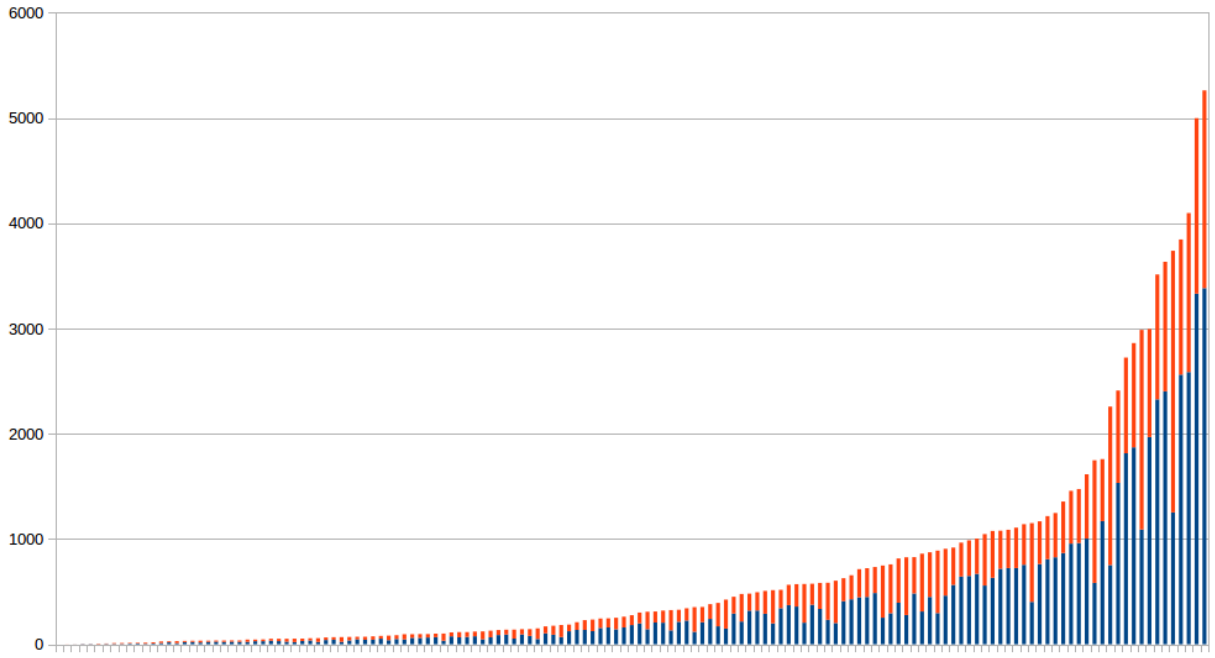


Figure 5.13: Stacked bargraphs representing the distance from the best query to the average (bottom) and the worst(top) for jEdit. Graph is ordered by distance from best to worst.

(c) LPMBV



(d) Flat

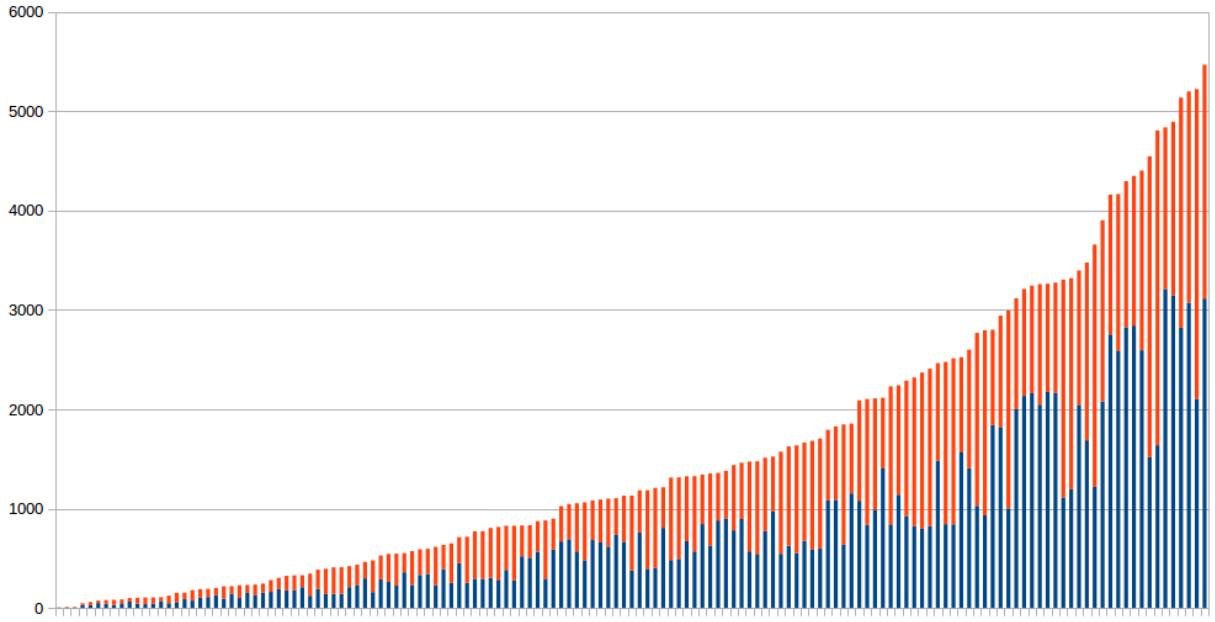


Figure 5.13: Stacked bargraphs representing the distance from the best query to the average (bottom) and the worst(top) for jEdit. Graph is ordered by distance from best to worst.

Query	CSB	ICL	LMPBV	Flat
Best	0.0400	0.0405	0.0154	0.0589
Average	0.0011	0.0013	0.0012	0.0035
Worst	0.0008	0.0009	0.0010	0.0018

Table 5.14: MRRs for choosing the best, average, and worst case for each feature for jEdit

Query	CSB	ICL	LMPBV	Flat
Title	56	56	71	57
Description	30	31	20	23
Combined	14	12	10	20

Table 5.15: Percentages for each query type where the best query was found for jEdit

four corpora, the Description query type had a higher percentage of the best queries when compared to the Combined queries. The smallest percentae for the Combined query type is for the LMPBV corpus. This corpus also has the lowest value for the Description. This is due to the high percentage of this corpus that is given to the Title query type.

I ran a Friedman test with a wilcoxon post-hoc analysis to determine whether there was any significant difference between the best, average, and worst queries for each feature. The results of my analysis found that for each of the four corpora, there was significant differences between each of the three cases.

As with JabRef, despite finding only small differences between the MRRs of the best queries and the MRRs of the best query type for three of the four corpora, there still exists the possibility that each of the four corpora perform significantly different. In this case, selecting the best queries from each corpus could lead to a significantly higher difference in the MRRs and the

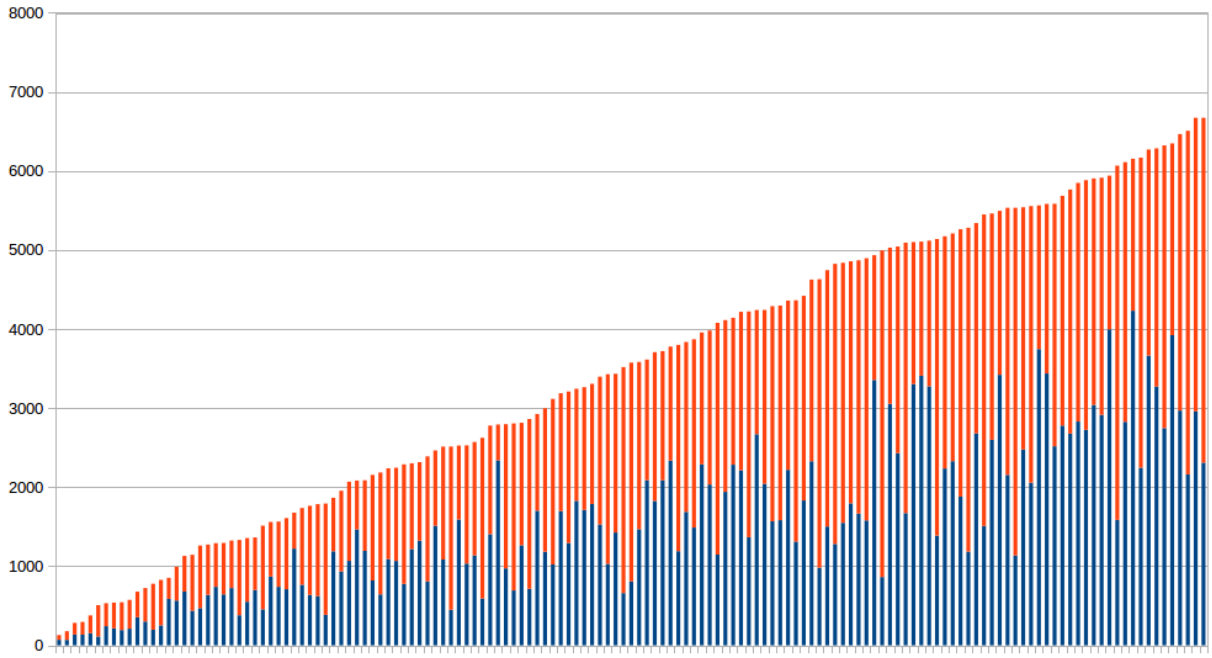


Figure 5.14: Stacked bargraphs representing the distance from the best query to the average (bottom) and the worst(top) for jEdit. Graph is ordered by distance from best to worst.

effectiveness measures. The distances between the best, average, and worst queries from all of the corpora can be found in Figure 5.14.

The greatest distance between the best and the worst query for all corpora is 6,672, only 230 higher than the greatest distance found for the four corpora individually. The smallest distance between the best and the worst query is 130 and the smallest distance between the best and the average query is 69. The greatest distance between the best and the average query is 4,233. The mean distance between the best and the worst queries is 3,491 which is almost two times the greatest distance found when looking at the four corpora individually. Furthermore, the mean distance between the best and the average query is found to be 1,561 which is higher than the mean distance found between the best and the average query of the individual corpora by 700.

I show the computed MRRs for the best, average, and worst case of each feature in Table

	Best	Average	Worst
MRR	0.0862	0.0010	0.0005

Table 5.16: MRRs for choosing the best, average, and worst case for each feature from all corpora for jEdit

	CSB	ICL	LMPBV	Flat
Percentage	20	11	18	43

Table 5.17: Percentages for each corpus where the best query was found from all corpora for jEdit

	Title	Description	Combined
Percentage	65	17	18

Table 5.18: Percentages for each query type where the best query was found from all corpora for jEdit

5.16. Again there is a substantial difference between the MRR when choosing the best query across all corpora with a difference of .0273 over the best MRR from the flat corpus. Table 5.17 shows the percentage of times the best query was selected from each corpus. The flat corpus had the highest percentage of 43%.The percentages for LMPBV and CSB are close to each other with less than a 2% difference between the two corpora. ICL had the smallest percentage of the four corpora. The percentage of times each query type resulted in the best query is found in Table 5.18. This table shows that once again Title had the highest percentage of queries that were selected as the best.

Finally, I performed a Friedman test with a wilcoxon post-hoc for the best, average, and worst case of each feature across all corpora. The results of the analysis showed that there is a significant difference between each case.

5.3.1.4 muCommander

The boxplots for muCommander can be found in Figure 5.15. For three of the four corpora, the spread for Title is smaller than the other two query types. The exception is found in the flat corpus where Title has the largest spread. For this corpus, the largest spread exists for flat corpus, however for each measure aside from the upper 1.5 IRQ and the 3Q value, the measures are lower for the Title corpus. For each of the four corpora, Description and Combined have a similar spread. When comparing the Description and Combined, the 3Q value is slightly lower in each corpus.

I calculated the MRRs for each query type for all four corpora and recorded their values in Table 5.19. The MRRs for the Title query type are significantly lower for muCommanderwhen compared to the other systems. For each of the four corpora, the MRR for the Title query type is significantly higher than the MRR values for the Description and the Combined query types. The highest MRR is found for the flat corpus with the Title query type while the lowest MRR is found for the ICL corpus. For three of the four corpora, the MRR for the Description query type is lower than the MRR for the Combined query type. For each query type, the flat corpus has a higher MRR than the corresponding query type for the other three corpora.

I ran a Friedman test with a wilcoxon post-hoc analysis to determine whether there were any significant differences between the three different query types for each of the four corpora. The results of the analysis found that there were significant differences for three of the four corpora between each query type. For the flat corpus, there were no significant difference found between any of the three query types.

I show the differences between the best, average, and worst cases in Figure 5.16. The

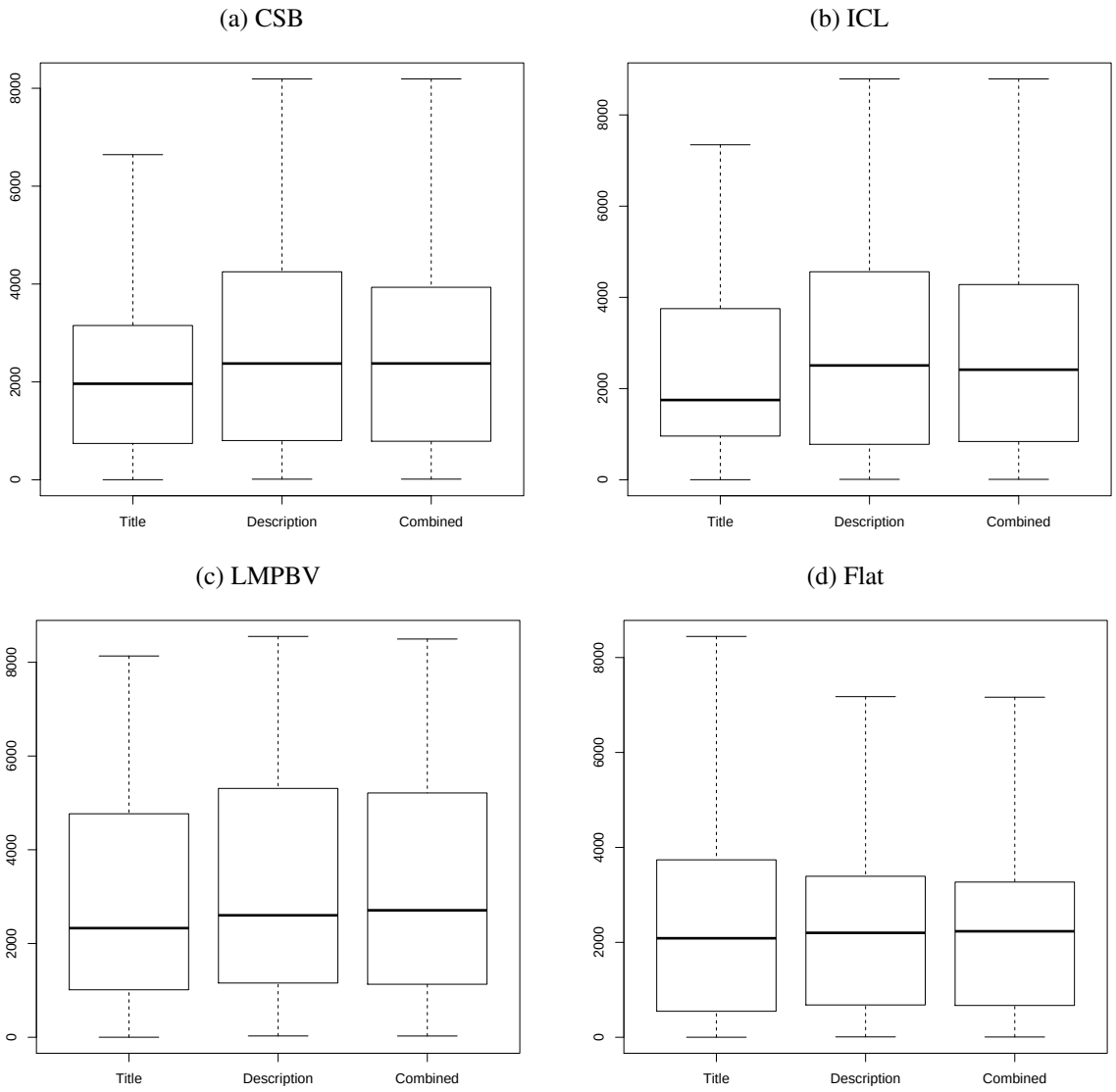


Figure 5.15: The Effectiveness Measures for the three different query types (Title, Description, Combined) for muCommander

Query	CSB	ICL	LMPBV	Flat
Title	0.0113	0.0103	0.0179	0.0237
Description	0.0021	0.0022	0.0011	0.0046
Combined	0.0024	0.0023	0.0012	0.0042

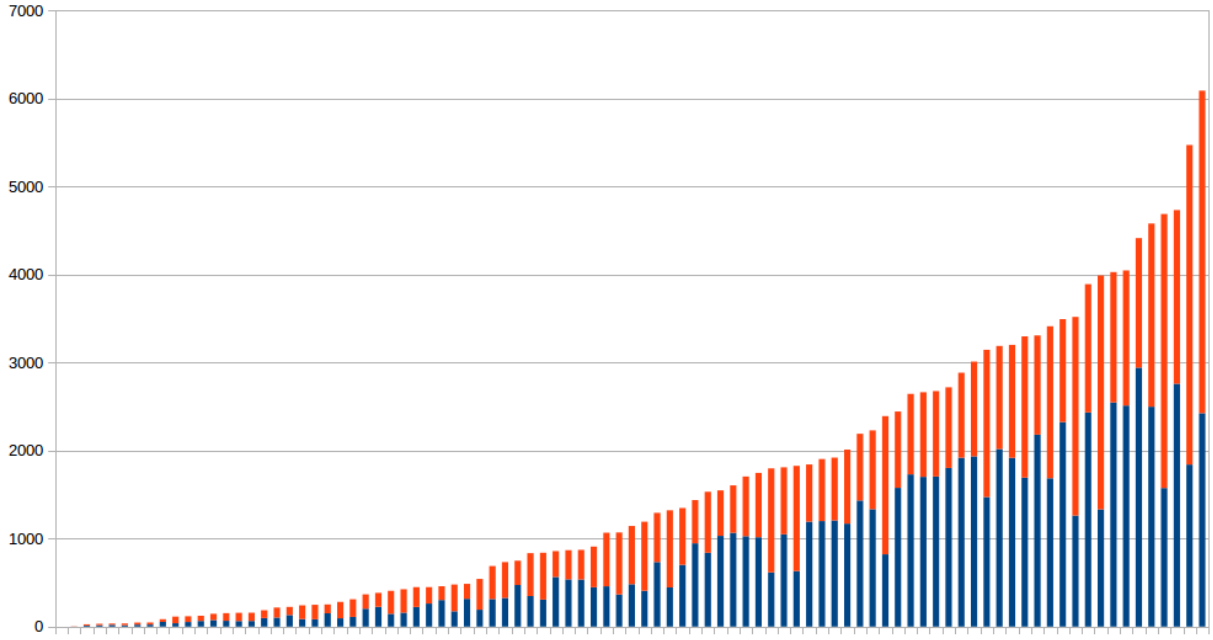
Table 5.19: MRRs for the three different query types (Title, Description, Combined) for muCommander

greatest distance between the best and worst query can be found in the flat corpus at 7,784. The greatest mean distance between the best query and the worst query can be found in the CSB corpus at 1,587. The greatest distance between the best query and the average query is 4,411 and it is found in the flat corpus, while the greatest mean distance between the best query and the average query is found in the ICL corpus at 850. The smallest mean distance between the best query and the worst query is 838 and it is found in the LMPBV corpus. The LMPBV corpus also has the smallest mean distance between the best query and the average query at 478. The LMPBV corpus has the smallest distance between the best and the worst query at 0. For each of the four corpora, the mean distance between the best and the average query is larger than the distance between the average and the worst query.

The MRRs for the best, average, and worst queries for each of the four corpora in muCommander can be found in Table 5.20. Once again, the differences between the MRRs for the best queries and the MRRs of the best query type for each corpus are small at less than or equal to .0025 for each of the four corpora. The largest difference is found in the flat corpus and is .0025. The smallest change is found for LMPBV at less than 0.0002. The greatest MRR between the four corpora is once again found for the flat corpus at .0262. The flat corpus also has the highest values for the average and worst MRRs, both of which are over .0020. Neither the average or the the worst is higher than .0020 for any other corpus.

I calculated the percentages of times that the best query was selected from each of the three query types for each of the four corpora. The results of these calculations can be found in Table 5.23. For each of the four corpora, the Title query was the best query type over 47% of the time. The highest percentage for Title was 64% and was found for the LMPBV corpus. For each of the four corpora, the percentages for Description are higher than the percentages for Combined. In

(a) CSB



(b) ICL

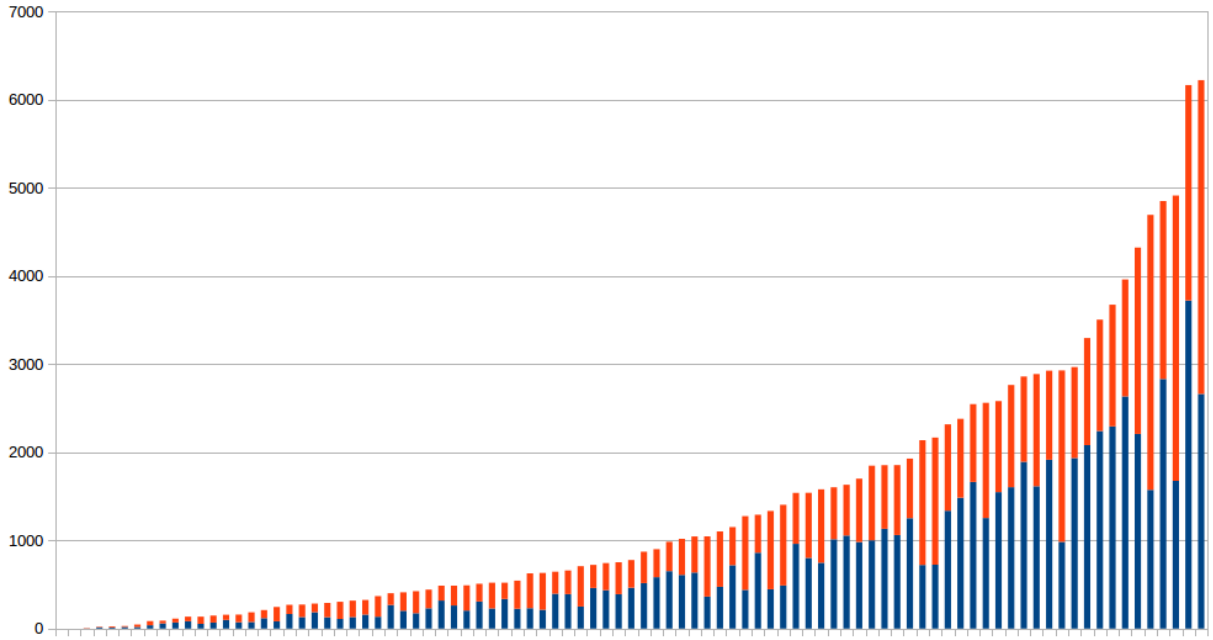
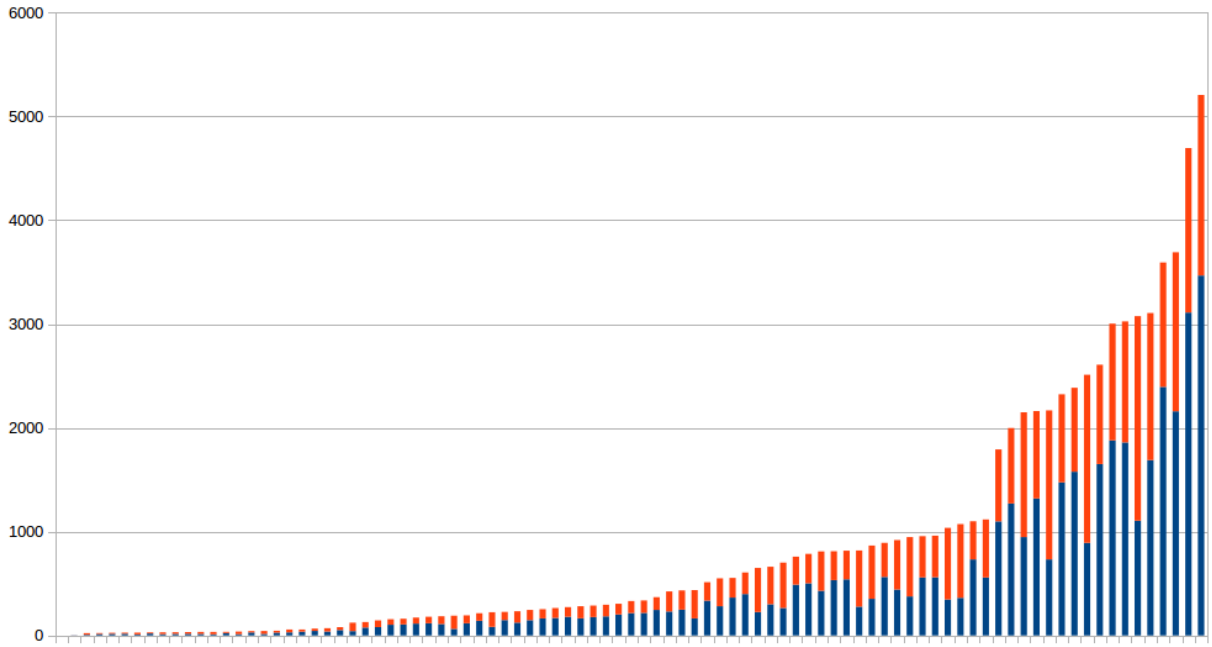


Figure 5.16: Stacked bargraphs representing the distance from the best query to the average (bottom) and the worst(top) for muCommander. Graph is ordered by distance from best to worst.

(c) LPMBV



(d) Flat

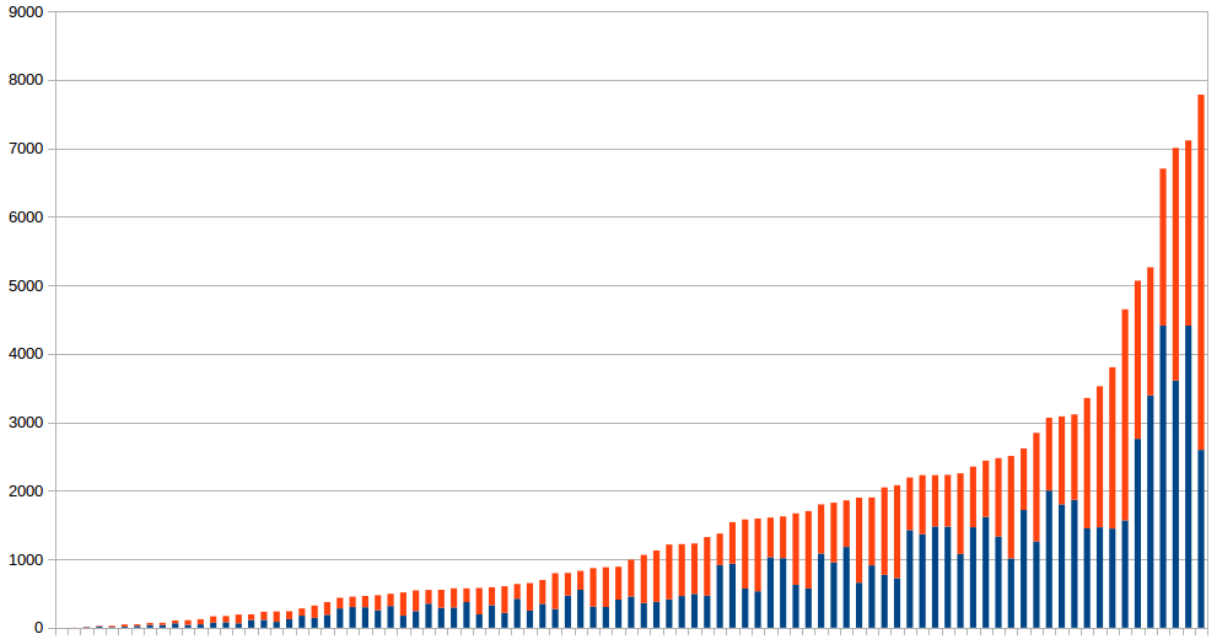


Figure 5.16: Stacked bargraphs representing the distance from the best query to the average (bottom) and the worst(top) for muCommander. Graph is ordered by distance from best to worst.

Query	CSB	ICL	LMPBV	Flat
Best	0.0121	0.0110	0.0180	0.0262
Average	0.0019	0.0019	0.0015	0.0029
Worst	0.0014	0.0014	0.0012	0.0022

Table 5.20: MRRs for choosing the best, average, and worst case for each feature for muCommander

Query	CSB	ICL	LMPBV	Flat
Title	58	62	64	47
Description	29	22	25	31
Combined	13	16	11	22

Table 5.21: Percentages for each query type where the best query was found for muCommander

each corpus, the percentages for Combined are less than 25% while for the Description query type, the percentages are above 20%. The highest percentage for the Description and for the Combined query types are found in the flat corpus, while this corpus has the lowest percentage for Title.

I ran a Friedman test with a wilcoxon post-hoc analysis to determine whether there was any significant difference between the best, average, and worst queries for each feature. The results of my analysis found that for each of the four corpora, there was significant differences between each of the three cases.

As with jEdit, despite finding only small differences between the MRRs of the best queries and the MRRs of the best query type for each corpus, there still exists the possibility that each of the four corpora perform significantly different. In this case, selecting the best queries from each corpus could lead to a significantly higher difference in the MRRs and the effectiveness measures.

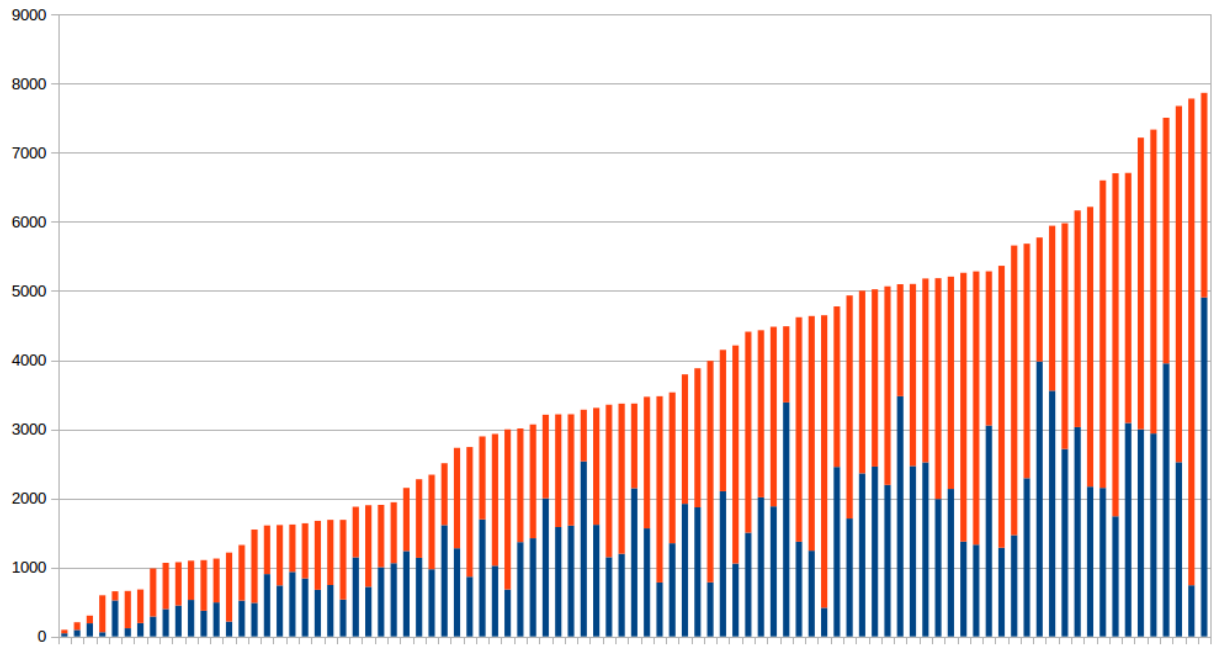


Figure 5.17: Stacked bargraphs representing the distance from the best query to the average (bottom) and the worst(top) for muCommander. Graph is ordered by distance from best to worst.

The distances between the best, average, and worst queries from all of the corpora can be found in Figure 5.17.

The greatest distance between the best and the worst query for all corpora is 7,866, slightly higher than any of the distances found for the four corpora individually. However, the difference between the greatest distance found overall and the greatest distance found by the individual corpora is only 82. The smallest distance between the best and the worst query is 97 and the smallest distance between the best and the average query is 45. Both of these values are higher than the values found for the corpora individually. The greatest distance between the best and the average query is 4,906, 495 highest than the value found for the corpora individually. The mean distance between the best and the worst queries is 3,622 which is over 2,000 more than the mean distance found when looking at the four corpora individually. Furthermore, the mean distance between

	Best	Average	Worst
MRR	0.0374	0.0010	0.0004

Table 5.22: MRRs for choosing the best, average, and worst case for each feature from all corpora for muCommander

	CSB	ICL	LMPBV	Flat
Percentage	16	10	16	48

Table 5.23: Percentages for each corpus where the best query was found from all corpora for muCommander

	Title	Description	Combined
Percentage	63	18	19

Table 5.24: Percentages for each query type where the best query was found from all corpora for muCommander

the best and the average query is found to be 1,534 which is higher than the mean distance when looking at the corpora individually.

I show the the computed MRRs for the best, average, and worst case of each feature in Table 5.22. muCommander shows the lowest MRR of any of the four subject systems, however choosing the best query across across all query types and corpora still shows a substantial change in the MRR value. Table 5.23 shows the percentage of times the best query was selected from each corpus. The flat corpus contained the highest percentage of the best queries selected. ICL contained the lowest percentage of the best queries at only 10%. The remaining two corpora, CSB and LMPBV, have the same percentage of best queries at 16% each. The percentage of times each query type resulted in the best query is found in Table 5.24. Similar to previous systems, this table shows that Title had the highest percentage of the best queries followed by Combined.

Finally, I performed a Friedman test with a wilcoxon post-hoc for the best, average, and worst case of each feature across all corpora. The results of the analysis showed that there is a significant difference between each case.

5.3.1.5 All Systems

The boxplots for all systems combined can be found in Figure 5.18. For three of the four corpora, the spread for Title is smaller than the other two query types. The exception is found in the flat corpus where Title has a larger spread than the Combined query type. However, for this corpus as with the other three corpora, Title has the lowest 1Q, median, and 3Q measures. For each of the four corpora, Description has the largest spread but for both ICL and LMPBV, this spread is close to the spread for Combined. ICL is the only corpus where the median measure for Combined is higher than for Description.

I calculated the MRRs for each query type for all four corpora and recorded their values in Table 5.25. For each of the four corpora, the MRR for the Title query type is significantly higher than the MRR values for the Description and the Combined query types. The highest MRR is found for the flat corpus with the Title query type while the lowest MRR is found for the LMPBV corpus with a tie between the Description and Combined query types. With the exception of LMPBV, the MRR for the Description query type is lower than the MRR for Combined. For both the Title and Combined query types, the flat corpus has a higher MRR than for the other three corpora. For Description, the ICL corpus has the highest MRR followed closely by the CSB corpus. For ICL, the difference between the Description query type and the Combined query type is less than .0001.

I ran a Friedman test with a wilcoxon post-hoc analysis to determine whether there were any significant differences between the three different query types for each of the four corpora. The results of the analysis found that there were significant differences for three of the four corpora

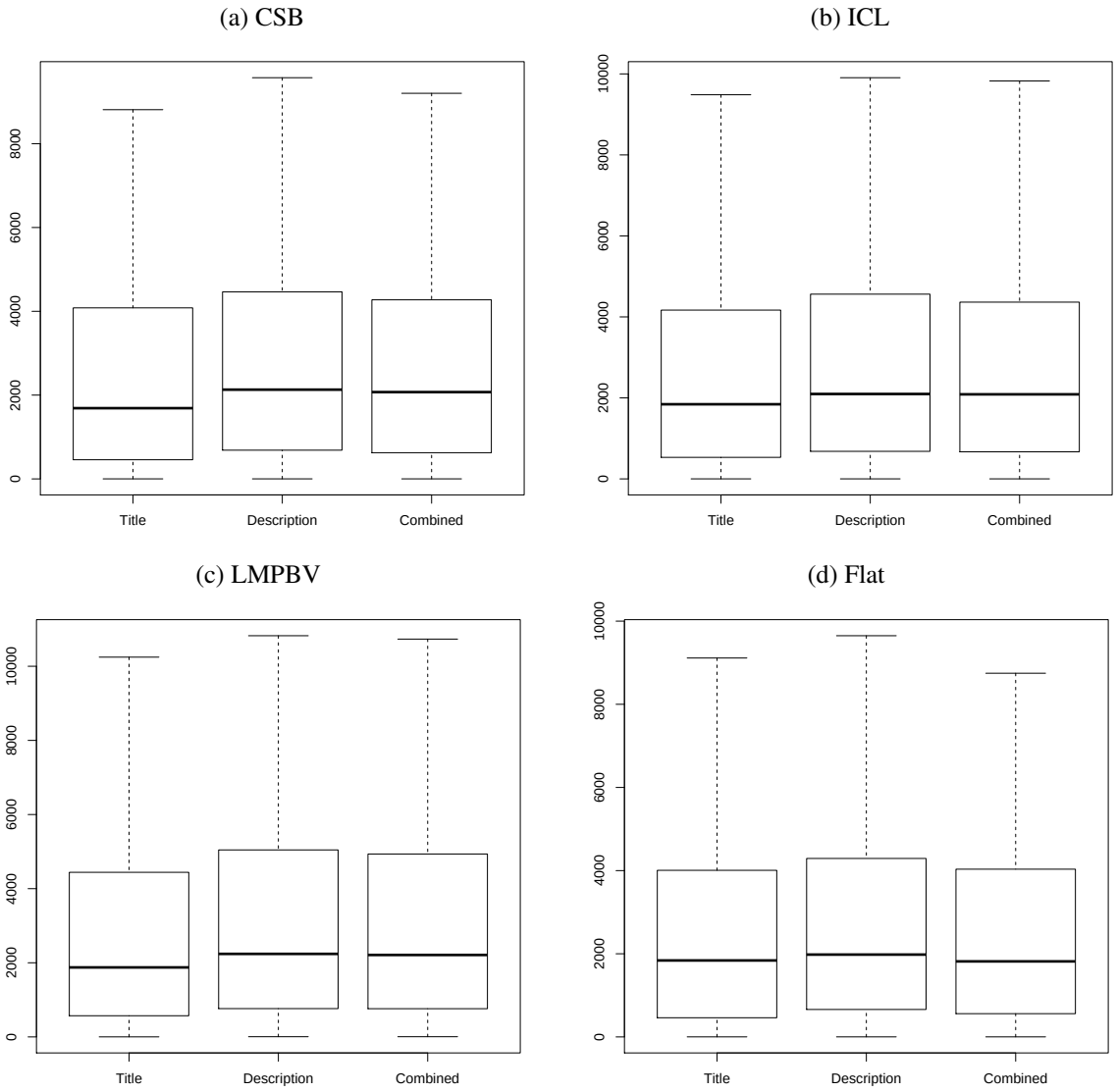


Figure 5.18: The Effectiveness Measures for the three different query types (Title, Description, Combined) for all systems

Query	CSB	ICL	LMPBV	Flat
Title	0.0395	0.0316	0.0176	0.0531
Description	0.0082	0.0085	0.0029	0.0064
Combined	0.0107	0.0086	0.0029	0.0119

Table 5.25: MRRs for the three different query types (Title, Description, Combined) for all systems

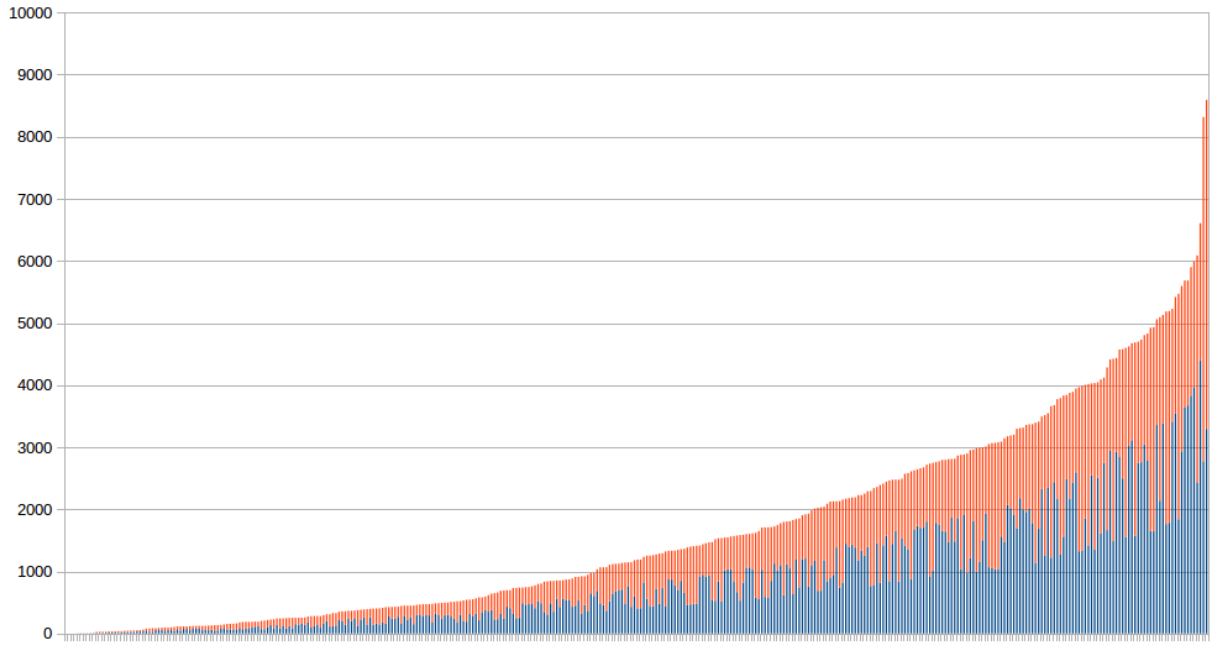
between each query type. For the flat corpus, there was a significant difference found between the Title and the Description query types.

I show the differences between the best, average, and worst cases in Figure 5.19. The greatest distance between the best and worst query can be found in the ICL corpus at 10,470. The greatest mean distance between the best query and the worst query can also be found in the ICL corpus at 1,672. The greatest distance between the best query and the average query is 6,971 and it is found in the ICL corpus, while the greatest mean distance between the best query and the average query is found in the CSB corpus at 859. The smallest mean distance between the best query and the worst query is 728 and it is found in the LMPBV corpus. The LMPBV corpus also has the smallest mean distance between the best query and the average query at 402. The LMPBV corpus has the smallest distance between the best and the worst query at 0. For three of the four corpora, the mean distance between the best and the average query is larger than the distance between the average and the worst query.

The MRRs for the best, average, and worst queries for each of the four corpora in all systems combined can be found in Table 5.26. Like the systems alone, the differences between the best queries and the highest query types are small. The largest difference is found in the flat corpus and is .0039. The smallest change is found for LMPBV at less than 0.0002. The greatest MRR between the four corpora is found for the flat corpus, while the smallest MRR is found for LMPBV.

I calculated the percentages of times that the best query was selected from each of the three query types for each of the four corpora. The results of these calculations can be found in Table 5.29. For each of the four corpora, the Title query was the best query type over 50% of the time. The highest percentage for Title was 64% and was found for the LMPBV corpus. For each of

(a) CSB



(b) ICL

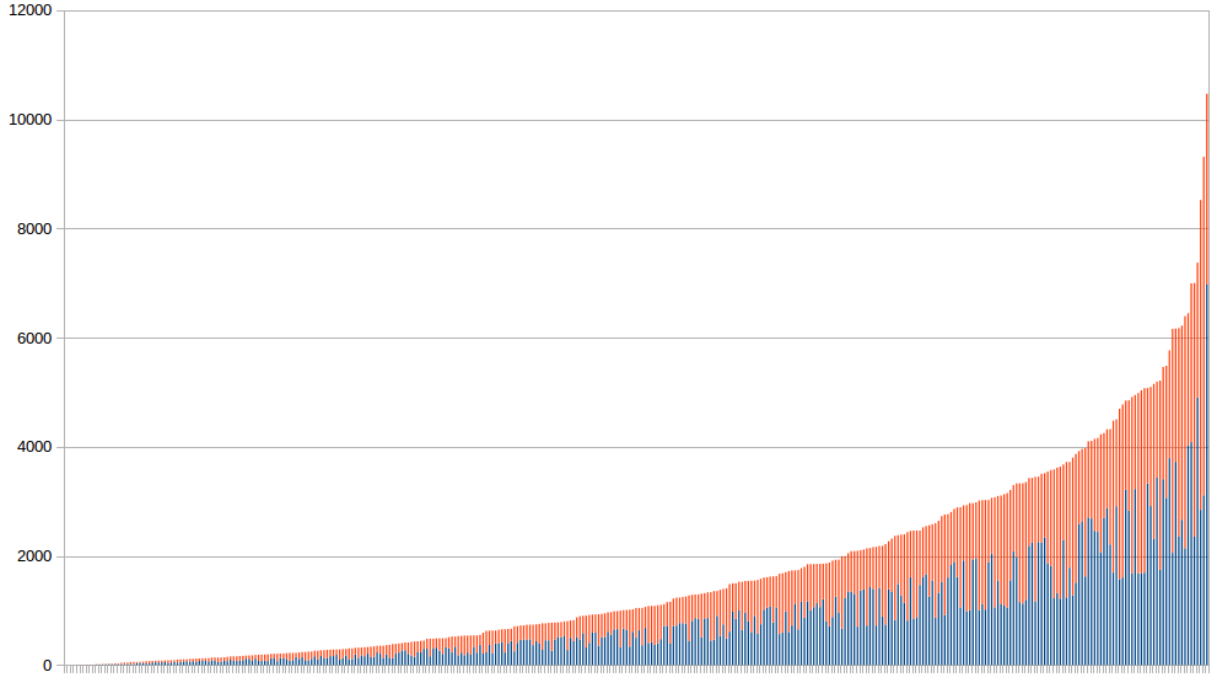
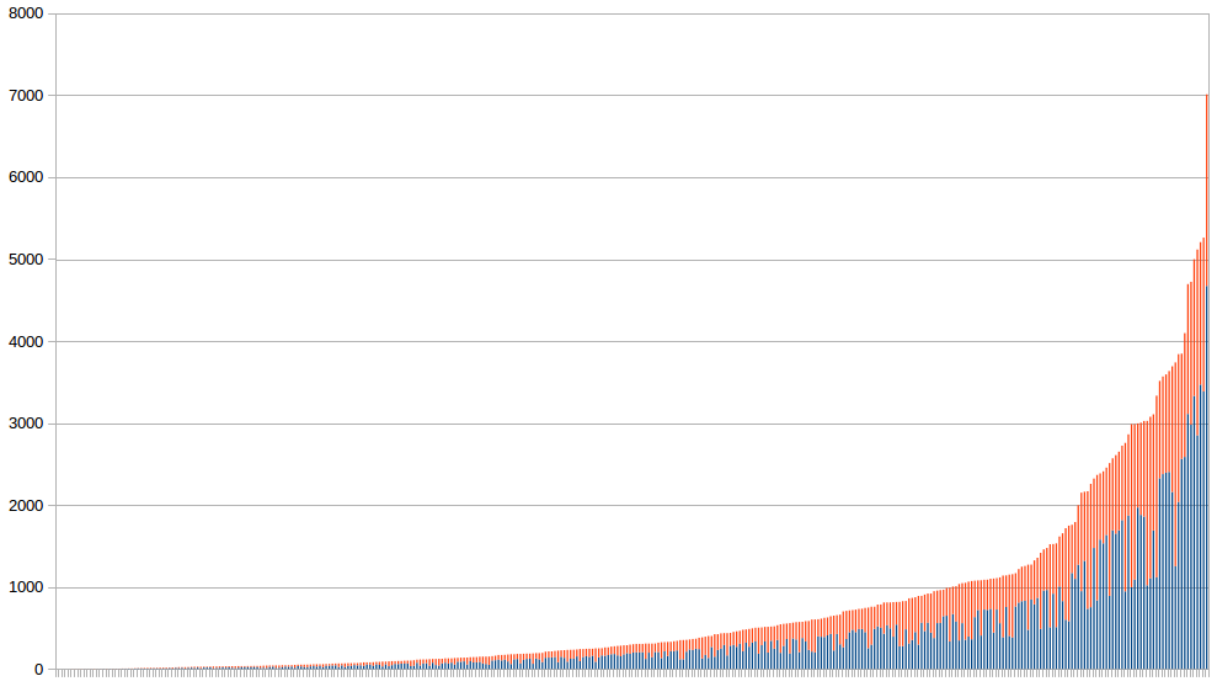


Figure 5.19: Stacked bargraphs representing the distance from the best query to the average (bottom) and the worst(top) for all systems. Graph is ordered by distance from best to worst.

(c) LPMBV



(d) Flat

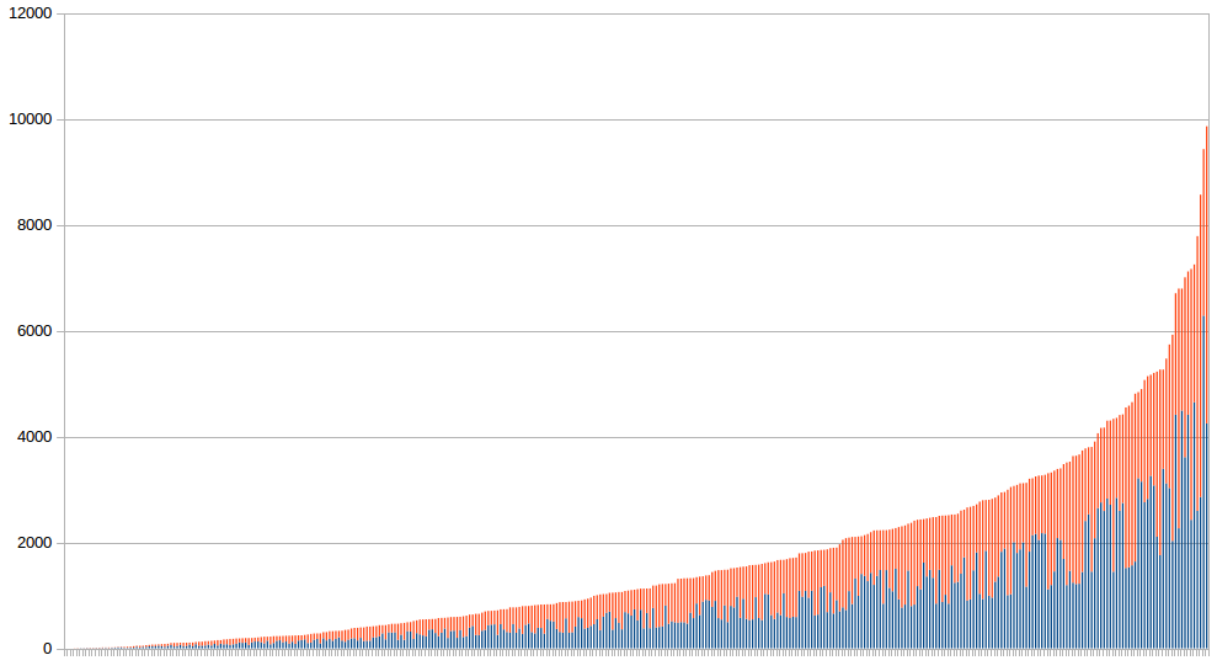


Figure 5.19: Stacked bargraphs representing the distance from the best query to the average (bottom) and the worst(top) for all systems. Graph is ordered by distance from best to worst.

Query	CSB	ICL	LMPBV	Flat
Best	0.0403	0.0328	0.0178	0.0570
Average	0.0087	0.0079	0.0029	0.0047
Worst	0.0073	0.0072	0.0025	0.0024

Table 5.26: MRRs for choosing the best, average, and worst case for each feature for all systems

Query	CSB	ICL	LMPBV	Flat
Title	58	57	64	52
Description	27	26	27	26
Combined	15	16	9	21

Table 5.27: Percentages for each query type where the best query was found for all systems

the four corpora, the percentage for Description was higher than the percentage for Combined. The highest percentage for Combined was 21% for the flat corpus, while the lowest percentage for Description was 26% for both the flat corpus and ICL. The lowest percentage of any query type is for Combined in the LMPBV corpus. This is the only percentage that drops below 10%.

I ran a Friedman test with a wilcoxon post-hoc analysis to determine whether there was any significant difference between the best, average, and worst queries for each feature. The results of my analysis found that for each of the four corpora, there was significant differences between each of the three cases.

As stated previously, the four corpora can act significantly different and this can lead to improvements in the overall results of the MRRs. The distances between the best, average, and worst queries from all of the corpora can be found in Figure 5.20.

The greatest distance between the best and the worst query for all corpora is 10,738, higher

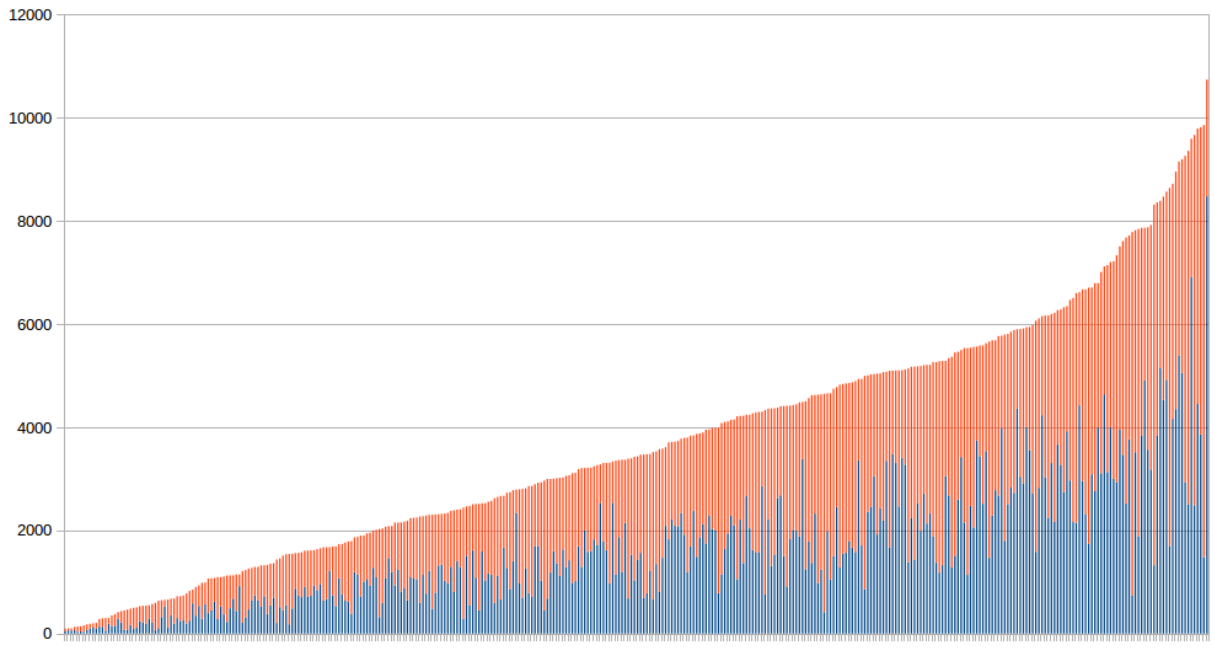


Figure 5.20: Stacked bargraphs representing the distance from the best query to the average (bottom) and the worst(top) for all systems. Graph is ordered by distance from best to worst.

than any of the distances found for the four corpora individually. However, this value is only 268 higher than the highest found for ICL. The smallest distance between the best and the worst query is 97 and the smallest distance between the best and the average query is 45. The greatest distance between the best and the average query is 8,477. The mean distance between the best and the worst queries is 3,707 which is over 2,000 more than the mean distance found when looking at the four corpora individually. Furthermore, the mean distance between the best and the average query is found to be 1,631 which is higher than the mean distance when looking at the corpora individually and only 41 less than the mean distance between the best query and the worst query when looking at the corpora individually. Overall, the mean distance between the best query and the average query is closer than the mean distance between the best query and the worst query.

I show the the computed MRRs for the best, average, and worst case of each feature in

	Best	Average	Worst
MRR	0.0793	0.0013	0.0005

Table 5.28: MRRs for choosing the best, average, and worst case for each feature from all corpora for all systems

	CSB	ICL	LMPBV	Flat
Percentage	18	11	16	48

Table 5.29: Percentages for each corpus where the best query was found from all corpora for all systems

	Title	Description	Combined
Percentage	62	17	21

Table 5.30: Percentages for each query type where the best query was found from all corpora for all systems

Table 5.28. There is a substantial increase in the MRR when compared to the MRRs of looking at the corpora individually. The difference between the best and the best for each query type is .0223. Table 5.29 shows the percentage of times the best query was selected from each corpus. Only 7% of the queries resulted in a tie between the corpora. The highest percentage for any of the corpora was found for the flat corpus, while the lowest percentage was found for the ICL corpus. The difference between the CSB and the LMPBV corpora is only 2%. The percentage of times each query type resulted in the best query is found in Table 5.30. Title had the highest percentage of the best queries at 62%, while Combined came in second with only 21% of the best queries.

Finally, I performed a Friedman test with a wilcoxon post-hoc for the best, average, and worst case of each feature across all corpora. The results of the analysis showed that there is a significant difference between each case.

5.3.2 Does changing the combination of included fields affect the accuracy of a structured retrieval-based FLT?

The purpose of this question is to address how using different combinations of the structural lexicons in a corpus may lead to differences in the MRRs for each corpus and overall. For this problem, I used both the Title and the Combined query types as they make up the shortest and the longest queries of the three types. Because there are no combinations for the flat corpus, I do not repeat the corpus's MRRs in this section, however I will use the flat corpus when I compute the best MRR overall.

5.3.2.1 ArgoUML

Due to the number of combinations in LMPBV, instead of using boxplots, I present the results of the MRR computation along with the spread in Table 5.31 for the Combined query type and in Table 5.32 for the Title query type. I list the results for ICL and CSB in similar tables.

Looking at the results of the Combined query for ArgoUML and the LMPBV corpus, there are two combinations tied for having the highest MRR. These combinations are the leading comments alone (L) and the leading comments combined with the method names (LM). In neither case do either of the combinations have the smallest spreads of effectiveness measures. However, LM does have the smallest median value of all the combinations, while L has the lowest minimum rank. The combination with the lowest MRR is the method names combined with the local variables (MV). This combination has the highest minimum rank out of all combinations. Interestingly, when looking at the results of the Title query type, the combination with the highest MRR

Combination	min	1Q	median	3Q	max	MRR
V	4	708	2942	6862	11006	0.0071
B	4	732	2921	7030	11695	0.0095
BV	4	735	2922	6997	11694	0.0081
P	5	633	2955	7055	11072	0.0076
PV	5	672	2941	7102	11078	0.0076
PB	5	711	2970	7061	11692	0.0075
PBV	5	712	2936	7062	11695	0.0075
M	5	782	2897	6884	10739	0.0046
MV	8	747	2913	7018	10783	0.0038
MB	4	736	2893	6881	10571	0.0074
MBV	4	753	2909	7177	11969	0.0065
MP	5	714	2918	6866	10780	0.0074
MPV	5	713	2914	7056	10796	0.0074
MPB	5	713	2929	7056	10642	0.0074
MPBV	5	712	2909	6910	10714	0.0074
L	2	698	2902	6995	10949	0.0109
LV	3	714	2965	7058	10867	0.0092
LB	4	683	2912	6937	11087	0.0080
LBV	4	700	3000	7054	11955	0.0080
LP	3	693	3004	6866	10867	0.0093
LPV	4	709	3006	7063	10926	0.0082
LPB	4	683	3016	6999	11459	0.0082
LPBV	5	698	3028	6874	11592	0.0073
LM	2	717	2869	7052	10848	0.0109
LMV	4	738	2944	7148	11983	0.0079
LMB	4	708	2893	7543	10590	0.0078
LMBV	4	735	3041	7136	11921	0.0078
LMP	4	697	2981	6734	10792	0.0080
LMPV	5	712	2975	6910	10802	0.0074
LMPB	4	686	2963	7172	10646	0.0079
LMPBV	5	710	3002	7002	10733	0.0072

Table 5.31: MRRs and descriptive statistics of the effectiveness measures for the LMPBV corpus and Combined query type for ArgoUML

Combination	min	1Q	median	3Q	max	MRR
V	1	673	2909	6702	11836	0.0169
B	1	674	2899	6554	11598	0.0183
BV	2	715	3074	6667	11759	0.0096
P	1	555	2860	6878	11633	0.0246
PV	1	604	2940	6602	11920	0.0182
PB	2	617	2964	6859	11302	0.0127
PBV	2	657	2923	6918	10796	0.0118
M	7	747	2834	6397	11986	0.0028
MV	9	706	2937	6169	11665	0.0029
MB	2	731	2889	6639	11815	0.0095
MBV	2	757	2900	6378	11690	0.0086
MP	5	688	2994	6284	11987	0.0045
MPV	5	689	2967	6177	11998	0.0049
MPB	2	626	2944	6340	11272	0.0118
MPBV	2	696	3052	6492	11934	0.0112
L	1	722	3047	7564	10907	0.0134
LV	1	734	3143	6683	11878	0.0134
LB	1	636	3043	7625	11335	0.0150
LBV	2	665	3332	6783	11647	0.0095
LP	1	640	3108	6912	10918	0.0139
LPV	2	623	3070	6212	11783	0.0079
LPB	2	591	3062	7156	11973	0.0098
LPBV	2	583	3036	6838	10600	0.0099
LM	1	638	2935	6437	11976	0.0134
LMV	2	607	3143	6331	11580	0.0079
LMB	1	533	2897	7238	11782	0.0150
LMBV	2	557	3139	6764	11445	0.0097
LMP	1	570	3013	6141	11144	0.0136
LMPV	2	597	2956	6393	11873	0.0079
LMPB	2	553	3040	6455	11571	0.0101
LMPBV	2	578	3140	6613	11977	0.0098

Table 5.32: MRRs and descriptive statistics of the effectiveness measures for the LMPBV corpus and Title query type for ArgoUML

Combination	min	1Q	median	3Q	max	MRR
B	1	456	2126	5235	11907	0.0256
S	3	664	2659	6139	10816	0.0080
SB	1	467	2136	5881	11848	0.0245
C	2	676	3134	6414	11794	0.0155
CB	1	575	2072	5364	11930	0.0386
CS	1	749	2259	6879	11643	0.0191
CSB	1	542	2416	5994	11745	0.0375

Table 5.33: MRRs and descriptive statistics of the effectiveness measures for the CSB corpus and Combined query type for ArgoUML

Combination	min	1Q	median	3Q	max	MRR
B	1	570	2148	5612	11868	0.0505
S	5	494	2548	5116	11851	0.0074
SB	1	380	1797	5389	11151	0.0470
C	1	490	3457	6821	11230	0.0413
CB	1	639	3309	6420	11818	0.0575
CS	1	413	3404	6387	11604	0.0419
CSB	1	383	2572	6207	11593	0.0688

Table 5.34: MRRs and descriptive statistics of the effectiveness measures for the CSB corpus and Title query type for ArgoUML

is actually the parameters alone (P). This combination is tied with multiple other combinations for having the lowest minimum rank. The combination with the lowest MRR is the method name by themselves (M). This combination does not have the highest value for any of the descriptive statistics.

The results of the experiment for the CSB corpus can be found in Tables 5.33 and 5.34. For the Combined query type, the combination with the highest MRR is comments combined with

Combination	min	1Q	median	3Q	max	MRR
L	1	725	3014	6533	11732	0.0263
C	2	676	3134	6414	11794	0.0155
CL	1	586	2819	5634	11772	0.0252
I	1	730	2864	5507	11940	0.0155
IL	1	680	2799	7286	11863	0.0208
IC	1	706	2162	5825	11999	0.0329
ICL	1	594	2482	5977	11384	0.0284

Table 5.35: MRRs and descriptive statistics of the effectiveness measures for the ICL corpus and Combined query type for ArgoUML

Combination	min	1Q	median	3Q	max	MRR
L	1	436	2132	6248	11321	0.0703
C	1	490	3457	6821	11230	0.0413
C	1	612	2272	5674	11367	0.0264
CL	1	457	2414	5456	11744	0.0410
I	1	612	2272	5674	11367	0.0264
IL	1	466	3488	6096	11908	0.0497
IC	1	717	3289	6291	11484	0.0381
ICL	1	560	3021	6143	11211	0.0387

Table 5.36: MRRs and descriptive statistics of the effectiveness measures for the ICL corpus and Title query type for ArgoUML

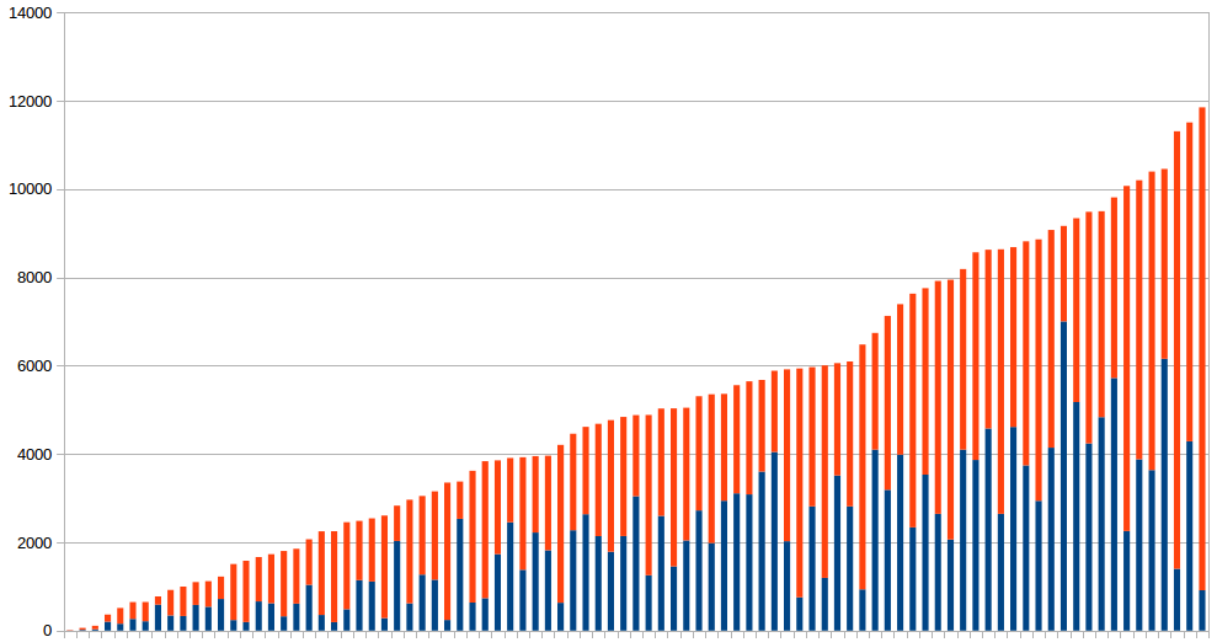
the body (CB). The MRR of this combination is followed closely by the entire corpus (CSB). CSB and CB are tied with three other combinations for having the lowest minimum rank, while CB also has the lowest median of all combinations. The combination with the lowest MRR is the signature alone (S). In the case of the Title query type, the CSB and CB have the highest MRRs once again, however in this case, the MRR for CSB is higher than the MRR for CB. Neither of these combinations have the lowest of any of the descriptive statistics. The lowest MRR is once

again for S. This combination is .0339 lowest than the next lowest combination. This combination has the highest minimum rank of any of the combinations.

In Tables 5.35 and 5.36, we can find the results of the experiment on the ICL corpus. For the Combined query type, the combination with the highest MRR is the identifiers combined with the comments (IC). This combination has the lowest median value of the combinations. Interestingly, there is a tie for the combination with the lowest MRR between the comments alone (C) and the identifiers alone (I). I has the highest 1Q value of any of the combinations while C has the highest median value. The Title query type shows literals alone (L) as having the highest MRR. This combination has the lowest 1Q and median values of any of the combinations. Another interesting find is that the combination with the lowest MRR when using the Title query is found to be the IC combination. In this case, the combination has the highest 1Q value.

I performed a Friedman test with a wilcoxon post-hoc for each of the corpora between each of the different structural combinations. This analysis was performed for both the Combined and the Title query types. For the Combined query type, a significant difference was found in both the ICL and the LMPBV corpora, however there was no significant difference found in the CSB corpus. For the LMPBV corpus, 25 pairs were found to have significant differences. For the ICL corpus, there were 6 significant differences found with significant difference found between L, IL, IC, and ICL. For the Title query type, more significant differences were found for CSB and LMPBV. For CSB, 9 significant differences were found. For the LMPBV corpus, 148 significant differences were found. ICL had a lower number of significant differences found with only 4. These significant differences are not as important for the individual combinations involved, but for understanding any improvements that will be seen when looking at choosing the best queries.

(a) CSB



(b) ICL

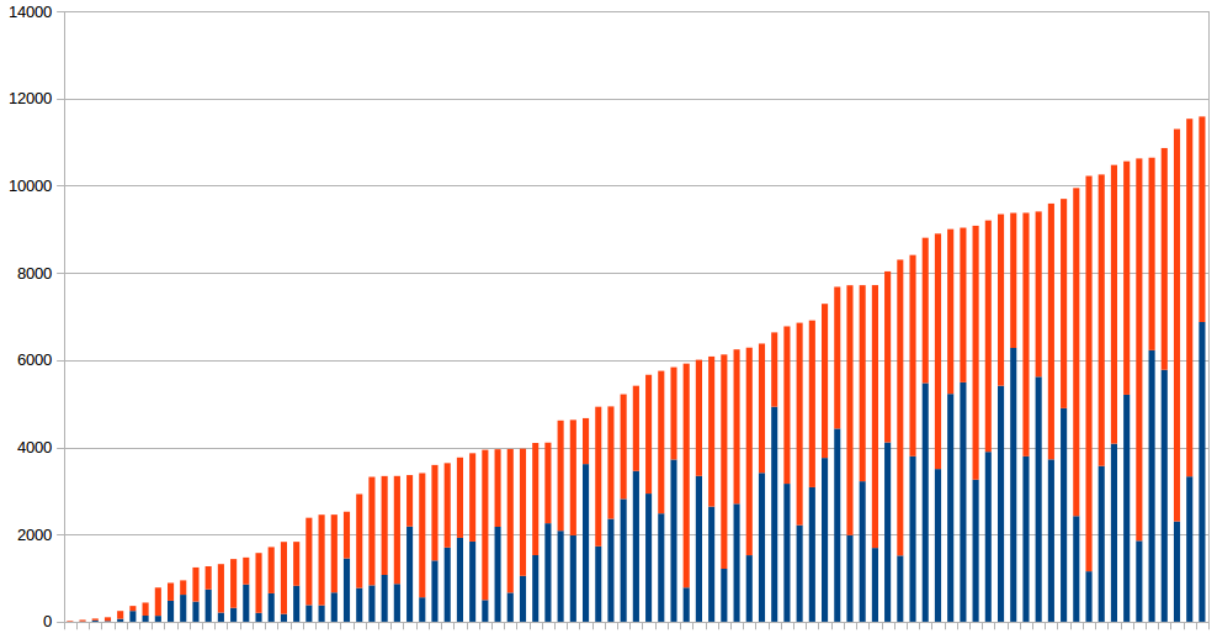


Figure 5.21: Stacked bargraphs representing the distance from the best query from structural combinations to the average (bottom) and the worst(top) for ArgoUML. Graph is ordered by distance from best to worst.

(c) LPMBV

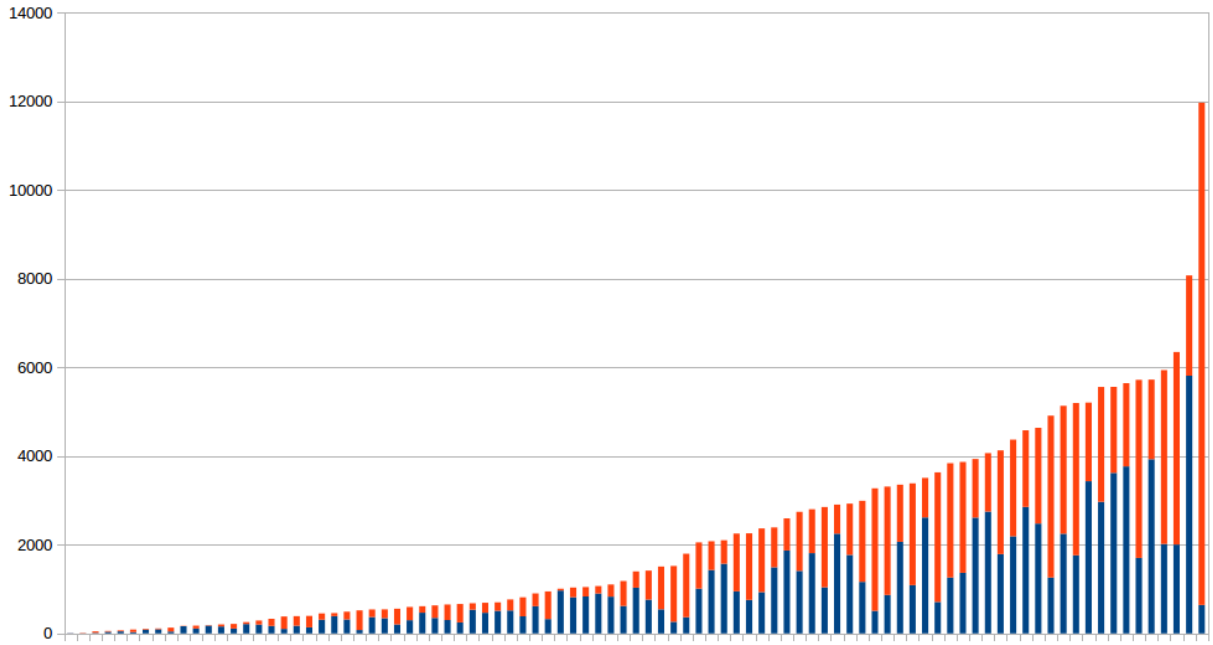


Figure 5.21: Stacked bargraphs representing the distance from the best query from structural combinations to the average (bottom) and the worst(top) for ArgoUML. Graph is ordered by distance from best to worst.

Similarly to how I selected the best query based on each query type in the previous problem, I selected the best query from each of the combinations for each feature. I selected from both the Combined and the Title query types. Figure 5.21 shows the differences between the best, average, and worst cases for each feature. The greatest distance between the best query and the worst query for the three corpora is found in the LMPBV corpus at 11,967, while the greatest mean distance between the best and the worst query is found in the CSB corpus at a distance of 7,004, while ICL showed the greatest mean distance between the best query and the average case is at 5,489. The smallest values in each case were found for the LMPBV corpus with a mean distance between the best and the worst queries of 2,198 and a mean distance between the best and the average case of 1,063. For each of the three corpora, the mean distance between the best and the average query is smaller than the mean between the average and the worst.

Query	CSB	ICL	LMPBV
Best	0.1120	0.1237	0.0612
Average	0.0045	0.0089	0.0027
Worst	0.0013	0.0014	0.0019

Table 5.37: MRRs for choosing the best, average, and worst case for each feature for ArgoUML from structural combinations

B	S	SB	C	CB	CS	CSB
40	16	6	17	13	4	1

(a) CSB

L	C	CL	I	IL	IC	ICL
25	20	5	25	10	10	1

(b) ICL

V	B	BV	P	PV	PB	PBV	M	MV	MB	MBV
10	9	1	18	0	1	0	14	3	0	3
MP	MPV	MPB	MPBV	L	LV	LB	LBV	LP	LPV	
2	1	0	0	4	0	0	2	3	1	
LPB	LPBV	LM	LMV	LMB	LMBV	LMP	LMPV	LMPB	LMPBV	
1	0	5	1	1	4	4	1	2	2	

(c) LMPBV

Table 5.38: Percentage of the best queries obtained from each structural combination for ArgoUML

The MRRs have been computed for each case and the results can be found in Table 5.37. Comparing the MRRs between the best cases and the results of the best structural combinations shows large increases. The largest increase occurs for the ICL corpus with a difference between

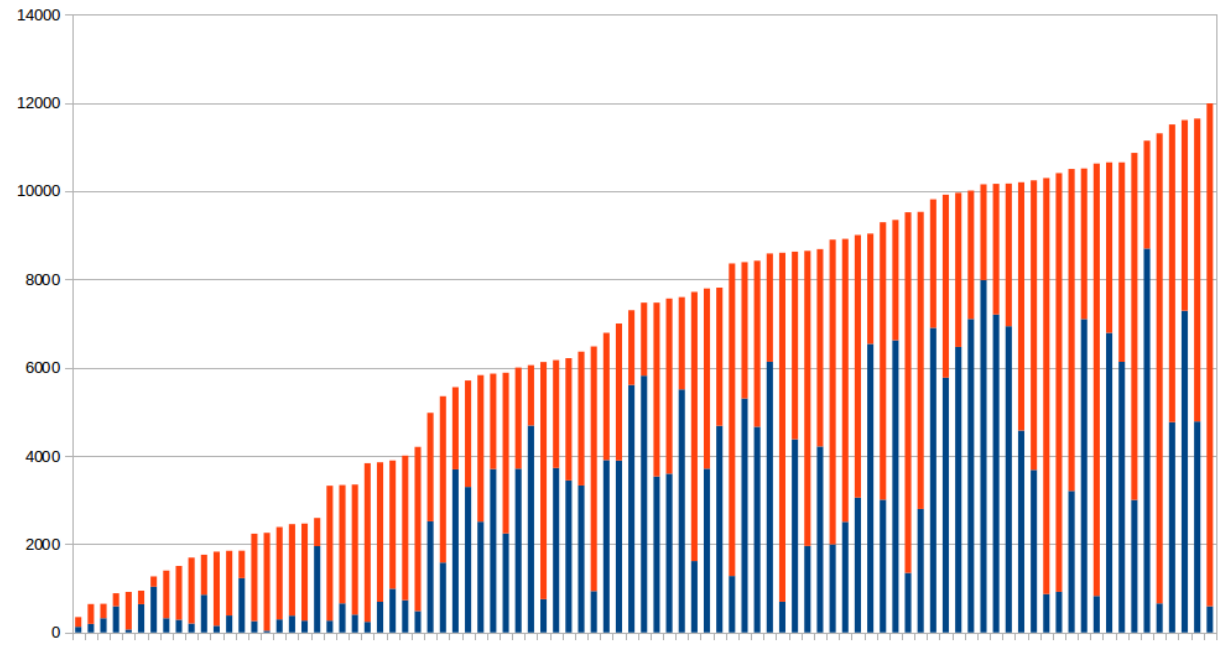


Figure 5.22: Stacked bargraphs representing the distance from the best query to the average (bottom) and the worst(top) from structural combinations for ArgoUML. Graph is ordered by distance from best to worst.

the best queries and the L combination of .0534. The smallest difference occurs for the LMPBV corpus with a difference of .0366. Of the three corpora, the largest MRR can also be found for the ICL corpus while the smallest MRR can be found for the LMPBV corpus.

In order to understand the combinations that make the greatest contribution to the MRR for each corpus, I computed the percentage of best queries that are found within each combination. The results of this computation can be found in Table 5.38. In the case of the CSB corpus, the highest percentage is found for the body combination alone (B) at 40%. The next closest percentage occurs for the comments alone (C) at 17%, less than half the percentage for B. The smallest percentage is found for the full corpus with the full corpus only representing 1% of the best queries. The percentages for the full ICL and LMPBV corpus are similar with ICL only representing 1% and LMPBV only representing 2%. Of the combinations in ICL, there is a tie for the highest percentage

	Best	Average	Worst
MRR	0.1759	0.0013	0.0003

Table 5.39: MRRs for choosing the best, average, and worst case for each feature from all corpora and all structural combinations for ArgoUML

	CSB	ICL	LMPBV	Flat
Percentage	49	21	11	19

Table 5.40: Percentages for each corpus where the best query was found from all corpora and all structural combinations for ArgoUML

between literals alone (L) and identifiers alone (I). These are followed by the comments alone (C). The highest percentages for LMPBV come from the parameters alone (P) and the method names alone (M). It should be noted that for each of the three corpora, the highest percentages come from combinations where only one structural lexicon is used.

I performed a Friedman test with a wilcoxon post-hoc for the best, average, and worst case of each feature for each of the four corpora. The results of the analysis showed that there is a significant difference between each case.

Similar to how choosing between the three query types for all corpora can lead to improvements in the results of the FLT, so can taking the combinations across the different corpora. In this computation I also included the best queries from the flat corpus. In Figure 5.22, you can see the difference between the best, average, and worst cases for each individual feature across all corpora. The greatest distance between the best and the worst query is 11,986. The mean distance between the best and worst query is 6,649 and the mean distance between the best query and the average case is 2,911 while the greatest distance between the best and the average case is 8,968. The mean distance distances between the best and the worst cases and the best and the average case are ac-

tually lower than the mean distance for the corpora individually. The mean distance from the best and the average case is smaller than the mean distance between the average and the worst case.

I computed the MRRs for these three cases and the results can be found in Table 5.39. The results of this computation show that there is another large increase from the individual corpora to taking the best across all queries. To understand which corpus is contributing the highest number of best queries from the corpora, I computed the percentage for each query and recorded the results in Table 5.40. From this table it can be seen that the CSB corpus contributed close half of the queries for the features. The corpus with the lowest percentage is the LMPBV corpus while ICL has a higher percentage than the flat corpus. The combinations with the highest percentages from the three corpora include literals only (L) from ICL with 13%, parameters only (P) from LMPBV with 4%, and the body alone (B) from CSB.

I performed a Friedman test with a wilcoxon post-hoc for the best, average, and worst case of each feature for each of the four corpora. The results of the analysis showed that there is a significant difference between each case.

5.3.2.2 JabRef

The MRRs for JabRef and the LMPBV corpus can be found in Tables 5.41 and 5.42. When looking at the results of the Combined query type, the majority of the MRRs for this corpus are less than .005. There are three exceptions to this. The first is found for the body comments alone(B), while the second and third come from the leading comments alone (L) and the leading comments with the local variables (LV), respectively. The highest MRR of the three comes from L with LV having less than a .001 difference. The three combinations make up the combinations with the three lowest minimum rank values. Additionally, L and LV have the lowest median values of any

Combination	min	1Q	median	3Q	max	MRR
V	26	544	1617	2990	4819	0.0031
B	2	476	1279	2806	5038	0.0164
BV	26	458	1149	2754	4760	0.0035
P	26	532	1133	2800	5136	0.0034
PV	26	518	1133	2763	4957	0.0031
PB	26	518	1150	2594	5073	0.0034
PBV	26	484	1133	2612	4884	0.0033
M	26	539	1133	2743	5150	0.0043
MV	26	508	1133	2844	4967	0.0037
MB	26	489	1194	2471	5075	0.0037
MBV	26	484	1133	2587	4868	0.0033
MP	26	540	1133	2485	5155	0.0033
MPV	26	525	1133	2516	5027	0.0030
MPB	26	523	1133	2336	5090	0.0034
MPBV	26	487	1133	2421	4912	0.0032
L	1	432	1079	2706	5106	0.0286
LV	1	469	1088	2765	4892	0.0285
LB	17	509	1263	2721	5089	0.0049
LBV	26	492	1124	2651	4832	0.0033
LP	26	493	1133	2723	5165	0.0034
LPV	26	509	1133	2552	4997	0.0031
LPB	26	527	1136	2556	5110	0.0035
LPBV	26	523	1133	2572	4919	0.0032
LM	26	494	1133	2481	5171	0.0042
LMV	26	499	1133	2582	5003	0.0037
LMB	26	520	1176	2426	5107	0.0035
LMBV	26	504	1133	2475	4915	0.0033
LMP	26	496	1133	2278	5171	0.0033
LMPV	26	517	1133	2292	5077	0.0031
LMPB	26	541	1133	2291	5119	0.0033
LMPBV	26	519	1133	2302	4998	0.0032

Table 5.41: MRRs and descriptive statistics of the effectiveness measures for the LMPBV corpus and Combined query type for JabRef

Combination	min	1Q	median	3Q	max	MRR
V	2	453	1211	2729	4572	0.0190
B	1	383	1219	2312	4404	0.0795
BV	1	392	1205	2240	4435	0.0793
P	1	445	1082	2327	5136	0.0598
PV	2	625	1065	2856	4707	0.0156
PB	1	347	1290	2231	4684	0.0558
PBV	1	329	1235	2230	4707	0.0560
M	2	482	1133	2273	4941	0.0201
MV	2	498	1082	2365	4545	0.0196
MB	1	375	1258	2401	4520	0.0485
MBV	1	363	1389	2204	4583	0.0439
MP	2	465	1092	2202	5089	0.0197
MPV	2	480	1121	2387	5035	0.0193
MPB	1	393	1383	2475	4979	0.0430
MPBV	1	363	1249	2410	4937	0.0429
L	2	416	1430	2590	5010	0.0217
LV	2	443	1153	2933	5010	0.0160
LB	1	353	1551	2325	4774	0.0792
LBV	1	411	1346	2166	4926	0.0791
LP	2	432	1344	2832	4982	0.0162
LPV	2	446	1188	2626	5055	0.0163
LPB	1	387	1517	2496	4669	0.0557
LPBV	1	410	1282	2574	4922	0.0559
LM	2	357	1387	2284	4033	0.0175
LMV	2	398	1220	2476	4499	0.0177
LMB	1	315	1505	2851	4694	0.0438
LMBV	1	372	1361	2403	4926	0.0436
LMP	2	405	1554	2426	5069	0.0172
LMPV	2	411	1235	2616	5019	0.0172
LMPB	1	371	1203	2600	5012	0.0428
LMPBV	1	387	1161	2530	4943	0.0430

Table 5.42: MRRs and descriptive statistics of the effectiveness measures for the LMPBV corpus and Title query type for JabRef

Combination	min	1Q	median	3Q	max	MRR
B	60	338	930	2419	4975	0.0032
S	26	539	1254	2199	5168	0.0034
SB	29	474	1121	2363	5105	0.0036
C	1	193	1144	2007	4949	0.0331
CB	6	235	1065	2034	5091	0.0079
CS	26	363	1218	2516	5077	0.0042
CSB	26	395	1118	2391	5101	0.0040

Table 5.43: MRRs and descriptive statistics of the effectiveness measures for the CSB corpus and Combined query type for JabRef

Combination	min	1Q	median	3Q	max	MRR
B	1	407	1271	2257	4968	0.0799
S	1	361	970	2242	4797	0.0451
SB	1	406	1046	2063	5078	0.0367
C	1	332	1025	2478	4676	0.0704
CB	1	309	1373	2565	4872	0.1043
CS	1	153	954	2541	4690	0.0717
CSB	1	253	683	2063	4845	0.0342

Table 5.44: MRRs and descriptive statistics of the effectiveness measures for the CSB corpus and Title query type for JabRef

of the combinations. The combination with the lowest MRR value is the method names combined with the parameters and the local variables (MPV), however this value is within a .0005 range of the majority of other combinations in this corpus. Looking at the Title query type, there are four top combinations that are within .0004 of each other. These include B, LB, BV, and LBV. The combination with the lowest MRR is PV which is tied for the highest minimum rank, has the highest 1Q, and the highest median value.

Combination	min	1Q	median	3Q	max	MRR
L	16	393	1294	2278	5000	0.0045
I	26	333	1109	2318	5104	0.0035
IL	21	496	1124	2471	4951	0.0039
C	1	193	1144	2007	4949	0.0331
CL	26	220	1086	2316	5116	0.0042
IC	24	299	1068	2076	4984	0.0044
ICL	21	384	969	2429	5115	0.0040

Table 5.45: MRRs and descriptive statistics of the effectiveness measures for the ICL corpus and Combined query type for JabRef

Combination	min	1Q	median	3Q	max	MRR
L	1	397	1428	2851	4847	0.0380
I	1	309	1046	2331	4768	0.0304
IL	1	500	1096	2186	5033	0.0308
C	1	332	1025	2478	4676	0.0704
CL	1	260	1233	2526	4920	0.0882
IC	1	369	1613	2621	4709	0.0574
ICL	1	253	1116	2478	4859	0.0316

Table 5.46: MRRs and descriptive statistics of the effectiveness measures for the ICL corpus and Title query type for JabRef

We can see the results of the CSB corpus in Tables 5.43 and 5.44. For the Combined query type, one MRR is significantly higher than the others. This MRR is for the comments alone (C) and is the only MRR to get over .008 with a value of .0331. This combination has the lowest minimum rank, 1Q, 3Q, and maximum rank of the combinations. The combination with the lowest MRR is the body alone (B) with the highest minimum rank, however interesting to note of this combination is that it has the lowest median value of any combination. This combination is slightly lower than the signature alone (S) and SB. Interestingly, for the Title query type, B actually has a higher MRR

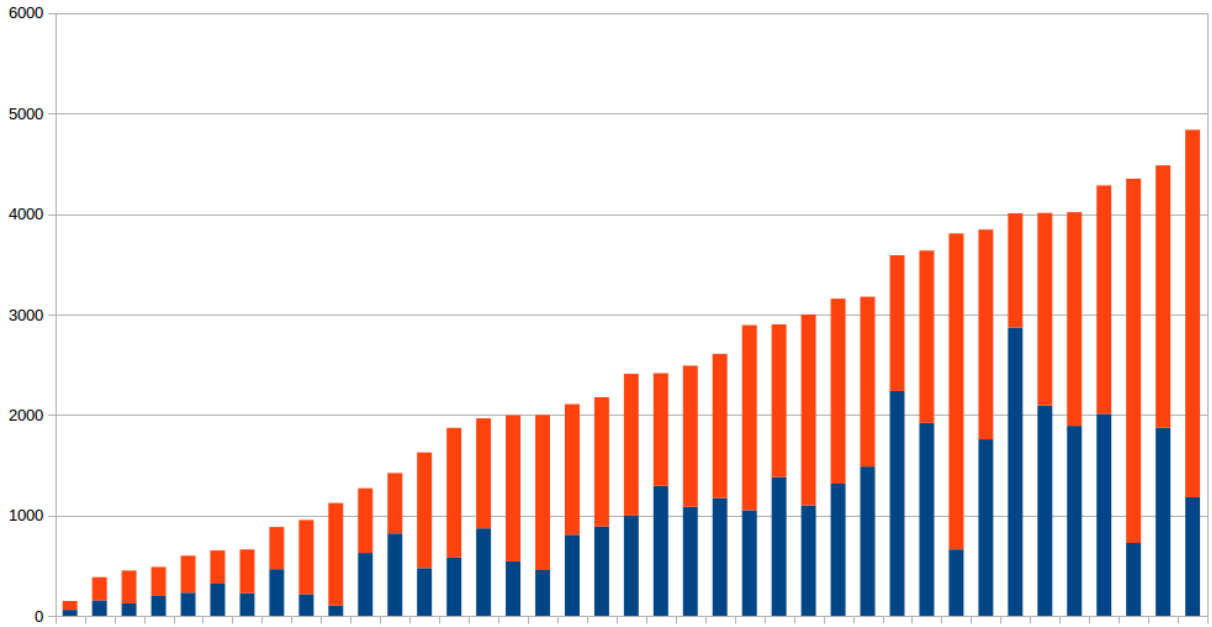
than C, however the highest MRR of the combinations is from CB, when the two are combined. The combination with the lowest MRR is the full corpus. However, the full corpus has the lowest median and 3Q values.

Tables 5.45 and 5.46 contain the results for the ICL corpus. Similar to the CSB corpus, C has the highest MRR for ICL and the Combined query type as well. This combination has the lowest of each value excepting the median. The corpus with the lowest MRR is the identifiers by themselves (I). For the Title query type, the comments combined with literals (CL) has the highest MRR. Once again, I has the lowest MRR of any of the combinations.

I performed a Friedman test with a wilcoxon post-hoc for each of the corpora between each of the different structural combinations. This analysis was performed for both the Combined and the Title query types. For the Combined query type, a significant difference was found in both the ICL and the LMPBV corpora, however there was no significant difference found in the CSB corpus. For the LMPBV corpus, 107 pairs were found to have significant differences. For the ICL corpus, there were only 2 significant differences found with significant difference found between IL and IC, and IL and ICL. Interestingly, for the Title query type, a far fewer number of significant differences were found. For CSB and ICL, no significant differences were found. Furthermore, for the LMPBV corpus, only 4 significant differences were found.

The results of the best, average, and worst cases for each feature for each corpus and both query types can be found in Figure 5.23. The greatest distance between the best query and the worst query for the three corpora is found in the LMPBV corpus at 5,054, while the greatest mean distance between the best and the worst query is found in the CSB corpus at a distance of 2,377. ICL showed the greatest mean distance between the best query and the average case is at 998. The smallest mean values in each case were found for the LMPBV corpus with a mean

(a) CSB



(b) ICL

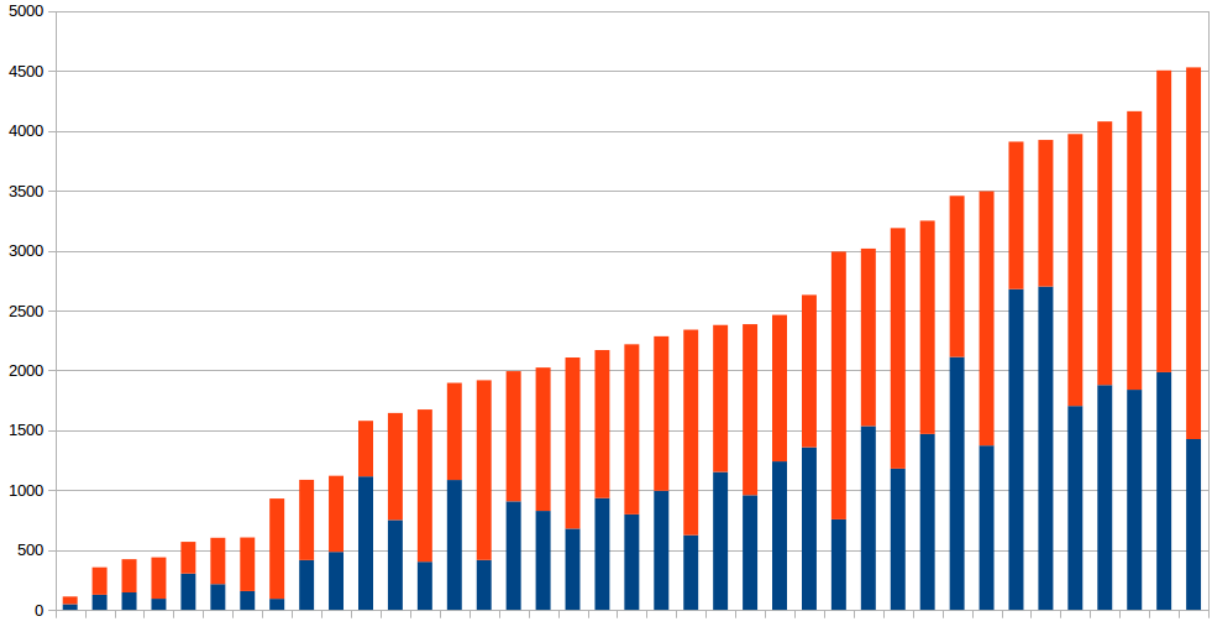


Figure 5.23: Stacked bargraphs representing the distance from the best query from structural combinations to the average (bottom) and the worst(top) for JabRef. Graph is ordered by distance from best to worst.

(c) LPMBV

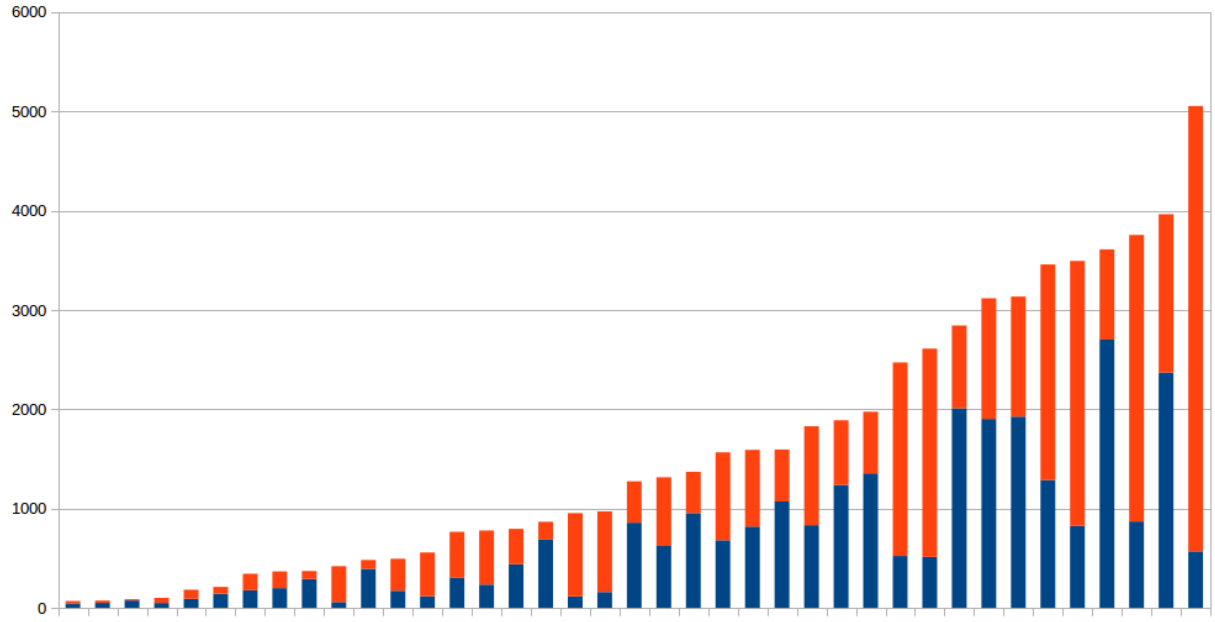


Figure 5.23: Stacked bargraphs representing the distance from the best query from structural combinations to the average (bottom) and the worst(top) for JabRef. Graph is ordered by distance from best to worst.

distance between the best and the worst queries of 1,559 and a mean distance between the best and the average case of 710. The smallest distance between the best and worst query is 111 and found in the ICL corpus. The smallest distance between the best and the average case is 47 and also found in the ICL corpus. For each of the three corpora, the mean distance between the best and the average query is smaller than the mean between the average and the worst.

The MRRs have been computed for each case and the results can be found in Table 5.47. Comparing the MRRs between the best cases and the results of the best structural combinations shows large increases. The largest increase occurs for the ICL corpus with a difference between the best queries and the CL combination of .1031. The smallest difference occurs for the CSB corpus with a difference of .0331. Of the three corpora, the largest MRR can also be found for the ICL corpus while the smallest MRR can be found for the LMPBV corpus.

Query	CSB	ICL	LMPBV
Best	0.1374	0.1913	0.1171
Average	0.0022	0.0020	0.0025
Worst	0.0008	0.0007	0.0014

Table 5.47: MRRs for choosing the best, average, and worst case for each feature for JabRef from structural combinations

B	S	SB	C	CB	CS	CSB
33	15	5	20	7	7	10

(a) CSB

L	C	CL	I	IL	IC	ICL
20	25	5	30	5	10	2

(b) ICL

V	B	BV	P	PV	PB	PBV	M	MV	MB	MBV
12	23	2	12	0	0	0	5	0	2	2
MP	MPV	MPB	MPBV	L	LV	LB	LBV	LP	LPV	
0	0	2	2	23	0	2	2	0	0	
LPB	LPBV	LM	LMV	LMB	LMBV	LMP	LMPV	LMPB	LMPBV	
0	0	0	0	0	5	0	0	0	0	

(c) LMPBV

Table 5.48: Percentage of the best queries obtained from each structural combination for JabRef

The percentages for JabRef can be found in Table 5.48. In the case of the CSB corpus, the highest percentage is found for the body combination alone (B) at 33%. The next closest percentage occurs for the comments alone (C) at 20%, 13% less than the percentage for B. The smallest percentage is found for the signature combined with the body (SB) with this combination

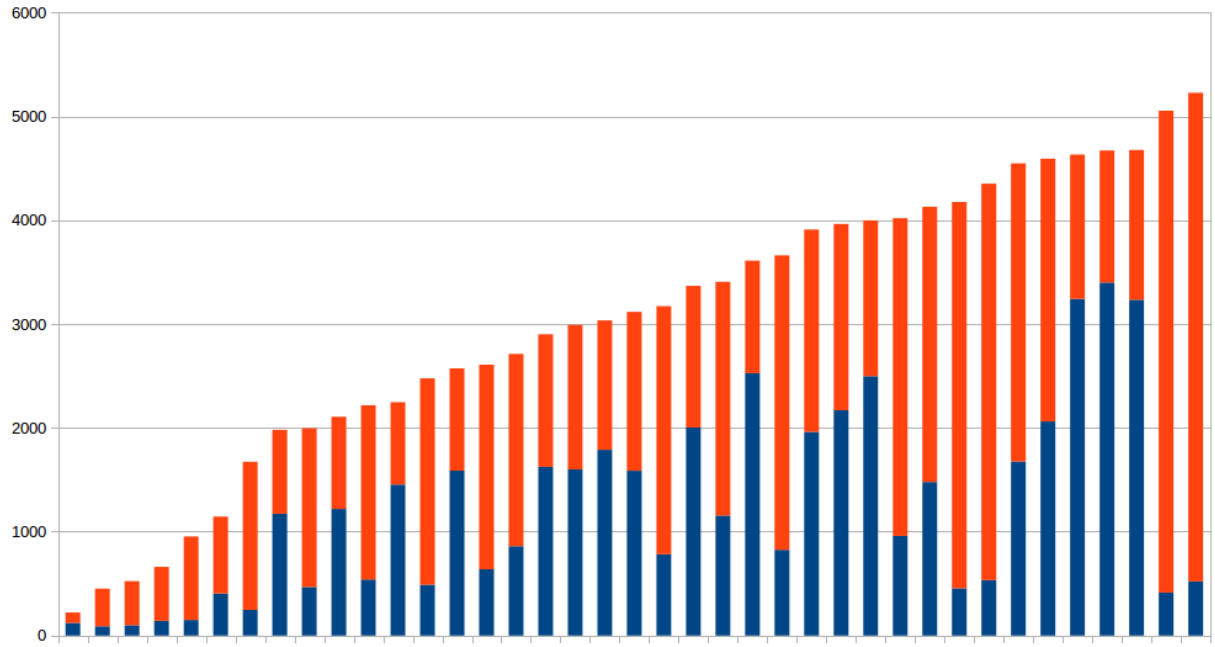


Figure 5.24: Stacked bargraphs representing the distance from the best query to the average (bottom) and the worst(top) from structural combinations for JabRef. Graph is ordered by distance from best to worst.

only representing 5% of the best queries. For both the ICL and LMPBV corpus, the full corpora perform poorly with ICL only representing 1% and LMPBV making up 0% of the best queries for the corpus. Of the combinations in ICL, the I outperforms the other combinations with 30%. The next closest combination is C with 25% and then L with 20%. The lowest percentage is found for the full corpus. The highest percentages for LMPBV come from the two types of comments (leading and body) alone. It should be noted that for each of the three corpora, the highest percentages come from combinations where only one structural lexicon is used.

I performed a Friedman test with a wilcoxon post-hoc for the best, average, and worst case of each feature for each of the four corpora. The results of the analysis showed that there is a significant difference between each case.

In Figure 5.24, you can see the difference between the best, average, and worst cases for

	Best	Average	Worst
MRR	0.2404	0.0018	0.0005

Table 5.49: MRRs for choosing the best, average, and worst case for each feature from all corpora and all structural combinations for JabRef

	CSB	ICL	LMPBV	Flat
Percentage	41	33	19	7

Table 5.50: Percentages for each corpus where the best query was found from all corpora and all structural combinations for JabRef

each individual feature across all corpora. The greatest distance between the best and the worst query is 5,227. The mean distance between the best and worst query is 3,019 and the mean distance between the best query and the average case is 1,235 while the greatest distance between the best and the average case is 3,400. Each of these values are higher than the values found for the same cases of the individual corpora. The mean distance from the best and the average case is smaller than the mean distance between the average and the worst case.

I computed the MRRs for these three cases and the results can be found in Table 5.49. The results of this computation show that there is another large increase from the individual corpora to taking the best across all queries. To understand which corpus is contributing the highest number of best queries from the corpora, I computed the percentage for each query and recorded the results in Table 5.50. From this table it can be seen that the CSB corpus contributed the largest percentage of the queries for the features. The corpus with the lowest percentage is the flat corpus while ICL has a higher percentage than the LMPBV corpus. The combinations with the highest percentages from the three corpora include the comments only (C) from ICL with 10%, the body comments only (B) from LMPBV with 7%, and the body alone (B) from CSB at 18%.

I performed a Friedman test with a wilcoxon post-hoc for the best, average, and worst case of each feature for each of the four corpora. The results of the analysis showed that there is a significant difference between each case.

5.3.2.3 jEdit

The MRRs for jEdit and the LMPBV corpus can be found in Tables 5.51 and 5.52. When looking at the results of the Combined query type, there is an interesting situation. All combinations for this corpus are less than .0015 and with the exception of three combinations, they have the same MRR at .0011. While this is true, each of the combinations have different spreads. The three exceptions are for body comments alone (B), parameters alone (P), and parameters combined with local variables (LV). Each of these combinations are higher than the .0011 shared by the other combinations. The combination with the highest MRR is P. This corpus also hass the lowest minimum rank of the combinations at 38. Looking at the Title query type, there is more variation between the MRRs of the different combinations. While the majority of MRRs are below .02, the exception exists for the highest MRR that was found for B at .0304. This combination shares the lowest minimum rank of 1 with multiple other combinations, but also has the lowest 1Q value for any of the combinations. The lowest MRR of the combinations exists for the parameters combined with the local variables (PV). PV has the highest minimum rank and the highest 3Q values of the combinations.

We can see the results of the CSB corpus in Tables 5.53 and 5.54. For the Combined query type, there is more variation between the MRRs than there were for the Combined query type and the LMPBV corpus. All MRRs range from .0010 and .0020. The highest MRR found for any of the combinations is .0020 and is found for the body alone (B) which has the lowest minimum

Combination	min	1Q	median	3Q	max	MRR
V	57	744	2035	4037	7279	0.0011
B	77	715	2120	3994	7275	0.0012
BV	126	754	2065	4040	7279	0.0011
P	38	753	1936	4011	7281	0.0013
PV	57	756	1915	3944	7281	0.0012
PB	131	773	1867	3975	7281	0.0011
PBV	126	785	1916	4014	7281	0.0011
M	86	748	2166	4163	7027	0.0011
MV	86	715	2085	4139	7071	0.0011
MB	131	701	1982	3906	7160	0.0011
MBV	131	719	2030	3829	7085	0.0011
MP	86	760	1923	4054	7100	0.0011
MPV	86	749	1958	4197	7077	0.0011
MPB	131	726	1836	3927	7124	0.0011
MPBV	131	730	1912	4104	7105	0.0011
L	86	730	1913	3882	7085	0.0011
LV	86	721	1915	4078	7088	0.0011
LB	131	714	2002	3703	7242	0.0011
LBV	131	720	1904	3816	7094	0.0011
LP	86	765	1793	4119	7081	0.0011
LPV	86	751	1874	4010	7086	0.0011
LPB	131	740	1837	3988	7175	0.0011
LPBV	131	741	1893	4048	7115	0.0011
LM	86	704	1886	4088	7148	0.0011
LMV	86	692	1911	3988	7061	0.0011
LMB	131	667	1894	3825	7231	0.0011
LMBV	131	682	1864	3842	7066	0.0011
LMP	86	715	1822	4178	7078	0.0011
LMPV	86	726	1792	4203	7085	0.0011
LMPB	131	686	1788	3941	7132	0.0011
LMPBV	131	700	1780	4097	7088	0.0011

Table 5.51: MRRs and descriptive statistics of the effectiveness measures for the LMPBV corpus and Combined query type for jEdit

Combination	min	1Q	median	3Q	max	MRR
V	1	655	1837	3652	6895	0.0093
B	1	503	1913	3491	7122	0.0304
BV	1	528	1854	3662	7045	0.0092
P	4	554	1889	3694	6924	0.0042
PV	18	578	1866	3771	6998	0.0021
PB	1	594	1797	3542	7045	0.0087
PBV	1	536	1816	3997	7027	0.0088
M	1	523	1745	3386	6925	0.0169
MV	1	529	1767	3493	7124	0.0167
MB	1	548	1612	3377	7006	0.0170
MBV	1	521	1491	3413	7163	0.0166
MP	1	555	1749	3303	6970	0.0158
MPV	1	564	1738	3630	7108	0.0156
MPB	1	587	1554	3456	6999	0.0156
MPBV	1	540	1657	3416	7059	0.0156
L	1	580	1653	3723	7169	0.0155
LV	1	544	1680	3718	7086	0.0119
LB	1	572	1684	3788	7116	0.0084
LBV	1	536	1739	3612	7099	0.0084
LP	1	581	1683	3852	7117	0.0085
LPV	1	576	1711	3709	7079	0.0085
LPB	1	588	1549	3728	7070	0.0083
LPBV	1	546	1640	3521	7141	0.0084
LM	1	564	1566	3342	7157	0.0156
LMV	1	535	1535	3466	7125	0.0155
LMB	1	548	1401	3446	7099	0.0154
LMBV	1	538	1368	3397	7175	0.0154
LMP	1	559	1625	3489	7122	0.0155
LMPV	1	557	1612	3485	7200	0.0155
LMPB	1	548	1368	3108	7110	0.0154
LMPBV	1	538	1584	3202	7191	0.0154

Table 5.52: MRRs and descriptive statistics of the effectiveness measures for the LMPBV corpus and Title query type for jEdit

Combination	min	1Q	median	3Q	max	MRR
B	28	582	1731	3934	7125	0.0020
S	86	792	1692	3592	7300	0.0011
SB	59	672	1895	3862	7255	0.0014
C	35	658	1685	4503	7101	0.0016
CB	59	606	1743	4154	7285	0.0016
CS	92	783	1455	3960	7179	0.0012
CSB	88	681	1855	4054	7248	0.0013

Table 5.53: MRRs and descriptive statistics of the effectiveness measures for the CSB corpus and Combined query type for jEdit

Combination	min	1Q	median	3Q	max	MRR
B	1	545	1694	3782	7121	0.0545
S	1	674	1481	3397	6651	0.0376
SB	1	539	1586	3705	7102	0.0378
C	1	624	1730	4070	7240	0.0386
CB	1	526	1969	3937	6991	0.0473
CS	1	665	1393	3237	7187	0.0196
CSB	1	480	1548	3921	7125	0.0398

Table 5.54: MRRs and descriptive statistics of the effectiveness measures for the CSB corpus and Title query type for jEdit

rank and 1Q of any of the combinations. The combination with the lowest MRR is found for the signature alone (S) with the highest maximum rank of the different combinations. The lowest MRRs for this corpus are found with any combination of S and the other structural lexicons. When looking at the Title query type, B is once again the combination with the highest MRR with the next closest being the combination of the comments with the body (CB). Neither of these combinations have the lowest of any of the values. The lowest MRR comes from comments combined

Combination	min	1Q	median	3Q	max	MRR
L	1	601	1582	4089	7166	0.0190
I	26	614	1660	4190	7280	0.0021
IL	50	708	1909	3895	7234	0.0016
C	35	658	1685	4503	7101	0.0016
CL	71	634	1803	4152	7263	0.0013
IC	40	661	1845	4079	7274	0.0017
ICL	70	669	1966	4295	7266	0.0015

Table 5.55: MRRs and descriptive statistics of the effectiveness measures for the ICL corpus and Combined query type for jEdit

Combination	min	1Q	median	3Q	max	MRR
L	1	547	1538	3915	7012	0.0648
I	1	453	1603	3650	7257	0.0586
IL	1	515	1722	3539	6960	0.0466
C	1	624	1730	4070	7240	0.0386
CL	1	484	1693	4043	7218	0.0282
IC	1	517	1419	3953	6998	0.0538
ICL	1	466	1811	3964	7217	0.0403

Table 5.56: MRRs and descriptive statistics of the effectiveness measures for the ICL corpus and Title query type for jEdit

with the signature (CS) which is the only MRR to drop below .02. This combination does not have the highest of any of the values, however it does have the lowest median of all combinations.

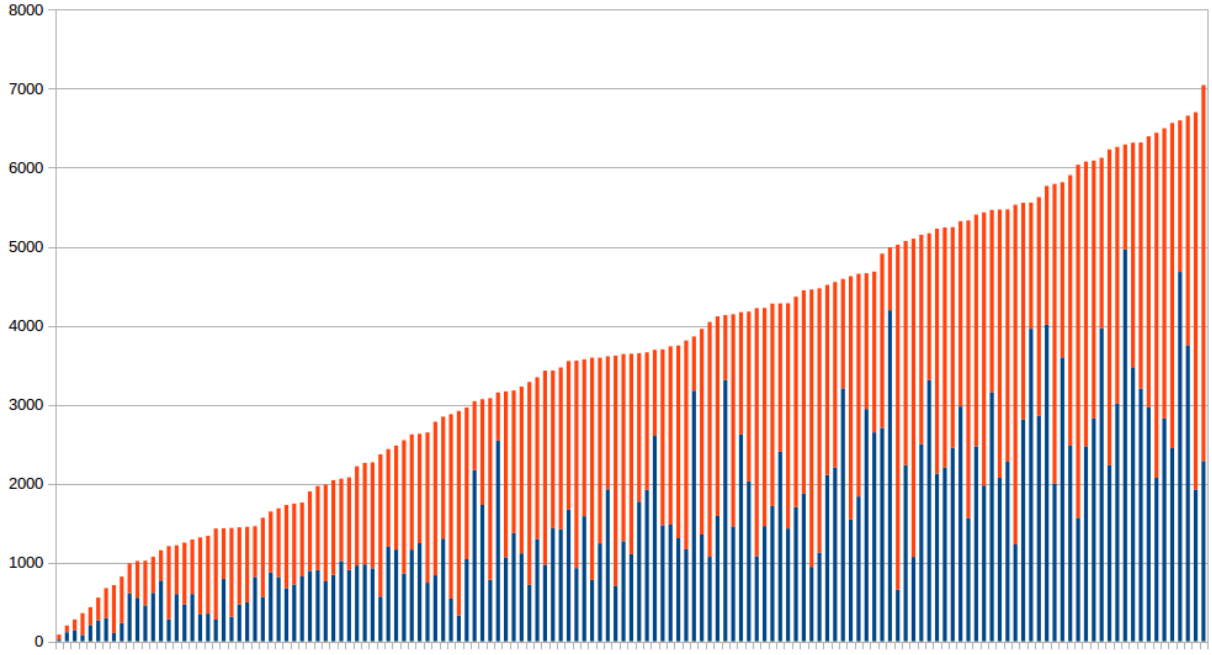
Tables 5.55 and 5.56 contain the results for the ICL corpus. When looking at the Combined query type, the literals alone (L) shows a substantial improvement over the other combinations. It is close to ten times the next closest MRR, has the lowest minimum rank, 1Q, and median values. The lowest MRR is found when combining the literals with the comments (CL). This combination has the highest minimum rank of the combinations. For the Title query type, the literals once again

have the highest MRR of the different combinations, however it does not have the highest of any of the descriptive statistics. Again, CL has the lowest MRR of the combinations with a difference from L of .0366.

I performed a Friedman test with a wilcoxon post-hoc for each of the corpora between each of the different structural combinations. This analysis was performed for both the Combined and the Title query types. For the Combined query type, a significant difference was found for each of the three corpora. For CSB, there was only a single significant difference between the CB and CSB combinations. For the LMPBV corpus, 65 pairs were found to have significant differences. For the ICL corpus, there were only 3 significant differences found with significant difference found between I and ICL, and CL and ICL. For the Title query type, no significant differences were found for CSB, while the only significant difference found for the ICL corpus was between CL and ICL. There was an increase in the number of significant differences found for the LMPBV corpus. For this corpus, a total of 104 significant differences were found.

The results of the best, average, and worst cases for each feature for each corpus and both query types can be found in Figure 5.25. The greatest distance between the best query and the worst query for the three corpora is found in the LMPBV corpus at 7,273, while the greatest mean distance between the best and the worst query is found in the CSB corpus at a distance of 3,612. This mean is only 4 higher than the mean distance for the ICL corpus. ICL showed the greatest mean distance between the best query and the average case at 1,612. The smallest mean values in each case were found for the LMPBV corpus with a mean distance between the best and the worst queries of 1,554 and a mean distance between the best and the average case of 834. The smallest distance between the best and worst query is 24 and found in the LMPBV corpus. The smallest distance between the best and the average case is 11 and also found in the LMPBV corpus.

(a) CSB



(b) ICL

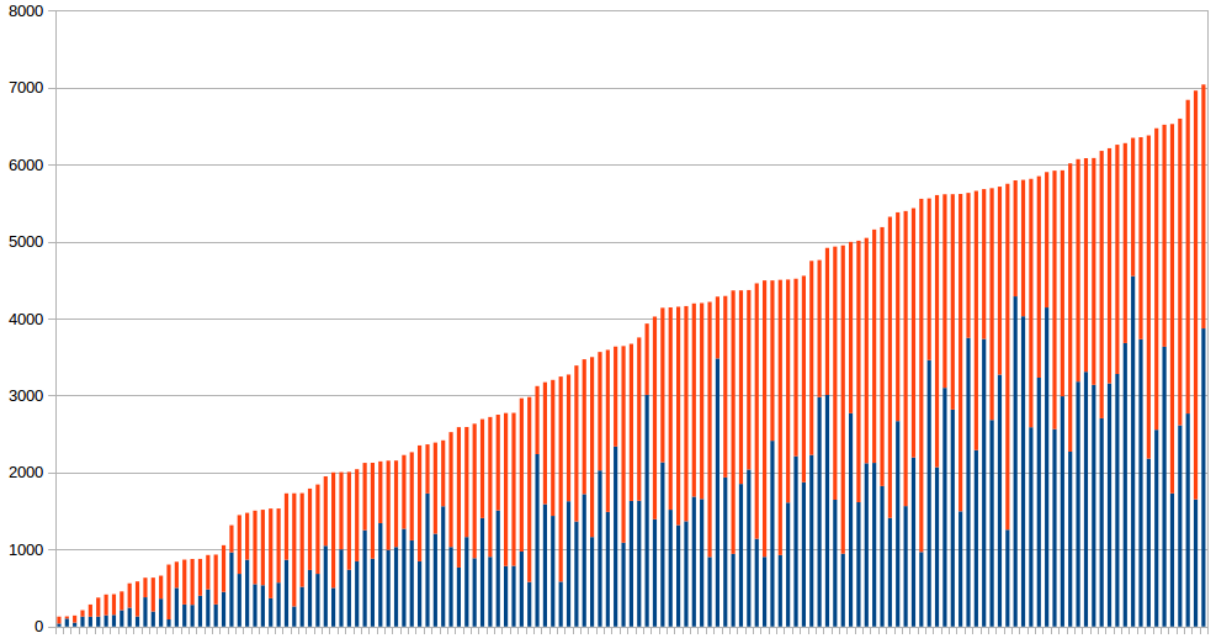


Figure 5.25: Stacked bargraphs representing the distance from the best query from structural combinations to the average (bottom) and the worst(top) for jEdit. Graph is ordered by distance from best to worst.

(c) LPMBV

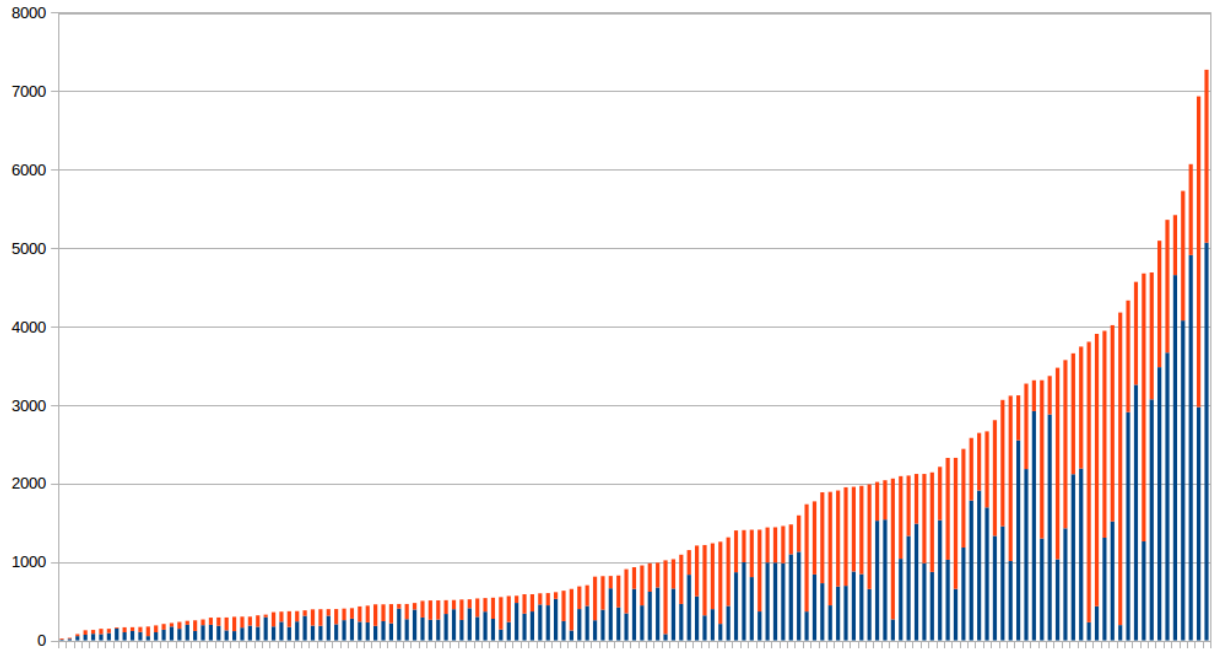


Figure 5.25: Stacked bargraphs representing the distance from the best query from structural combinations to the average (bottom) and the worst(top) for jEdit. Graph is ordered by distance from best to worst.

For two of the three corpora, the mean distance between the best and the average query is smaller than the mean between the average and the worst. The exception is for the LMPBV corpus.

The MRRs have been computed for each case and the results can be found in Table 5.57. Comparing the MRRs between the best cases and the results of the best structural combinations shows large increases. The largest increase occurs for the ICL corpus with a difference between the best queries and the L combination of .0781. The smallest difference occurs for the LMPBV corpus with a difference of .0339. Of the three corpora, the largest MRR can also be found for the ICL corpus while the smallest MRR can be found for the LMPBV corpus.

The percentages for jEdit can be found in Table 5.58. In the case of the CSB corpus, the highest percentage is found for the body combination alone (B) at 30%. There is a tie for the next closest percentage which occurs between the comments alone (C) and the signature alone

Query	CSB	ICL	LMPBV
Best	0.1047	0.1429	0.0643
Average	0.0009	0.0012	0.0012
Worst	0.0004	0.0005	0.0008

Table 5.57: MRRs for choosing the best, average, and worst case for each feature for jEdit from structural combinations

B	S	SB	C	CB	CS	CSB
30	23	7	23	5	7	1

(a) CSB

L	C	CL	I	IL	IC	ICL
29	18	6	31	4	6	2

(b) ICL

V	B	BV	P	PV	PB	PBV	M	MV	MB	MBV
14	17	2	8	0	1	1	10	0	2	1
MP	MPV	MPB	MPBV	L	LV	LB	LBV	LP	LPV	
1	0	1	1	8	1	0	0	2	0	
LPB	LPBV	LM	LMV	LMB	LMBV	LMP	LMPV	LMPB	LMPBV	
1	2	7	1	4	0	0	0	2	2	

(c) LMPBV

Table 5.58: Percentage of the best queries obtained from each structural combination for jEdit

(S) at 23% each, 7% less than the percentage for B. The smallest percentage is found for the full corpus with this combination only representing 1% of the best queries. For both the ICL and LMPBV corpus, the full corpora perform poorly with both corpora only contributing 2% of the best queries for the corpus. Of the combinations in ICL, the I combination outperforms the other

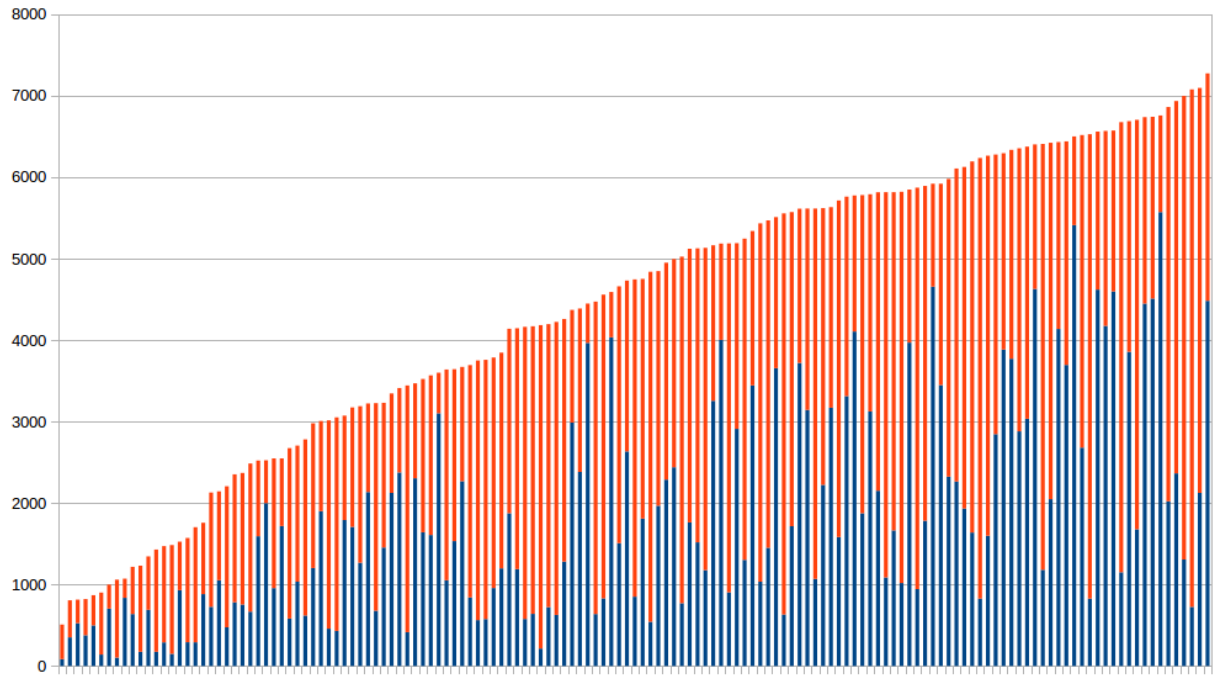


Figure 5.26: Stacked bargraphs representing the distance from the best query to the average (bottom) and the worst(top) from structural combinations for jEdit. Graph is ordered by distance from best to worst.

combinations with 31%. The next closest combination is L with 29% and then C with 18%. The lowest percentage is found for the full corpus. The highest percentages for LMPBV come from the variables, body comments, and method names alone. It should be noted that for each of the three corpora, the highest percentages come from combinations where only one structural lexicon is used.

I performed a Friedman test with a wilcoxon post-hoc for the best, average, and worst case of each feature for each of the four corpora. The results of the analysis showed that there is a significant difference between each case.

In Figure 5.26, you can see the difference between the best, average, and worst cases for each individual feature across all corpora. The greatest distance between the best and the worst

	Best	Average	Worst
MRR	0.2032	0.0010	0.0003

Table 5.59: MRRs for choosing the best, average, and worst case for each feature from all corpora and all structural combinations for jEdit

	CSB	ICL	LMPBV	Flat
Percentage	32	33	21	14

Table 5.60: Percentages for each corpus where the best query was found from all corpora and all structural combinations for jEdit

query is 7,273. The mean distance between the best and worst query is 4,421 and the mean distance between the best query and the average case is 1,832 while the greatest distance between the best and the average case is 5,571. Each of these values are higher than the values found for the same cases of the individual corpora. The mean distance from the best and the average case is smaller than the mean distance between the average and the worst case.

I computed the MRRs for these three cases and the results can be found in Table 5.59. The results of this computation show that there is another large increase from the individual corpora to taking the best across all queries. To understand which corpus is contributing the highest number of best queries from the corpora, I computed the percentage for each query and recorded the results in Table 5.60. From this table it can be seen that the ICL corpus contributed the largest percentage of the queries for the features. This corpus was followed closely by the CSB corpus with a difference of less than 1%. The corpus with the lowest percentage is the flat corpus. The combinations with the highest percentages from the three corpora include the identifiers only (I) from ICL with 12%, the body comments only (B) from LMPBV with 7%, and the signature alone (S) from CSB at 12%.

For the ICL corpus, the literals and the comments alone follow closely with a tie at 10%. None of the full corpora contributed to the best queries.

I performed a Friedman test with a wilcoxon post-hoc for the best, average, and worst case of each feature for each of the four corpora. The results of the analysis showed that there is a significant difference between each case.

5.3.2.4 muCommander

The MRRs for muCommander and the LMPBV corpus can be found in Tables 5.61 and 5.62. When looking at the results of the Combined query type, the highest MRR is found for the parameters only (P) combination with .0070. This combination has the lowest minimum rank of all the combinations. The next closest combination is for body comments only (B) which is the only other combination that is above .002. This combination has the second lowest minimum rank of all the combinations. There are three combinations with the lowest MRR of .0011, the local variables alone (V), the leading comments alone (L), and the leading comments combined with the parameters (LP). When looking at the Title query type, the highest MRR is for combining the parameters with the local variables (PV). This is followed by the leading comments combined with PV (LPV). The combination with the lowest MRR is the leading comments combined with the local variables and the body comments (LPB). This is the only combination with an MRR less than .01, however this combination does not have the highest of any of the descriptive statistics.

We can see the results of the CSB corpus in Tables 5.63 and 5.64. For the Combined query type, the only combination that is higher than 0.91 is for the comments alone (C). This combination has the lowest minimum rank, 1Q, and median of all the combinations. The next closest MRR is .0101 below C at .0035 for the comments combined with the signature (CS). The lowest MRR for

Combination	min	1Q	median	3Q	max	MRR
V	32	1233	2686	5763	8544	0.0011
B	11	1053	2720	6023	8531	0.0021
BV	18	1042	2743	6000	8560	0.0017
P	2	1178	2685	5994	8479	0.0070
PV	19	1229	2683	5884	8516	0.0016
PB	19	1138	2666	5923	8506	0.0016
PBV	25	1198	2703	5846	8526	0.0015
M	18	1143	2658	5406	8479	0.0014
MV	31	1195	2653	5530	8513	0.0012
MB	16	1062	2728	5288	8503	0.0015
MBV	24	1056	2719	5457	8530	0.0013
MP	17	1148	2666	5422	8510	0.0015
MPV	31	1222	2612	5398	8481	0.0012
MPB	25	1129	2606	5328	8480	0.0013
MPBV	26	1187	2602	5459	8497	0.0012
L	42	1200	2729	5817	8510	0.0011
LV	34	1196	2728	5799	8566	0.0012
LB	25	964	2728	5623	8520	0.0013
LBV	26	1029	2667	5691	8780	0.0013
LP	41	1199	2728	5956	8602	0.0011
LPV	32	1229	2724	5791	8488	0.0012
LPB	27	1104	2681	5560	8495	0.0013
LPBV	27	1140	2651	5654	8511	0.0013
LM	34	1208	2723	5366	8509	0.0012
LMV	32	1186	2685	5503	8507	0.0012
LMB	26	1005	2681	5084	8497	0.0013
LMBV	27	1030	2716	5179	8511	0.0012
LMP	33	1237	2683	5321	8487	0.0012
LMPV	31	1241	2690	5444	8484	0.0012
LMPB	27	1142	2647	5145	8486	0.0012
LMPBV	27	1131	2709	5212	8496	0.0012

Table 5.61: MRRs and descriptive statistics of the effectiveness measures for the LMPBV corpus and Combined query type for muCommander

Combination	min	1Q	median	3Q	max	MRR
V	1	964	2625	5492	8071	0.0196
B	1	1009	2625	5482	8362	0.0127
BV	1	1012	2701	5581	8341	0.0128
P	1	1135	2572	5444	8288	0.0142
PV	1	1229	2542	5297	8197	0.0256
PB	19	1115	2296	5293	8308	0.0015
PBV	1	1151	2348	5578	8153	0.0128
M	2	914	2578	4889	8549	0.0165
MV	1	969	2641	4987	8375	0.0180
MB	2	918	2368	4951	8573	0.0106
MBV	1	976	2691	4947	8377	0.0178
MP	2	966	2645	4917	8569	0.0108
MPV	1	994	2429	4891	8384	0.0181
MPB	2	1030	2328	5041	8575	0.0103
MPBV	1	1076	2278	4945	8433	0.0178
L	1	1105	2775	5147	7955	0.0126
LV	1	988	2739	5209	8294	0.0139
LB	19	902	2363	5204	8195	0.0016
LBV	1	951	2461	4949	8207	0.0127
LP	1	1071	2991	4926	7810	0.0129
LPV	1	1025	2913	4939	8199	0.0241
LPB	11	1032	2518	4740	7959	0.0024
LPBV	1	1032	2352	4936	8118	0.0128
LM	1	979	2592	4787	8306	0.0217
LMV	1	964	2553	4830	8245	0.0180
LMB	2	899	2227	4886	8557	0.0114
LMBV	1	934	2333	4855	8205	0.0177
LMP	1	1035	2585	4825	8228	0.0215
LMPV	1	1028	2489	4882	8161	0.0183
LMPB	1	986	2446	4803	8352	0.0213
LMPBV	1	1022	2328	4761	8134	0.0180

Table 5.62: MRRs and descriptive statistics of the effectiveness measures for the LMPBV corpus and Title query type for muCommander

Combination	min	1Q	median	3Q	max	MRR
B	13	977	2276	5141	8348	0.0031
S	12	956	2358	4632	8469	0.0020
SB	17	1044	2661	4485	8260	0.0022
C	1	600	2069	3959	8284	0.0136
CB	13	895	2306	4137	8188	0.0029
CS	5	678	2093	4114	8506	0.0035
CSB	14	788	2377	3934	8191	0.0024

Table 5.63: MRRs and descriptive statistics of the effectiveness measures for the CSB corpus and Combined query type for muCommander

Combination	min	1Q	median	3Q	max	MRR
B	1	716	2099	4009	8058	0.0218
S	1	741	1791	3560	8789	0.0284
SB	2	671	1986	3656	8107	0.0115
C	2	787	2127	3895	8372	0.0113
CB	1	846	2200	3577	8160	0.0218
CS	1	510	1893	3682	8668	0.0222
CSB	2	763	1963	3116	8440	0.0114

Table 5.64: MRRs and descriptive statistics of the effectiveness measures for the CSB corpus and Title query type for muCommander

the combinations is for the signature alone (S), and this combination does not have the highest of any of the descriptive statistic values. Aside from the comments combined with the body (CB) that has a lower MRR than CS, the lowest MRRs for this corpus are found with any combination of S and the other structural lexicons. However, when looking at the Title query type, S has the highest MRR of any of the combinations. This combination has the lowest median value of any of the combinations. The next closest combination is CS which is .0062 lowest than S. CS has the

Combination	min	1Q	median	3Q	max	MRR
L	1	978	2641	6135	8587	0.0128
I	16	1047	2371	4182	8156	0.0027
IL	18	967	2385	4937	8549	0.0017
C	1	600	2069	3959	8284	0.0136
CL	11	750	2273	4894	8777	0.0023
IC	14	858	2143	3936	8270	0.0023
ICL	10	840	2414	4281	8789	0.0023

Table 5.65: MRRs and descriptive statistics of the effectiveness measures for the ICL corpus and Combined query type for muCommander

Combination	min	1Q	median	3Q	max	MRR
L	1	1092	2789	4715	8687	0.0307
I	1	820	2144	3900	7795	0.0204
IL	2	839	2199	4354	8763	0.0104
C	2	787	2127	3895	8372	0.0113
CL	2	860	2256	4050	8486	0.0099
IC	1	806	2208	3772	8469	0.0206
ICL	2	971	1749	3735	8512	0.0103

Table 5.66: MRRs and descriptive statistics of the effectiveness measures for the ICL corpus and Title query type for muCommander

lowest 1Q value of the combinations. The lowest MRR comes from the comments alone (C) at .0113. This combination is tied for having the highest minimum rank.

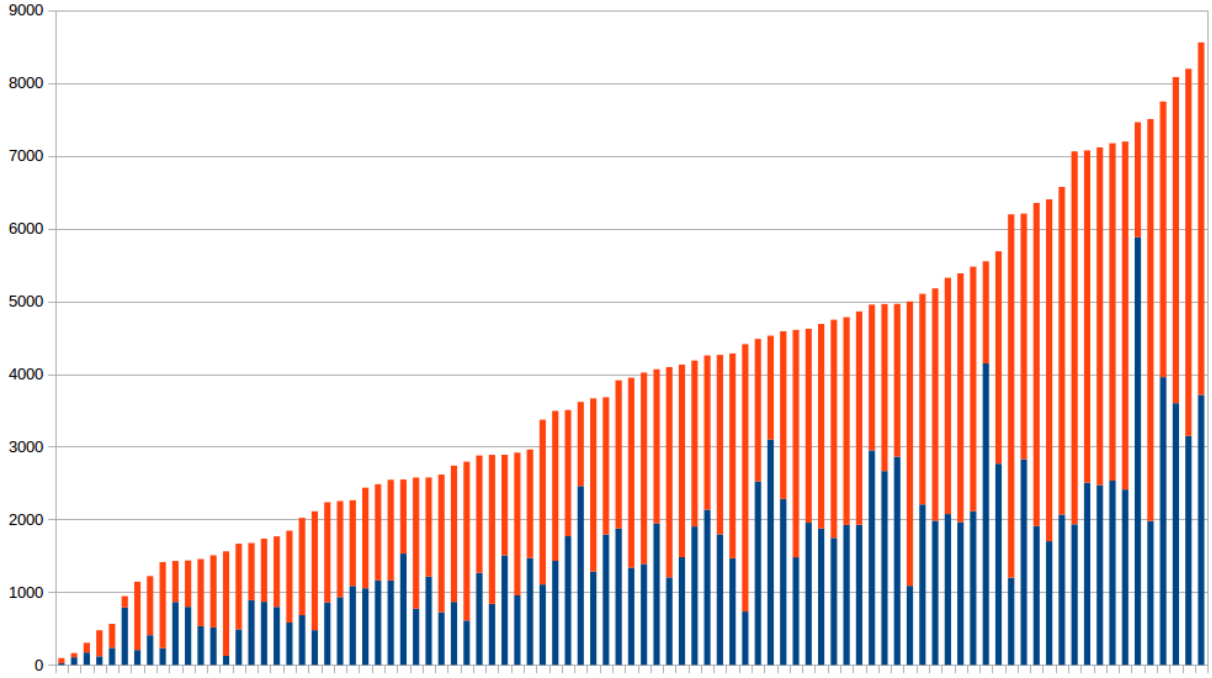
Tables 5.65 and 5.66 contain the results for the ICL corpus. When looking at the Combined query type, there are two combinations that are significantly higher than the other combinations. These combinations include the literals alone (L) and the comments alone (C), with C having the higher MRR of the two. C is tied with L for the lowest minimum rank, and has the lowest 1Q and median values of all the combinations. L does not have the lowest of any of the descriptive

statistics. The lowest MRR comes from the identifiers combined with the literals and has the highest minimum rank of any of the combinations. For the Title query type, the literals alone (L) have the highest MRR of the different combinations, however it does not have the highest of any of the descriptive statistics. CL has the lowest MRR of the combinations with a difference from L of .0208.

I performed a Friedman test with a wilcoxon post-hoc for each of the corpora between each of the different structural combinations. This analysis was performed for both the Combined and the Title query types. For the Combined query type, a significant difference was found for each of the three corpora. For CSB, there were four significant differences found between the combinations. For both LMPBV and ICL, a large proportion of the pairings resulted in significant differences. For the LMPBV corpus, 211 pairs were found to have significant differences. For the ICL corpus, there were 13 significant differences found with significant difference found with every combination being significantly different compared to L. For the Title query type, no significant differences were found for CSB, while the number of significant differences for ICL and LMPBV were reduced. For ICL, only 4 significant differences were found with L being significantly different from the C, the comments combined with L (CL), and the full corpus, a significant difference was also found between ICL and the identifiers combined with L (IL). For LMPBV, the total number of pairings found with significant differences is 78. This is a reduction of 133 pairings.

The results of the best, average, and worst cases for each feature for each corpus and both query types can be found in Figure 5.27. The greatest distance between the best query and the worst query for the three corpora is found in the CSB corpus at 8,558, while the greatest mean distance between the best and the worst query is found in the ICL corpus at a distance of 3,920.

(a) CSB



(b) ICL

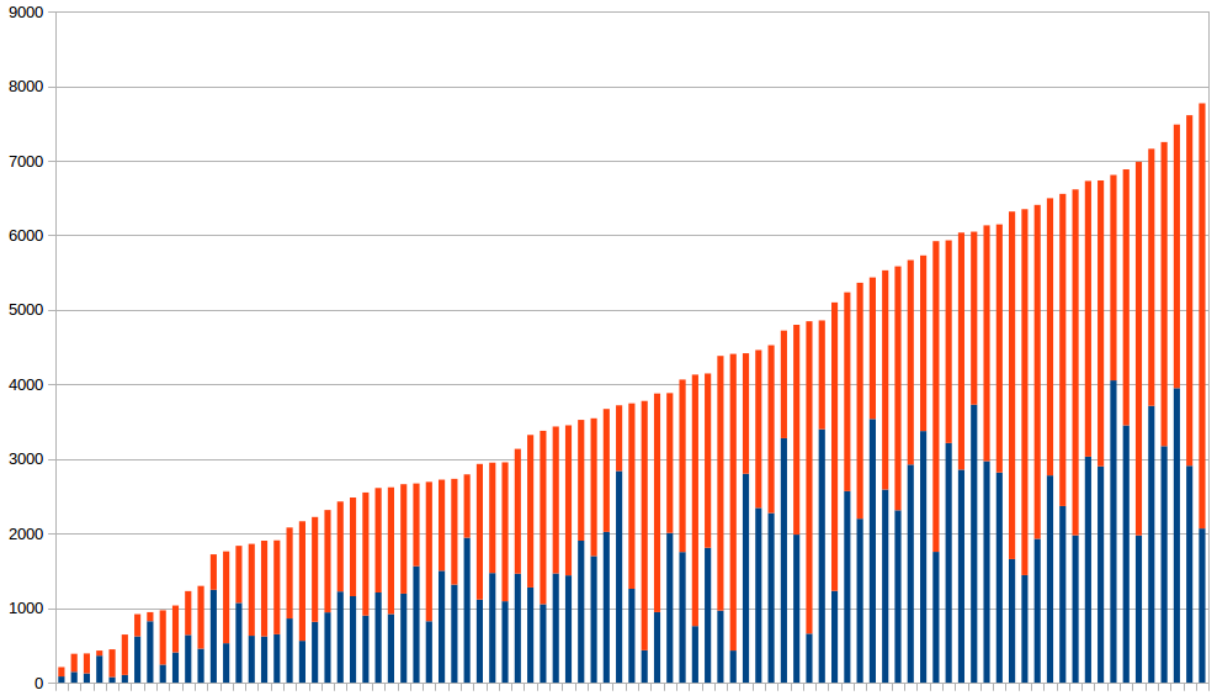


Figure 5.27: Stacked bargraphs representing the distance from the best query from structural combinations to the average (bottom) and the worst(top) for muCommander. Graph is ordered by distance from best to worst.

(c) LPMBV

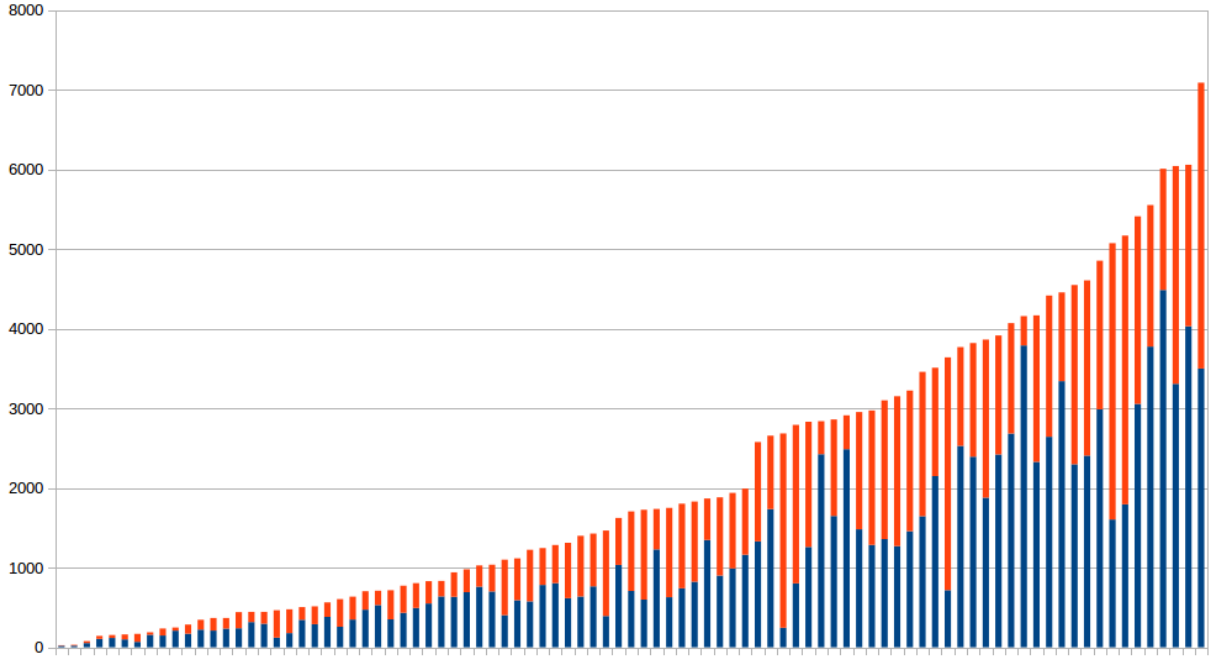


Figure 5.27: Stacked bargraphs representing the distance from the best query from structural combinations to the average (bottom) and the worst(top) for muCommander. Graph is ordered by distance from best to worst.

ICL also showed the greatest mean distance between the best query and the average case at 1,682. The smallest mean values in each case were found for the LMPBV corpus with a mean distance between the best and the worst queries of 2,129 and a mean distance between the best and the average case of 1,176. The smallest distance between the best and worst query is 25 and found in the LMPBV corpus. The smallest distance between the best and the average case is 19 and also found in the LMPBV corpus. For two of the three corpora, the mean distance between the best and the average query is smaller than the mean between the average and the worst. The exception is for the LMPBV corpus.

The MRRs have been computed for each case and the results can be found in Table 5.67. Comparing the MRRs between the best cases and the results of the best structural combinations shows large increases, however these differences are the smallest of the four subject systems. The

Query	CSB	ICL	LMPBV
Best	0.0545	0.0635	0.0571
Average	0.0013	0.0009	0.0009
Worst	0.0005	0.0004	0.0007

Table 5.67: MRRs for choosing the best, average, and worst case for each feature for muCommander from structural combinations

B	S	SB	C	CB	CS	CSB
19	19	5	26	12	13	3

(a) CSB

L	C	CL	I	IL	IC	ICL
19	31	9	19	5	9	3

(b) ICL

V	B	BV	P	PV	PB	PBV	M	MV	MB	MBV
8	12	0	10	0	2	2	12	0	3	0
MP	MPV	MPB	MPBV	L	LV	LB	LBV	LP	LPV	
3	1	1	1	10	0	5	0	2	0	
LPB	LPBV	LM	LMV	LMB	LMBV	LMP	LMPV	LMPB	LMPBV	
1	0	4	1	10	1	1	0	2	1	

(c) LMPBV

Table 5.68: Percentage of the best queries obtained from each structural combination for muCommander

largest increase occurs for the ICL corpus with a difference between the best queries and the L combination of .0328. The smallest difference occurs for the CSB corpus with a difference of

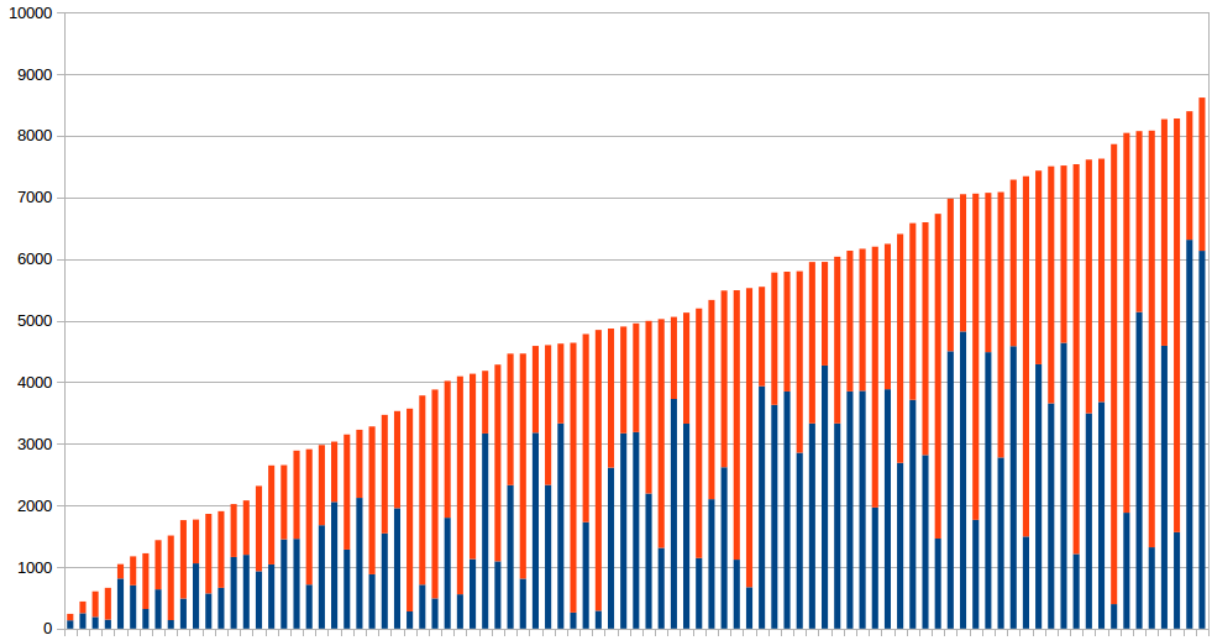


Figure 5.28: Stacked bargraphs representing the distance from the best query to the average (bottom) and the worst(top) from structural combinations for muCommander. Graph is ordered by distance from best to worst.

.0261. Of the three corpora, the largest MRR can also be found for the ICL corpus while the smallest MRR can be found for the CSB corpus.

The percentages for muCommander can be found in Table 5.68. In the case of the CSB corpus, the highest percentage is found for the comments combination alone (C) at 26%. There is a tie for the next closest percentage which occurs between the body alone (B) and the signature alone (S) at 19% each, 7% less than the percentage for C. The smallest percentage is found for the full corpus with this combination only representing 3% of the best queries. For both the ICL and LMPBV corpus, the full corpora perform poorly with the ICL corpus only contributing 3% of the best queries and LMPBV only contributing 1% of the best queries for the corpus. Of the combinations in ICL, the C combination outperforms the other combinations with 31%. There is a tie between L and I for the next closest combination at 19%. The lowest percentage is found

	Best	Average	Worst
MRR	0.1039	0.0008	0.0003

Table 5.69: MRRs for choosing the best, average, and worst case for each feature from all corpora and all structural combinations for muCommander

	CSB	ICL	LMPBV	Flat
Percentage	41	30	13	16

Table 5.70: Percentages for each corpus where the best query was found from all corpora and all structural combinations for muCommander

for the full corpus. The highest percentages for LMPBV come from the body comments alone. It should be noted that for each of the three corpora, the highest percentages come from combinations where only one structural lexicon is used.

I performed a Friedman test with a wilcoxon post-hoc for the best, average, and worst case of each feature for each of the four corpora. The results of the analysis showed that there is a significant difference between each case.

In Figure 5.28, you can see the difference between the best, average, and worst cases for each individual feature across all corpora. The greatest distance between the best and the worst query is 8,619. The mean distance between the best and worst query is 4,828 and the mean distance between the best query and the average case is 2,179 while the greatest distance between the best and the average case is 6,311. Each of these values are higher than the values found for the same cases of the individual corpora. The mean distance from the best and the average case is smaller than the mean distance between the average and the worst case.

I computed the MRRs for these three cases and the results can be found in Table 5.69. The results of this computation show that there is another large increase from the individual corpora to

taking the best across all queries, but this MRR is the lowest of all subject systems. To understand which corpus is contributing the highest number of best queries from the corpora, I computed the percentage for each query and recorded the results in Table 5.70. From this table it can be seen that the CSB corpus contributed the largest percentage of the queries for the features. This corpus was followed by the ICL corpus with a difference of 11%. The corpus with the lowest percentage is the LMPBV corpus. The combinations with the highest percentages from the three corpora include the comments only (C) from ICL and CSB with 16% and the parameters only (P) from LMPBV with 4%. None of the full corpora contributed to the best queries.

I performed a Friedman test with a wilcoxon post-hoc for the best, average, and worst case of each feature for each of the four corpora. The results of the analysis showed that there is a significant difference between each case.

5.3.2.5 All Systems

The MRRs for all the systems combined and the LMPBV corpus can be found in Tables 5.71 and 5.72. When looking at the results of the Combined query type, all MRRs are below .005 with the exception of three, the body comments alone (B), the leading comments alone (L), and the leading comments combined with the local variables (LV). The highest MRR is found for L which is tied for the lowest minimum rank of the combinations. The combination with the lowest MRR is the method name combined with the local variables (MV) which has the highest minimum rank of any of the combinations and an MRR of .0021. When looking at the Title query type, the highest MRR is for the body comments alone (B) which has an MRR of .0284, the only MRR with a value higher than .02. The next closest MRR is from the combination of the leading comments, method names, parameters, and body comments (LMPB), and has an MRR of .0185. This is a difference

Combination	min	1Q	median	3Q	max	MRR
V	4	813	2316	5066	11006	0.0028
B	2	766	2279	5163	11695	0.0051
BV	4	783	2273	4963	11694	0.0032
P	2	777	2289	5146	11072	0.0044
PV	5	780	2267	5072	11078	0.0031
PB	5	782	2260	5180	11692	0.0031
PBV	5	802	2237	5160	11695	0.0030
M	5	777	2276	5027	10739	0.0024
MV	8	780	2285	5019	10783	0.0021
MB	4	756	2257	5059	10571	0.0030
MBV	4	791	2279	4944	11969	0.0027
MP	5	782	2265	5082	10780	0.0030
MPV	5	780	2268	5073	10796	0.0029
MPB	5	771	2231	5107	10642	0.0029
MPBV	5	785	2259	5022	10714	0.0029
L	1	738	2240	5139	10949	0.0064
LV	1	764	2266	5152	10867	0.0060
LB	4	721	2263	4996	11087	0.0032
LBV	4	755	2237	4893	11955	0.0031
LP	3	784	2265	5185	10867	0.0034
LPV	4	789	2241	5151	10926	0.0031
LPB	4	742	2270	5129	11459	0.0031
LPBV	5	772	2251	5069	11592	0.0029
LM	2	739	2214	4879	10848	0.0038
LMV	4	762	2242	4944	11983	0.0031
LMB	4	710	2219	4833	10590	0.0031
LMBV	4	735	2215	4881	11921	0.0030
LMP	4	762	2175	5049	10792	0.0030
LMPV	5	767	2206	5007	10802	0.0029
LMPB	4	730	2135	4987	10646	0.0030
LMPBV	5	761	2211	4934	10733	0.0029

Table 5.71: MRRs and descriptive statistics of the effectiveness measures for the LMPBV corpus and Combined query type for all systems

Combination	min	1Q	median	3Q	max	MRR
V	1	688	2166	4673	11836	0.0147
B	1	672	2068	4954	11598	0.0284
BV	1	697	2018	4838	11759	0.0178
P	1	600	2127	4778	11633	0.0178
PV	1	686	2183	4728	11920	0.0133
PB	1	687	2092	4923	11302	0.0130
PBV	1	699	2012	4907	10796	0.0156
M	1	593	2080	4433	11986	0.0136
MV	1	621	2038	4468	11665	0.0138
MB	1	631	1976	4647	11815	0.0169
MBV	1	675	1946	4589	11690	0.0178
MP	1	643	2045	4379	11987	0.0121
MPV	1	659	1973	4418	11998	0.0140
MPB	1	657	1961	4606	11272	0.0163
MPBV	1	669	1854	4622	11934	0.0179
L	1	654	2029	4773	10907	0.0149
LV	1	624	2149	4854	11878	0.0132
LB	1	635	2041	4741	11335	0.0160
LBV	1	624	2034	4671	11647	0.0173
LP	1	626	2110	4742	10918	0.0118
LPV	1	623	2115	4661	11783	0.0131
LPB	1	651	2069	4667	11973	0.0124
LPBV	1	601	1958	4659	10600	0.0149
LM	1	560	1995	4413	11976	0.0168
LMV	1	573	1963	4455	11580	0.0144
LMB	1	548	1933	4540	11782	0.0174
LMBV	1	559	1860	4583	11445	0.0176
LMP	1	570	2049	4392	11144	0.0167
LMPV	1	588	2035	4398	11873	0.0145
LMPB	1	566	1810	4404	11571	0.0185
LMPBV	1	571	1875	4441	11977	0.0176

Table 5.72: MRRs and descriptive statistics of the effectiveness measures for the LMPBV corpus and Title query type for all systems

Combination	min	1Q	median	3Q	max	MRR
B	1	557	1862	4019	11907	0.0081
S	3	766	1907	4120	10816	0.0033
SB	1	665	2043	4371	11848	0.0075
C	1	570	1857	4414	11794	0.0113
CB	1	605	1823	4168	11930	0.0116
CS	1	688	1735	4302	11643	0.0065
CSB	1	625	2072	4274	11745	0.0107

Table 5.73: MRRs and descriptive statistics of the effectiveness measures for the CSB corpus and Combined query type for all systems

Combination	min	1Q	median	3Q	max	MRR
B	1	563	1765	4019	11868	0.0482
S	1	573	1679	3707	11851	0.0285
SB	1	526	1665	3805	11151	0.0335
C	1	519	2116	4681	11230	0.0360
CB	1	526	2019	4128	11818	0.0497
CS	1	517	1699	4172	11604	0.0314
CSB	1	458	1688	4078	11593	0.0394

Table 5.74: MRRs and descriptive statistics of the effectiveness measures for the CSB corpus and Title query type for all systems

in .0099 from the highest MRR. The combination with the lowest MRR is the combination of the leading comments and the parameters (LP) with an MRR of .0118.

We can see the results of the CSB corpus in Tables 5.73 and 5.74. For the Combined query type, there are three combinations with MRRs higher than .01. These combinations include the comments alone (C), the comments combined with the body (CB), and the full corpus (CSB). Of the three, the highest MRR is for CB with a value of .0116 which is .0003 higher than the MRR

Combination	min	1Q	median	3Q	max	MRR
L	1	689	1964	5039	11732	0.0177
I	1	676	1869	4262	11940	0.0057
IL	1	688	2146	4567	11772	0.0077
C	1	570	1857	4414	11794	0.0113
CL	1	606	1925	4874	11863	0.0066
IC	1	681	1953	4189	11999	0.0098
ICL	1	669	2088	4364	11384	0.0086

Table 5.75: MRRs and descriptive statistics of the effectiveness measures for the ICL corpus and Combined query type for all systems

Combination	min	1Q	median	3Q	max	MRR
L	1	577	2023	4534	11321	0.0552
I	1	521	1858	4132	11367	0.0382
IL	1	573	1913	4161	11744	0.0346
C	1	519	2116	4681	11230	0.0360
CL	1	523	2053	4507	11908	0.0354
IC	1	536	1969	4182	11484	0.0421
ICL	1	533	1843	4165	11211	0.0316

Table 5.76: MRRs and descriptive statistics of the effectiveness measures for the ICL corpus and Title query type for all systems

for C. The lowest MRR for the combinations is for the signature alone (S) which has the highest minimum rank and the highest 1Q of all combinations. Aside from the body comments alone (B) which has a lower MRR than the full corpus, the lowest MRRs are for any combination that includes S. When looking at the Title query type, the highest MRR is also found for CB. This combination is followed by the body comments alone with a difference of .0015 between the two combinations. These are the only two MRRs above .04. The lowest MRR is found for S which is

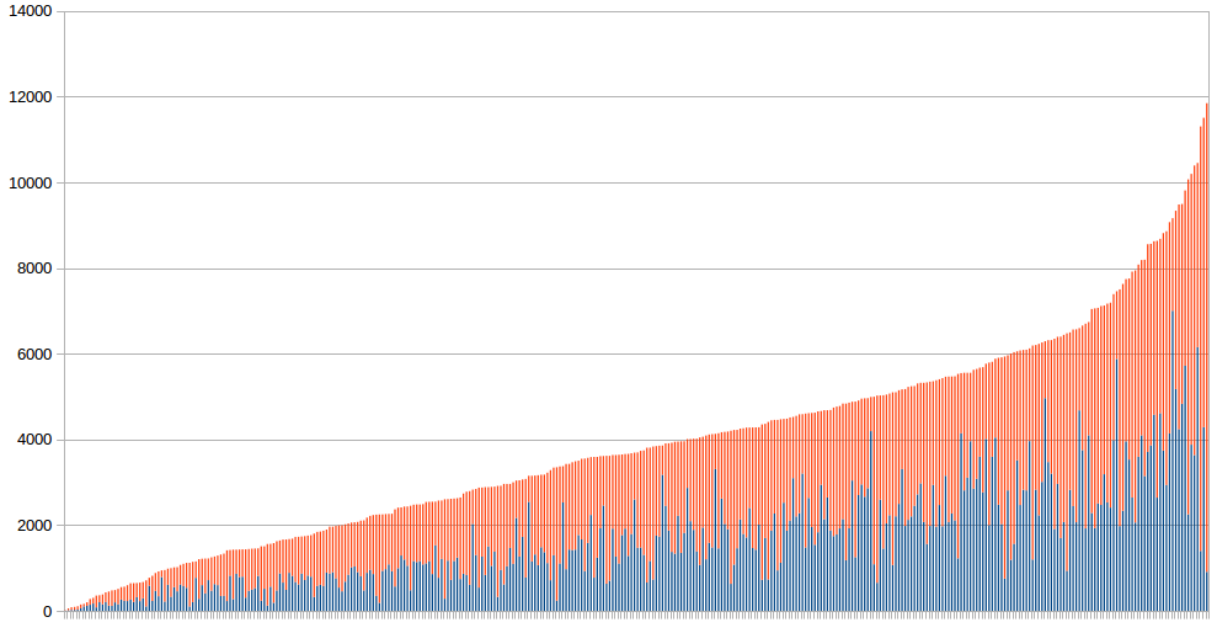
also the only combination with an MRR less than .03. This combination also has the highest 1Q value of all combinations.

Tables 5.75 and 5.76 contain the results for the ICL corpus. When looking at the Combined query type, there are two combinations that are higher than the other combinations. These combinations include the literals alone (L) and the comments alone (C), with L having the higher MRR of the two. C has the lowest 1Q and median values of the combinations. L does not have the lowest of any of the descriptive statistics. The lowest MRR comes from the identifiers alone (I). For the Title query type, the literals alone (L) have the highest MRR of the different combinations with the only MRR higher than .05, however the combination has the highest 1Q of the combinations. The full corpus has the lowest MRR with a difference from L of .0236.

I performed a Friedman test with a wilcoxon post-hoc for each of the corpora between each of the different structural combinations. This analysis was performed for both the Combined and the Title query types. For the Combined query type, significant differences were found for each of the three corpora. For CSB, there were four significant differences found between the combinations. For both LMPBV and ICL, a large proportion of the pairings resulted in significant differences. For the LMPBV corpus, 231 pairs were found to have significant differences. For the ICL corpus, there were 14 significant differences found with significant difference found with every combination being significantly different compared to L. For the Title query type, the CSB corpus had an increase to 7 significant differences, while the number of significant differences for ICL and LMPBV were reduced. For ICL, only 2 significant differences were found with C and I being significantly different from IL. For LMPBV, the total number of pairings found with significant differences is 179.

The results of the best, average, and worst cases for each feature for each corpus and both

(a) CSB



(b) ICL

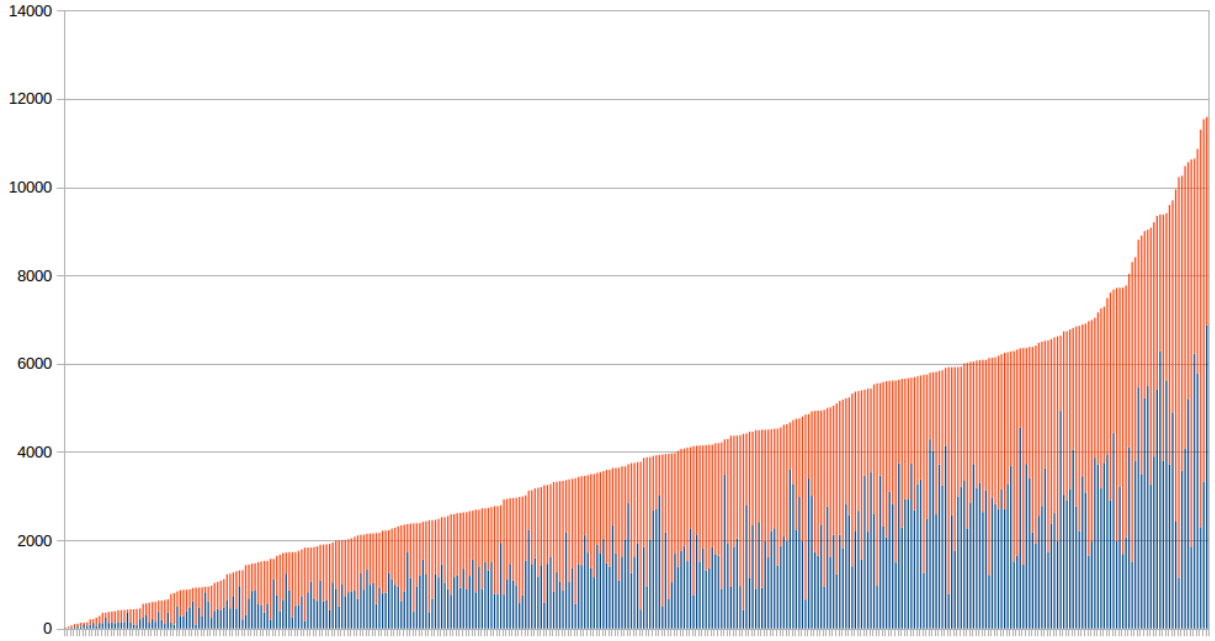


Figure 5.29: Stacked bargraphs representing the distance from the best query from structural combinations to the average (bottom) and the worst(top) for all systems. Graph is ordered by distance from best to worst.

(c) LPMBV

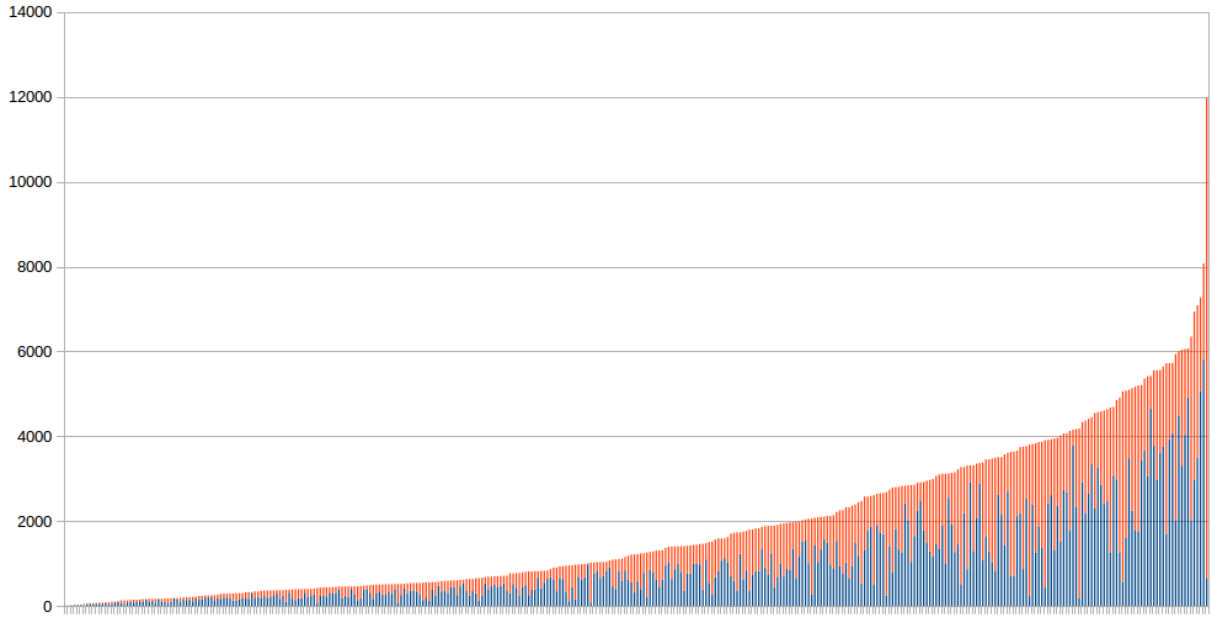


Figure 5.29: Stacked bargraphs representing the distance from the best query from structural combinations to the average (bottom) and the worst(top) for all systems. Graph is ordered by distance from best to worst.

query types can be found in Figure 5.29. The greatest distance between the best query and the worst query for the three corpora is found in the LMPBV corpus at 11,967, while the greatest mean distance between the best and the worst query is found in the ICL corpus at a distance of 4,008. ICL also showed the greatest mean distance between the best query and the average case at 1,737. The smallest mean values in each case were found for the LMPBV corpus with a mean distance between the best and the worst queries of 1,856 and a mean distance between the best and the average case of 962. The smallest distance between the best and worst query is 10 and found in the LMPBV corpus. The smallest distance between the best and the average case is 1 and found in the ICL corpus. For two of the three corpora, the mean distance between the best and the average query is smaller than the mean between the average and the worst. The exception is for the LMPBV corpus.

Query	CSB	ICL	LMPBV
Best	0.1033	0.1180	0.0674
Average	0.0020	0.0031	0.0016
Worst	0.0007	0.0007	0.0011

Table 5.77: MRRs for choosing the best, average, and worst case for each feature for all systems from structural combinations

B	S	SB	C	CB	CS	CSB
30	20	6	22	9	8	2

(a) CSB

L	C	CL	I	IL	IC	ICL
25	23	7	26	6	8	2

(b) ICL

V	B	BV	P	PV	PB	PBV	M	MV	MB	MBV
11	14	1	11	0	1	1	11	1	2	1
MP	MPV	MPB	MPBV	L	LV	LB	LBV	LP	LPV	
1	0	1	1	9	0	1	0	2	0	
LPB	LPBV	LM	LMV	LMB	LMBV	LMP	LMPV	LMPB	LMPBV	
1	0	5	1	4	1	1	0	1	1	

(c) LMPBV

Table 5.78: Percentage of the best queries obtained from each structural combination for all systems

The MRRs have been computed for each case and the results can be found in Table 5.77. Comparing the MRRs between the best cases and the results of the best structural combinations shows large increases from the individual corpora. The largest increase occurs for the ICL corpus

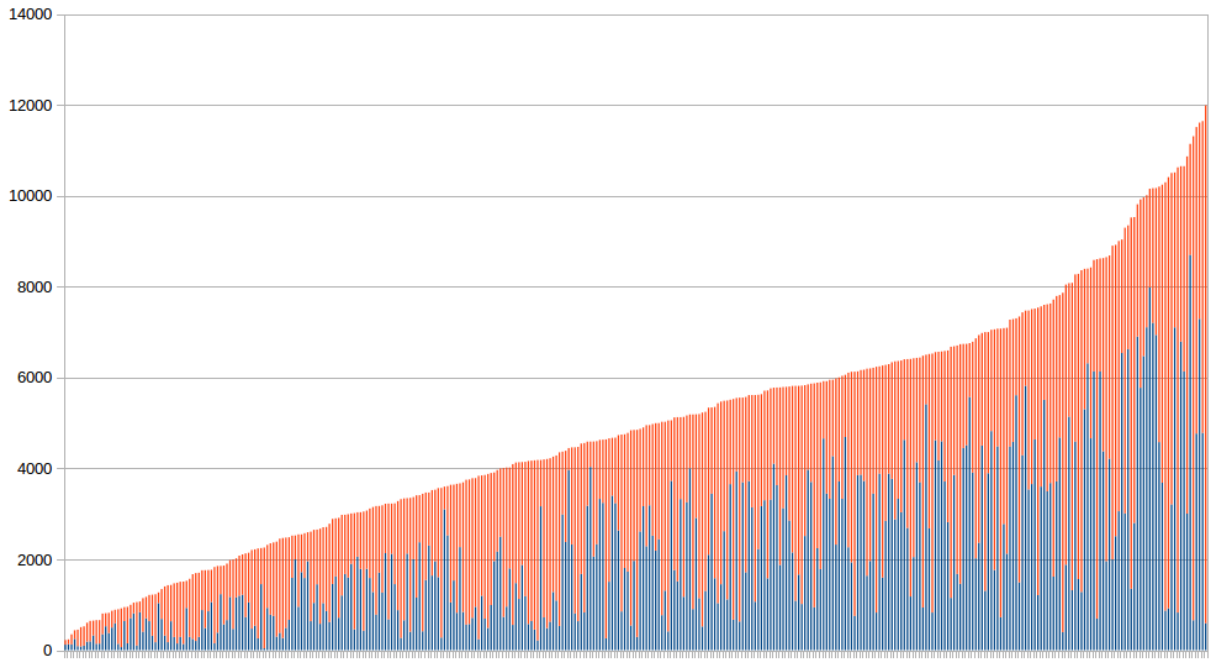


Figure 5.30: Stacked bargraphs representing the distance from the best query to the average (bottom) and the worst(top) from structural combinations for all systems. Graph is ordered by distance from best to worst.

with a difference between the best queries and the L combination of .0628. The smallest difference occurs for the LMPBV corpus with a difference of .0390. Of the three corpora, the largest MRR can also be found for the ICL corpus while the smallest MRR can be found for the LMPBV corpus.

The percentages for all systems combined can be found in Table 5.78. In the case of the CSB corpus, the highest percentage is found for the body combination alone (B) at 30%. The next closest percentage is for the comments alone (C) at 22% followed by the signature alone (S) at 20%. The smallest percentage is found for the full corpus with this combination only representing 2% of the best queries. For both the ICL and LMPBV corpus, the full corpora perform poorly with the ICL corpus only contributing 2% of the best queries and LMPBV only contributing 1% of the best queries for the corpus. Of the combinations in ICL, the identifiers alone (I) outperforms the other combinations with 26%. The next two combinations are the literals alone (L) at 25% and

	Best	Average	Worst
MRR	0.1759	0.0011	0.0003

Table 5.79: MRRs for choosing the best, average, and worst case for each feature from all corpora and all structural combinations for all systems

	CSB	ICL	LMPBV	Flat
Percentage	40	29	16	15

Table 5.80: Percentages for each corpus where the best query was found from all corpora and all structural combinations for all systems

C again with a percentage of 23%. The lowest percentage is found for the full corpus. The highest percentage for LMPBV comes from the body comments alone (B). It should be noted that for each of the three corpora, the highest percentages come from combinations where only one structural lexicon is used.

I performed a Friedman test with a wilcoxon post-hoc for the best, average, and worst case of each feature for each of the four corpora. The results of the analysis showed that there is a significant difference between each case.

In Figure 5.30, you can see the difference between the best, average, and worst cases for each individual feature across all corpora. The greatest distance between the best and the worst query is 11,986. The mean distance between the best and worst query is 4,924 and the mean distance between the best query and the average case is 2,121 while the greatest distance between the best and the average case is 8,698. Each of these values are higher than the values found for the same cases of the individual corpora. The mean distance from the best and the average case is smaller than the mean distance between the average and the worst case.

I computed the MRRs for these three cases and the results can be found in Table 5.79. The

results of this computation show that there is another large increase from the individual corpora to taking the best across all queries. To understand which corpus is contributing the highest number of best queries from the corpora, I computed the percentage for each query and recorded the results in Table 5.80. From this table it can be seen that the CSB corpus contributed the largest percentage of the queries for the features. This corpus was followed by the ICL corpus with a difference of 11%. The corpus with the lowest percentage is the flat corpus. The combinations with the highest percentages from the three corpora include a tie between comments only (C) from ICL and CSB and literals only (L) from ICL with 11% and the body comments only (B) from LMPBV with 4%. None of the full corpora contributed to the best queries.

I performed a Friedman test with a wilcoxon post-hoc for the best, average, and worst case of each feature for each of the four corpora. The results of the analysis showed that there is a significant difference between each case.

5.3.3 Does structural weighting affect the accuracy of a structured retrieval-based FLT?

The purpose of this question is to isolate the weighting from the structural combination in the query, and to determine what type of effects weighting alone can have on the results. I used the full corpus in each case (ICL, CSB, and LMPBV) and both the Combined and the Title query type. This allows me to see if the query type has any effect on the top weightings in each corpus.

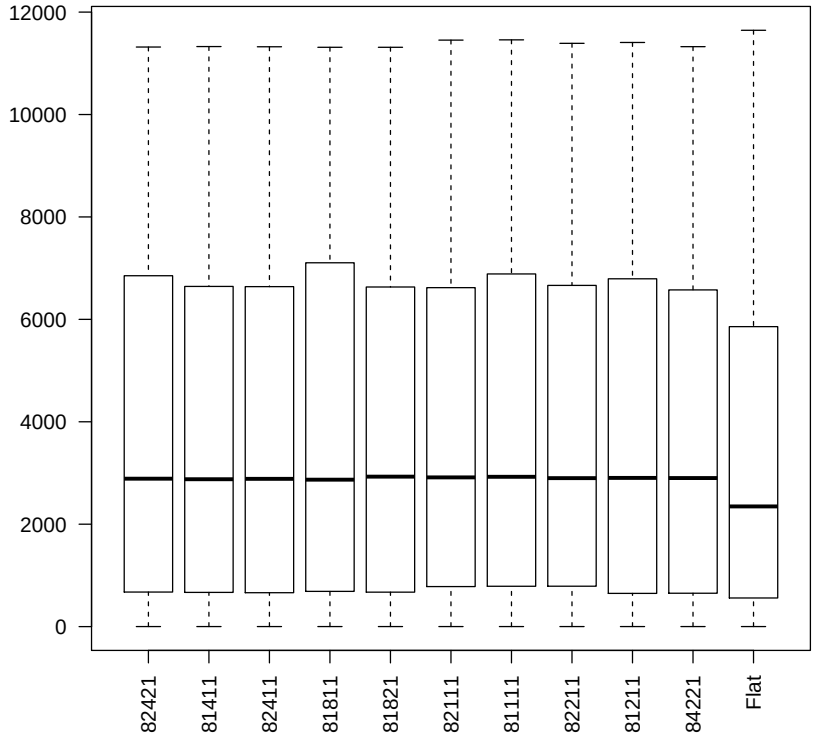
5.3.3.1 ArgoUML

I start by creating boxplots of the effectiveness measures. The boxplots for the LMPBV corpus and each query type can be found in Figure 5.31. These are the boxplots for the ten weighting configuration with the highest MRRs. The resulting MRRs can be found in Table 5.81. All MRRs can be found within a range of .0002 from one another. For each of the top configurations for Combined, the leading comments have the highest weighting factor of 8 while the local vari-

ables have the lowest weighting factor of 1. Looking at the boxplots, while the flat corpus has the highest upper 1.5 IRQ, it has the smallest spread otherwise. The remaining weighting configurations have similar spreads to one another with only slight fluctuations. The Title query type shows a bit more fluctuation and has MRRs with a distance of .0047. While the flat configuration still has a lower 1Q value than the weighted configurations, it has a higher median value than all but two of the configurations. When using the Title query, leading comments have lower weighting factors and now body comments have a weighting factor of 8 in all but one of the top configurations. The weightings factors for parameters are slightly lower, but the weighting factors for the method names and local variables stay similar to their Combined counterparts.

The boxplots for the CSB corpus can be found in Figure 5.32, while the MRRs can be found in Table 5.82. For the top configurations for the Combined query type, there is actually a three-way tie for the top position. The three configurations with a tie actually represent variants of one another as they are all configurations that weight the comments and the body equally. Even though it has a lower MRR, the fourth configuration also falls into this category, however it also weights the signature equally which appears to lower the MRR. Configurations 5, 7, and 9, are also variants of one another with a 2:1 ratio for comments to the body, however these variants do result in different MRRs. Overall, the top configurations are a distance of .0109 from one another. There is fluctuation in the spreads of the top configurations and the flat corpus, and while the flat corpus has some of the lowest values, it does not have the lowest value at any point when compared to the weighting configurations. When looking at the Title query type, the top five configurations do not weight the body, but the top four weight the comments, and the top two give some weighting to the signature. There is a difference of .01 between the configurations. When looking at the boxplots,

(a) LMPBV Combined



(b) LMPBV Title

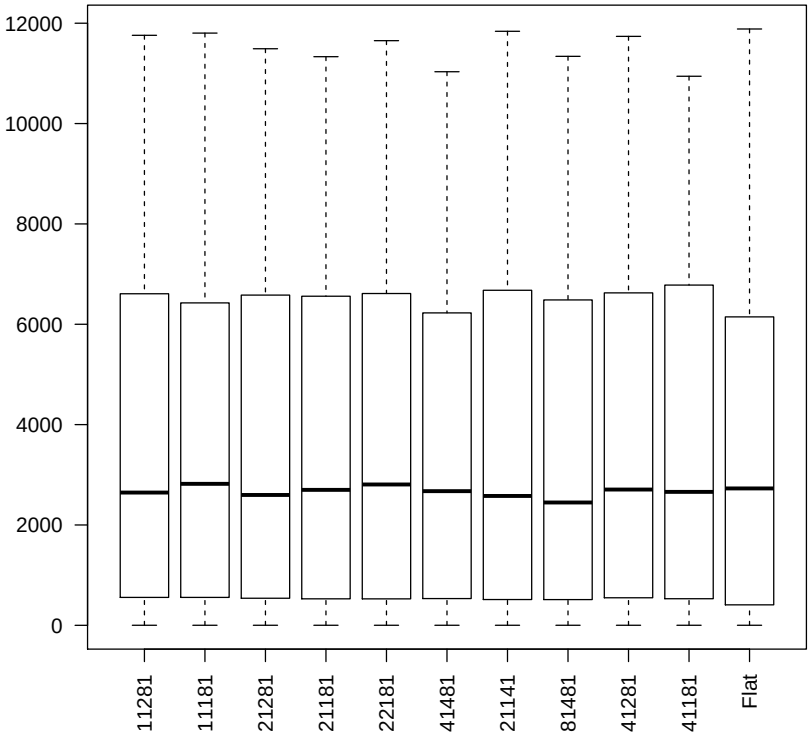


Figure 5.31: The top weighting configurations and flat configuration for ArgoUML and the LMPBV corpus. The x axis is the configuration, while the y axis is the effectiveness measure.

Config	MRR
$C\{L=8,M=2,P=4,B=2,V=1\}$	0.0284
$C\{L=8,M=1,P=4,B=1,V=1\}$	0.0284
$C\{L=8,M=2,P=4,B=1,V=1\}$	0.0284
$C\{L=8,M=1,P=8,B=1,V=1\}$	0.0284
$C\{L=8,M=1,P=8,B=2,V=1\}$	0.0284
$C\{L=8,M=2,P=1,B=1,V=1\}$	0.0284
$C\{L=8,M=1,P=1,B=1,V=1\}$	0.0283
$C\{L=8,M=2,P=2,B=1,V=1\}$	0.0283
$C\{L=8,M=1,P=2,B=1,V=1\}$	0.0283
$C\{L=8,M=4,P=2,B=2,V=1\}$	0.0283

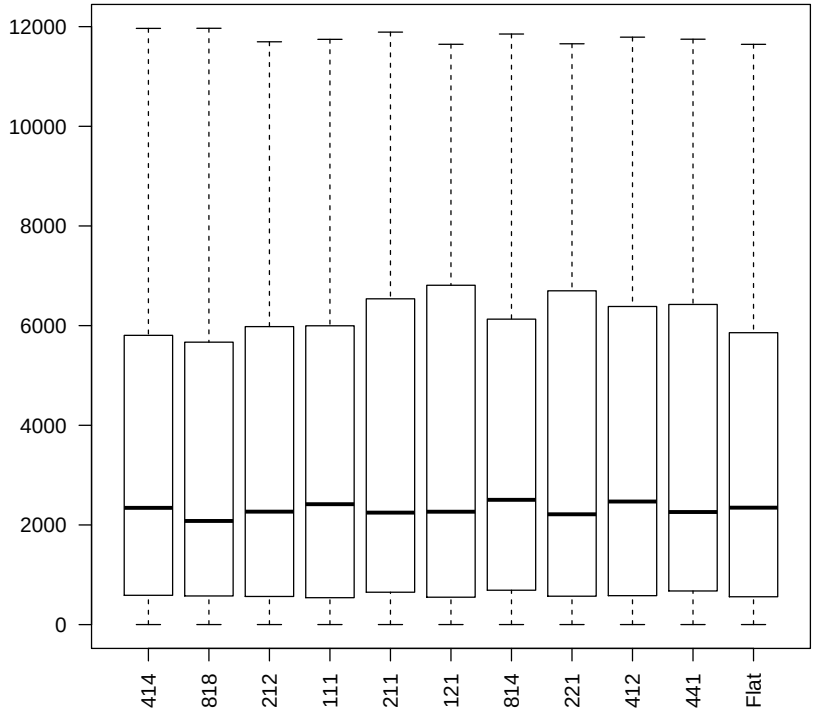
(a) LMPBV Combined

Config	MRR
$C\{L=1,M=1,P=2,B=8,V=1\}$	0.0320
$C\{L=1,M=1,P=1,B=8,V=1\}$	0.0305
$C\{L=2,M=1,P=2,B=8,V=1\}$	0.0281
$C\{L=2,M=1,P=1,B=8,V=1\}$	0.0281
$C\{L=2,M=2,P=1,B=8,V=1\}$	0.0280
$C\{L=4,M=1,P=4,B=8,V=1\}$	0.0276
$C\{L=2,M=1,P=1,B=4,V=1\}$	0.0275
$C\{L=8,M=1,P=4,B=8,V=1\}$	0.0274
$C\{L=4,M=1,P=2,B=8,V=1\}$	0.0273
$C\{L=4,M=1,P=1,B=8,V=1\}$	0.0273

(b) LMPBV Title

Table 5.81: Top 10 configurations for ArgoUML

(a) CSB Combined



(b) CSB Title

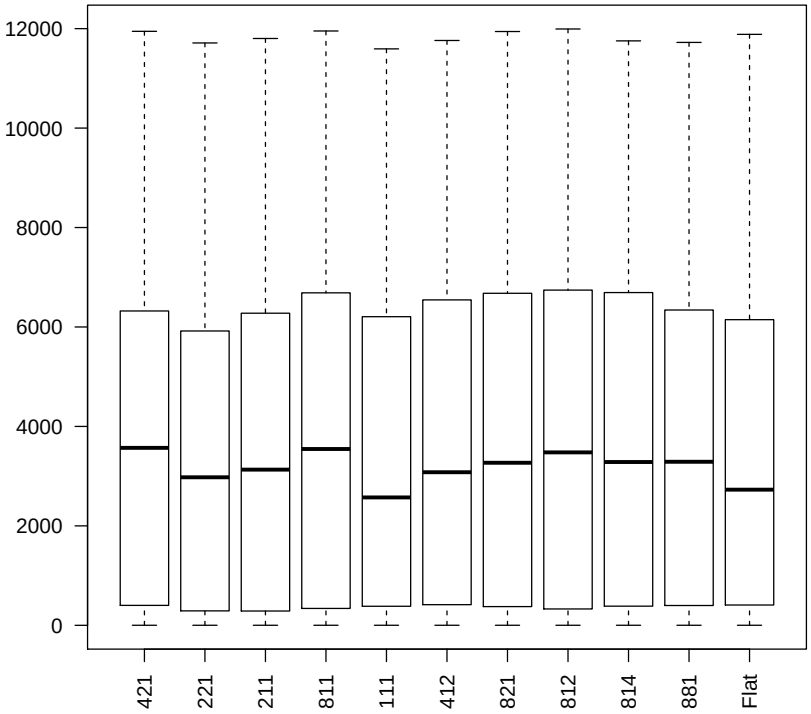


Figure 5.32: The top weighting configurations and flat configuration for ArgoUML and the CSB corpus. The x axis is the configuration, while the y axis is the effectiveness measure.

Config	MRR
$C\{C=4,S=1,B=4\}$	0.0385
$C\{C=8,S=1,B=8\}$	0.0385
$C\{C=2,S=1,B=2\}$	0.0385
$C\{C=1,S=1,B=1\}$	0.0375
$C\{C=2,S=1,B=1\}$	0.0302
$C\{C=1,S=2,B=1\}$	0.0289
$C\{C=8,S=1,B=4\}$	0.0288
$C\{C=2,S=2,B=1\}$	0.0284
$C\{C=4,S=1,B=2\}$	0.0282
$C\{C=4,S=4,B=1\}$	0.0276

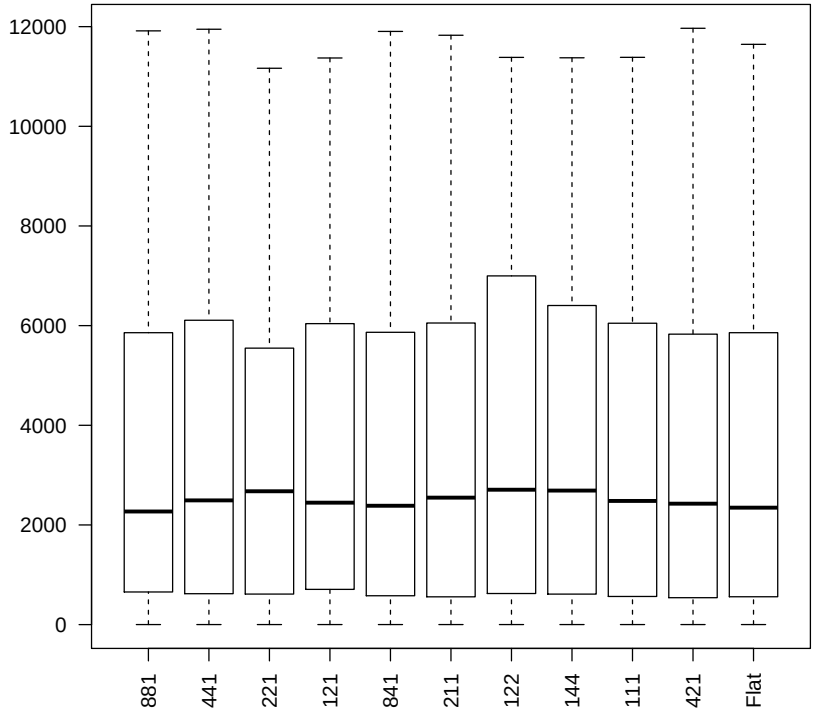
(a) CSB Combined

Config	MRR
$C\{C=4,S=2,B=1\}$	0.0713
$C\{C=2,S=2,B=1\}$	0.0709
$C\{C=2,S=1,B=1\}$	0.0695
$C\{C=8,S=1,B=1\}$	0.0693
$C\{C=1,S=1,B=1\}$	0.0688
$C\{C=4,S=1,B=2\}$	0.0688
$C\{C=8,S=2,B=1\}$	0.0636
$C\{C=8,S=1,B=2\}$	0.0634
$C\{C=8,S=1,B=4\}$	0.0631
$C\{C=8,S=8,B=1\}$	0.0613

(b) CSB Title

Table 5.82: Top 10 configurations for ArgoUML

(a) ICL Combined



(b) ICL Title

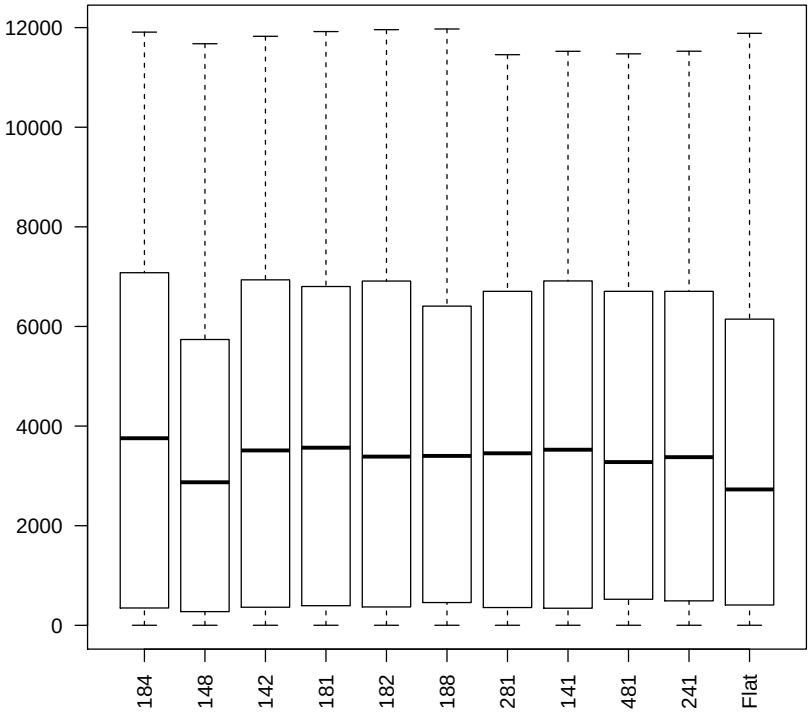


Figure 5.33: The top weighting configurations and flat configuration for ArgoUML and the ICL corpus. The x axis is the configuration, while the y axis is the effectiveness measure.

Config	MRR
$C\{I=8,C=8,L=1\}$	0.0412
$C\{I=4,C=4,L=1\}$	0.0402
$C\{I=2,C=2,L=1\}$	0.0351
$C\{I=1,C=2,L=1\}$	0.0298
$C\{I=8,C=4,L=1\}$	0.0291
$C\{I=2,C=1,L=1\}$	0.0289
$C\{I=1,C=2,L=2\}$	0.0286
$C\{I=1,C=4,L=4\}$	0.0285
$C\{I=1,C=1,L=1\}$	0.0284
$C\{I=4,C=2,L=1\}$	0.0283

(a) ICL Combined

Config	MRR
$C\{I=1,C=8,L=4\}$	0.0613
$C\{I=1,C=4,L=8\}$	0.0609
$C\{I=1,C=4,L=2\}$	0.0601
$C\{I=1,C=8,L=1\}$	0.0588
$C\{I=1,C=8,L=2\}$	0.0550
$C\{I=1,C=8,L=8\}$	0.0543
$C\{I=2,C=8,L=1\}$	0.0537
$C\{I=1,C=4,L=1\}$	0.0508
$C\{I=4,C=8,L=1\}$	0.0496
$C\{I=2,C=4,L=1\}$	0.0494

(b) ICL Title

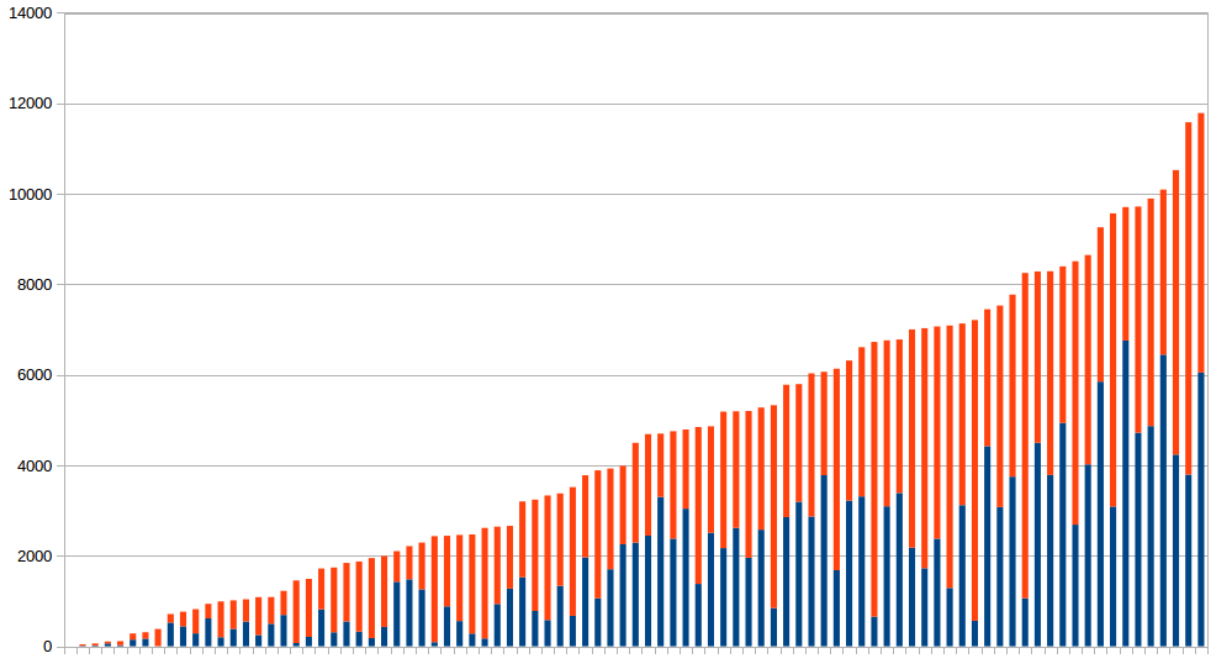
Table 5.83: Top 10 configurations for ArgoUML

we can see that the lowest spreads are given for the unweighted configuration (weighting factor of 1 for all lexicons) and the configuration $C\{C=2,S=2,B=1\}$.

In Figure 5.33 and Table 5.83, you can see the boxplots and MRRs for the ICL corpus. The top three configurations are variations of one another with a 1:1 ratio between the identifiers and the comments, but unlike with the CSB corpus, there is a difference of .0051 between these configurations. There is a difference of .0129 between the top configurations. Of the top configurations, $C\{I=2,C=2,L=1\}$ has the smallest spread. For the Title query type, the top three configurations weight the comments and the literals, with both the top and the third highest configurations having a ratio of 2:1. Each configuration amongst the top gives the comments some weight, while for the top six configurations there is no weighting given to the identifiers. There is a difference of .0119 between the top configurations. Aside from the upper 1.5 IRQ, the smallest spread is for $C\{I=1,C=4,L=8\}$. The second smallest spread is seen for the flat corpus.

I conducted a Friedman test with a post-hoc analysis on the top configurations and the flat corpus. I chose to only look at the top configurations and the flat corpus to lower the number of possible pairings that need to be compared. For the Combined query type, the results of the analysis did not show any significant differences for the CSB corpus, however significant differences were found for both the ICL and the LMPBV corpora. For the ICL corpus, there were 30 significant differences found between the weighting configurations, however of these 30 differences, none were between the flat corpus and the weighting configurations. For LMPBV, only 2 significant differences were found with one between $C\{L=8,M=2,P=4,B=2,V=1\}$ and $C\{L=8,M=1,P=8,B=1,V=1\}$, and the other between $C\{L=8,M=1,P=8,B=1,V=1\}$ and $C\{L=8,M=4,P=2,B=2,V=1\}$. For the Title query type, each of the three corpora had significant differences. For the CSB corpus, there was a total of 21 significant differences discovered with

(a) CSB



(b) ICL

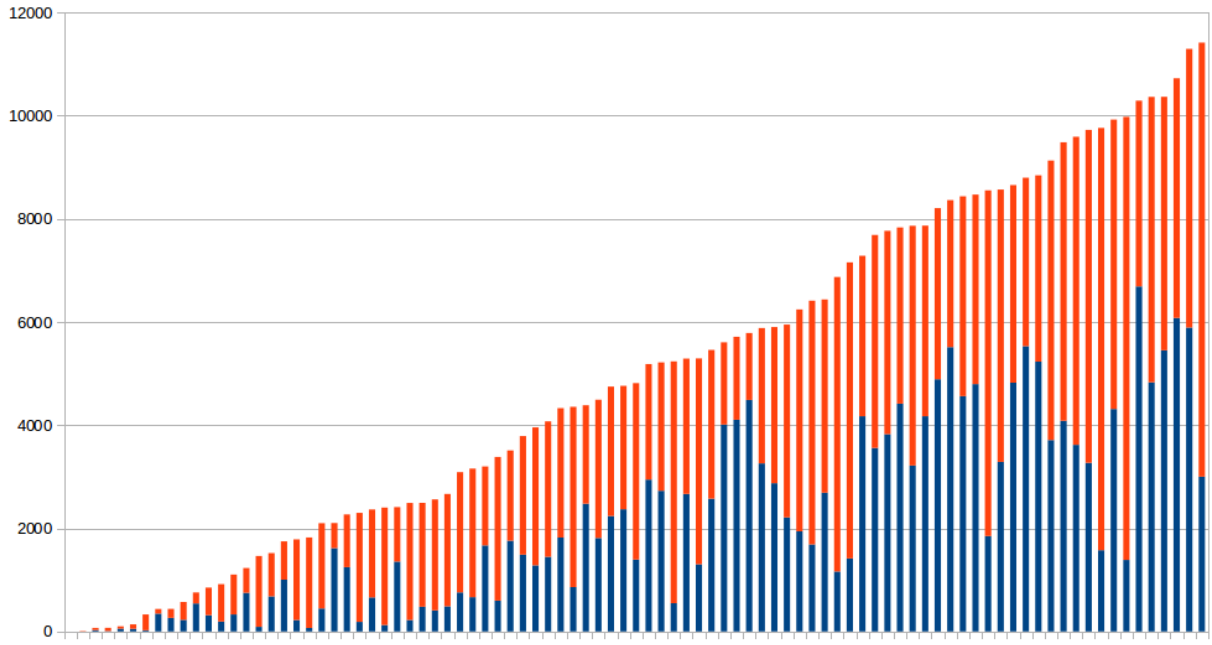


Figure 5.34: Stacked bargraphs representing the distance from the best query from structural weighting to the average (bottom) and the worst(top) for ArgoUML. Graph is ordered by distance from best to worst.

(c) LPMBV

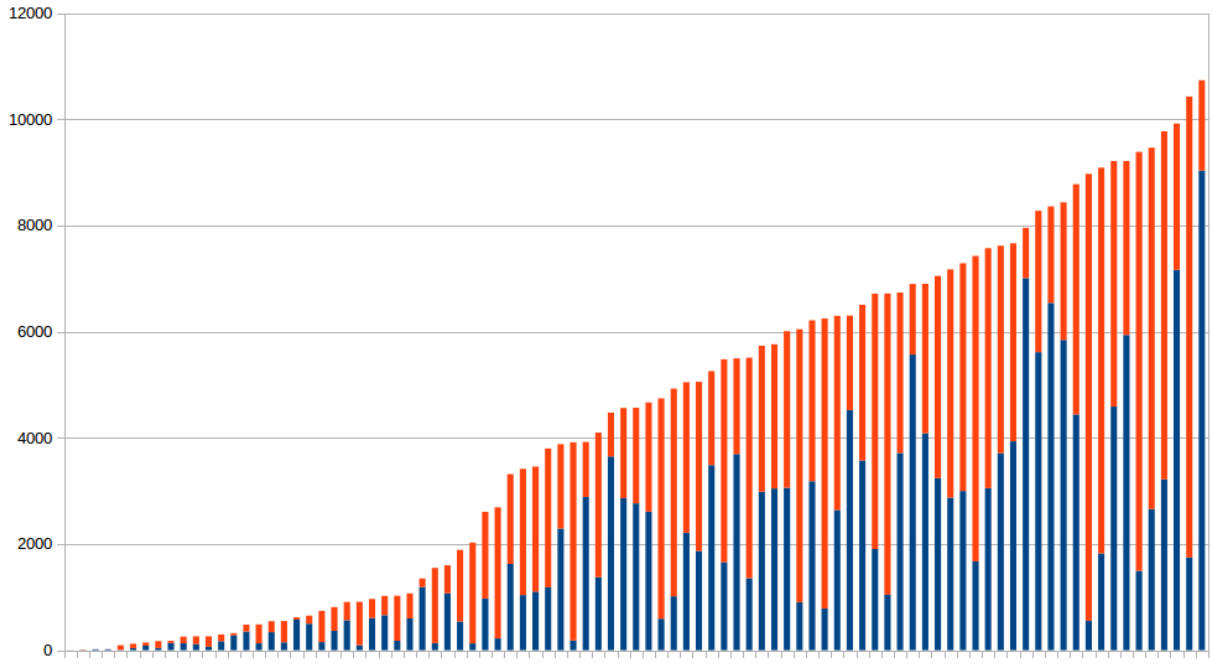


Figure 5.34: Stacked bargraphs representing the distance from the best query from structural weighting to the average (bottom) and the worst(top) for ArgoUML. Graph is ordered by distance from best to worst.

no significant differences betweent he flat corpus and the weighting configurations. For ICL, there were 6 significant differences, the flat corpus was found to be significantly different from the $C\{I = 1, C = 8, L = 4\}$ and $C\{I = 1, C = 4, L = 2\}$ configurations. For the LMPBV corpus, 11 significant differences were found with no significant differences between the flat corpus and the weighting configurations.

I computed the best, worst, and average cases for each feature for each corpus using both query types. The differences between each case can be found in Figure 5.34. The greatest distance between the best query and the worst query for the three corpora is found in the CSB corpus at 11,785, while the greatest mean distance between the best and the worst query is found in the ICL corpus at a distance of 5,015. ICL also showed the greatest mean distance between the best query and the average case at 2,147. The smallest mean distance between the best and the worst

Query	CSB	ICL	LMPBV
Best	0.1109	0.1123	0.0463
Average	0.0168	0.0249	0.0049
Worst	0.0119	0.0127	0.0025

Table 5.84: MRRs for choosing the best, average, and worst case for each feature for ArgoUML from structural weighting

	CSB	ICL	LMPBV
Weighted	97	97	97

Table 5.85: The percentage of time that weighting each corpus improved the results for ArgoUML

case is found for the LMPBV with a mean distance of 4,340. The smallest mean distance between the best and the average case was found for the CSB corpus with a mean distance of 1,898. The smallest distances are found for both the ICL and CSB corpus with a distance of 0. For each of the three corpora, the mean distance between the best and the average query is smaller than the mean between the average and the worst.

The MRRs have been computed for each case and the results can be found in Table 5.84. Comparing the MRRs between the best cases and the results of the best structural combinations shows large increases from the individual corpora. The largest increase occurs for the ICL corpus with a difference between the best queries and the top weighting configuration of .0510. The smallest difference occurs for the LMPBV corpus with a difference of only .0143. Of the three corpora, the largest MRR can also be found for the ICL corpus while the smallest MRR can be found for the LMPBV corpus.

I also computed the percentage of times that the best query was a result of one of the

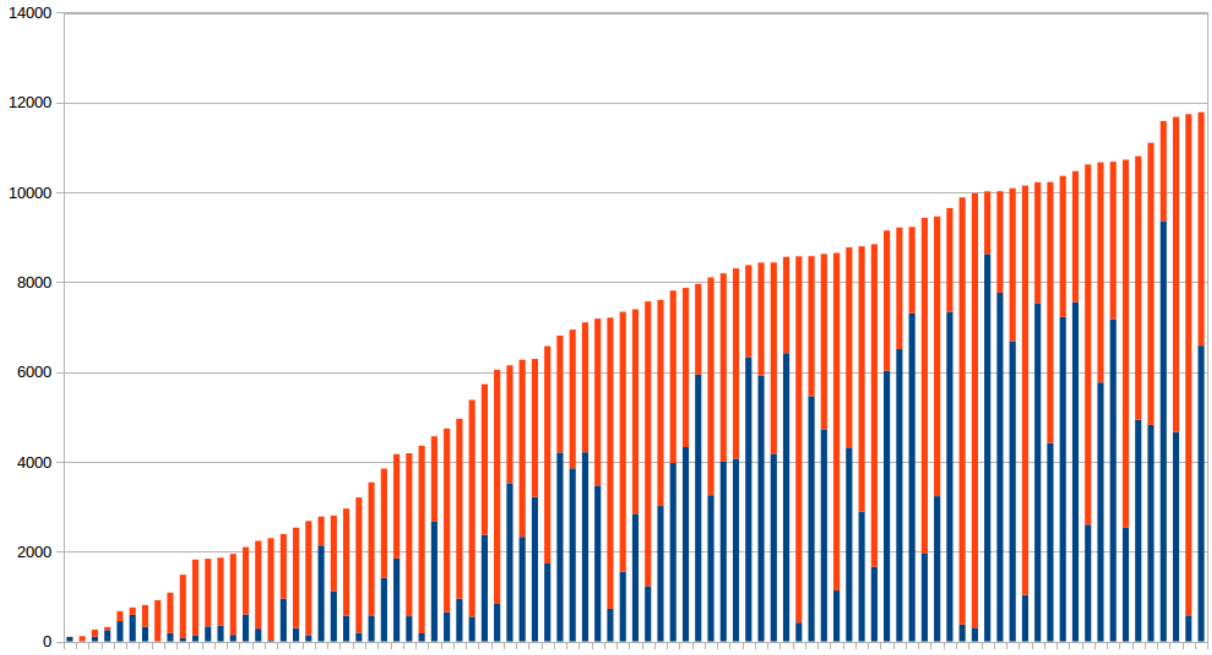


Figure 5.35: Stacked bargraphs representing the distance from the best query to the average (bottom) and the worst(top) from structural weighting for ArgoUML. Graph is ordered by distance from best to worst.

weighting configurations. This helps to understand whether the weighted or unweighted configurations are having a bigger impact on the results for each corpus. The results of this calculation can be found in Table 5.85. For each of the three corpora, the weighting configurations had the best queries 97% of the time, indicating that the unweighted configuration only had 3% of the best queries in any corpus.

I performed a Friedman test with a wilcoxon post-hoc for the best, average, and worst case of each feature for each of the four corpora. The results of the analysis showed that there is a significant difference between each case.

I computed the best, worst, and average cases for each feature across all corpora using both query types. The differences between each case can be found in Figure 5.35. The greatest distance between the best query and the worst query for the three corpora is 11,785, while the

	Best	Average	Worst
MRR	0.1146	0.0035	0.0005

Table 5.86: MRRs for choosing the best, average, and worst case for each feature from all corpora and all structural weighting for ArgoUML

	CSB	ICL	LMPBV	Flat
Percentage	33	23	17	27

Table 5.87: Percentages for each corpus where the best query was found from all corpora and all structural weighting for ArgoUML

mean distance between the best and the worst query is 6,525. The mean distance between the best query and the average case was 2,855. The smallest distance betweent the best and worst case was 105, while the smallest distance betweent the best and average cases was found to be 97. Each of these values are greater than the values for the individual corpora. The mean distance between the best and the average query is smaller than the mean between the average and the worst.

I computed the MRRs for these three cases and the results can be found in Table 5.86. Surprisingly, the results of this computation only show a small increase, of only .0026, from the individual corpora to taking the best across all queries. To understand which corpus is contributing the highest number of best queries from the corpora, I computed the percentage for each query and recorded the results in Table 5.87. Despite the small increase, there is a balanced spread across the four corpora. From this table it can be seen that the CSB corpus contributed the largest percentage of the queries for the features. This corpus was followed by the flat corpus with a difference of 6%. The corpus with the lowest percentage is the LMPBV corpus. A weighting configurations was used for 73%, indicating that the only queries not weighted were those coming from the flat corpus.

I performed a Friedman test with a wilcoxon post-hoc for the best, average, and worst case of each feature for each of the four corpora. The results of the analysis showed that there is a significant difference between each case.

5.3.3.2 JabRef

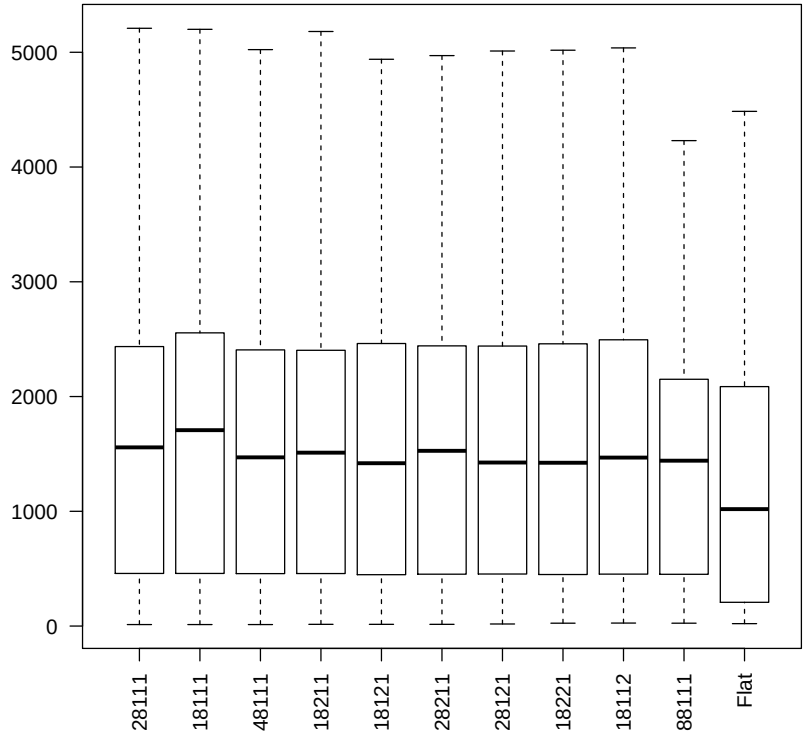
The boxplots for the LMPBV corpus and each query type can be found in Figure 5.36. These are the boxplots for the ten weighting configuration with the highest MRRs. The resulting MRRs can be found in Table 5.88. All MRRs can be found within a range of .001 from one another. For each of the top configurations for Combined, the method names have the highest weighting factor of 8 while the local variables have the lowest weighting factor of 1 for all but one configuration. Looking at the boxplots, while the flat corpus has a slightly higher upper 1.5 IRQ when compared to $C\{L=1,M=8,P=1,B=1,V=2\}$, it has the smallest spread otherwise. The remaining weighting configurations have similar spreads to one another except for $C\{L=1,M=8,P=1,B=1,V=2\}$ which has the smallest spread of the weighting configurations. The Title query type shows a bit more fluctuation and has MRRs with a distance of .0027. The flat configuration actually has the highest spread of the configurations. When using the Title query, method names still have a weighting factor of 8 in seven of the configurations. Local variables continue to have low weighting factors, as well as parameters and body comments, however in several cases the weighting factors on leading comments have increased.

The boxplots for the CSB corpus can be found in Figure 5.37, while the MRRs can be found in Table 5.89. For the top configurations for the Combined query type, the difference between the configurations is .0014. Amongst the weighting configurations, in nine out of the ten configurations, the signature is not weighted, however the comments and the body are. The highest performing ratio between comments and the body is a 1:1 ratio with higher weighting factors

outperforming smaller ones. Another common ratio is 2:1, seen in three of the ten top configurations with the emphasis being on the comments. Of the configurations, the flat configuration has he second smallest spread behind $C\{C = 8, S = 1, B = 2\}$. However, $C\{C = 8, S = 1, B = 4\}$ has a similar spread with the lowest median value. For the Title query type, there is a difference of .0417 between the configurations. For the top five configurations, the comments are given a weighting factor of 8. While the signature is not weighted in seven of the configurations, for the top configuration, the signature is given a weighting factor of 4. The remaining two weighting factors for the signature are 2. The body is weighted with the comments in five of the ten configurations. The spreads of the different configurations are similar except for a large amount of fluctuation around the median. The flat configuration has the largest spread of all configurations, while $C\{C = 2, S = 2, B = 1\}$ has the lowest median value, and $C\{C = 8, S = 4, B = 1\}$ has the lowest 1Q value.

In Figure 5.38 and Table 5.90, you can see the boxplots and MRRs for the ICL corpus. For the Combined query type, there is a difference between the configurations of only .0003. For each of the configurations, the comments have been weighted with a weighting factor of 8 used in six of the top ten configurations. This includes the top five configurations. The top two configurations give a smaller weighting factor to identifiers, with the top configuration having a ratio of 1:2 between the identifiers and the commments, and the second configuration having a ratio of 1:4. Three of the top configurations only weight the comments and leave both the identifiers and the literals unweighted. Of all configurations, literals only receive a weighting factor in three configurations. Despite the small difference in MRR, there is fluctuation in the size of the spreads with the smallest spread being for $C\{I = 1, C = 8, L = 1\}$. When looking at the Title query type, there is a difference between the top configurations of .0253. Comments are heavily weighted

(a) LMPBV Combined



(b) LMPBV Title

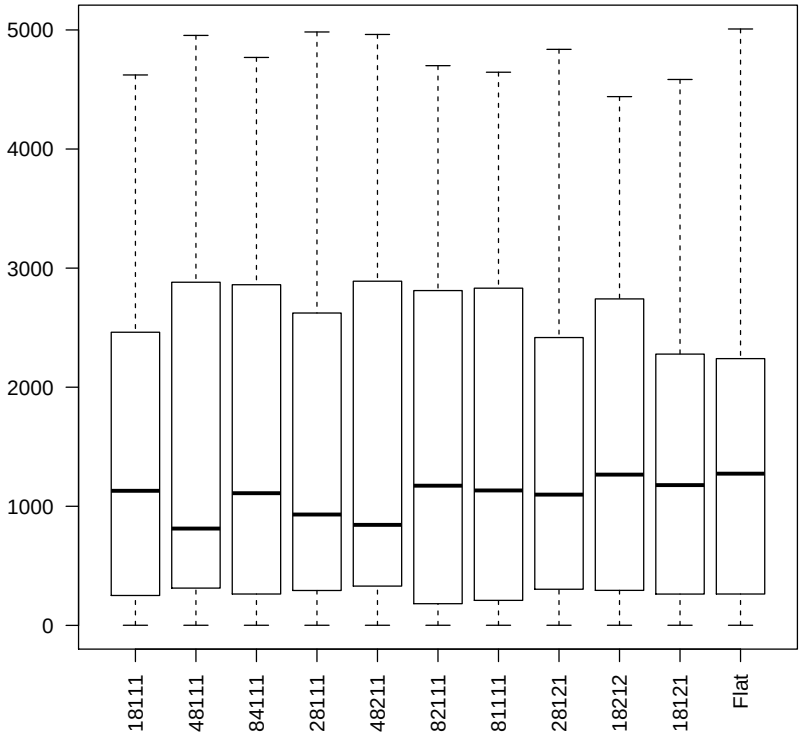


Figure 5.36: The top weighting configurations and flat configuration for JabRef and the LMPBV corpus. The x axis is the configuration, while the y axis is the effectiveness measure.

Config	MRR
$C\{L=2,M=8,P=1,B=1,V=1\}$	0.0050
$C\{L=1,M=8,P=1,B=1,V=1\}$	0.0050
$C\{L=4,M=8,P=1,B=1,V=1\}$	0.0050
$C\{L=1,M=8,P=2,B=1,V=1\}$	0.0048
$C\{L=1,M=8,P=1,B=2,V=1\}$	0.0047
$C\{L=2,M=8,P=2,B=1,V=1\}$	0.0047
$C\{L=2,M=8,P=1,B=2,V=1\}$	0.0044
$C\{L=1,M=8,P=2,B=2,V=1\}$	0.0040
$C\{L=1,M=8,P=1,B=1,V=2\}$	0.0040
$C\{L=8,M=8,P=1,B=1,V=1\}$	0.0040

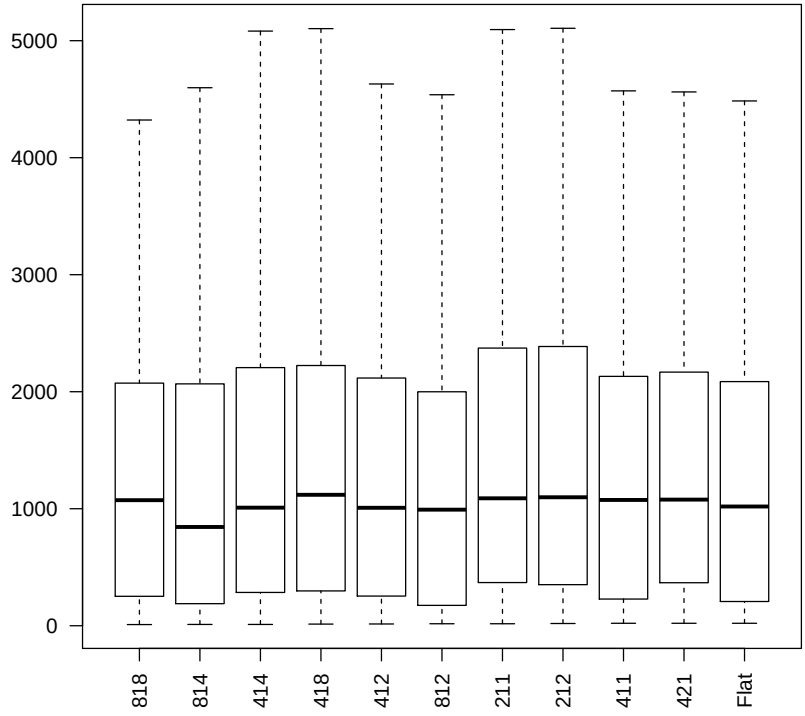
(a) LMPBV Combined

Config	MRR
$C\{L=1,M=8,P=1,B=1,V=1\}$	0.0597
$C\{L=4,M=8,P=1,B=1,V=1\}$	0.0597
$C\{L=8,M=4,P=1,B=1,V=1\}$	0.0593
$C\{L=2,M=8,P=1,B=1,V=1\}$	0.0590
$C\{L=4,M=8,P=2,B=1,V=1\}$	0.0589
$C\{L=8,M=2,P=1,B=1,V=1\}$	0.0587
$C\{L=8,M=1,P=1,B=1,V=1\}$	0.0586
$C\{L=2,M=8,P=1,B=2,V=1\}$	0.0581
$C\{L=1,M=8,P=2,B=1,V=2\}$	0.0577
$C\{L=1,M=8,P=1,B=2,V=1\}$	0.0570

(b) LMPBV Title

Table 5.88: Top 10 configurations for JabRef

(a) CSB Combined



(b) CSB Title

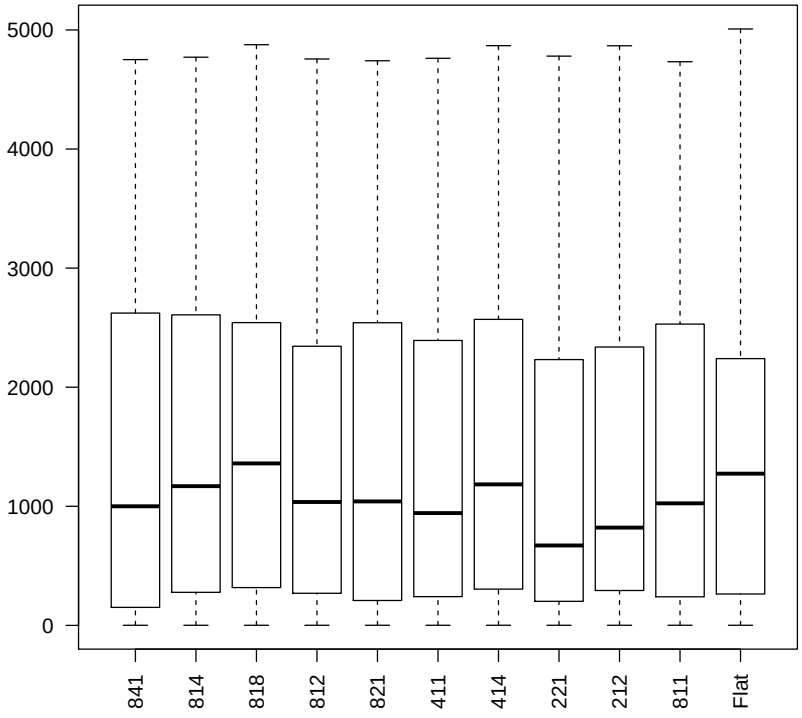


Figure 5.37: The top weighting configurations and flat configuration for JabRef and the CSB corpus. The x axis is the configuration, while the y axis is the effectiveness measure.

Config	MRR
$C\{C=8,S=1,B=8\}$	0.0061
$C\{C=8,S=1,B=4\}$	0.0060
$C\{C=4,S=1,B=4\}$	0.0057
$C\{C=4,S=1,B=8\}$	0.0053
$C\{C=4,S=1,B=2\}$	0.0052
$C\{C=8,S=1,B=2\}$	0.0052
$C\{C=2,S=1,B=1\}$	0.0048
$C\{C=2,S=1,B=2\}$	0.0047
$C\{C=4,S=1,B=1\}$	0.0047
$C\{C=4,S=2,B=1\}$	0.0047

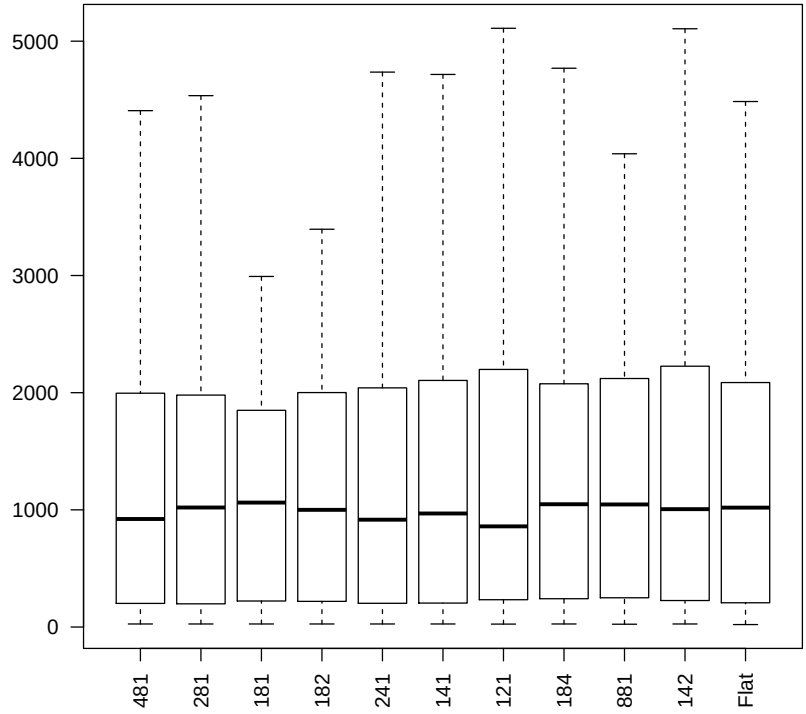
(a) CSB Combined

Config	MRR
$C\{C=8,S=4,B=1\}$	0.1069
$C\{C=8,S=1,B=4\}$	0.0926
$C\{C=8,S=1,B=8\}$	0.0874
$C\{C=8,S=1,B=2\}$	0.0862
$C\{C=8,S=2,B=1\}$	0.0849
$C\{C=4,S=1,B=1\}$	0.0835
$C\{C=4,S=1,B=4\}$	0.0830
$C\{C=2,S=2,B=1\}$	0.0814
$C\{C=2,S=1,B=2\}$	0.0814
$C\{C=8,S=1,B=1\}$	0.0652

(b) CSB Title

Table 5.89: Top 10 configurations for JabRef

(a) ICL Combined



(b) ICL Title

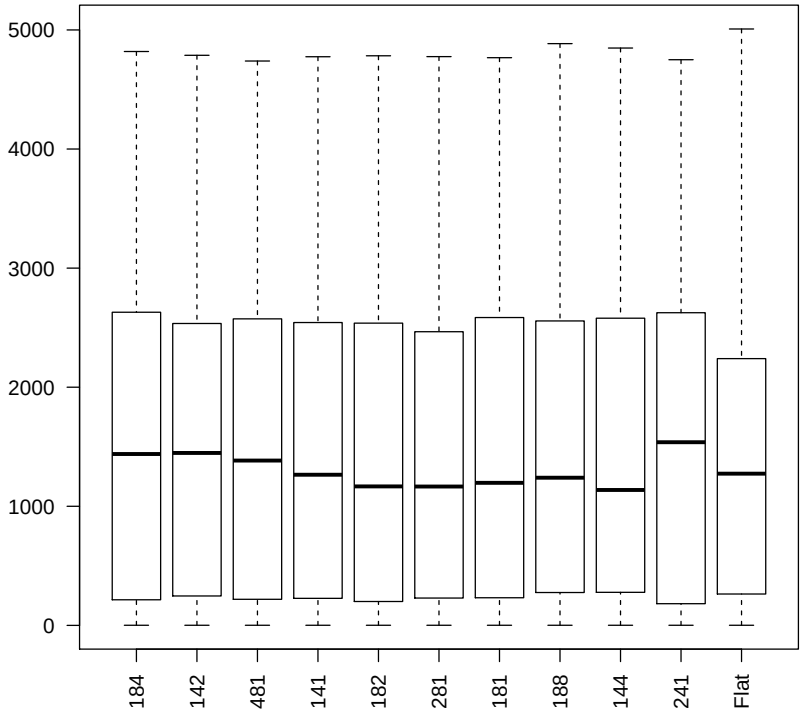


Figure 5.38: The top weighting configurations and flat configuration for JabRef and the ICL corpus. The x axis is the configuration, while the y axis is the effectiveness measure.

Config	MRR
$C\{I=4,C=8,L=1\}$	0.0045
$C\{I=2,C=8,L=1\}$	0.0045
$C\{I=1,C=8,L=1\}$	0.0044
$C\{I=1,C=8,L=2\}$	0.0044
$C\{I=2,C=4,L=1\}$	0.0044
$C\{I=1,C=4,L=1\}$	0.0043
$C\{I=1,C=2,L=1\}$	0.0042
$C\{I=1,C=8,L=4\}$	0.0042
$C\{I=8,C=8,L=1\}$	0.0042
$C\{I=1,C=4,L=2\}$	0.0042

(a) ICL Combined

Config	MRR
$C\{I=1,C=8,L=4\}$	0.0847
$C\{I=1,C=4,L=2\}$	0.0845
$C\{I=4,C=8,L=1\}$	0.0835
$C\{I=1,C=4,L=1\}$	0.0833
$C\{I=1,C=8,L=2\}$	0.0671
$C\{I=2,C=8,L=1\}$	0.0667
$C\{I=1,C=8,L=1\}$	0.0651
$C\{I=1,C=8,L=8\}$	0.0616
$C\{I=1,C=4,L=4\}$	0.0601
$C\{I=2,C=4,L=1\}$	0.0594

(b) ICL Title

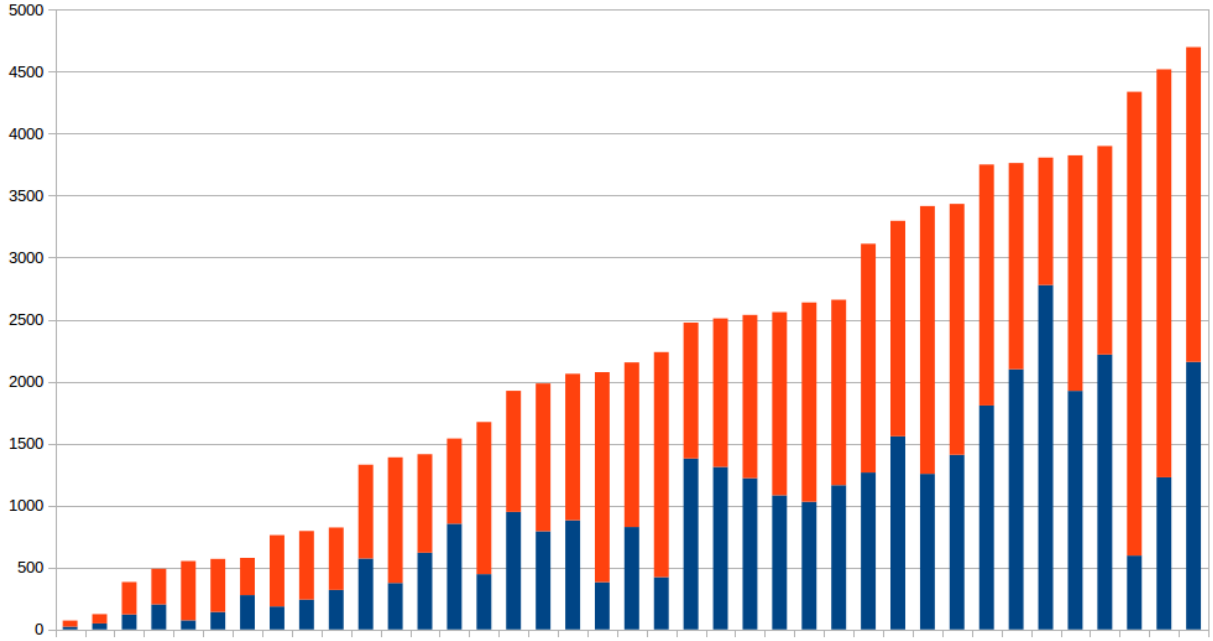
Table 5.90: Top 10 configurations for JabRef

again, but literals have been given higher weighting factors. In the top configuration, there is a 2:1 ratio between the comments and the literals. The top configuration from the Combined query type is still listed, but has dropped down to the third configuration. There are similar spreads for the top configurations of the Title query type, with the flat configuration having the largest spread, but the lowest 3Q value.

I conducted a Friedman test with a post-hoc analysis on the top configurations and the flat corpus. I chose to only look at the top configurations and the flat corpus to lower the number of possible pairings that need to be compared. For the Combined query type, the results of the analysis did not show any significant differences for the CSB of the ICL corpus, however significant differences were found for the LMPBV corpora. There were a total of 5 different significant differences identified for the LMPBV corpus, with 3 of the differences being between the flat corpus and the top four weighting configurations for the corpus. The only significant difference found between the weighting configurations was identified between $C\{L = 1, M = 8, P = 1, B = 1, V = 1\}$ and $C\{L = 1, M = 8, P = 1, B = 1, V = 2\}$. For the Title query type, there were no significant differences identified for either the ICL or the LMPBV corpus. For the CSB corpus there was only a single significant difference found. This significant difference was identified between the configurations $C\{C = 2, S = 2, B = 1\}$ and $C\{C = 8, S = 1, B = 1\}$.

I computed the best, worst, and average cases for each feature for each corpus using both query types. The differences between each case can be found in Figure 5.39. The greatest distance between the best query and the worst query for the three corpora is found in the CSB corpus at 4,697, but this distance is less than 30 higher than the greatest distance for the ICL corpus. The greatest mean distance between the best and the worst query is also found in the CSB corpus at a mean distance of 2,209. CSB also showed the greatest mean distance between the best query and

(a) CSB



(b) ICL

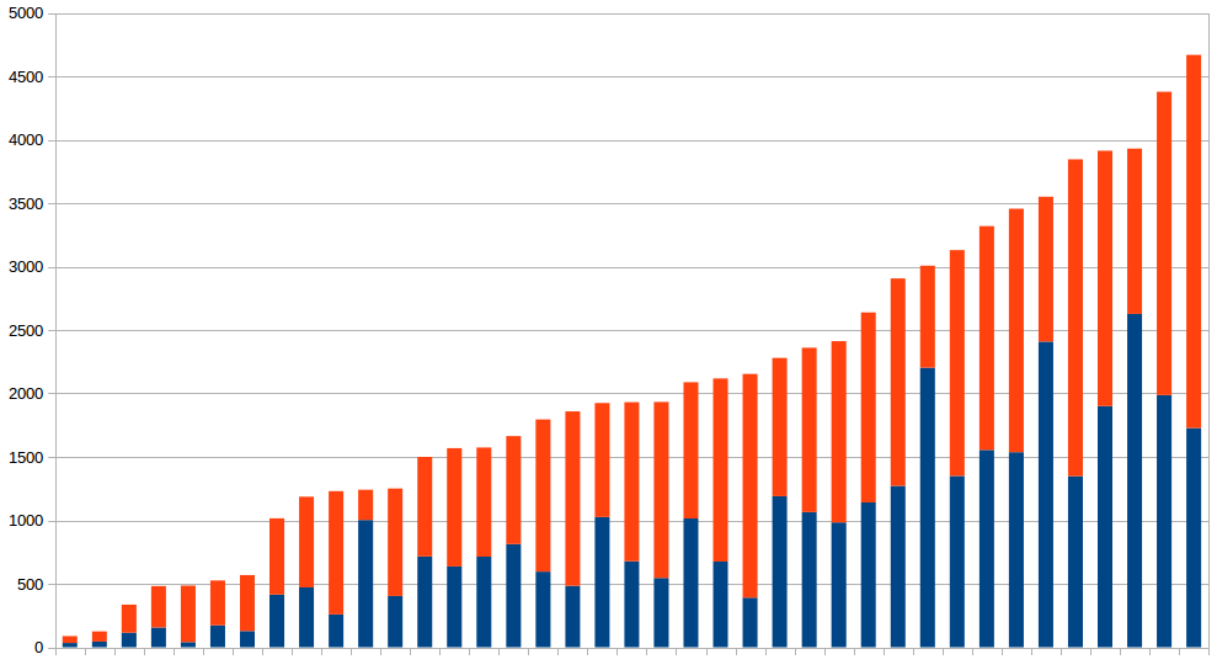


Figure 5.39: Stacked bargraphs representing the distance from the best query from structural weighting to the average (bottom) and the worst(top) for JabRef. Graph is ordered by distance from best to worst.

(c) LPMBV

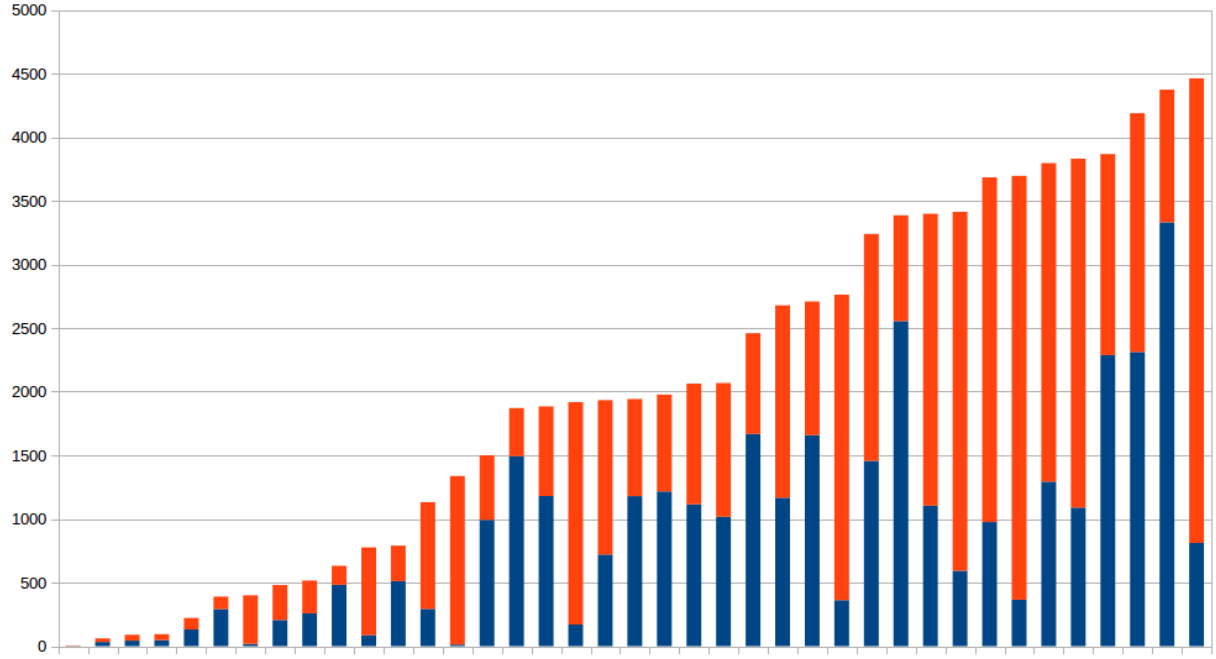


Figure 5.39: Stacked bargraphs representing the distance from the best query from structural weighting to the average (bottom) and the worst(top) for JabRef. Graph is ordered by distance from best to worst.

the average case at 928, a value that is only 10 higher than the mean distance between the best query and the average case for ICL. The smallest mean distance between the best and the worst case is found for the LMPBV with a mean distance of 2,051. This corpus also had the smallest mean distance between the best and the average case with a mean distance of 885. In addition the smallest mean distances, the smallest distances were found in the LMPBV corpus with a distance between the best and the worst queries of 6 and a distance between the best query and the average case of 2. For each of the three corpora, the mean distance between the best and the average query is smaller than the mean between the average and the worst.

The MRRs have been computed for each case and the results can be found in Table 5.91. Comparing the MRRs between the best cases and the results of the best structural combinations shows large increases from the individual corpora. The largest increase occurs for the ICL corpus

Query	CSB	ICL	LMPBV
Best	0.1545	0.1464	0.1180
Average	0.0030	0.0029	0.0034
Worst	0.0011	0.0010	0.0014

Table 5.91: MRRs for choosing the best, average, and worst case for each feature for JabRef from structural weighting

	CSB	ICL	LMPBV
Weighted	95	95	90

Table 5.92: The percentage of time that weighting each corpus improved the results for JabRef

with a difference between the best queries and the top weighting configuration of .0617. The smallest difference occurs for the CSB corpus with a difference of .0476. Of the three corpora, the largest MRR can also be found for the CSB corpus while the smallest MRR can be found for the LMPBV corpus.

I also computed the percentage of times that the best query was a result of one of the weighting configurations. This helps to understand whether the weighted or unweighted configurations are having a bigger impact on the results for each corpus. The results of this calculation can be found in Table 5.92. For both the CSB and the ICL corpora, the unweighted configuration has the best queries only 5% of the time. The percentage for the LMPBV corpus is not much higher. The percentage of time that the unweighted configuration has the best queries is 10% for this corpus.

I performed a Friedman test with a wilcoxon post-hoc for the best, average, and worst case

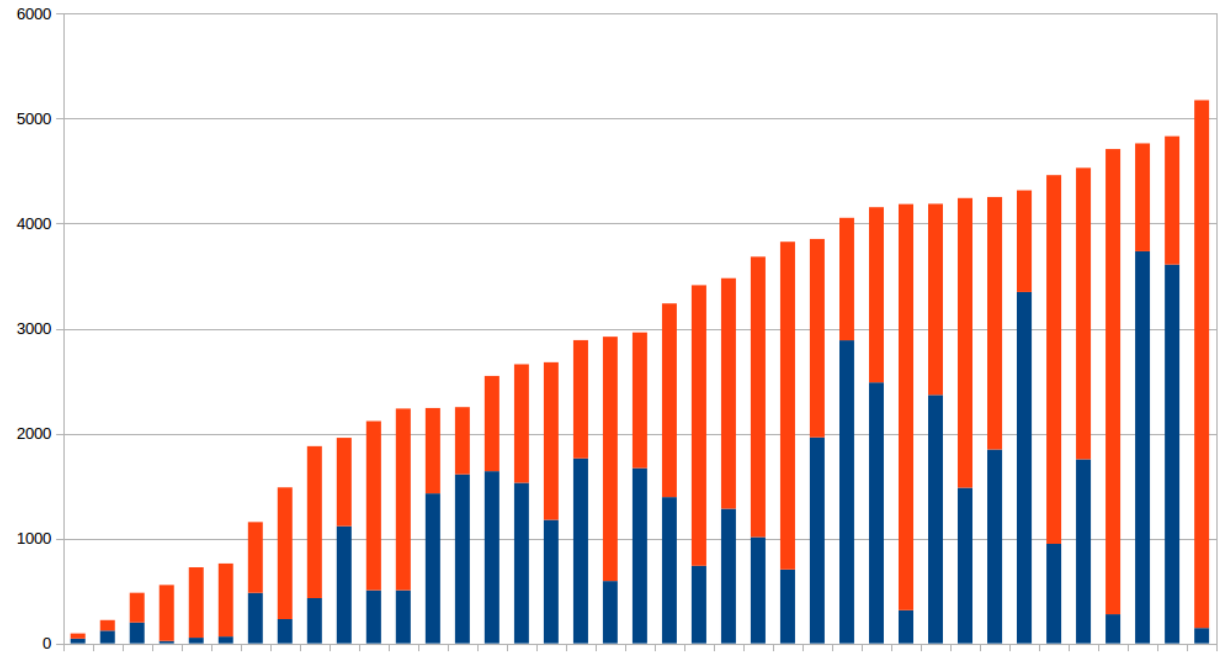


Figure 5.40: Stacked bargraphs representing the distance from the best query to the average (bottom) and the worst(top) from structural weighting for JabRef. Graph is ordered by distance from best to worst.

of each feature for each of the four corpora. The results of the analysis showed that there is a significant difference between each case.

I computed the best, worst, and average cases for each feature across all corpora using both query types. The differences between each case can be found in Figure 5.40. The greatest distance between the best query and the worst query for the three corpora is 5,176, while the mean distance between the best and the worst query is 2,928. The mean distance between the best query and the average case was 1,217. The smallest distance betweent the best and worst case was 95, while the smallest distance betweent the best and average cases was found to be 45. Each of these values are greater than the values for the individual corpora. The mean distance between the best and the average query is smaller than the mean between the average and the worst.

I computed the MRRs for these three cases and the results can be found in Table 5.93.

	Best	Average	Worst
MRR	0.2081	0.0030	0.0008

Table 5.93: MRRs for choosing the best, average, and worst case for each feature from all corpora and all structural weighting for JabRef

	CSB	ICL	LMPBV	Flat
Percentage	35	17	31	17

Table 5.94: Percentages for each corpus where the best query was found from all corpora and all structural weighting for JabRef

Unlike for ArgoUML where only a small increase was seen for the overall results of the system, the results of this computation show a large increase, of .0536, from the individual corpora to taking the best across all queries. To understand which corpus is contributing the highest number of best queries from the corpora, I computed the percentage for each query and recorded the results in Table 5.94. From this table it can be seen that the CSB corpus contributed the largest percentage of the queries for the features. This corpus was followed by the LMPBV corpus with a difference of 4%. There is a tie between the ICL and the flat corpora for the lowest percentages. Each of these corpora contributed 17% of the best queries to the results. A weighting configuration was used for 74% of the best queries. Adjusting for the flat corpus shows that either a weighting configuration of the flat corpus was used 91% of the time. This indicates that the unweighted configurations of one of the others corpora was used for 9% of the best queries in the results.

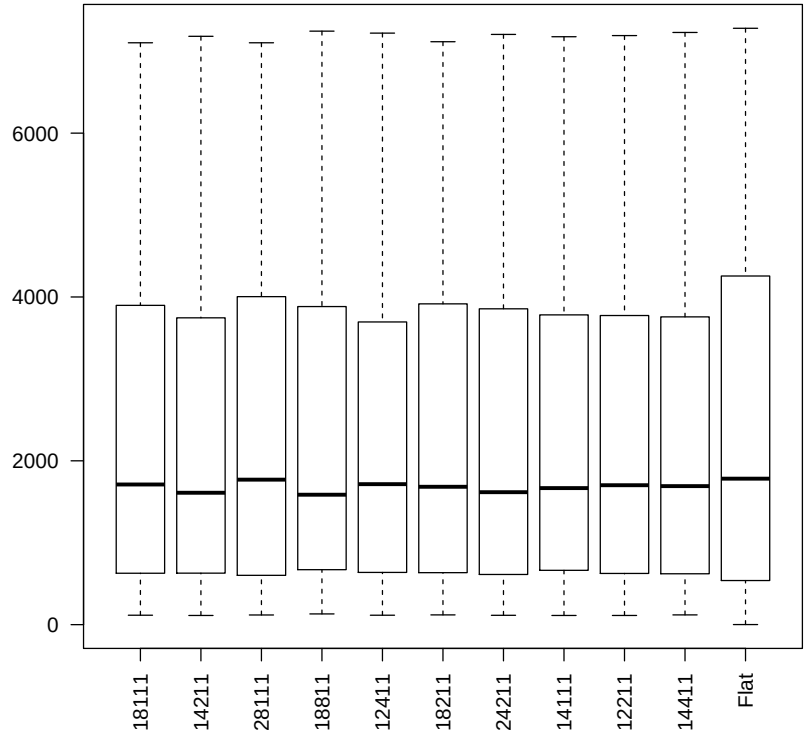
I performed a Friedman test with a wilcoxon post-hoc for the best, average, and worst case of each feature for each of the four corpora. The results of the analysis showed that there is a significant difference between each case.

5.3.3.3 jEdit

The boxplots for the LMPBV corpus and each query type can be found in Figure 5.41. These are the boxplots for the ten weighting configuration with the highest MRRs. The resulting MRRs can be found in Table 5.95. When looking at the Combined query type, there is less than a .0001 difference between all the top ten configurations. For each of the ten configurations, neither the body comments or the local variables are weighted. The most weighted lexicon in the corpus is the method names with it being the only lexicon in the top configuration and in the ninth configuration. Other lexicons that were weighted include the parameters which were weighted in seven of the ten configurations and leading comments which were weighted twice. The boxplots only show minor fluctuations in the effectiveness measures of the ten weighting configurations, however the flat corpus has a slightly larger spread than the weighting configurations. When looking at the Title query type, there is only a slight increase in the difference for the ten configurations at .0003. For all but one of the configurations, the leading comments were given a weighting factor of 8 while parameters were given a weighting factor of 8 for the top four configurations and then a weighting factor of 4 for all other configurations. Local variables were not weighted in any of the top ten configurations. Body comments were given a weighting factor of 2 for seven configurations with the other three either giving a weighting factor of 4 or leaving the lexicon unweighted. While method names were unweighted in the top configuration, they were weighted in seven configurations with a weighting factor of 8 for three configurations. The most common ratio was a ratio of 1:1 between leading comments and parameters for the top five configurations. When looking at the boxplots, there are slight fluctuations in the 3Q measure and the median, but the spreads were close in other values. The flat corpus had the smallest spread of the configurations.

The boxplots for the CSB corpus can be found in Figure 5.42, while the MRRs can be found in Table 5.96. For the Combined query type, there was a similar closeness as with the LMPBV corpus. There was only a difference of .0003 in the top ten configurations. For each of the ten configurations, the body was given a weighting factor of 8 or 4 with the top two configurations having a weighting factor of 8. The second and third highest weighting configurations had a ratio of 1:4 between the comments and the body in the corpus. There was also a ratio of 1:2 for the fifth and sixth configurations between the comments and the body, while the ninth and the tenth highest configurations had a ratio of 1:1 between the comments and the body. The comments were weighted in five of the top ten configurations with three of the weightings being lower than the weight for the body. The signature had similar ratios with the body. In the third and fourth configurations, there was a ratio of 1:4 between the signature and the body, while in the seventh and eighth configurations there was a ratio of 1:2 between the signature and the body. The signature always had a lower weighting configuration than the body. The boxplots show little difference in the spreads across all of the configurations. When looking at the Title query type, there is a larger difference between the configurations of .0094. While the body is still weighted in each of the ten configurations, there is now more balance between the signature and the body, and the comments and the body. For five of the top six configurations, there is a ratio of 1:1 between the signature and the body or the comments and the body with the top two configurations giving equal weighting to the signature and the body. The boxplots still show similar spreads for each of the weighting configurations, however the flat corpus now has a smaller spread than any of the other configurations.

(a) LMPBV Combined



(b) LMPBV Title

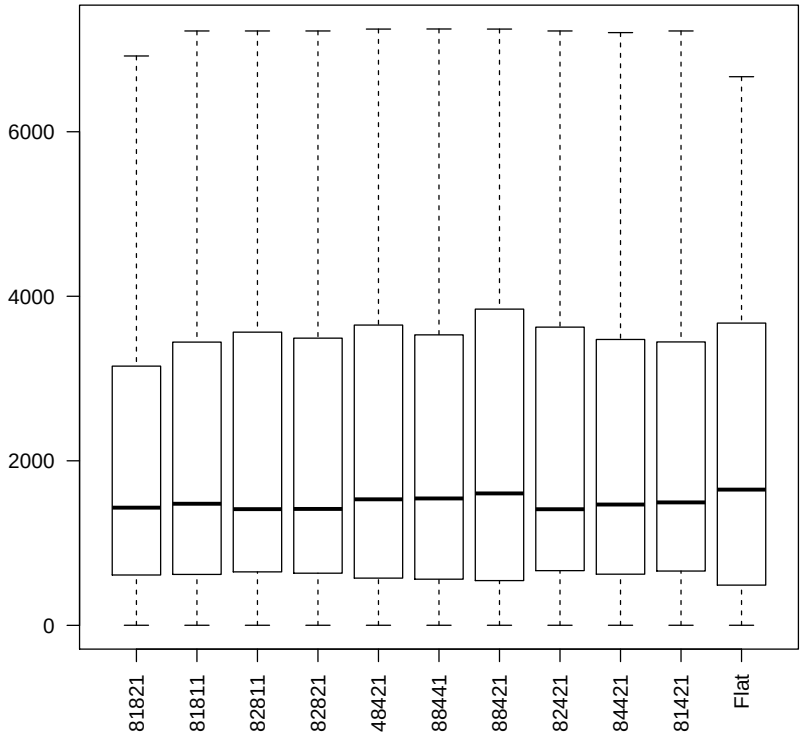


Figure 5.41: The top weighting configurations and flat configuration for jEdit and the LMPBV corpus. The x axis is the configuration, while the y axis is the effectiveness measure.

Config	MRR
$C\{L=1,M=8,P=1,B=1,V=1\}$	0.0011
$C\{L=1,M=4,P=2,B=1,V=1\}$	0.0011
$C\{L=2,M=8,P=1,B=1,V=1\}$	0.0011
$C\{L=1,M=8,P=8,B=1,V=1\}$	0.0011
$C\{L=1,M=2,P=4,B=1,V=1\}$	0.0011
$C\{L=1,M=8,P=2,B=1,V=1\}$	0.0011
$C\{L=2,M=4,P=2,B=1,V=1\}$	0.0011
$C\{L=1,M=4,P=1,B=1,V=1\}$	0.0011
$C\{L=1,M=2,P=2,B=1,V=1\}$	0.0011
$C\{L=1,M=4,P=4,B=1,V=1\}$	0.0011

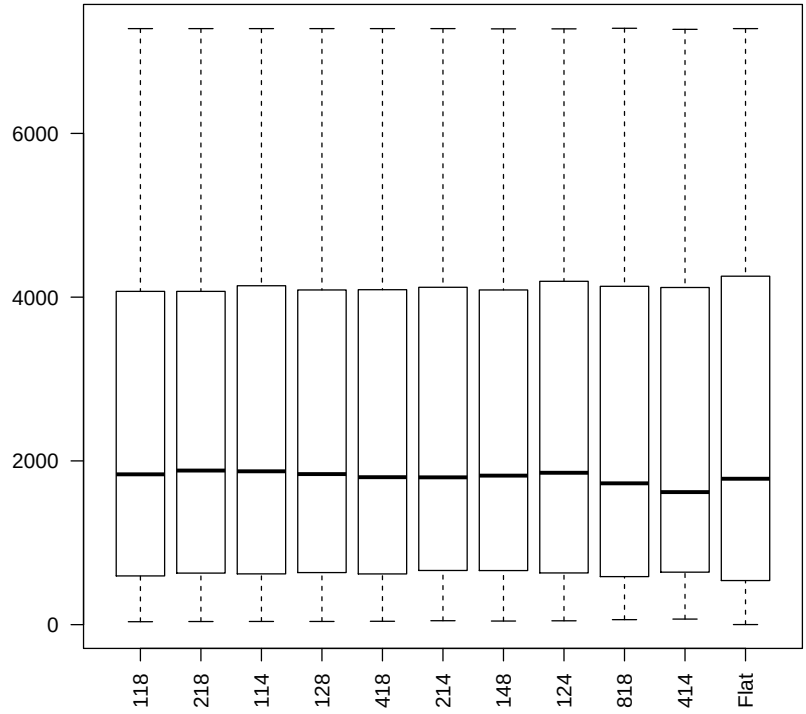
(a) LMPBV Combined

Config	MRR
$C\{L=8,M=1,P=8,B=2,V=1\}$	0.0161
$C\{L=8,M=1,P=8,B=1,V=1\}$	0.0161
$C\{L=8,M=2,P=8,B=1,V=1\}$	0.0160
$C\{L=8,M=2,P=8,B=2,V=1\}$	0.0160
$C\{L=4,M=8,P=4,B=2,V=1\}$	0.0159
$C\{L=8,M=8,P=4,B=4,V=1\}$	0.0159
$C\{L=8,M=8,P=4,B=2,V=1\}$	0.0159
$C\{L=8,M=2,P=4,B=2,V=1\}$	0.0158
$C\{L=8,M=4,P=4,B=2,V=1\}$	0.0158
$C\{L=8,M=1,P=4,B=2,V=1\}$	0.0158

(b) LMPBV Title

Table 5.95: Top 10 configurations for jEdit

(a) CSB Combined



(b) CSB Title

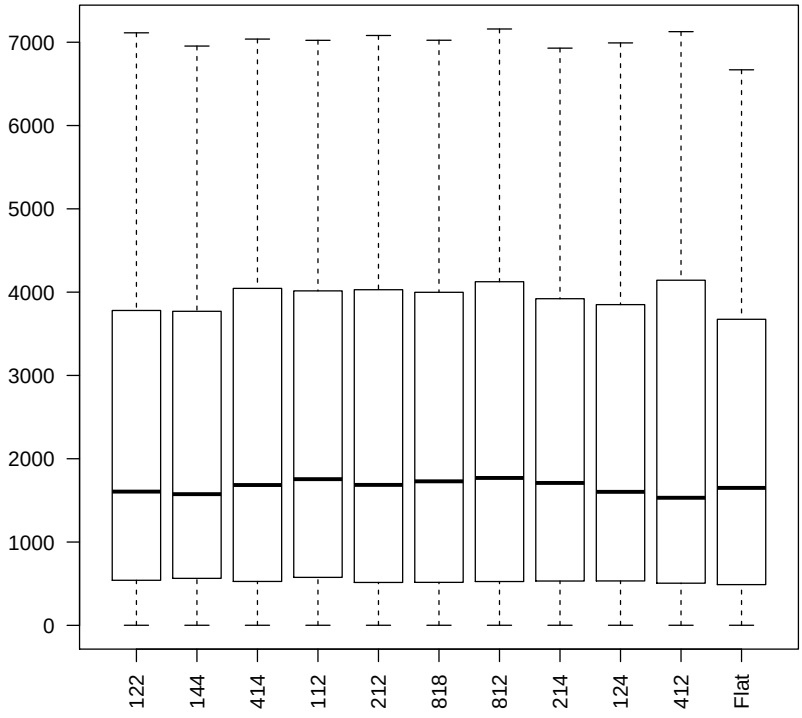


Figure 5.42: The top weighting configurations and flat configuration for jEdit and the CSB corpus. The x axis is the configuration, while the y axis is the effectiveness measure.

Config	MRR
$C\{C=1,S=1,B=8\}$	0.0018
$C\{C=2,S=1,B=8\}$	0.0018
$C\{C=1,S=1,B=4\}$	0.0017
$C\{C=1,S=2,B=8\}$	0.0017
$C\{C=4,S=1,B=8\}$	0.0017
$C\{C=2,S=1,B=4\}$	0.0016
$C\{C=1,S=4,B=8\}$	0.0016
$C\{C=1,S=2,B=4\}$	0.0016
$C\{C=8,S=1,B=8\}$	0.0016
$C\{C=4,S=1,B=4\}$	0.0015

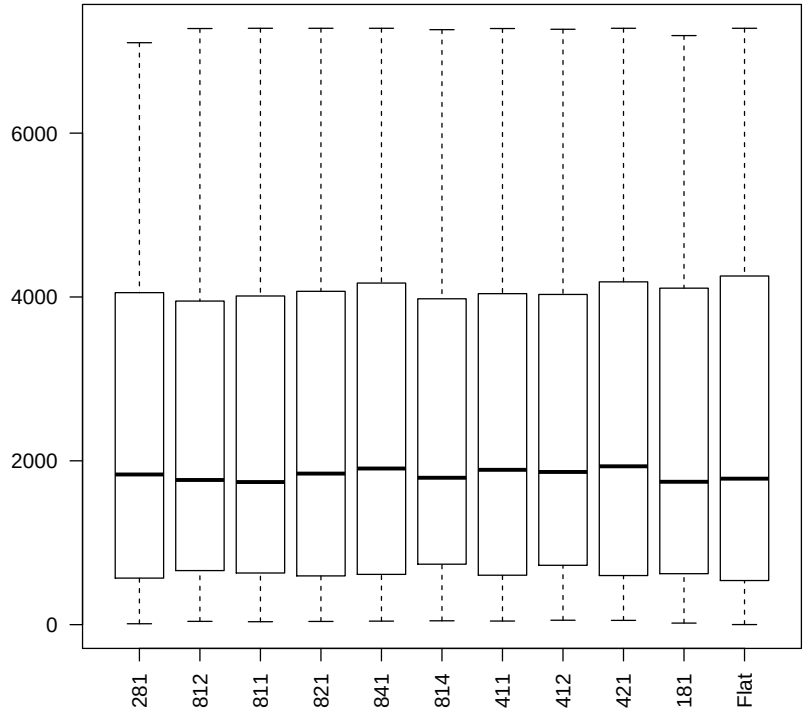
(a) CSB Combined

Config	MRR
$C\{C=1,S=2,B=2\}$	0.0528
$C\{C=1,S=4,B=4\}$	0.0526
$C\{C=4,S=1,B=4\}$	0.0494
$C\{C=1,S=1,B=2\}$	0.0479
$C\{C=2,S=1,B=2\}$	0.0477
$C\{C=8,S=1,B=8\}$	0.0477
$C\{C=8,S=1,B=2\}$	0.0439
$C\{C=2,S=1,B=4\}$	0.0438
$C\{C=1,S=2,B=4\}$	0.0438
$C\{C=4,S=1,B=2\}$	0.0434

(b) CSB Title

Table 5.96: Top 10 configurations for jEdit

(a) ICL Combined



(b) ICL Title

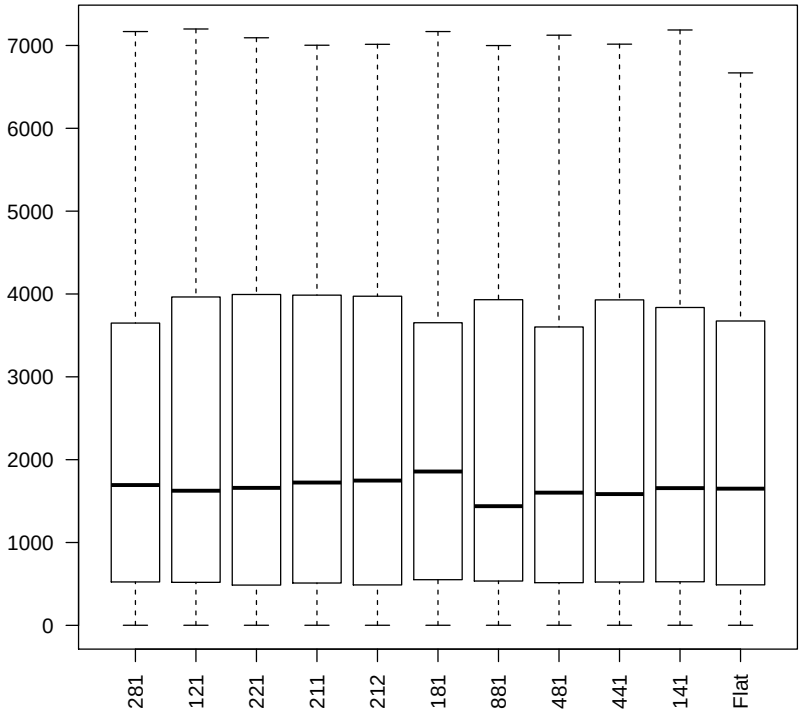


Figure 5.43: The top weighting configurations and flat configuration for jEdit and the ICL corpus. The x axis is the configuration, while the y axis is the effectiveness measure.

Config	MRR
$C\{I=2,C=8,L=1\}$	0.0020
$C\{I=8,C=1,L=2\}$	0.0019
$C\{I=8,C=1,L=1\}$	0.0018
$C\{I=8,C=2,L=1\}$	0.0018
$C\{I=8,C=4,L=1\}$	0.0018
$C\{I=8,C=1,L=4\}$	0.0018
$C\{I=4,C=1,L=1\}$	0.0017
$C\{I=4,C=1,L=2\}$	0.0017
$C\{I=4,C=2,L=1\}$	0.0017
$C\{I=1,C=8,L=1\}$	0.0017

(a) ICL Combined

Config	MRR
$C\{I=2,C=8,L=1\}$	0.0495
$C\{I=1,C=2,L=1\}$	0.0473
$C\{I=2,C=2,L=1\}$	0.0472
$C\{I=2,C=1,L=1\}$	0.0459
$C\{I=2,C=1,L=2\}$	0.0456
$C\{I=1,C=8,L=1\}$	0.0453
$C\{I=8,C=8,L=1\}$	0.0450
$C\{I=4,C=8,L=1\}$	0.0446
$C\{I=4,C=4,L=1\}$	0.0438
$C\{I=1,C=4,L=1\}$	0.0435

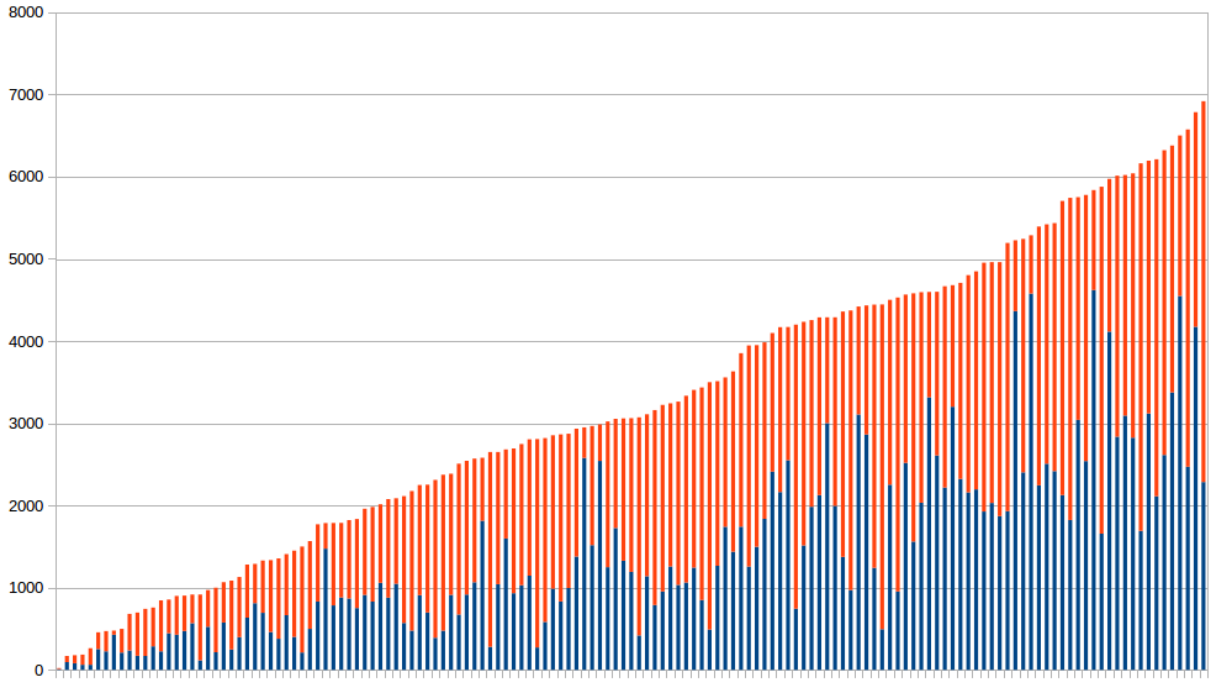
(b) ICL Title

Table 5.97: Top 10 configurations for jEdit

In Figure 5.43 and Table 5.97, you can see the boxplots and MRRs for the ICL corpus. Once again, for the Combined query type there is only a small difference of .0003 between the top ten weighting configurations. The top configuration gives a weighting factor of 8 to the comments with a 1:4 ratio between identifiers and the comments. This is the only configuration until the tenth configuration where any lexicon has a higher weighting than the identifiers. For the second through the sixth configuration, the identifiers have a weighting factor of 8 with the third through the fifth configuration be decreasing ratios of 8:1, 4:1, and 2:1 between the identifiers and the comments. The literals are only weighted in three of the top ten configurations. The boxplots for the effectiveness measures are similar to the boxplots from the previous two corpora with little difference between the spreads of each of the configurations. The Title query type has a wider difference of .006 between the top ten configurations. For comments the weightings have increased from what they were for the Combined query type. The top configuration is the same as the top configuration for the Combined query type, however for the top six configurations, the identifiers only get a weighting factor of 1 or 2. For the second through the fifth configuration, the highest weighting factors are 2 with the second configuration having a 1:2 ratio between identifiers and comments, the third configuration having a 1:1 ratio between the identifiers and the comments, and the fourth configuration having a 2:1 ratio between the identifiers and the comments. The spreads of the boxplots are similar for to the 1Q value, however afterward there is fluctuation for the medians, 3Q, and upper 1.5IRQ. The smallest spread of the configuration is for the flat corpus.

I conducted a Friedman test with a post-hoc analysis on the top configurations and the flat corpus. I chose to only look at the top configurations and the flat corpus to lower the number of possible pairings that need to be compared. For the Combined query type, there were 13 significant differences found for the CSB corpus. None of these differences were identified between the flat

(a) CSB



(b) ICL

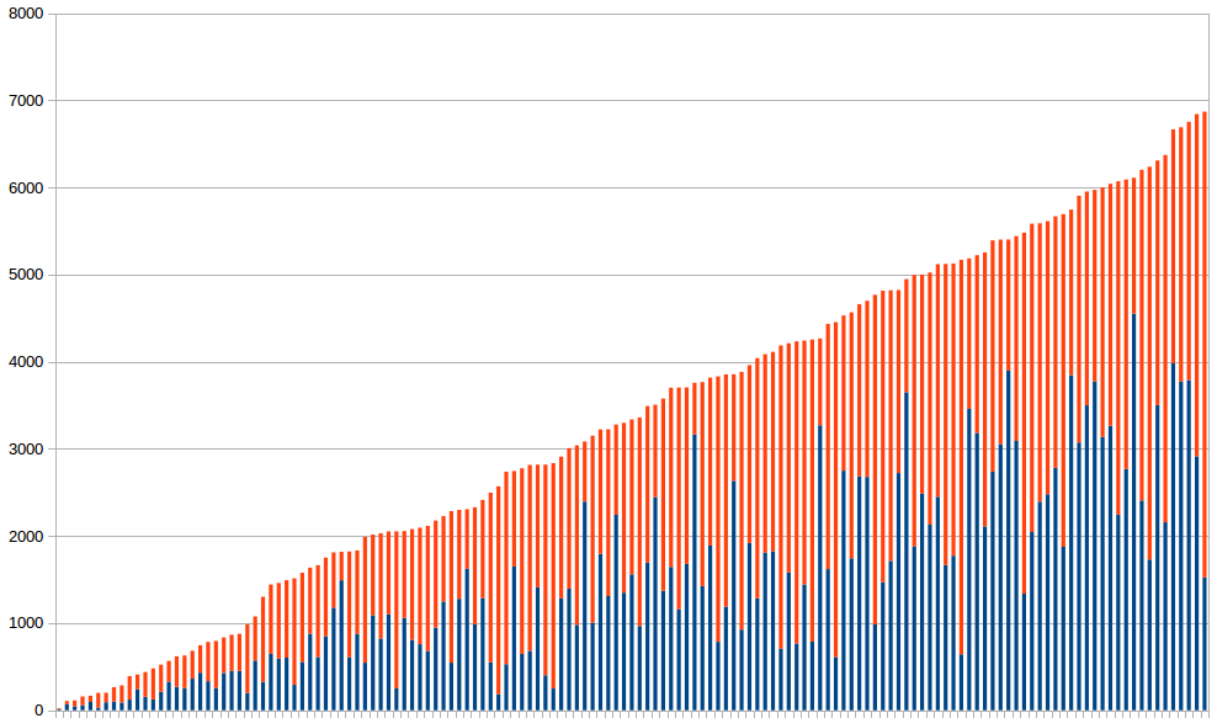


Figure 5.44: Stacked bargraphs representing the distance from the best query from structural weighting to the average (bottom) and the worst(top) for jEdit. Graph is ordered by distance from best to worst.

(c) LPMBV

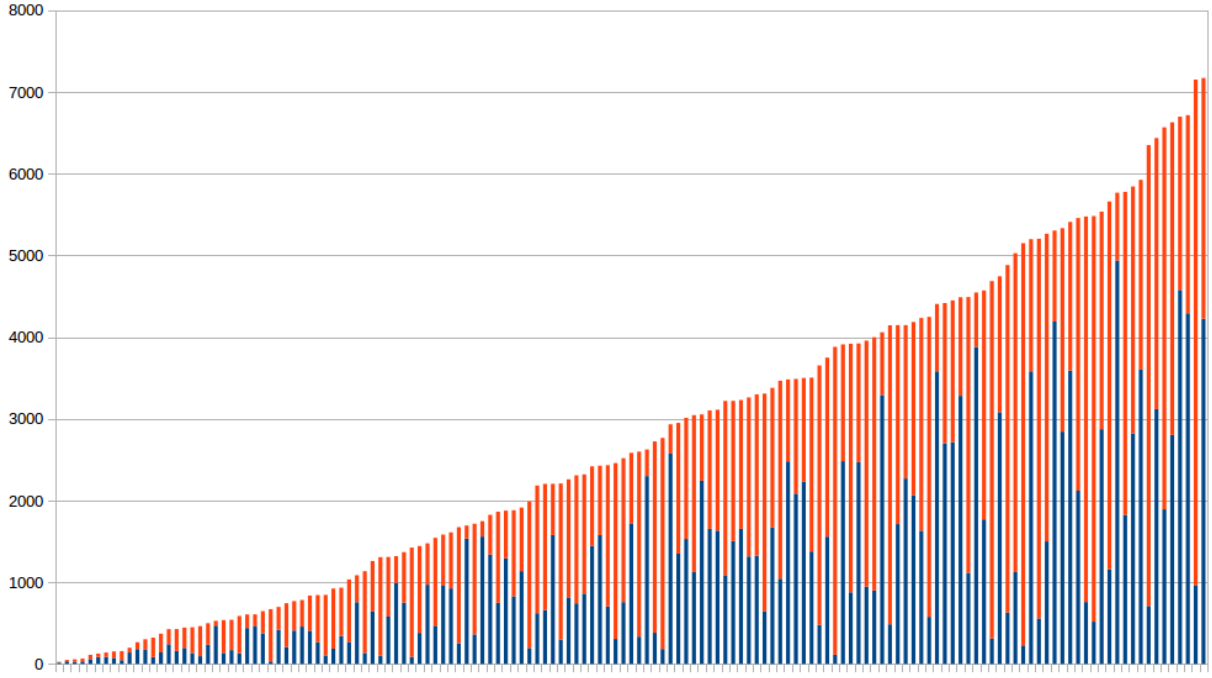


Figure 5.44: Stacked bargraphs representing the distance from the best query from structural weighting to the average (bottom) and the worst(top) for jEdit. Graph is ordered by distance from best to worst.

configuration and the weighting configurations. For the ICL corpus, 17 significant differences were identified with none of the differences being between the flat configuration and the weighting configurations. There differences were found despite the small difference in the MRRs and the spreads of effectiveness measures. There were no significant differences identified for the LMPBV corpus. For the Title query type, no significant differences were found for the CSB corpus, while the number of significant differences for the ICL corpus decreases, and 12 significant differences were identified between the weighting configurations for LMPBV. For the ICL corpus, 10 significant differences were found with 4 of the 10 being between the flat corpus and the configurations: $C\{I=1,C=2,L=1\}$, $C\{I=2,C=1,L=1\}$, $C\{I=2,C=1,L=2\}$, and $C\{I=1,C=4,L=1\}$.

I computed the best, worst, and average cases for each feature for each corpus using both

Query	CSB	ICL	LMPBV
Best	0.0639	0.0775	0.0204
Average	0.0011	0.0013	0.0011
Worst	0.0005	0.0006	0.0006

Table 5.98: MRRs for choosing the best, average, and worst case for each feature for jEdit from structural weighting

	CSB	ICL	LMPBV
Weighted	96	97	96

Table 5.99: The percentage of time that weighting each corpus improved the results for jEdit

query types. The differences between each case can be found in Figure 5.44. The greatest distance between the best query and the worst query for the three corpora is found in the LMPBV corpus at 7,171. The greatest mean distance between the best and the worst query is found in the ICL corpus at a mean distance of 3,328. ICL also showed the greatest mean distance between the best query and the average case at 1,471, a value that is only 37 higher than the mean distance between the best query and the average case for CSB. The smallest mean distance between the best and the worst case is found for the LMPBV with a mean distance of 2,784. This corpus also had the smallest mean distance between the best and the average case with a mean distance of 1,197. There was a tie for the smallest distance between the best and the worst queries between the ICL and the CSB corpora. Both corpora had a minimum distance of 22. ICL also had the smallest distance between the best query and the average case at 9. For each of the three corpora, the mean distance between the best and the average query is smaller than the mean between the average and the worst.

The MRRs have been computed for each case and the results can be found in Table 5.98.

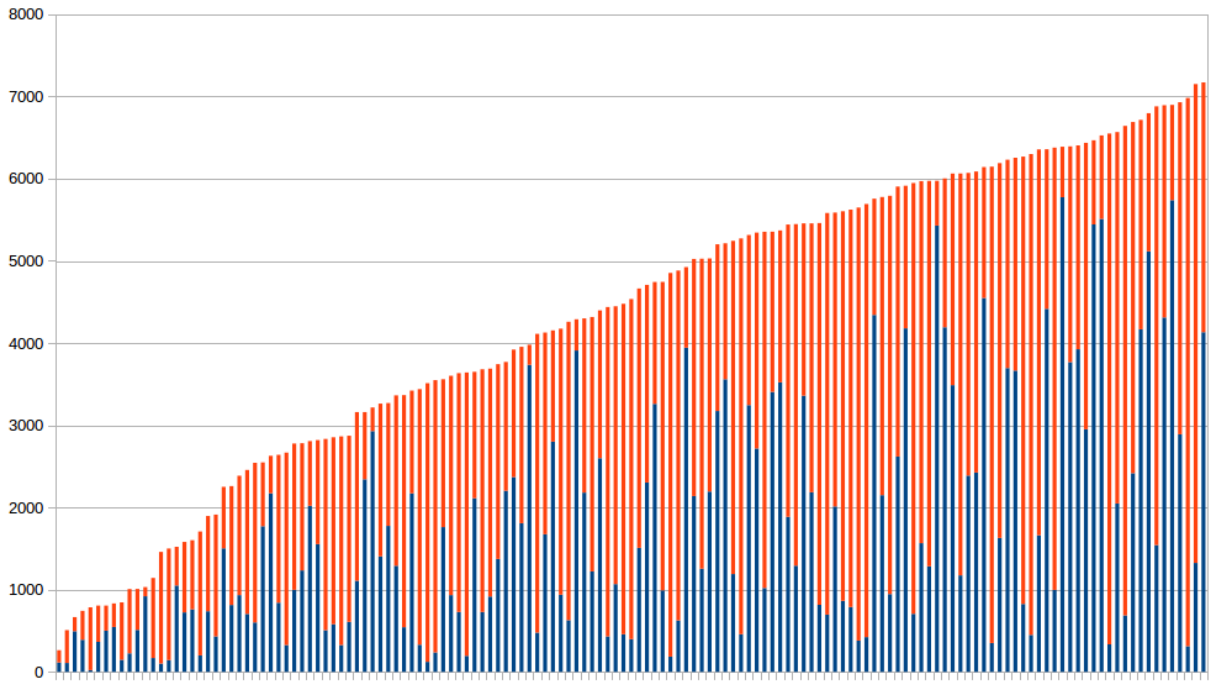


Figure 5.45: Stacked bargraphs representing the distance from the best query to the average (bottom) and the worst(top) from structural weighting for jEdit. Graph is ordered by distance from best to worst.

Comparing the MRRs between the best cases and the results of the best structural combinations shows only moderate increases from the individual corpora. The largest increase occurs for the ICL corpus with a difference between the best queries and the top weighting configuration of .0280. The smallest difference occurs for the LMPBV corpus with only a difference of .0043. Of the three corpora, the largest MRR can also be found for the ICL corpus while the smallest MRR can be found for the LMPBV corpus.

I also computed the percentage of times that the best query was a result of one of the weighting configurations. This helps to understand whether the weighted or unweighted configurations are having a bigger impact on the results for each corpus. The results of this calculation

	Best	Average	Worst
MRR	0.1044	0.0010	0.0003

Table 5.100: MRRs for choosing the best, average, and worst case for each feature from all corpora and all structural weighting for jEdit

	CSB	ICL	LMPBV	Flat
Percentage	28	26	24	22

Table 5.101: Percentages for each corpus where the best query was found from all corpora and all structural weighting for jEdit

can be found in Table 5.99. For each of three corpora, the unweighted configuration has the best queries at most 4% of the time.

I performed a Friedman test with a wilcoxon post-hoc for the best, average, and worst case of each feature for each of the four corpora. The results of the analysis showed that there is a significant difference between each case.

I computed the best, worst, and average cases for each feature across all corpora using both query types. The differences between each case can be found in Figure 5.45. The greatest distance between the best query and the worst query is 7,171, while the mean distance between the best and the worst query is 4,332. The mean distance between the best query and the average case was 1,734. The smallest distance between the best and worst case was 263, while the smallest distance betweent the best and average cases was found to be 114. Each of these values are greater than the values for the individual corpora. The mean distance between the best and the average query is smaller than the mean between the average and the worst.

I computed the MRRs for these three cases and the results can be found in Table 5.100. The results of this computation show a large increase, of .0269, from the individual corpora to taking

the best across all queries. To understand which corpus is contributing the highest number of best queries from the corpora, I computed the percentage for each query and recorded the results in Table 5.101. From this table it can be seen that the CSB corpus contributed the largest percentage of the queries for the features. This corpus was followed closely by the ICL corpus with a difference of 2%. The flat corpus has the lowest percentage with 22% only a 6% difference from the highest percentage. A weighting configuration was used for 76% of the best queries. Adjusting for the flat corpus shows that either a weighting configuration of the flat corpus was used 98% of the time. This indicates that the unweighted configurations of one of the others corpora was used for only 2% of the best queries in the results.

I performed a Friedman test with a wilcoxon post-hoc for the best, average, and worst case of each feature for each of the four corpora. The results of the analysis showed that there is a significant difference between each case.

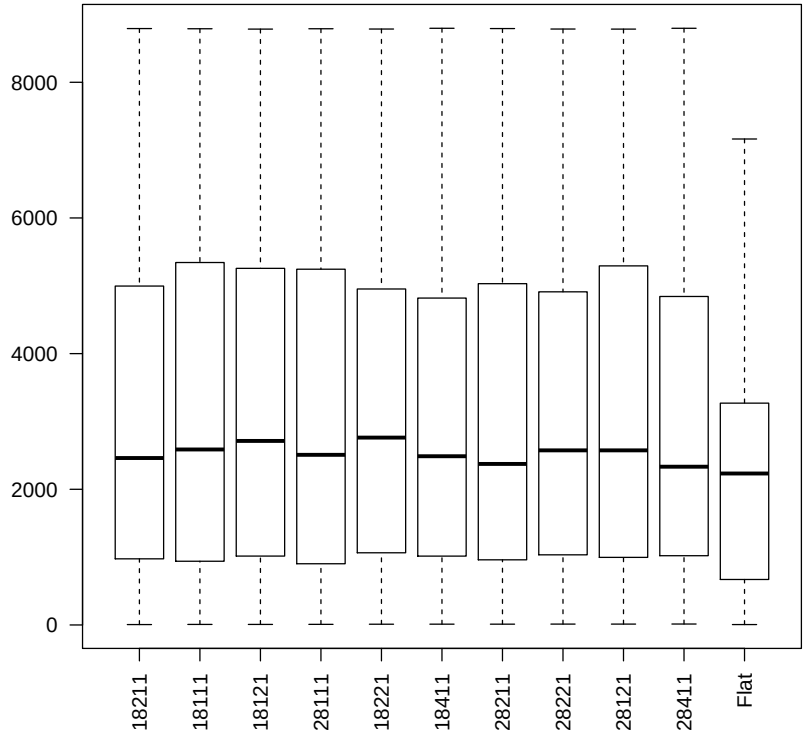
5.3.3.4 muCommander

The boxplots for the LMPBV corpus and each query type can be found in Figure 5.46. These are the boxplots for the ten weighting configuration with the highest MRRs. The resulting MRRs can be found in Table 5.102. When looking at the Combined query type, there is a difference of .001 between the configuations. Each configuration has a weighting factor of 8 for the method names while also having a weighting factor of 1 for the local variables. The top configuration gives the parameters a weighting factor of 2 while the second configuration only has the method names weighted. Other lexicons that receive weighting during the configurations include the leading and body comments which receive weighting factors of 2 in a subset of the configurations. The only lexicon to get a weighting factor higher than 2 other than the method names are the parameters in two of the configurations. The most common ratios of the configurations include a 1:4 between

leading comments and the method names and a 4:1 ratio between the method names and the parameters. When looking at the boxplots, the weighting configurations have a similar spread, while the flat corpus has a reduced spread when compared to the other configurations. The Title query type gives leading comments a weighting factor of 8 in all but one of the top configurations. The second configuration is the only configuration with a weighting factor of 4 for the leading comments. The weighting factors for the method names are reduced when compared to the Combined query type. The method names only receive a weighting factor in four of the ten configurations, while only receiving a weighting factor of 8 in two configurations. The parameters, local variables, and body comments also receive weighting factors of 8 in a majority of the configurations. The ten configurations are close in distance with only a .0012 difference between the configurations. When looking at the boxplots, the flat corpus has a smaller spread when compared to the other configurations with the exception of the upper 1.5IRQ.

The boxplots for the CSB corpus can be found in Figure 5.47, while the MRRs can be found in Table 5.103. For the Combined query type, the difference between the top ten configurations is .0011. The highest weighting factors are given for the comments with a weighting factor of 8 being given to comments in the top five configurations. The body is the least weighted lexicon with only two configurations being weighted, receiving a weighting factor of 2 in one case and a weighting factor of 8 in the other. The signature receives weighting in five of the top ten configuration with the top configuration giving signature a weighting factor of 4 and a 2:1 ratio with the comments, while the second configuration gives the signature a weighting factor of 8 for a 1:1 ratio with the comments. The flat corpus has the smallest spread of any of the configurations when looking at the boxplots. For the weighting configurations there are only small fluctuations in the spreads. For the Title query type, there is a difference of .0059 in the top ten configurations.

(a) LMPBV Combined



(b) LMPBV Title

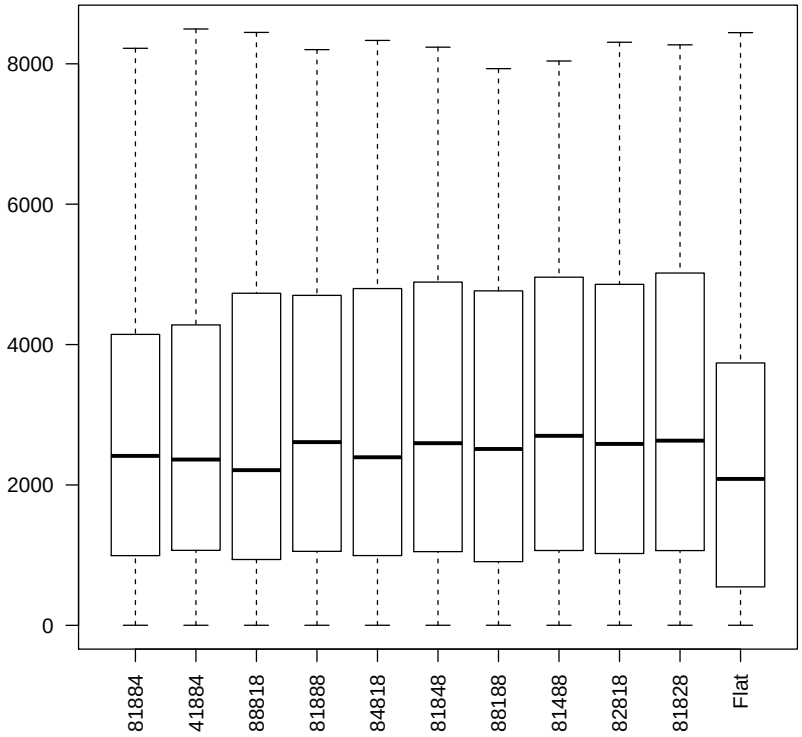


Figure 5.46: The top weighting configurations and flat configuration for muCommander and the LMPBV corpus. The x axis is the configuration, while the y axis is the effectiveness measure.

Config	MRR
$C\{L=1,M=8,P=2,B=1,V=1\}$	0.0028
$C\{L=1,M=8,P=1,B=1,V=1\}$	0.0023
$C\{L=1,M=8,P=1,B=2,V=1\}$	0.0023
$C\{L=2,M=8,P=1,B=1,V=1\}$	0.0022
$C\{L=1,M=8,P=2,B=2,V=1\}$	0.0019
$C\{L=1,M=8,P=4,B=1,V=1\}$	0.0019
$C\{L=2,M=8,P=2,B=1,V=1\}$	0.0018
$C\{L=2,M=8,P=2,B=2,V=1\}$	0.0018
$C\{L=2,M=8,P=1,B=2,V=1\}$	0.0018
$C\{L=2,M=8,P=4,B=1,V=1\}$	0.0018

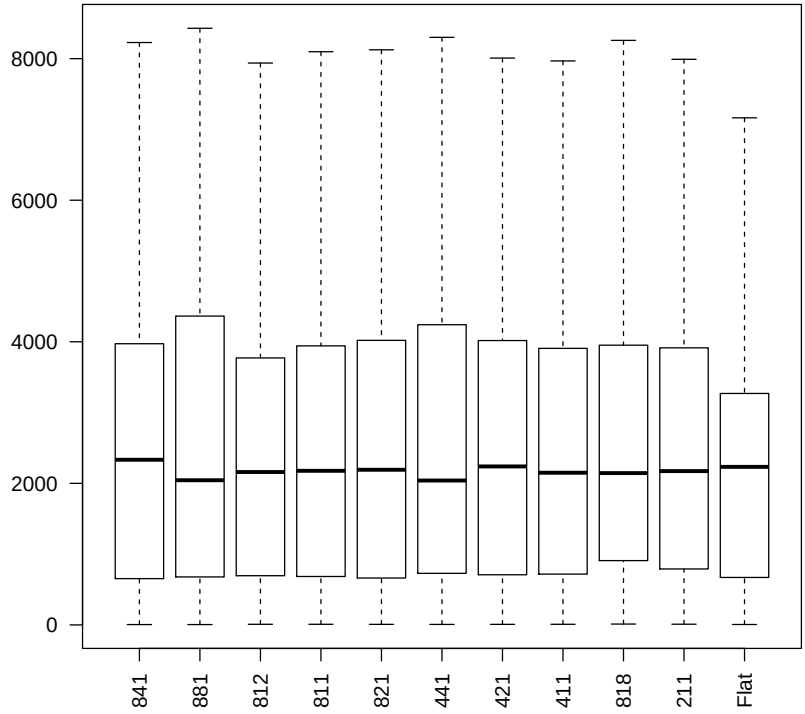
(a) LMPBV Combined

Config	MRR
$C\{L=8,M=1,P=8,B=8,V=4\}$	0.0189
$C\{L=4,M=1,P=8,B=8,V=4\}$	0.0189
$C\{L=8,M=8,P=8,B=1,V=8\}$	0.0179
$C\{L=8,M=1,P=8,B=8,V=8\}$	0.0178
$C\{L=8,M=4,P=8,B=1,V=8\}$	0.0178
$C\{L=8,M=1,P=8,B=4,V=8\}$	0.0178
$C\{L=8,M=8,P=1,B=8,V=8\}$	0.0178
$C\{L=8,M=1,P=4,B=8,V=8\}$	0.0177
$C\{L=8,M=2,P=8,B=1,V=8\}$	0.0177
$C\{L=8,M=1,P=8,B=2,V=8\}$	0.0177

(b) LMPBV Title

Table 5.102: Top 10 configurations for muCommander

(a) CSB Combined



(b) CSB Title

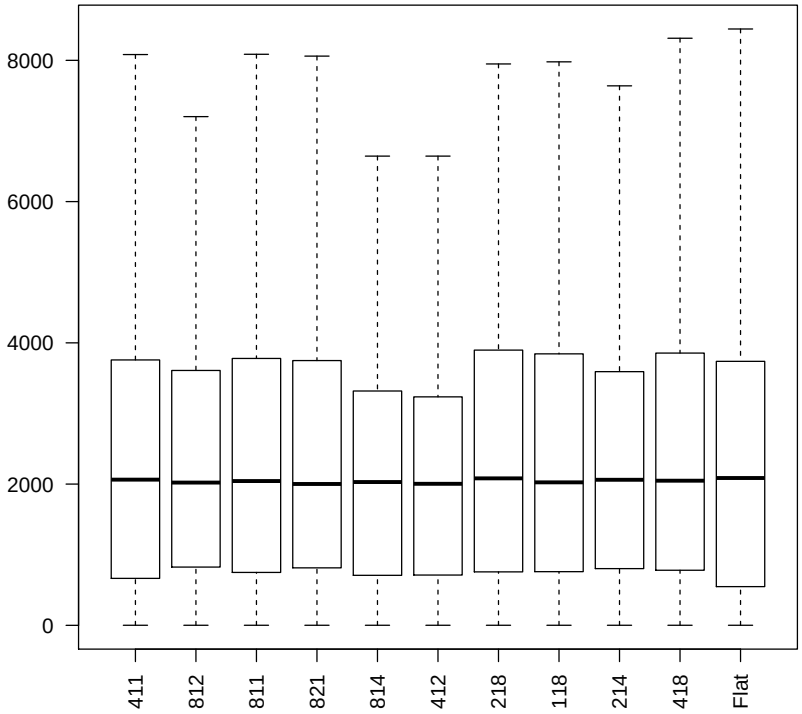


Figure 5.47: The top weighting configurations and flat configuration for muCommander and the CSB corpus. The x axis is the configuration, while the y axis is the effectiveness measure.

Config	MRR
$C\{C=8,S=4,B=1\}$	0.0039
$C\{C=8,S=8,B=1\}$	0.0036
$C\{C=8,S=1,B=2\}$	0.0035
$C\{C=8,S=1,B=1\}$	0.0034
$C\{C=8,S=2,B=1\}$	0.0034
$C\{C=4,S=4,B=1\}$	0.0030
$C\{C=4,S=2,B=1\}$	0.0030
$C\{C=4,S=1,B=1\}$	0.0030
$C\{C=8,S=1,B=8\}$	0.0029
$C\{C=2,S=1,B=1\}$	0.0028

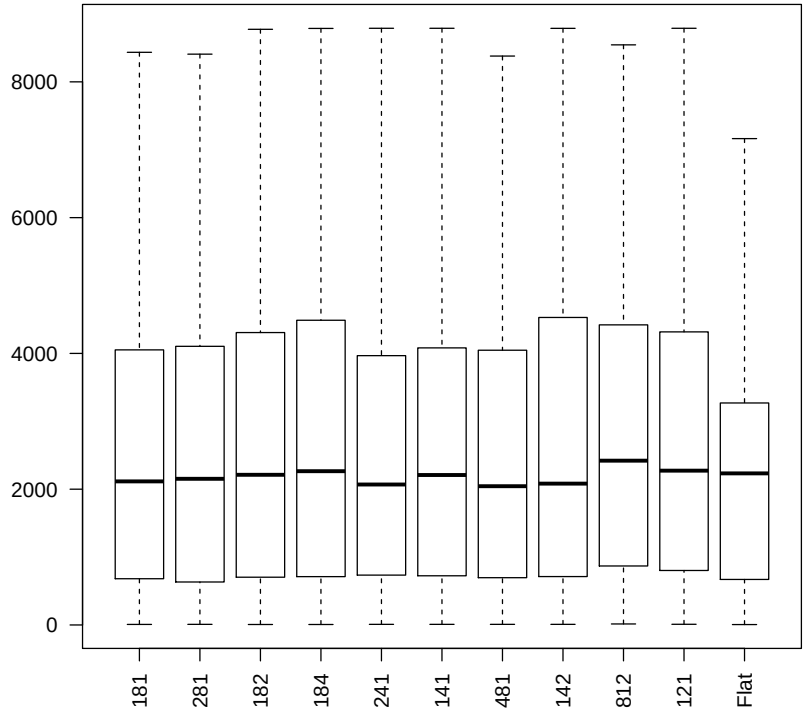
(a) CSB Combined

Config	MRR
$C\{C=4,S=1,B=1\}$	0.0277
$C\{C=8,S=1,B=2\}$	0.0252
$C\{C=8,S=1,B=1\}$	0.0244
$C\{C=8,S=2,B=1\}$	0.0241
$C\{C=8,S=1,B=4\}$	0.0236
$C\{C=4,S=1,B=2\}$	0.0227
$C\{C=2,S=1,B=8\}$	0.0218
$C\{C=1,S=1,B=8\}$	0.0218
$C\{C=2,S=1,B=4\}$	0.0218
$C\{C=4,S=1,B=8\}$	0.0218

(b) CSB Title

Table 5.103: Top 10 configurations for muCommander

(a) ICL Combined



(b) ICL Title

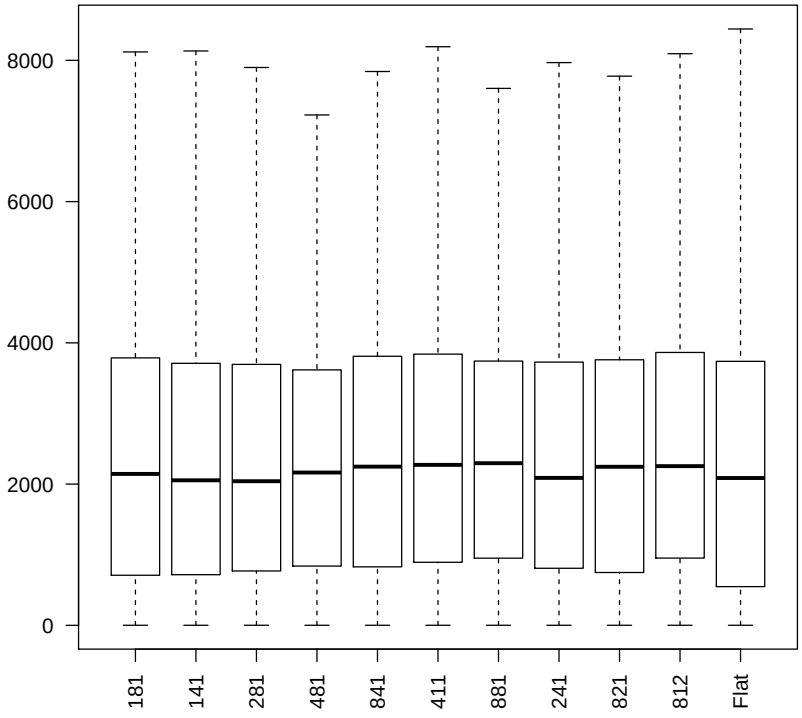


Figure 5.48: The top weighting configurations and flat configuration for muCommander and the ICL corpus. The x axis is the configuration, while the y axis is the effectiveness measure.

Config	MRR
$C\{I=1,C=8,L=1\}$	0.0033
$C\{I=2,C=8,L=1\}$	0.0032
$C\{I=1,C=8,L=2\}$	0.0032
$C\{I=1,C=8,L=4\}$	0.0030
$C\{I=2,C=4,L=1\}$	0.0029
$C\{I=1,C=4,L=1\}$	0.0029
$C\{I=4,C=8,L=1\}$	0.0029
$C\{I=1,C=4,L=2\}$	0.0027
$C\{I=8,C=1,L=2\}$	0.0026
$C\{I=1,C=2,L=1\}$	0.0026

(a) ICL Combined

Config	MRR
$C\{I=1,C=8,L=1\}$	0.0246
$C\{I=1,C=4,L=1\}$	0.0245
$C\{I=2,C=8,L=1\}$	0.0232
$C\{I=4,C=8,L=1\}$	0.0221
$C\{I=8,C=4,L=1\}$	0.0200
$C\{I=4,C=1,L=1\}$	0.0199
$C\{I=8,C=8,L=1\}$	0.0199
$C\{I=2,C=4,L=1\}$	0.0199
$C\{I=8,C=2,L=1\}$	0.0198
$C\{I=8,C=1,L=2\}$	0.0198

(b) ICL Title

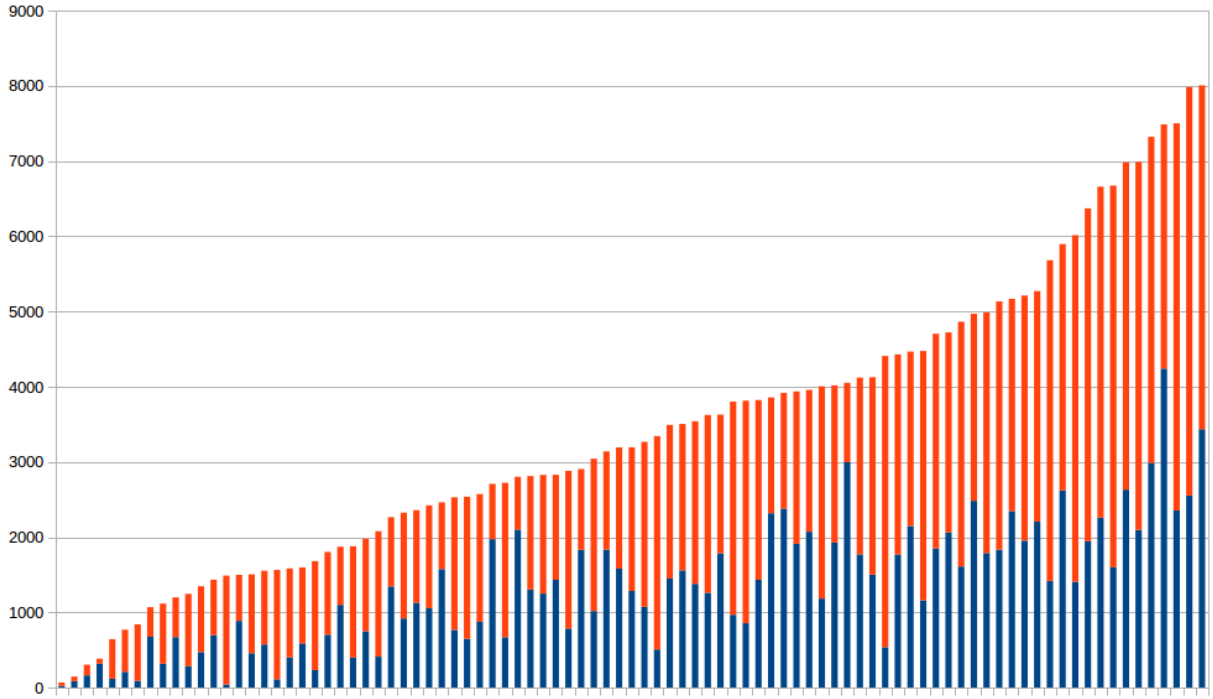
Table 5.104: Top 10 configurations for muCommander

The comments still receive the most weighting of any of the configurations with a weighting factor of 4 in the first configuration and a weighting factor of 8 for the second configuration through the fifth. The signature receives reduced weighting from the Combined query type with only one configuration giving signature a weighting factor of 2. Two of the top configurations only weight the comments with the first configuration having a 4:1:1 ratio and the third configuration having an 8:1:1 ratio. Looking at the boxplots shows two configurations with similar spreads that are smaller than the others. These spreads are for the $C\{C=8,S=1,B=4\}$ and the $C\{C=4,S=1,B=2\}$ configurations. Both of these configurations have a 2:1 ratio between the comments and the body.

In Figure 5.48 and Table 5.104, you can see the boxplots and MRRs for the ICL corpus. There is only a .0007 difference between the top configurations for the Combined query type. For the top four configurations, the comments get a weighting factor of 8, while for eight of the ten configurations they receive a weighting factor of 8 or 4. The weighting factors for both identifiers and literals are substantially lower. The top configuration does not weight either the identifiers or the literals, making the comments the only lexicon with weighting. This occurs for three of the top ten configurations. When looking at the boxplots, the flat corpus has a smaller spread than any of the weighting configurations. The weighting configurations show small fluctuations. For the Title query type, there is a difference of .0048 between the top configurations. Comments maintain high weighting factors, however for eight of the configurations, the weighting for the identifiers has increased. Literals continue to have low weighting and have even been further reduced to only having a weighting factor of 2 in the tenth configuration. The boxplots show that the flat corpus has the smallest 1Q value, but the greatest spread of any of the configurations. The smallest spread comes for the $C\{I=4,C=8,L=1\}$ configuration.

I conducted a Friedman test with a post-hoc analysis on the top configurations and the flat

(a) CSB



(b) ICL

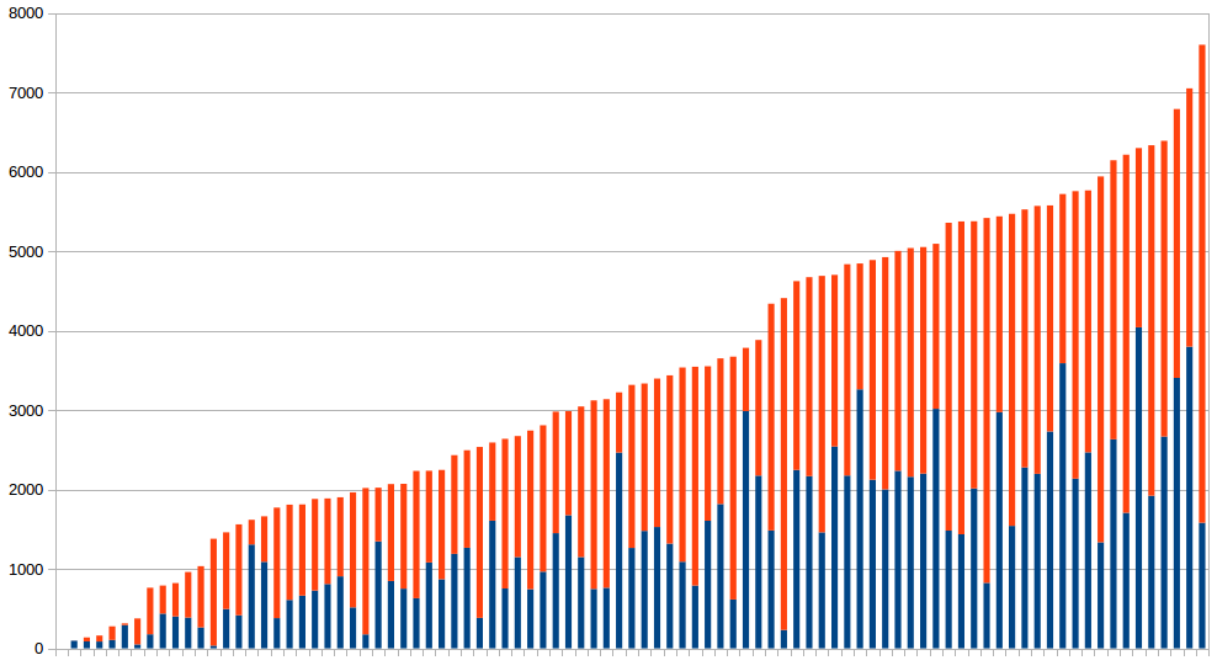


Figure 5.49: Stacked bargraphs representing the distance from the best query from structural weighting to the average (bottom) and the worst(top) for muCommander. Graph is ordered by distance from best to worst.

(c) LPMBV

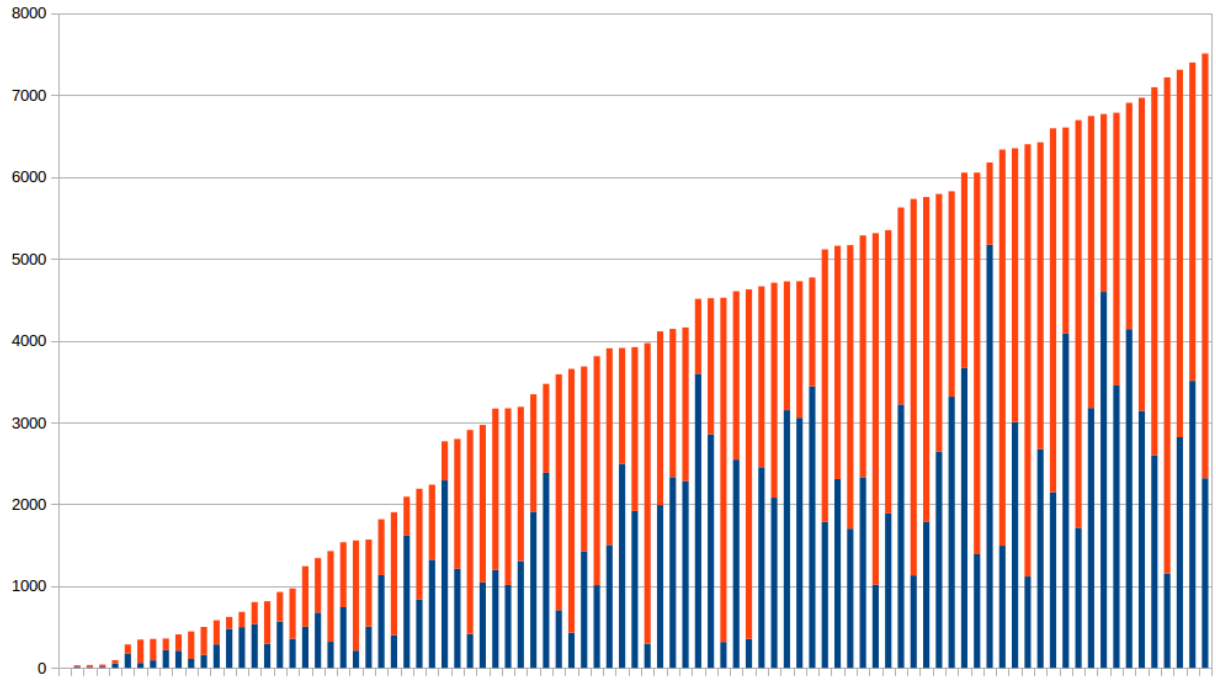


Figure 5.49: Stacked bargraphs representing the distance from the best query from structural weighting to the average (bottom) and the worst(top) for muCommander. Graph is ordered by distance from best to worst.

corpus. I chose to only look at the top configurations and the flat corpus to lower the number of possible pairings that need to be compared. For the Combined query type, there was only 1 significant difference for the CSB corpus between the $C\{C=8,S=4,B=1\}$ and the $C\{C=4,S=1,B=1\}$ configurations. For the ICL corpus there were 29 significant differences found with only one difference with the flat corpus. This difference was found for the configuration $C\{I=1,C=8,L=4\}$. For the LMPBV corpus, there were 27 significant differences discovered. Ten of those differences were from the flat corpus and each of the weighting configurations. The Title query type resulted in 1 significant difference for the CSB corpus between the $C\{C=2,S=1,B=8\}$ and the $C\{C=1,S=1,B=8\}$ configurations. For the ICL corpus, only two significant differences were found between the $C\{I=4,C=1,L=1\}$ and the $C\{I=8,C=2,L=1\}$ configurations,

Query	CSB	ICL	LMPBV
Best	0.0464	0.0288	0.0255
Average	0.0015	0.0013	0.0016
Worst	0.0005	0.0005	0.0008

Table 5.105: MRRs for choosing the best, average, and worst case for each feature for muCommander from structural weighting

	CSB	ICL	LMPBV
Weighted	95	95	95

Table 5.106: The percentage of time that weighting each corpus improved the results for muCommander

and the $C\{I = 8, C = 2, L = 1\}$ and the $C\{I = 8, C = 1, L = 2\}$ configurations. For LMPBV, 25 significant differences were found with there once again being significant differences between the flat corpus and each of the ten configurations.

I computed the best, worst, and average cases for each feature for each corpus using both query types. The differences between each case can be found in Figure 5.49. The greatest distance between the best query and the worst query for the three corpora is found in the CSB corpus at 8,008. The greatest mean distance between the best and the worst query is found in the LMPBV corpus at a mean distance of 3,722. LMPBV also showed the greatest mean distance between the best query and the average case at 1,603. The smallest mean distance between the best and the worst case is found for the CSB corpus with a mean distance of 3,422. This corpus also had the smallest mean distance between the best and the average case with a mean distance of 1,338. These values are close to the values for the ICL corpus. The smallest distance between the best and the worst query was found for the ICL corpus with a distance of 0. There was a tie for the

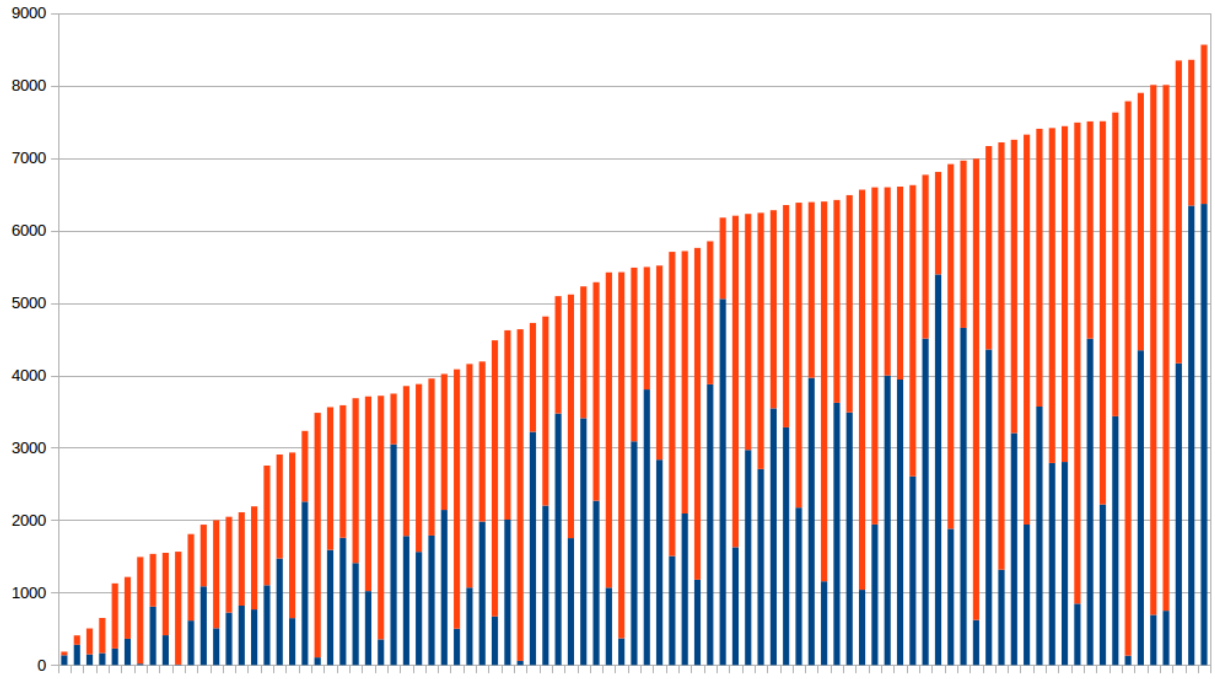


Figure 5.50: Stacked bargraphs representing the distance from the best query to the average (bottom) and the worst(top) from structural weighting for muCommander. Graph is ordered by distance from best to worst.

smallest distance between the best query and the average case between the ICL and the LMPBV corpora. Both corpora had a minimum distance of 0. For each of the three corpora, the mean distance between the best and the average query is smaller than the mean between the average and the worst.

The MRRs have been computed for each case and the results can be found in Table 5.105. Comparing the MRRs between the best cases and the results of the best structural combinations shows only moderate increases from the individual corpora. The largest increase occurs for the CSB corpus with a difference between the best queries and the top weighting configuration of .0187. The smallest difference occurs for the ICL corpus with only a difference of .0042. Of the

	Best	Average	Worst
MRR	0.0611	0.0014	0.0003

Table 5.107: MRRs for choosing the best, average, and worst case for each feature from all corpora and all structural weighting for muCommander

	CSB	ICL	LMPBV	Flat
Percentage	40	12	25	23

Table 5.108: Percentages for each corpus where the best query was found from all corpora and all structural weighting for muCommander

three corpora, the largest MRR can also be found for the CSB corpus while the smallest MRR can be found for the LMPBV corpus.

I also computed the percentage of times that the best query was a result of one of the weighting configurations. This helps to understand whether the weighted or unweighted configurations are having a bigger impact on the results for each corpus. The results of this calculation can be found in Table 5.106. For each of three corpora, the unweighted configuration has the best queries at most 5% of the time.

I performed a Friedman test with a wilcoxon post-hoc for the best, average, and worst case of each feature for each of the four corpora. The results of the analysis showed that there is a significant difference between each case.

I computed the best, worst, and average cases for each feature across all corpora using both query types. The differences between each case can be found in Figure 5.50. The greatest distance between the best query and the worst query is 8,564, while the mean distance between the best and the worst query is 5,026. The mean distance between the best query and the average case was 2,081. The smallest distance between the best and worst case was 184, while the smallest distance

between the best and average cases was found to be 135. Each of these values are greater than the values for the individual corpora. The mean distance between the best and the average query is smaller than the mean between the average and the worst.

I computed the MRRs for these three cases and the results can be found in Table 5.107. While muCommander has the lowest MRR of the four subject systems, the results of this computation show a moderate increase, of .0147, from the individual corpora to taking the best across all queries. To understand which corpus is contributing the highest number of best queries from the corpora, I computed the percentage for each query and recorded the results in Table 5.108. From this table it can be seen that the CSB corpus contributed the largest percentage of the queries for the features. This corpus was followed by the LMPBV corpus with a difference of 15%. The ICL corpus has the lowest percentage with 12%. A weighting configuration was used for 74% of the best queries. Adjusting for the flat corpus shows that either a weighting configuration of the flat corpus was used 97% of the time. This indicates that the unweighted configurations of one of the others corpora was used for only 3% of the best queries in the results.

I performed a Friedman test with a wilcoxon post-hoc for the best, average, and worst case of each feature for each of the four corpora. The results of the analysis showed that there is a significant difference between each case.

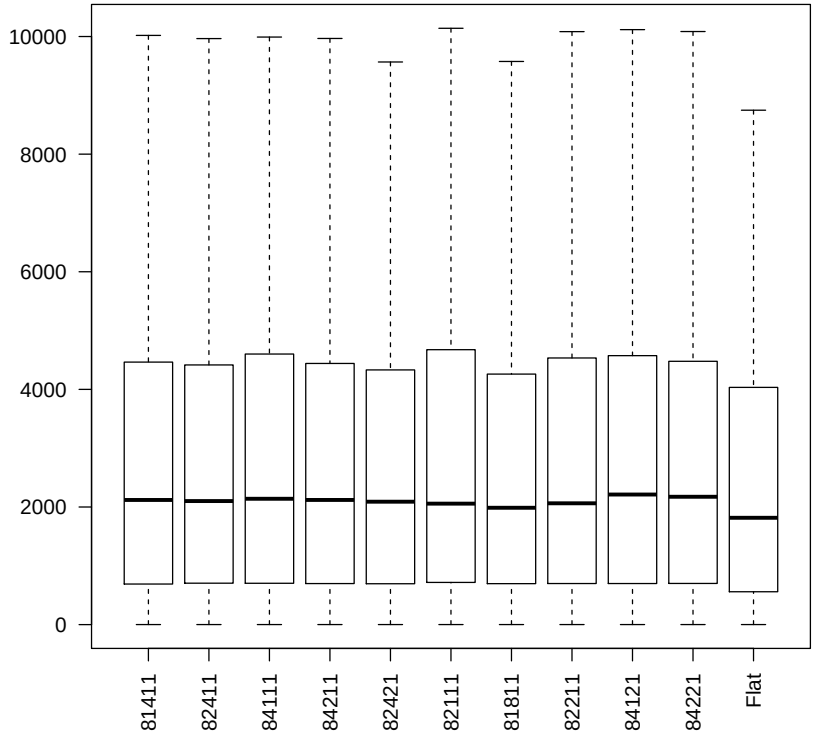
5.3.3.5 All Systems

The boxplots for the LMPBV corpus and each query type can be found in Figure 5.51. These are the boxplots for the ten weighting configuration with the highest MRRs. The resulting MRRs can be found in Table 5.109. For the Combined query type, each of the top ten configurations are less and a distance of .0001 apart. In addition, each of the top ten configurations uses a weighting factor of 8 for each of the configurations and a weighting factor of 1 for local

variables. The method names and the parameters are each weighted in the majority of the configurations while the body comments are only weighting in three of the ten configurations. Method names have a weighting factor that is greater than or equal to the weighting factor for parameters in six of the ten configurations. When looking at the boxplots, the flat corpus has the lowest spread of the configurations, while the smallest spreads for the weighting configurations are seen for $C\{L = 8, M = 2, P = 4, B = 2, V = 1\}$ and $C\{L = 8, M = 1, P = 8, B = 1, V = 1\}$. For the Title query type, the difference between the top ten configurations is now .0012. Each of the configurations once again give leading comments a weighting factor of 8, while method names are weighted in seven of the ten configurations and parameters are weighted in four of the ten configurations. Body comments are weighted more than they were in the Combined query type, where they have a weighting factor of 4 or 8 in eight of the top ten configurations. Local variables once again have a weighting factor of 1 in each of the top configurations. The smallest spread is seen for the configuration $C\{L = 8, M = 1, P = 1, B = 4, V = 1\}$, while the flat corpus has the smallest 1Q value. Important to note is that for each of the subject systems and for all systems combined, there did not exists a weighting configuration that weighted every lexicon.

The boxplots for the CSB corpus can be found in Figure 5.52, while the MRRs can be found in Table 5.110. For the Combined query type, there is a difference of .0029 between the top configurations. The top four configurations show a ratio of 1:1 between the comments and the body, beginning with a weighting factor of 8 for each lexicon and then decreasing by each weighting factor until reaching the unweighted configuration in the fourth position. This is followed by a ratio of 2:1 between comments and the body in the fifth, sixth, and eighth positions. For the signature, no weighting factors were used except in the ninth and tenth configurations where the signature was given a weighting factor of 2. The flat corpus has the smallest spread of the effectiveness

(a) LMPBV Combined



(b) LMPBV Title

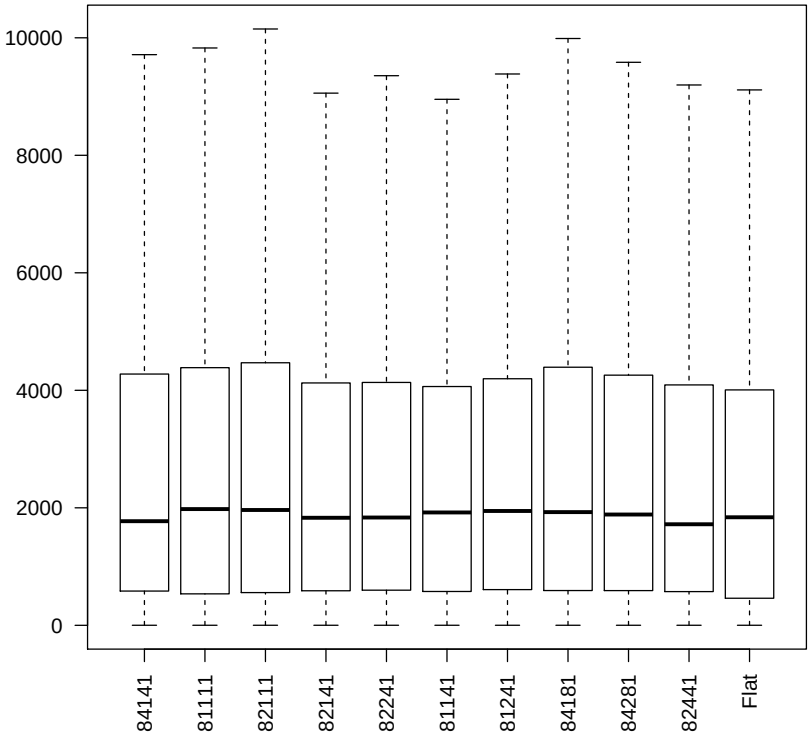


Figure 5.51: The top weighting configurations and flat configuration for all systems and the LMPBV corpus. The x axis is the configuration, while the y axis is the effectiveness measure.

Config	MRR
$C\{L=8,M=1,P=4,B=1,V=1\}$	0.0081
$C\{L=8,M=2,P=4,B=1,V=1\}$	0.0081
$C\{L=8,M=4,P=1,B=1,V=1\}$	0.0081
$C\{L=8,M=4,P=2,B=1,V=1\}$	0.0081
$C\{L=8,M=2,P=4,B=2,V=1\}$	0.0081
$C\{L=8,M=2,P=1,B=1,V=1\}$	0.0081
$C\{L=8,M=1,P=8,B=1,V=1\}$	0.0081
$C\{L=8,M=2,P=2,B=1,V=1\}$	0.0081
$C\{L=8,M=4,P=1,B=2,V=1\}$	0.0081
$C\{L=8,M=4,P=2,B=2,V=1\}$	0.0081

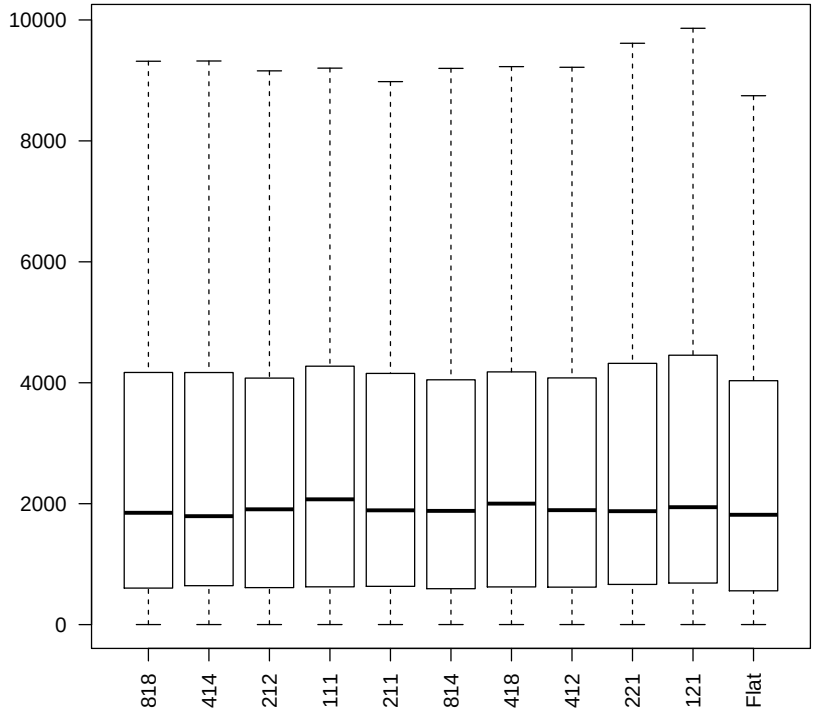
(a) LMPBV Combined

Config	MRR
$C\{L=8,M=4,P=1,B=4,V=1\}$	0.0205
$C\{L=8,M=1,P=1,B=1,V=1\}$	0.0204
$C\{L=8,M=2,P=1,B=1,V=1\}$	0.0204
$C\{L=8,M=2,P=1,B=4,V=1\}$	0.0203
$C\{L=8,M=2,P=2,B=4,V=1\}$	0.0203
$C\{L=8,M=1,P=1,B=4,V=1\}$	0.0200
$C\{L=8,M=1,P=2,B=4,V=1\}$	0.0199
$C\{L=8,M=4,P=1,B=8,V=1\}$	0.0199
$C\{L=8,M=4,P=2,B=8,V=1\}$	0.0198
$C\{L=8,M=2,P=4,B=4,V=1\}$	0.0193

(b) LMPBV Title

Table 5.109: Top 10 configurations for all systems

(a) CSB Combined



(b) CSB Title

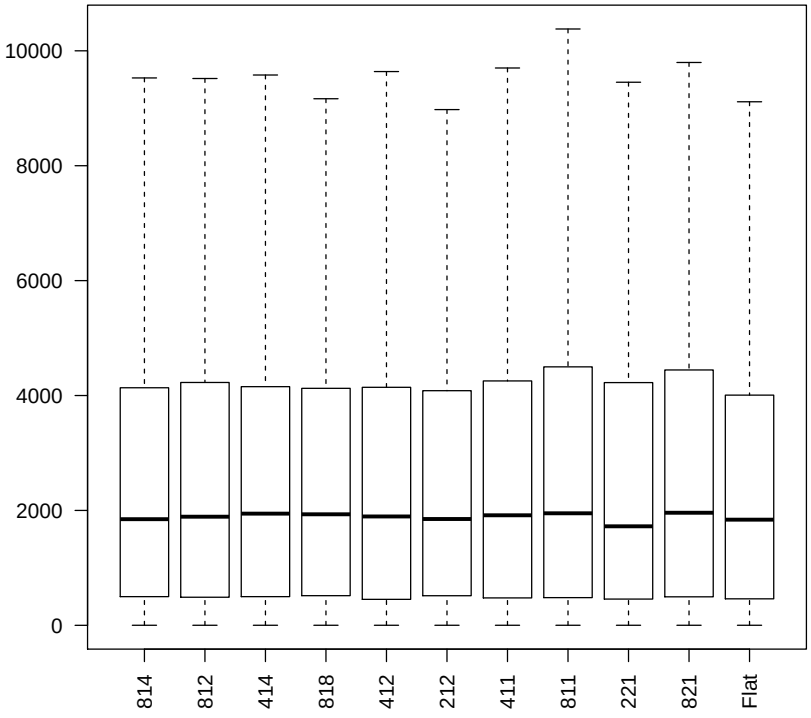


Figure 5.52: The top weighting configurations and flat configuration for all systems and the CSB corpus. The x axis is the configuration, while the y axis is the effectiveness measure.

Config	MRR
$C\{C=8,S=1,B=8\}$	0.0114
$C\{C=4,S=1,B=4\}$	0.0113
$C\{C=2,S=1,B=2\}$	0.0111
$C\{C=1,S=1,B=1\}$	0.0107
$C\{C=2,S=1,B=1\}$	0.0092
$C\{C=8,S=1,B=4\}$	0.0089
$C\{C=4,S=1,B=8\}$	0.0086
$C\{C=4,S=1,B=2\}$	0.0086
$C\{C=2,S=2,B=1\}$	0.0085
$C\{C=1,S=2,B=1\}$	0.0085

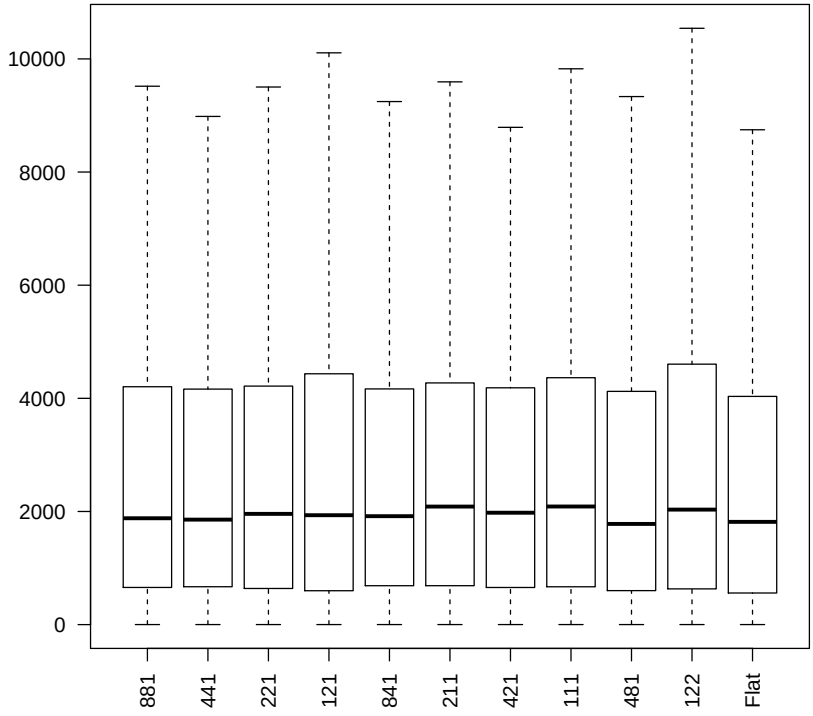
(a) CSB Combined

Config	MRR
$C\{C=8,S=1,B=4\}$	0.0487
$C\{C=8,S=1,B=2\}$	0.0486
$C\{C=4,S=1,B=4\}$	0.0482
$C\{C=8,S=1,B=8\}$	0.0480
$C\{C=4,S=1,B=2\}$	0.0465
$C\{C=2,S=1,B=2\}$	0.0460
$C\{C=4,S=1,B=1\}$	0.0454
$C\{C=8,S=1,B=1\}$	0.0451
$C\{C=2,S=2,B=1\}$	0.0428
$C\{C=8,S=2,B=1\}$	0.0420

(b) CSB Title

Table 5.110: Top 10 configurations for all systems

(a) ICL Combined



(b) ICL Title

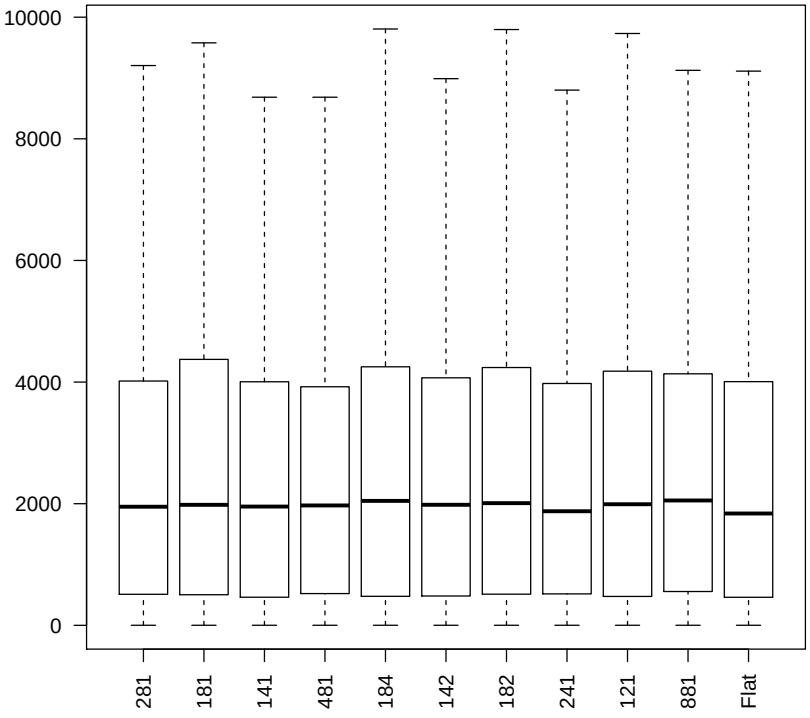


Figure 5.53: The top weighting configurations and flat configuration for all systems and the ICL corpus. The x axis is the configuration, while the y axis is the effectiveness measure.

Config	MRR
$C\{I=8,C=8,L=1\}$	0.0118
$C\{I=4,C=4,L=1\}$	0.0115
$C\{I=2,C=2,L=1\}$	0.0102
$C\{I=1,C=2,L=1\}$	0.0089
$C\{I=8,C=4,L=1\}$	0.0089
$C\{I=2,C=1,L=1\}$	0.0087
$C\{I=4,C=2,L=1\}$	0.0086
$C\{I=1,C=1,L=1\}$	0.0086
$C\{I=4,C=8,L=1\}$	0.0085
$C\{I=1,C=2,L=2\}$	0.0085

(a) ICL Combined

Config	MRR
$C\{I=2,C=8,L=1\}$	0.0459
$C\{I=1,C=8,L=1\}$	0.0457
$C\{I=1,C=4,L=1\}$	0.0449
$C\{I=4,C=8,L=1\}$	0.0444
$C\{I=1,C=8,L=4\}$	0.0423
$C\{I=1,C=4,L=2\}$	0.0413
$C\{I=1,C=8,L=2\}$	0.0402
$C\{I=2,C=4,L=1\}$	0.0402
$C\{I=1,C=2,L=1\}$	0.0399
$C\{I=8,C=8,L=1\}$	0.0395

(b) ICL Title

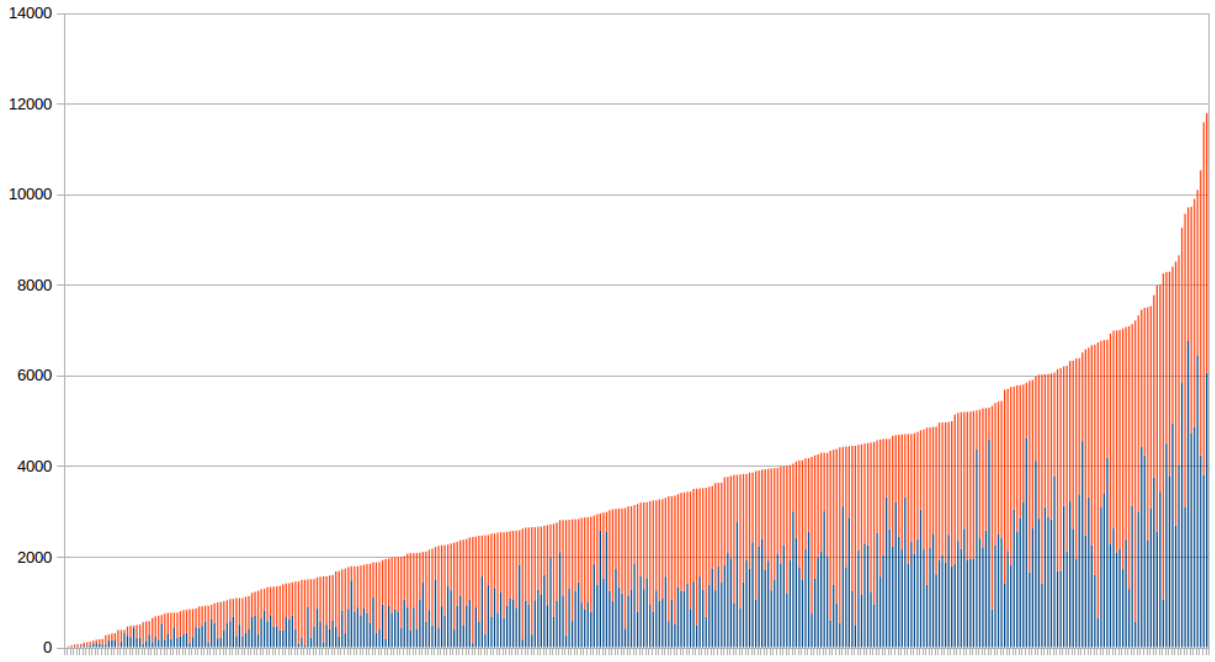
Table 5.111: Top 10 configurations for all systems

measures. For the Title query type, there is a difference between the top ten configurations of .0067. Comments and the body continue to be the most weighted lexicons while the signature is once again only weighting in the ninth and tenth positions. For the top two configurations, the ratio of comments to body has increase to 2:1 and then 4:1, while in the third and fourth positions, the comments and body have a 1:1 ratio again. The unweighted configuration is no longer amongst the top ten configurations. The $C\{C = 2, S = 1, B = 2\}$ configuration now has the smallest spread of the ten configurations.

In Figure 5.53 and Table 5.111, you can see the boxplots and MRRs for the ICL corpus. There is a .0033 difference between the top configurations for the Combined query type. For the top three configurations there is a 1:1 ratio between the identifiers and the comments. In the fourth position there is a 1:2 ratios between identifiers and comments which switches to a 2:1 ratio for the fifth through the seventh positions. The literals are only given a weighting factor of 2 in the tenth position. The boxplots show that the flat corpus has a similar spread to $C\{I = 4, C = 2, L = 1\}$. The two make up the smallest spreads for the configurations. For the Title query type, there is a difference of .0064 between the top configurations. For the top nine configurations, the comments have a higher weighting factor than either the identifiers or the literals. The identifiers are only weighted in four of the ten configurations, while the literals are only weighting in three of the top ten configurations. The first and third configurations have a ratio of 1:4 between the identifiers and the comments with the second and third having comments as the only weighted lexicon. The configurations $C\{I = 1, C = 4, L = 1\}$, $C\{I = 4, C = 8, L = 1\}$, and $C\{I = 2, C = 4, L = 1\}$, have the similar spreads and make up the smallest spreads of the top configurations.

I conducted a Friedman test with a post-hoc analysis on the top configurations and the flat

(a) CSB



(b) ICL

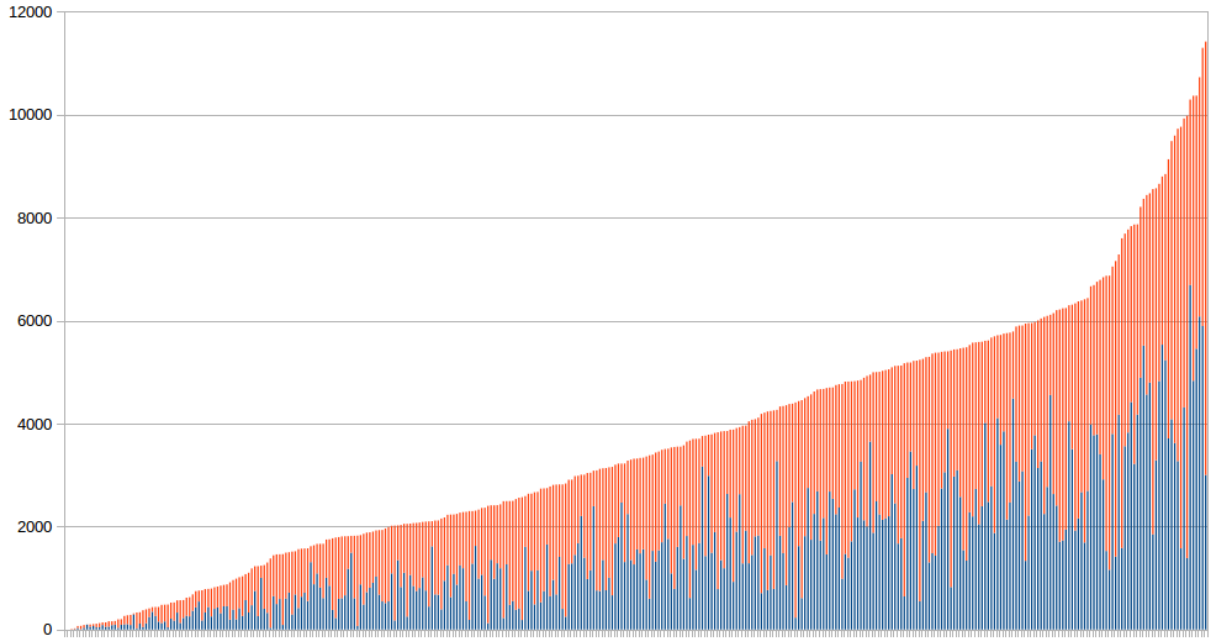


Figure 5.54: Stacked bargraphs representing the distance from the best query from structural weighting to the average (bottom) and the worst(top) for all systems. Graph is ordered by distance from best to worst.

(c) LPMBV

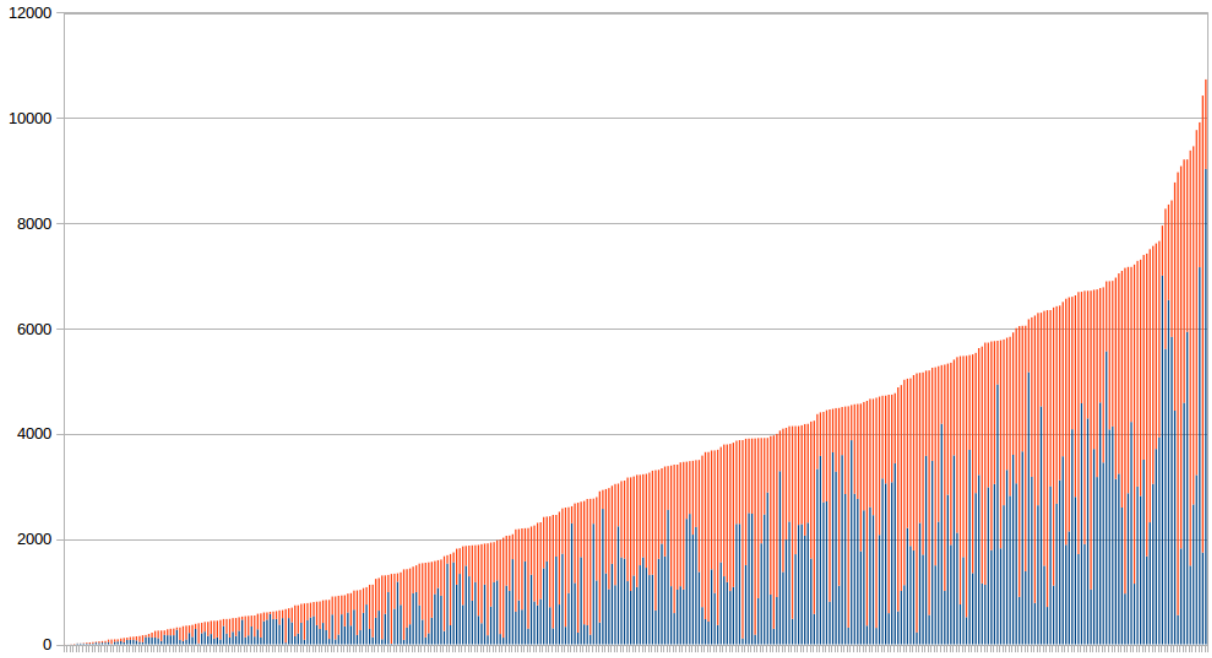


Figure 5.54: Stacked bargraphs representing the distance from the best query from structural weighting to the average (bottom) and the worst(top) for all systems. Graph is ordered by distance from best to worst.

corpus. I chose to only look at the top configurations and the flat corpus to lower the number of possible pairings that need to be compared. For the Combined query type, there were a large number of significant differences for each of the corpora. For CSB, 41 significant differences were found. For the ICL corpus, there were 45 significant differences identified, while for the LMPBV corpus there were 24 significant differences found. For each of the three corpora, the flat corpus was identified to be significantly different from all weighting configurations. For the Title query type, the number of significant differences were decreased for the CSB and ICL corpora with only 11 identifiers for the CSB corpus and 24 for the ICL corpus. For the LMPBV corpus there was an increase in the number of significant differences to a total of 44. For both ICL and LMPBV the flat corpus was significantly different from the weighting configurations, while for the CSB corpus the flat corpus was not significantly different from the weighting configurations.

Query	CSB	ICL	LMPBV
Best	0.0808	0.0814	0.0384
Average	0.0053	0.0073	0.0024
Worst	0.0034	0.0036	0.0012

Table 5.112: MRRs for choosing the best, average, and worst case for each feature for all systems from structural weighting

	CSB	ICL	LMPBV
Weighted	96	96	95

Table 5.113: The percentage of time that weighting each corpus improved the results for all systems

I computed the best, worst, and average cases for each feature for each corpus using both query types. The differences between each case can be found in Figure 5.54. The greatest distance between the best query and the worst query for the three corpora is found in the CSB corpus at 11,785. The greatest mean distance between the best and the worst query is found in the ICL corpus at a mean distance of 3,636. ICL also showed the greatest mean distance between the best query and the average case at 1,561. The smallest mean distance between the best and the worst case is found for the LMPBV corpus with a mean distance of 3,323. This corpus also had the smallest mean distance between the best and the average case with a mean distance of 1,458. There was a tie between the smallest distance between the best and worst query with a 0 for both ICL and CSB. LMPBV also had minimum distance of 0 between the best and the average case. For each of the three corpora, the mean distance between the best and the average query is smaller than the mean between the average and the worst.

The MRRs have been computed for each case and the results can be found in Table 5.112.

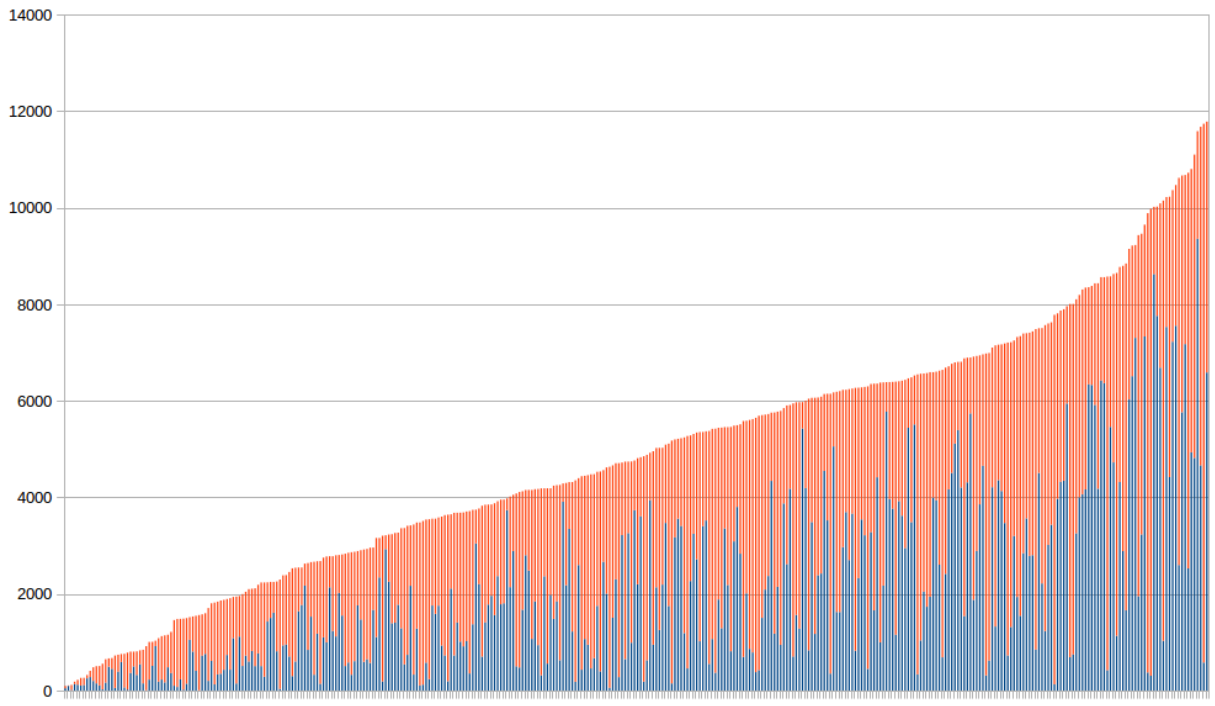


Figure 5.55: Stacked bargraphs representing the distance from the best query to the average (bottom) and the worst(top) from structural weighting for all systems. Graph is ordered by distance from best to worst.

Comparing the MRRs between the best cases and the results of the best structural combinations shows large increases from the individual corpora. The largest increase occurs for the ICL corpus with a difference between the best queries and the top weighting configuration of .0355. The smallest difference occurs for the LMPBV corpus with a difference of .0179. Of the three corpora, the largest MRR can also be found for the ICL corpus while the smallest MRR can be found for the LMPBV corpus.

I also computed the percentage of times that the best query was a result of one of the weighting configurations. This helps to understand whether the weighted or unweighted configurations are having a bigger impact on the results for each corpus. The results of this calculation

	Best	Average	Worst
MRR	0.1072	0.0019	0.0004

Table 5.114: MRRs for choosing the best, average, and worst case for each feature from all corpora and all structural weighting for all systems

	CSB	ICL	LMPBV	Flat
Percentage	33	21	23	23

Table 5.115: Percentages for each corpus where the best query was found from all corpora and all structural weighting for all systems

can be found in Table 5.113. For each of three corpora, the unweighted configuration has the best queries at most 5% of the time.

I performed a Friedman test with a wilcoxon post-hoc for the best, average, and worst case of each feature for each of the four corpora. The results of the analysis showed that there is a significant difference between each case.

I computed the best, worst, and average cases for each feature across all corpora using both query types. The differences between each case can be found in Figure 5.55. The greatest distance between the best query and the worst query is 11,785, while the mean distance between the best and the worst query is 4,897. The mean distance between the best query and the average case was 2,042. The smallest distance between the best and worst case was 95, while the smallest distance between the best and average cases was found to be 45. Each of these values are greater than the values for the individual corpora. The mean distance between the best and the average query is smaller than the mean between the average and the worst.

I computed the MRRs for these three cases and the results can be found in Table 5.114. The results of this computation show a large increase, of .0258, from the individual corpora to taking

the best across all queries. To understand which corpus is contributing the highest number of best queries from the corpora, I computed the percentage for each query and recorded the results in Table 5.115. From this table it can be seen that the CSB corpus contributed the largest percentage of the queries for the features. This corpus was followed by a tie between the LMPBV corpus and the flat corpus with a difference of 10%. The ICL corpus has the lowest percentage with 21%. A weighting configuration was used for 74% of the best queries. Adjusting for the flat corpus shows that either a weighting configuration of the flat corpus was used 97% of the time. This indicates that the unweighted configurations of one of the others corpora was used for only 3% of the best queries in the results.

I performed a Friedman test with a wilcoxon post-hoc for the best, average, and worst case of each feature for each of the four corpora. The results of the analysis showed that there is a significant difference between each case.

5.3.4 How does the best configuration of structural field combination and weighting affect the accuracy of a structured retrieval-based FLT?

In the previous questions, I looked at the different dimensions that can affect the query results in the structured retrieval process. In this question I look at the structural combination together with the weighting configuration. Each of these questions only provide a piece of the picture when looking at the effects of this process, however taking them together can give a better understand than any one question would. It is important to note that in this question, I am not looking for the best possible results. This would require optimizing the α and β values for the language model and finding the best text to include in the query. Instead, for feasibility of the study, I am using only the Title query type and looking at the relative differences. In the previous questions, approximately 25% of the best queries came from the Combined query type.

5.3.4.1 ArgoUML

The top combinations for ArgoUML and the LMPBV were computed. I show boxplots of the effectiveness measures in Figure 5.56 and the MRRs in Table 5.116. To save space, I list the configurations in the boxplots with the combination and the weighting schemes separated by a '_'. There is a difference of .0119 between the top configurations. For each of the top configurations, the leading comments are included. The only combination to include local variables is the top configuration, while the body comments are included in eight configurations, the method names in five, and the parameters in four. No configuration uses a weighting factor of 8 for any lexicon, and the most common weighting factors are 1 and 2. The smallest spread comes from the $C\{L = 2, M = 1, B = 1\}$ with the flat corpus being amongst the largest.

The boxplots for the CSB corpus can be found in Figure 5.57, while the MRRs can be found in Table 5.117. There is a difference of .0079 in the top configurations. The most common combination across the top configurations uses the full corpus. In seven of the configurations where the full corpus is used, the comments are given a heavier weighting than the signature or the body. Each of the top six configurations give the body a weighting factor of 1, while the top two configurations give the signature a weighting factor of 2. Looking at the boxplots, there is not a clear winner for the smallest spread. The full corpus with no weighting has the smallest median value, while the configuration $C\{C = 2, S = 2, B = 1\}$ has the smallest 1Q and 3Q values.

For the ICL corpus, the boxplots can be found in Figure 5.58, while the MRRs can be found in Table 5.118. The difference between the top configurations is .0153. Literals are present in each of the top configurations with the top configuration being the unweighted literals alone. Comments are the second most common lexicon, being present in nine of the ten configurations

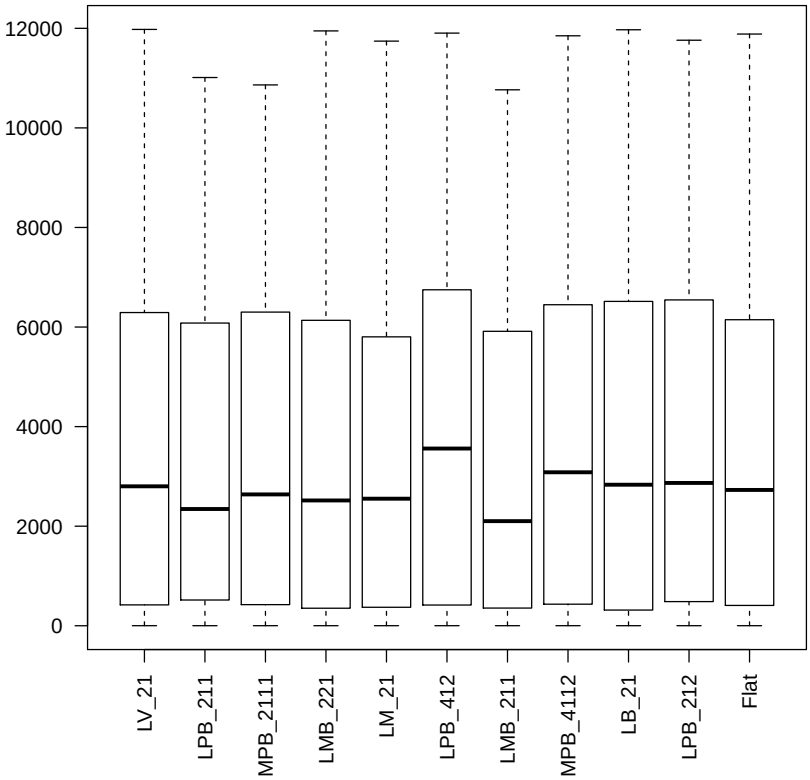


Figure 5.56: The top configurations and flat configuration for ArgoUML and the LMPBV corpus. The x axis is the configuration, while the y axis is the effectiveness measure.

Config	MRR
$C\{L=2,V=1\}$	0.0704
$C\{L=2,P=1,B=1\}$	0.0703
$C\{L=2,M=1,P=1,B=1\}$	0.0696
$C\{L=2,M=2,B=1\}$	0.0630
$C\{L=2,M=1\}$	0.0607
$C\{L=4,P=1,B=2\}$	0.0601
$C\{L=2,M=1,B=1\}$	0.0600
$C\{L=4,M=1,P=1,B=2\}$	0.0599
$C\{L=2,B=1\}$	0.0598
$C\{L=2,P=1,B=2\}$	0.0585

Table 5.116: Top 10 configurations from all combinations for ArgoUML and the LMPBV corpus

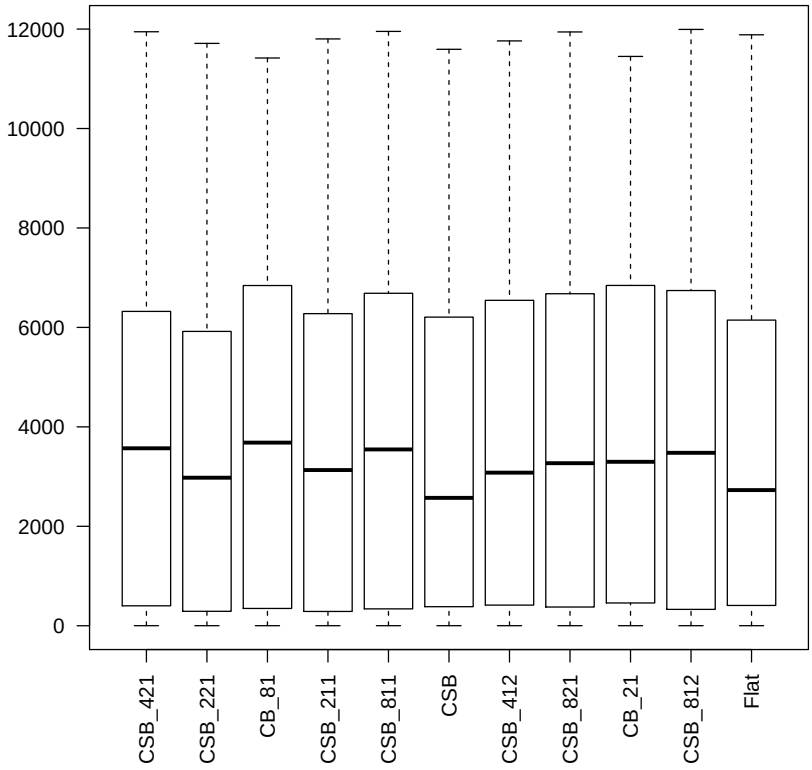


Figure 5.57: The top configurations and flat configuration for ArgoUML and the CSB corpus. The x axis is the configuration, while the y axis is the effectiveness measure.

Config	MRR
$C\{C=4,S=2,B=1\}$	0.0713
$C\{C=2,S=2,B=1\}$	0.0709
$C\{C=8,B=1\}$	0.0698
$C\{C=2,S=1,B=1\}$	0.0695
$C\{C=8,S=1,B=1\}$	0.0693
$C\{C=1,S=1,B=1\}$	0.0688
$C\{C=4,S=1,B=2\}$	0.0688
$C\{C=8,S=2,B=1\}$	0.0636
$C\{C=2,B=1\}$	0.0634
$C\{C=8,S=1,B=2\}$	0.0634

Table 5.117: Top 10 configurations from all combinations for ArgoUML and the CSB corpus

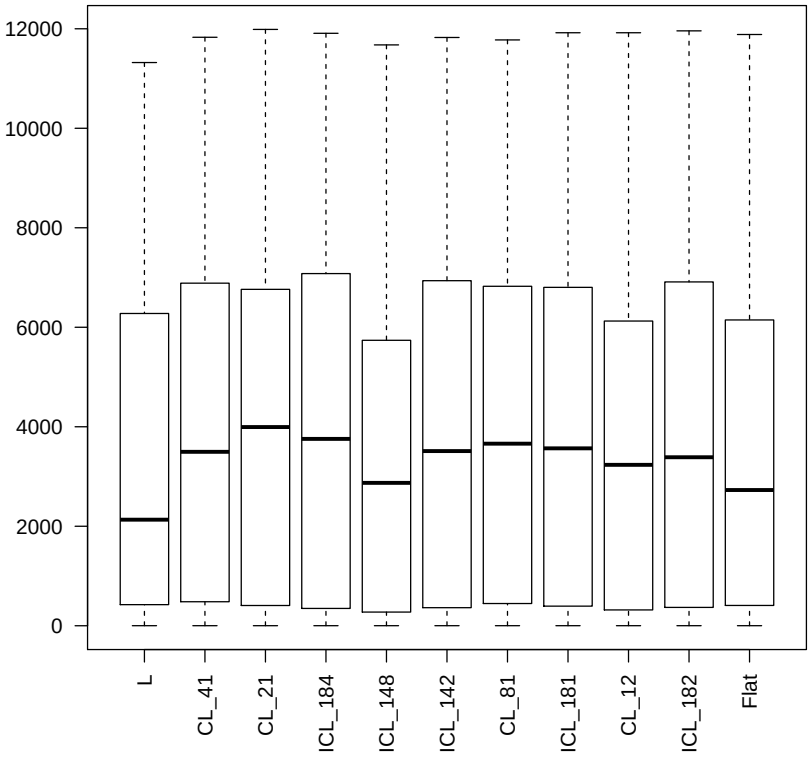


Figure 5.58: The top configurations and flat configuration for ArgoUML and the ICL corpus. The x axis is the configuration, while the y axis is the effectiveness measure.

Config	MRR
$C\{L=1\}$	0.0703
$C\{C=4,L=1\}$	0.0697
$C\{C=2,L=1\}$	0.0624
$C\{I=1,C=8,L=4\}$	0.0613
$C\{I=1,C=4,L=8\}$	0.0609
$C\{I=1,C=4,L=2\}$	0.0601
$C\{C=8,L=1\}$	0.0592
$C\{I=1,C=8,L=1\}$	0.0588
$C\{C=1,L=2\}$	0.0555
$C\{I=1,C=8,L=2\}$	0.0550

Table 5.118: Top 10 configurations from all combinations for ArgoUML and the ICL corpus

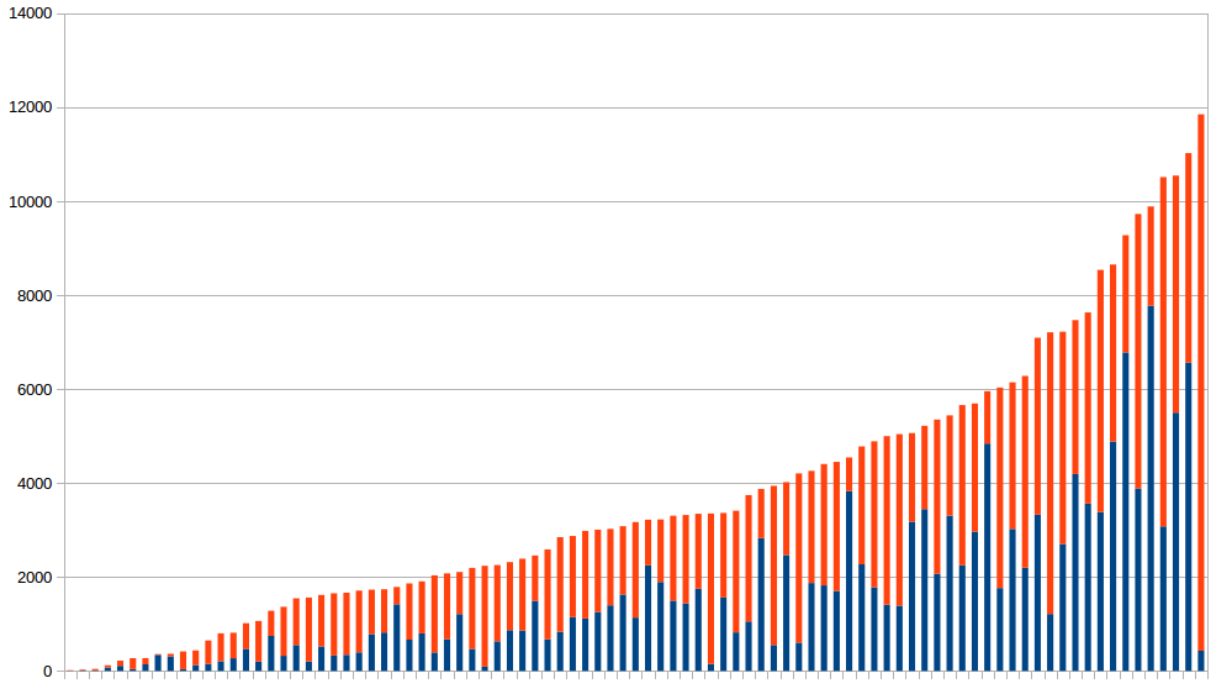
and often having a higher weighting factor than the literals. Identifiers are present in only five of the configurations, but are not weighted in any of the combinations. The boxplots show that the smallest spread comes for the $C\{I = 1, C = 4, L = 8\}$ configuration.

I conducted a Friedman test with a post-hoc analysis on the top configurations and the flat corpus. I chose to only look at the top configurations and the flat corpus to lower the number of possible pairings that need to be compared. For the CSB corpus, 20 significant differences were found between the top configurations with none found with the flat corpus and the other configurations. For the ICL corpus, 7 significant differences were found, with 6 of the differences between the flat corpus and the other configurations. Finally for the LMPBV corpus, 21 significant differences were found with none between the flat corpus and the other configurations.

I computed the best, worst, and average cases for each feature for each corpus. The differences between each case can be found in Figure 5.59. The greatest distance between the best query and the worst query for the three corpora is found in the LMPBV corpus at 11,973. The greatest mean distance between the best and the worst query is found in the LMPBV corpus at a mean distance of 6,037. LMPBV also showed the greatest mean distance between the best query and the average case at 2,596. The smallest mean distance between the best and the worst case is found for the CSB corpus with a mean distance of 3,726. This corpus also had the smallest mean distance between the best and the average case with a mean distance of 1,620. There was a tie between the smallest distance between the best and worst query with a 14 for both ICL and CSB. These corpora also had the minimum distance of 2 between the best and the average case. For each of the three corpora, the mean distance between the best and the average query is smaller than the mean between the average and the worst.

The MRRs have been computed for each case and the results can be found in Table 5.119.

(a) CSB



(b) ICL

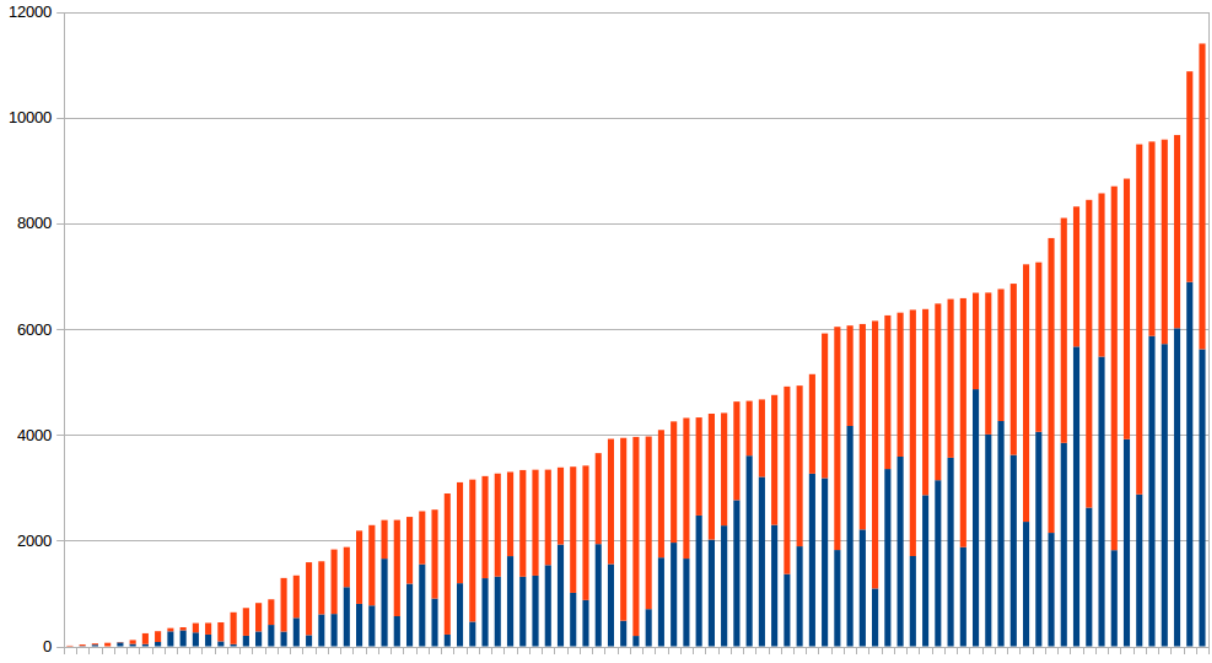


Figure 5.59: Stacked bargraphs representing the distance from the best query from all combinations to the average (bottom) and the worst(top) for ArgoUML. Graph is ordered by distance from best to worst.

(c) LPMBV

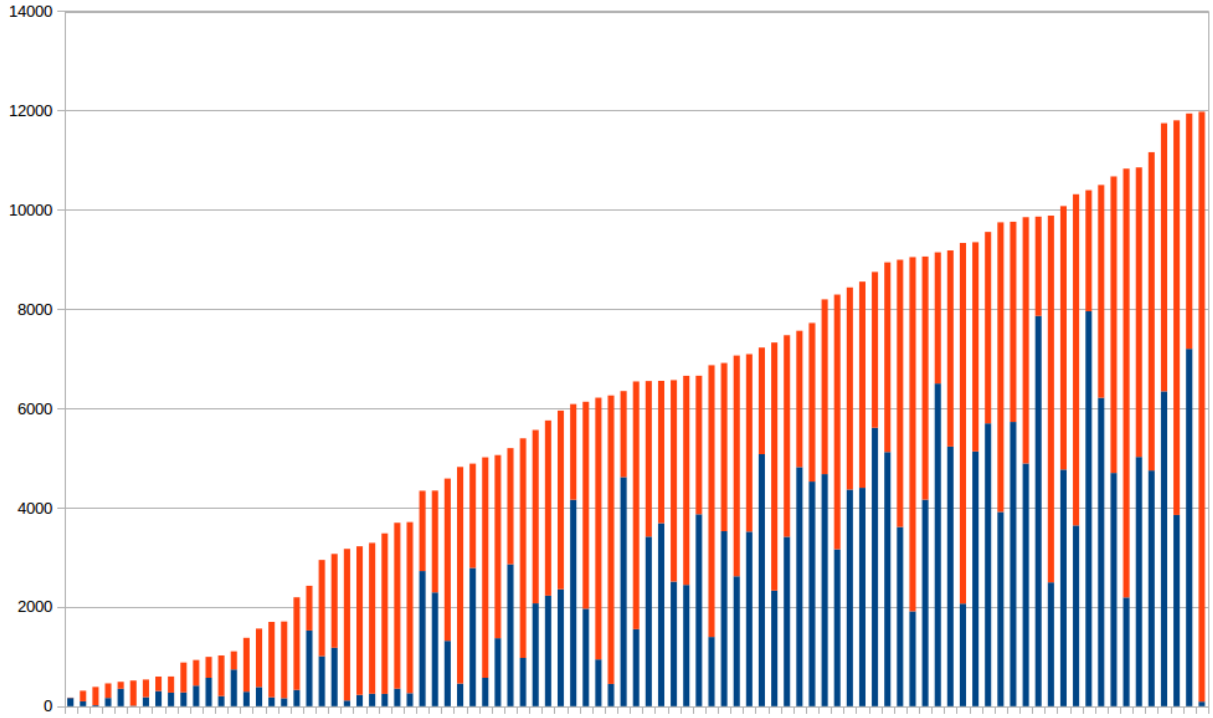


Figure 5.59: Stacked bargraphs representing the distance from the best query from all combinations to the average (bottom) and the worst(top) for ArgoUML. Graph is ordered by distance from best to worst.

Comparing the MRRs between the best cases and the results of the best configurations shows large increases from the individual corpora. The largest increase occurs for the LMPBV corpus with a difference between the best queries and the top configuration of .1093. The smallest difference occurs for the CSB corpus with a difference of .0455. Of the three corpora, the largest MRR can also be found for the LMPBV corpus while the smallest MRR can be found for the CSB corpus.

I also computed the percentage of times that the best query was a result of one of the weighting configurations. This helps to understand whether the weighted or unweighted configurations are having a bigger impact on the results for each corpus. The results of this calculation can be found in Table 5.120. For both ICL and CSB, a weighting configuration was used 100% of

Query	CSB	ICL	LMPBV
Best	0.1168	0.1341	0.1797
Average	0.0071	0.0174	0.0029
Worst	0.0018	0.0017	0.0004

Table 5.119: MRRs for choosing the best, average, and worst case for each feature for ArgoUML from all combinations

	CSB	ICL	LMPBV
Weighted	100	100	92

Table 5.120: The percentage of time that weighting each corpus improved the results for ArgoUML

B	S	SB	C	CB	CS	CSB
0	0	30	0	23	15	30

(a) CSB

L	C	CL	I	IL	IC	ICL
0	0	21	0	28	14	35

(b) ICL

V	B	BV	P	PV	PB	PBV	M	MV	MB	MBV
2	4	9	11	6	4	0	9	9	4	0
MP	MPV	MPB	MPBV	L	LV	LB	LBV	LP	LPV	
0	0	2	0	6	6	9	0	2	0	
LPB	LPBV	LM	LMV	LMB	LMBV	LMP	LMPV	LMPB	LMPBV	
0	0	2	0	4	0	2	0	0	0	

(c) LMPBV

Table 5.121: Percentage of the best queries obtained from each structural combination for ArgoUML

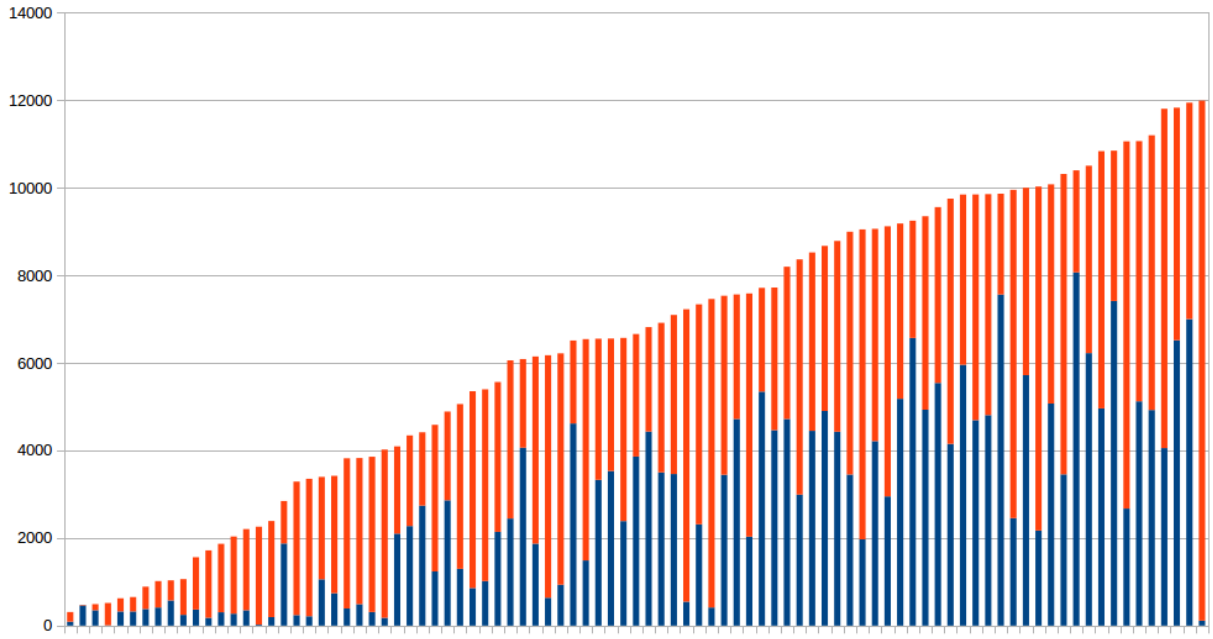


Figure 5.60: Stacked bargraphs representing the distance from the best query to the average (bottom) and the worst(top) from all combinations for ArgoUML. Graph is ordered by distance from best to worst.

the time, while for LMPBV a weighting configuration was used 92% of the time. This indicates that the unweighted configuration only had 8% of the best queries in the LMPBV corpus.

The percentages for the individual combinations having the best queries regardless of weighting can be found in Table 5.121. In the case of the CSB corpus, there is a tie for the highest percentage between the structure combined with the body (SB) and the full corpus. While each of the combinations that combined two lexicons contributed to the best queries, the individual lexicons by themselves did not. For the ICL corpus, the highest percentage was for the full corpus, with each of the two lexicon combinations contributing to the best queries and none of the one lexicon combinations contributing. For the LMPBV corpus, the highest percentage actually came from the parameters alone (P). The full corpus did not contribute to the best queries.

	Best	Average	Worst
MRR	0.2196	0.0028	0.0003

Table 5.122: MRRs for choosing the best, average, and worst case for each feature from all corpora and all combinations for ArgoUML

	CSB	ICL	LMPBV	Flat
Percentage	2	7	80	11

Table 5.123: Percentages for each corpus where the best query was found from all corpora and all combinations for ArgoUML

The percentages decreased for contributing to the best queries as the number of lexicons in the combination went up.

I performed a Friedman test with a wilcoxon post-hoc for the best, average, and worst case of each feature for each of the four corpora. The results of the analysis showed that there is a significant difference between each case.

I computed the best, worst, and average cases for each feature across all corpora using both query types. The differences between each case can be found in Figure 5.60. The greatest distance between the best query and the worst query is 11,986, while the mean distance between the best and the worst query is 6,445. The mean distance between the best query and the average case was 2,747. The smallest distance between the best and worst case was 311, while the smallest distance between the best and average cases was found to be 90. Each of these values are greater than the values for the individual corpora. The mean distance between the best and the average query is smaller than the mean between the average and the worst.

I computed the MRRs for these three cases and the results can be found in Table 5.122. The results of this computation show that there is another large increase from the individual corpora to

taking the best across all queries. To understand which corpus is contributing the highest number of best queries from the corpora, I computed the percentage for each query and recorded the results in Table 5.123. From this table it can be seen that the LMPBV corpus contributed the largest percentage of the queries for the features by far. The next closest corpus was the flat corpus with only 11% of the best queries. The combinations with the highest percentages from the three corpora include the combination of the signature and body (SB) from CSB with 2%, the comments combined with the literals (CL) with 4%, and the parameters only (P) from LMPBV with 10%. None of the full corpora contributed to the best queries. A weighting configurations was used for 80% of the best queries, after adjusting for the flat corpus, the percentage of best queries that do not use a weighting configuration or the flat corpus totals 9%.

I performed a Friedman test with a wilcoxon post-hoc for the best, average, and worst case of each feature for each of the four corpora. The results of the analysis showed that there is a significant difference between each case.

5.3.4.2 JabRef

The top combinations for JabRef and the LMPBV were computed. I show boxplots of the effectiveness measures in Figure 5.61 and the MRRs in Table 5.124. To save space, I list the configurations in the boxplots with the combination and the weighting schemes separated by a '_'. There is a difference of .0249 between the top configurations. For each of the top configurations, the leading comments are included with a weighting factor of 8 or 4. Method names are not included in any of the top configurations. Body comments are also included in all configurations with the most common weighting factors of 2 and 1. The parameters are included in four configurations, while local variables are used in five. In none of the configurations is there a lexicon given a higher

weighting factor than the leading comments. The largest spread comes from the flat corpus, while there is not a clear winner for the smallest spread.

The boxplots for the CSB corpus can be found in Figure 5.62, while the MRRs can be found in Table 5.125. There is a difference of .0297 in the top configurations. The most common combination across the top configurations uses the full corpus. In each of the configurations where the full corpus is used, the comments are given a heavier weighting than the signature or the body. The top two configurations are two lexicon configurations with the comments and the signature (CS) and the comments with the body (CB). CB appears in three of the top six configurations. Looking at the boxplots, there is not a clear winner for the smallest spread. The flat corpus has the largest spread of the configurations. The $C\{C = 2, S = 1\}$ configuration has the lowest 1Q value, while $C\{C = 4, S = 1, B = 1\}$ has the lowest median value.

For the ICL corpus, the boxplots can be found in Figure 5.63, while the MRRs can be found in Table 5.126. The difference between the top configurations is .0196. The two most common combinations used include the full corpus and the combination of the comments with the literals (CL). Comments are the most common lexicons in any of the configurations while literals are the second most common. In seven of the configurations that contain both comments and literals, comments are weighted heavier than literals. Identifiers are present in five of the top configurations and only receive a weighting in one of the configurations. $C\{C = 8, L = 1\}$ and $C\{C = 1\}$ have the smallest spreads, however have higher 1Q values than other configurations. The flat corpus has the largest spread of the configurations.

I conducted a Friedman test with a post-hoc analysis on the top configurations and the flat corpus. I chose to only look at the top configurations and the flat corpus to lower the number of possible pairings that need to be compared. No significant differences were identified for the

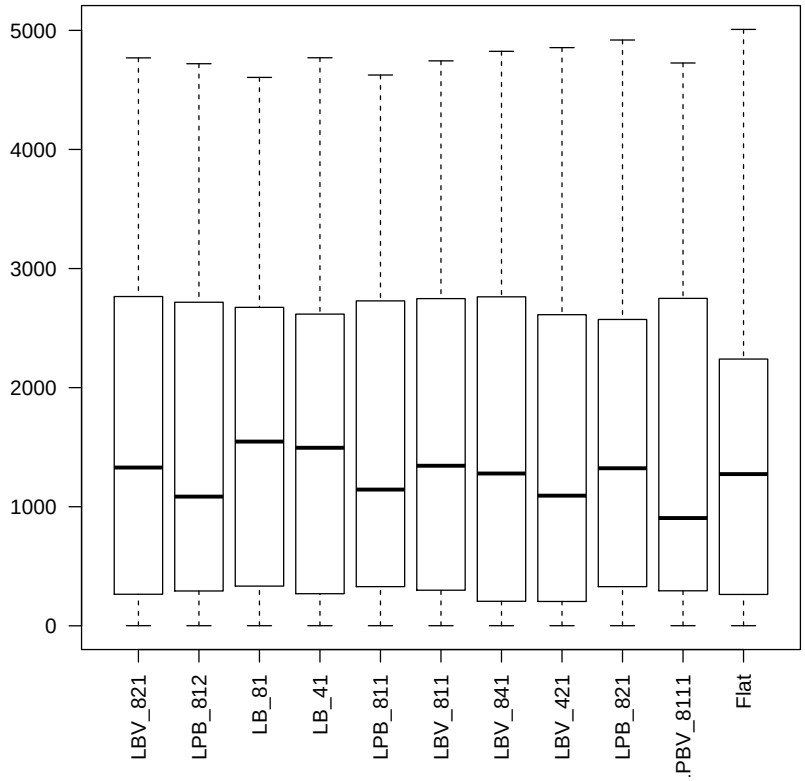


Figure 5.61: The top configurations and flat configuration for JabRef and the LMPBV corpus. The x axis is the configuration, while the y axis is the effectiveness measure.

Config	MRR
$C\{L=8,B=2,V=1\}$	0.1424
$C\{L=8,P=1,B=2\}$	0.1380
$C\{L=8,B=1\}$	0.1331
$C\{L=4,B=1\}$	0.1321
$C\{L=8,P=1,B=1\}$	0.1314
$C\{L=8,B=1,V=1\}$	0.1186
$C\{L=8,B=4,V=1\}$	0.1178
$C\{L=4,B=2,V=1\}$	0.1176
$C\{L=8,P=2,B=1\}$	0.1175
$C\{L=8,P=1,B=1,V=1\}$	0.1175

Table 5.124: Top 10 configurations from all combinations for JabRef and the LMPBV corpus

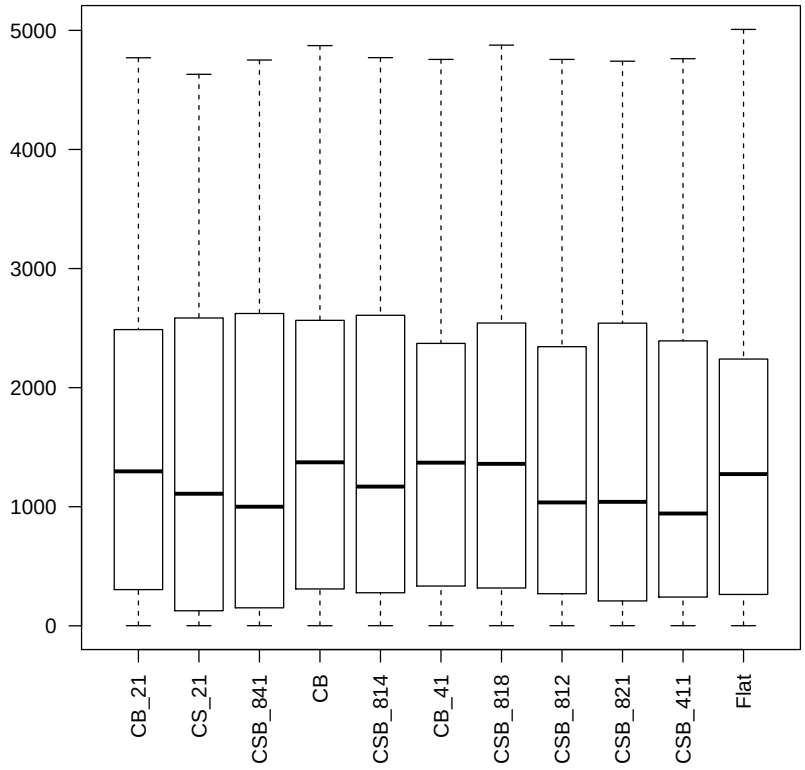


Figure 5.62: The top configurations and flat configuration for JabRef and the CSB corpus. The x axis is the configuration, while the y axis is the effectiveness measure.

Config	MRR
$C\{C=2,B=1\}$	0.1132
$C\{C=2,S=1\}$	0.1085
$C\{C=8,S=4,B=1\}$	0.1069
$C\{C=1,B=1\}$	0.1043
$C\{C=8,S=1,B=4\}$	0.0926
$C\{C=4,B=1\}$	0.0884
$C\{C=8,S=1,B=8\}$	0.0874
$C\{C=8,S=1,B=2\}$	0.0862
$C\{C=8,S=2,B=1\}$	0.0849
$C\{C=4,S=1,B=1\}$	0.0835

Table 5.125: Top 10 configurations from all combinations for JabRef and the CSB corpus

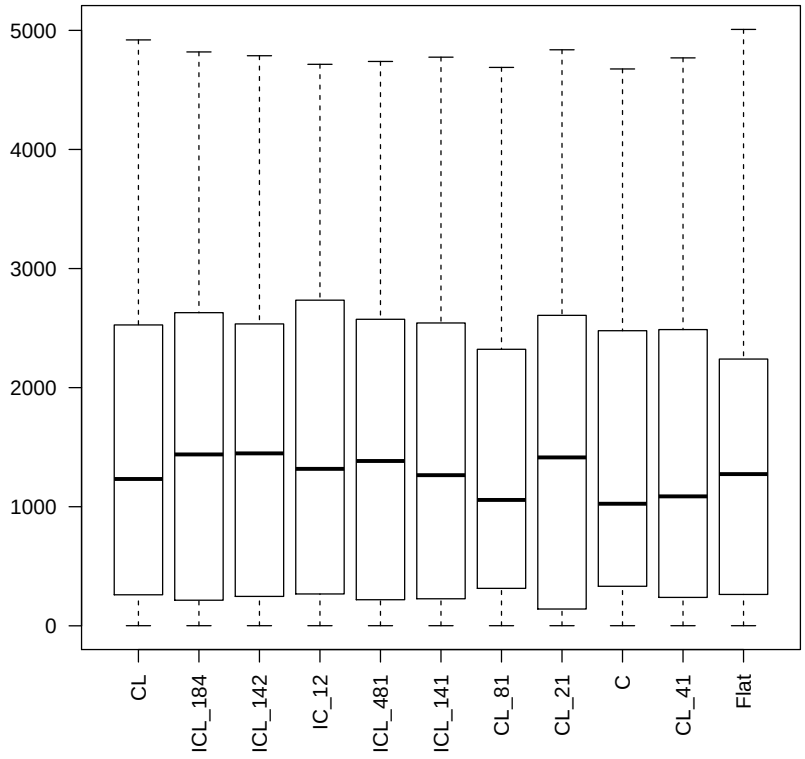


Figure 5.63: The top configurations and flat configuration for JabRef and the ICL corpus. The x axis is the configuration, while the y axis is the effectiveness measure.

Config	MRR
$C\{C=1,L=1\}$	0.0882
$C\{I=1,C=8,L=4\}$	0.0847
$C\{I=1,C=4,L=2\}$	0.0845
$C\{I=1,C=2\}$	0.0842
$C\{I=4,C=8,L=1\}$	0.0835
$C\{I=1,C=4,L=1\}$	0.0833
$C\{C=8,L=1\}$	0.0800
$C\{C=2,L=1\}$	0.0757
$C\{C=1\}$	0.0704
$C\{C=4,L=1\}$	0.0686

Table 5.126: Top 10 configurations from all combinations for JabRef and the ICL corpus

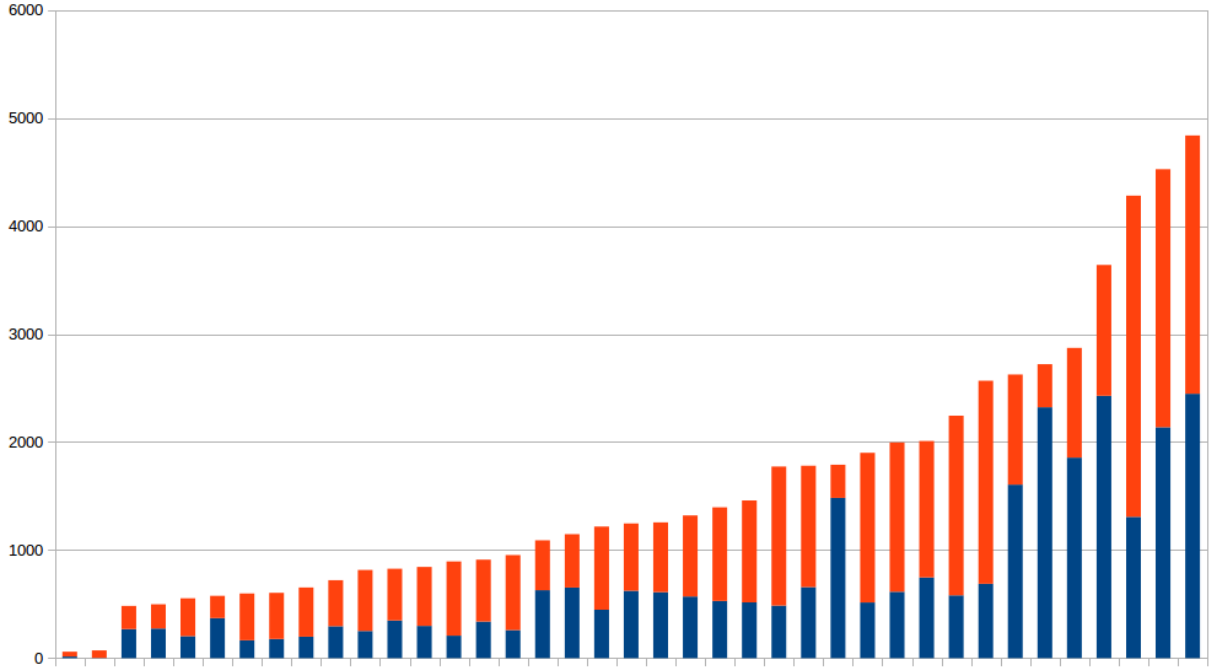
ICL or the CSB corpora, while only a single significant difference was found for the LMPBV configuration between the $C\{L = 8, P = 1, B = 1\}$ and $C\{L = 4, B = 2, V = 1\}$ configurations.

I computed the best, worst, and average cases for each feature for each corpus. The differences between each case can be found in Figure 5.64. The greatest distance between the best query and the worst query for the three corpora is found in the LMPBV corpus at 5,054. The greatest mean distance between the best and the worst query is found in the LMPBV corpus at a mean distance of 2,673. LMPBV also showed the greatest mean distance between the best query and the average case at 1,142. The smallest mean distance between the best and the worst case is found for the ICL corpus with a mean distance of 1,530. This corpus also had the smallest mean distance between the best and the average case with a mean distance of 693. The ICL corpus had the smallest distances with a distance between the best and worst query of 32 and a distance between the best query and the average case of 9. For each of the three corpora, the mean distance between the best and the average query is smaller than the mean between the average and the worst.

The MRRs have been computed for each case and the results can be found in Table 5.127. Comparing the MRRs between the best cases and the results of the best configurations shows large increases from the individual corpora. The largest increase occurs for the LMPBV corpus with a difference between the best queries and the top configuration of .1250. The smallest difference occurs for the ICL corpus with a difference of .0818. Of the three corpora, the largest MRR can also be found for the LMPBV corpus while the smallest MRR can be found for the ICL corpus.

I also computed the percentage of times that the best query was a result of one of the weighting configurations. This helps to understand whether the weighted or unweighted configurations are having a bigger impact on the results for each corpus. The results of this calculation can be found in Table 5.128. For both ICL and CSB, a weighting configuration was used 100% of

(a) CSB



(b) ICL

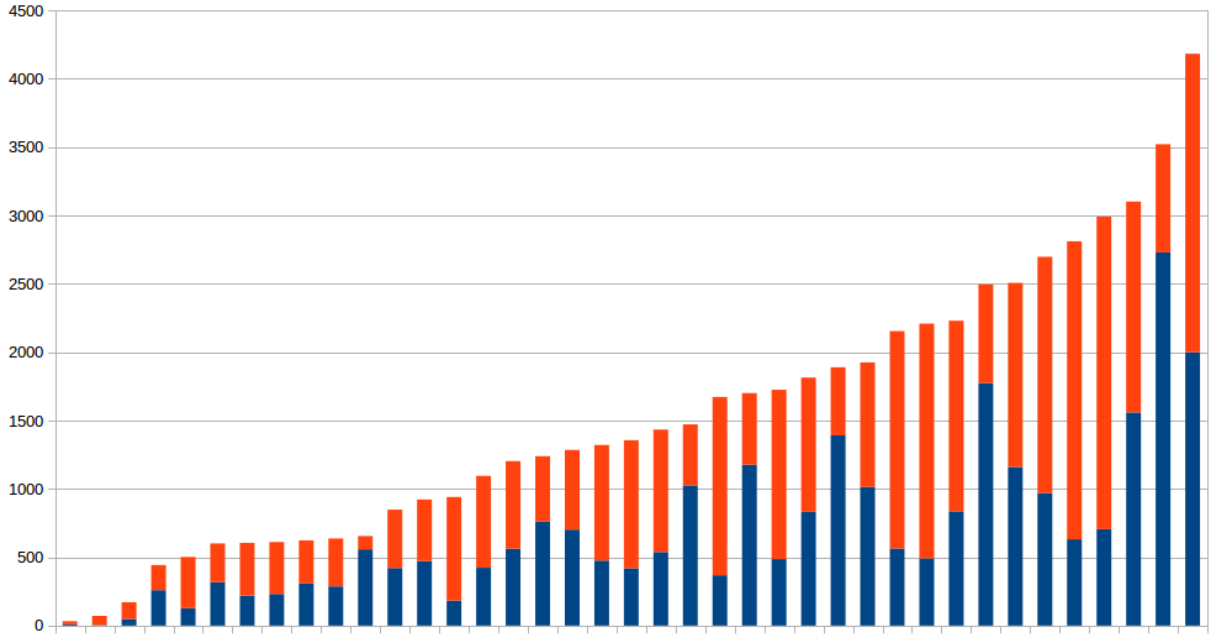


Figure 5.64: Stacked bargraphs representing the distance from the best query from all combinations to the average (bottom) and the worst(top) for JabRef. Graph is ordered by distance from best to worst.

(c) LPMBV

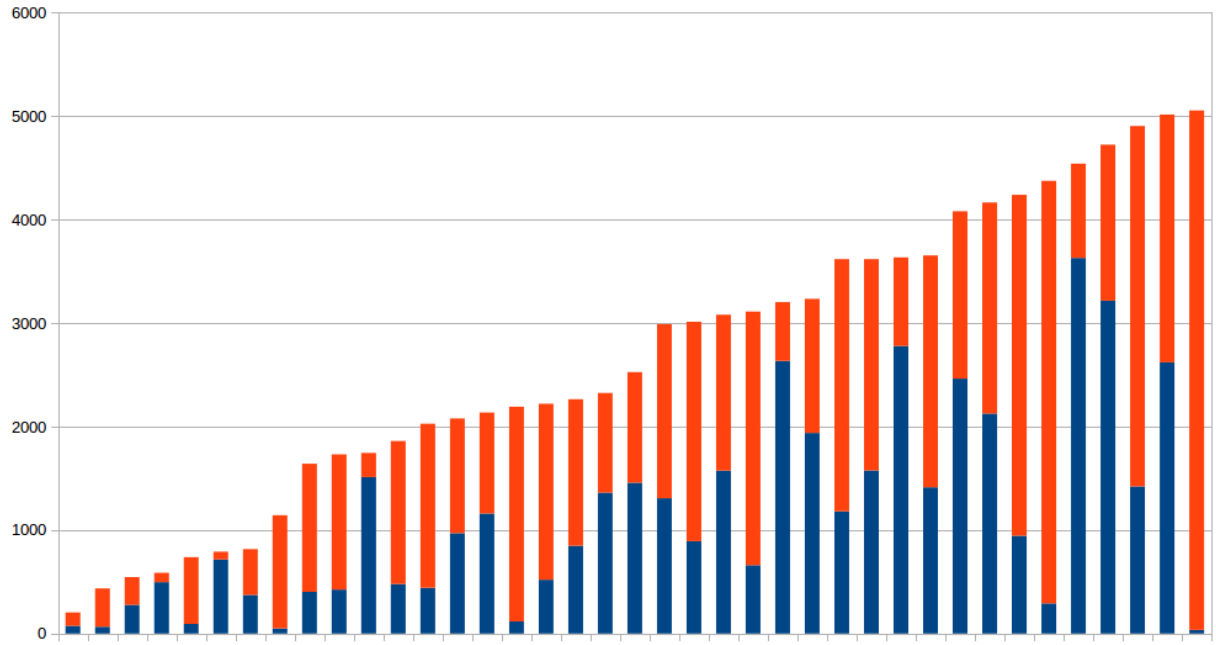


Figure 5.64: Stacked bargraphs representing the distance from the best query from all combinations to the average (bottom) and the worst(top) for JabRef. Graph is ordered by distance from best to worst.

the time, while for LMPBV a weighting configuration was used 87% of the time. This indicates that the unweighted configuration only had 13% of the best queries in the LMPBV corpus.

The percentages for the individual combinations having the best queries regardless of weighting can be found in Table 5.129. In the case of the CSB corpus, there is a tie for the highest percentage between the structure combined with the body (SB) and the full corpus. These two corpora make up 100% of the best queries for the CSB corpus. For the ICL corpus, the highest percentage was for the full corpus, with a tie for the remaining percentage between the comments combined with the literals and the identifiers combined with the literals. For the LMPBV corpus, the highest percentage came from the parameters alone (P). The full corpus did not contribute to the best queries.

I performed a Friedman test with a wilcoxon post-hoc for the best, average, and worst case

Query	CSB	ICL	LMPBV
Best	0.2015	0.1700	0.2674
Average	0.0117	0.0106	0.0030
Worst	0.0014	0.0018	0.0006

Table 5.127: MRRs for choosing the best, average, and worst case for each feature for JabRef from all combinations

	CSB	ICL	LMPBV
Weighted	100	100	87

Table 5.128: The percentage of time that weighting each corpus improved the results for JabRef

B	S	SB	C	CB	CS	CSB
0	0	50	0	0	0	50

(a) CSB

L	C	CL	I	IL	IC	ICL
0	0	28	0	28	0	42

(b) ICL

V	B	BV	P	PV	PB	PBV	M	MV	MB	MBV
0	12	6	18	12	6	12	0	6	6	0
MP	MPV	MPB	MPBV	L	LV	LB	LBV	LP	LPV	
0	6	0	0	0	6	0	6	0	0	
LPB	LPBV	LM	LMV	LMB	LMBV	LMP	LMPV	LMPB	LMPBV	
0	0	0	0	0	0	0	0	0	0	

(c) LMPBV

Table 5.129: Percentage of the best queries obtained from each structural combination for JabRef

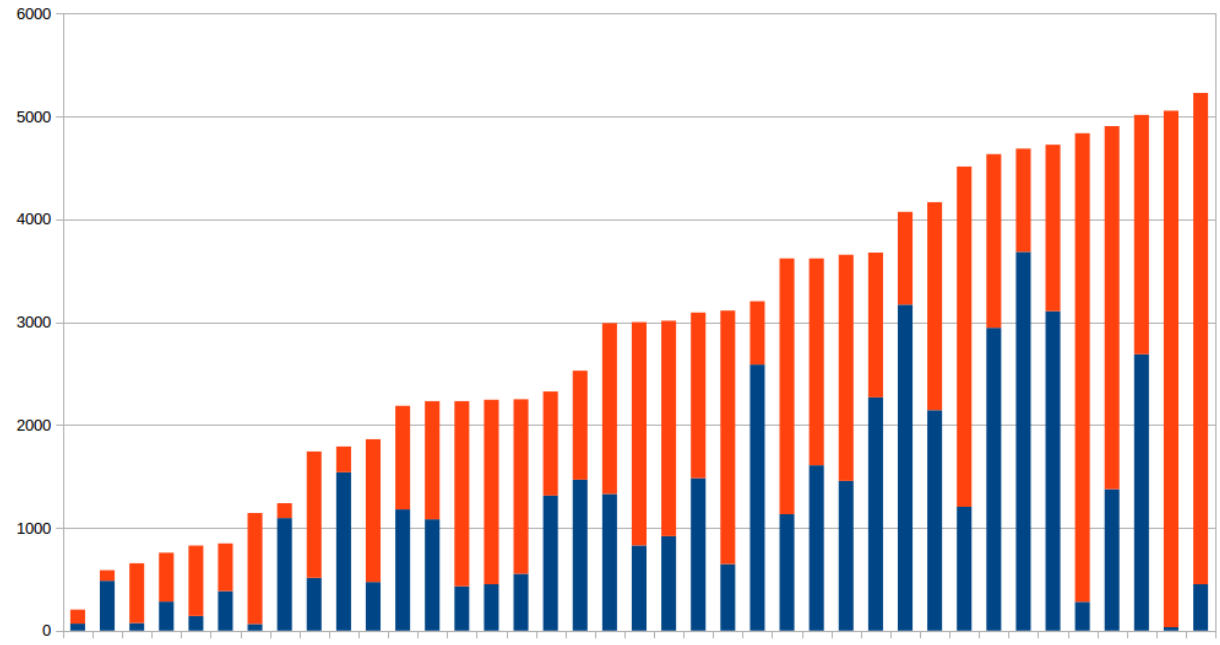


Figure 5.65: Stacked bargraphs representing the distance from the best query to the average (bottom) and the worst(top) from all combinations for JabRef. Graph is ordered by distance from best to worst.

of each feature for each of the four corpora. The results of the analysis showed that there is a significant difference between each case.

I computed the best, worst, and average cases for each feature across all corpora using both query types. The differences between each case can be found in Figure 5.65. The greatest distance between the best query and the worst query is 5,227, while the mean distance between the best and the worst query is 2,882. The mean distance between the best query and the average case was 1,202. The smallest distance between the best and worst case was 204, while the smallest distance between the best and average cases was found to be 68. Each of these values are greater than the values for the individual corpora. The mean distance between the best and the average query is smaller than the mean between the average and the worst.

I computed the MRRs for these three cases and the results can be found in Table 5.130. The

	Best	Average	Worst
MRR	0.3193	0.0029	0.0006

Table 5.130: MRRs for choosing the best, average, and worst case for each feature from all corpora and all combinations for JabRef

	CSB	ICL	LMPBV	Flat
Percentage	6	0	94	0

Table 5.131: Percentages for each corpus where the best query was found from all corpora and all combinations for JabRef

results of this computation show that there is another large increase from the individual corpora to taking the best across all queries. To understand which corpus is contributing the highest number of best queries from the corpora, I computed the percentage for each query and recorded the results in Table 5.131. From this table it can be seen that the LMPBV and the CSB corpora were the only two corpora to contribute to the top results. The LMPBV corpus contributed the largest percentage of the queries for the features by far. The CSB corpus only contributed 6% of the best queries. The combinations with the highest percentages from the two corpora include the full CSB corpus with 6% and the parameters only (P) from LMPBV with 18%. The CSB corpus was the only full corpus to contribute to the best queries. A weighting configurations was used for 79% of the best queries, since the flat corpus does not contribute to the best queries, the percentage of best queries that do not use a weighting configuration or the flat corpus totals 21%.

I performed a Friedman test with a wilcoxon post-hoc for the best, average, and worst case of each feature for each of the four corpora. The results of the analysis showed that there is a significant difference between each case.

5.3.4.3 jEdit

The top combinations for jEdit and the LMPBV were computed. I show boxplots of the effectiveness measures in Figure 5.66 and the MRRs in Table 5.132. To save space, I list the configurations in the boxplots with the combination and the weighting schemes separated by a '_'. There is a difference of only .0038 between the top configurations. The top configuration comes from using the unweighted body comments alone, however the body comments are not used again until the seventh configuration. The most common lexicons are the leading comments, parameters, and the local variables, with the leading comments appearing in eight of the ten configurations. Looking at the boxplots, the smallest spread comes from the flat corpus.

The boxplots for the CSB corpus can be found in Figure 5.67, while the MRRs can be found in Table 5.133. There is a difference of .0113 in the top configurations. The two most common lexicons include the comments and the body with body lexicons appearing in nine of the top ten configurations. The two most common combinations include the comments combined with the body and the full corpus. There is not a clear winner for the smallest spread among the configurations, however the flat corpus has the lowest 1.5IRQ of the configurations and a similar value with the smallest 1Q, median, and 3Q values.

For the ICL corpus, the boxplots can be found in Figure 5.68, while the MRRs can be found in Table 5.134. The difference between the top configurations is .0176. The unweighted literals are the only lexicon present in the top configuration. In the second configuration, the only lexicon is the unweighted identifiers. The two most common combinations include the identifiers with the literals and the full corpus. In each configuration where identifiers and literals are together, the identifiers are given a higher weighting than the literals. Comments only appear

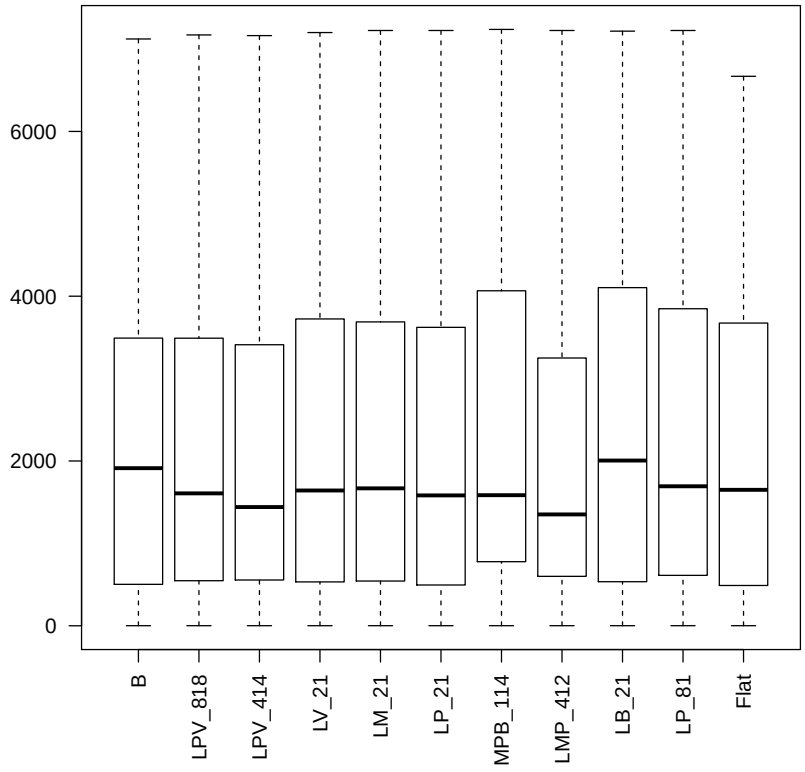


Figure 5.66: The top configurations and flat configuration for jEdit and the LMPBV corpus. The x axis is the configuration, while the y axis is the effectiveness measure.

Config	MRR
$C\{B=1\}$	0.0304
$C\{L=8,P=1,V=8\}$	0.0296
$C\{L=4,P=1,V=4\}$	0.0295
$C\{L=2,V=1\}$	0.0286
$C\{L=2,M=1\}$	0.0281
$C\{L=2,P=1\}$	0.0278
$C\{M=1,P=1,B=4\}$	0.0278
$C\{L=4,M=1,P=2\}$	0.0272
$C\{L=2,B=1\}$	0.0272
$C\{L=8,P=1\}$	0.0266

Table 5.132: Top 10 configurations from all combinations for jEdit and the LMPBV corpus

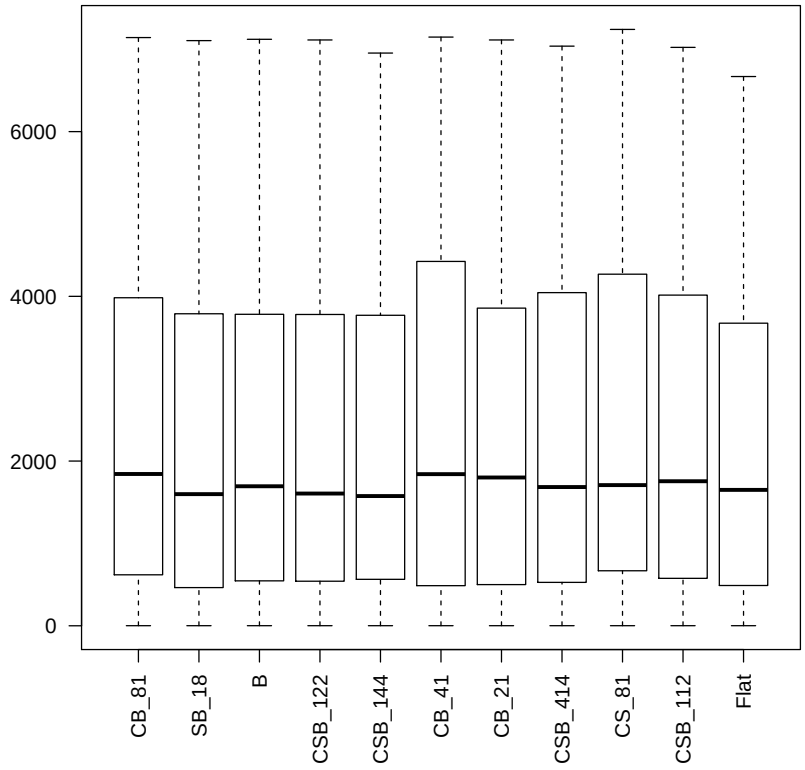


Figure 5.67: The top configurations and flat configuration for jEdit and the CSB corpus. The x axis is the configuration, while the y axis is the effectiveness measure.

Config	MRR
$C\{C=8,B=1\}$	0.0592
$C\{S=1,B=8\}$	0.0574
$C\{B=1\}$	0.0545
$C\{C=1,S=2,B=2\}$	0.0528
$C\{C=1,S=4,B=4\}$	0.0526
$C\{C=4,B=1\}$	0.0498
$C\{C=2,B=1\}$	0.0496
$C\{C=4,S=1,B=4\}$	0.0494
$C\{C=8,S=1\}$	0.0480
$C\{C=1,S=1,B=2\}$	0.0479

Table 5.133: Top 10 configurations from all combinations for jEdit and the CSB corpus

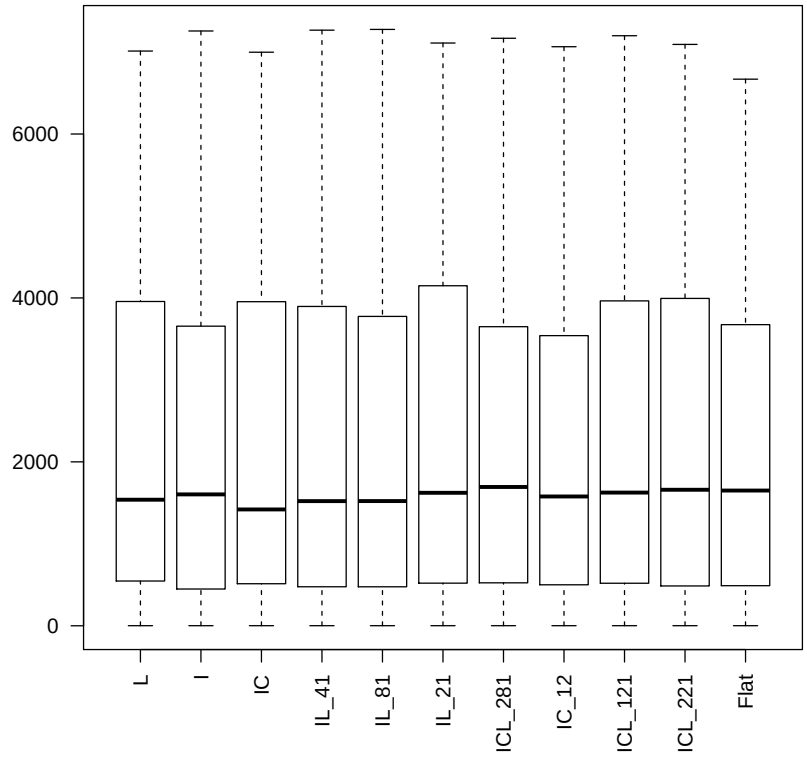


Figure 5.68: The top configurations and flat configuration for jEdit and the ICL corpus. The x axis is the configuration, while the y axis is the effectiveness measure.

Config	MRR
$C\{L=1\}$	0.0648
$C\{I=1\}$	0.0586
$C\{I=1,C=1\}$	0.0538
$C\{I=4,L=1\}$	0.0519
$C\{I=8,L=1\}$	0.0510
$C\{I=2,L=1\}$	0.0501
$C\{I=2,C=8,L=1\}$	0.0495
$C\{I=1,C=2\}$	0.0476
$C\{I=1,C=2,L=1\}$	0.0473
$C\{I=2,C=2,L=1\}$	0.0472

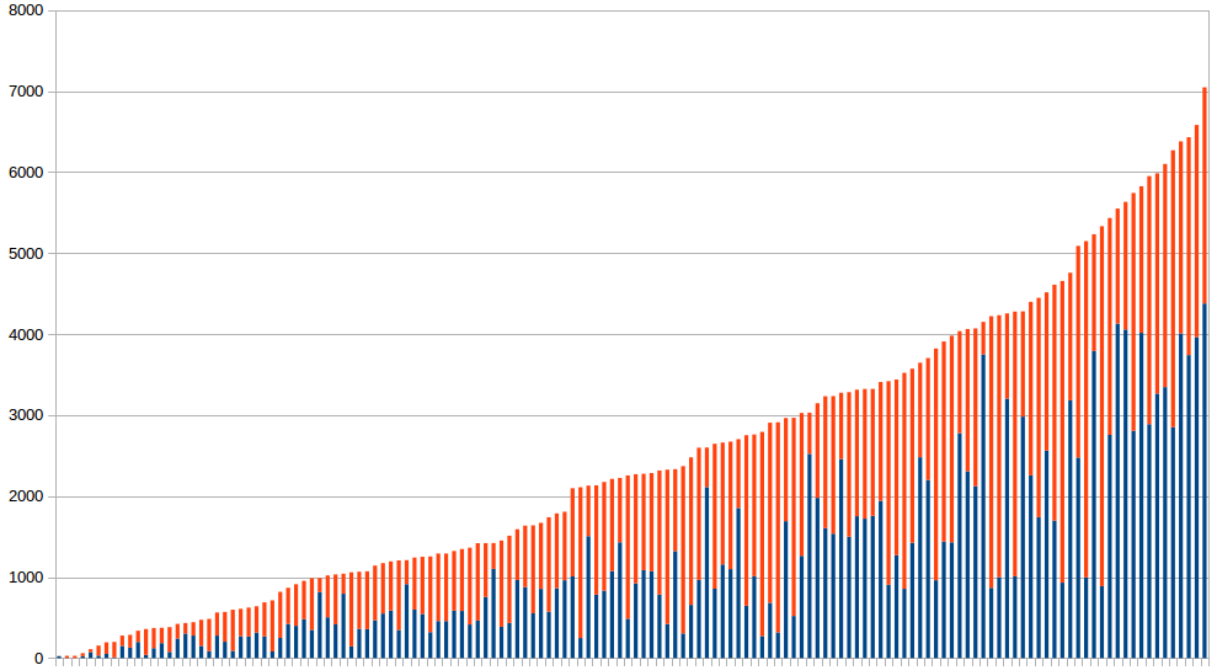
Table 5.134: Top 10 configurations from all combinations for jEdit and the ICL corpus

once in the top six configurations, while they appear in the sixth through the tenth configuration. For the top three configurations, no weighting is used. Again, there is no clear smallest spread amongst the configurations, however the flat corpus has the smallest upper 1.5IRQ and $C\{I = 1, C = 1\}$ has the smallest median.

I conducted a Friedman test with a post-hoc analysis on the top configurations and the flat corpus. I chose to only look at the top configurations and the flat corpus to lower the number of possible pairings that need to be compared. No significant differences were found for the CSB corpus. For the ICL corpus, 4 significant differences were found, with 2 of the differences between the flat corpus and the $C\{I = 2, L = 1\}$ and $C\{I = 1, C = 2, L = 1\}$ configurations. Finally for the LMPBV corpus, only 2 significant differences were found with each being between the flat corpus and the $C\{L = 2, V = 1\}$ and $C\{L = 2, B = 1\}$ configurations.

I computed the best, worst, and average cases for each feature for each corpus. The differences between each case can be found in Figure 5.69. The greatest distance between the best query and the worst query for the three corpora is found in the CSB corpus at 7,046. The greatest mean distance between the best and the worst query is found in the LMPBV corpus at a mean distance of 3,887. LMPBV also showed the greatest mean distance between the best query and the average case at 1,458. The smallest mean distance between the best and the worst case is found for the CSB corpus with a mean distance of 2,484. This corpus also had the smallest mean distance between the best and the average case with a mean distance of 1,175. The smallest distance between the best and worst query was found in the CSB corpus with a 25. The smallest distance between the best query and the average case was found for the LMPBV corpus at 5. For each of the three corpora, the mean distance between the best and the average query is smaller than the mean between the average and the worst.

(a) CSB



(b) ICL

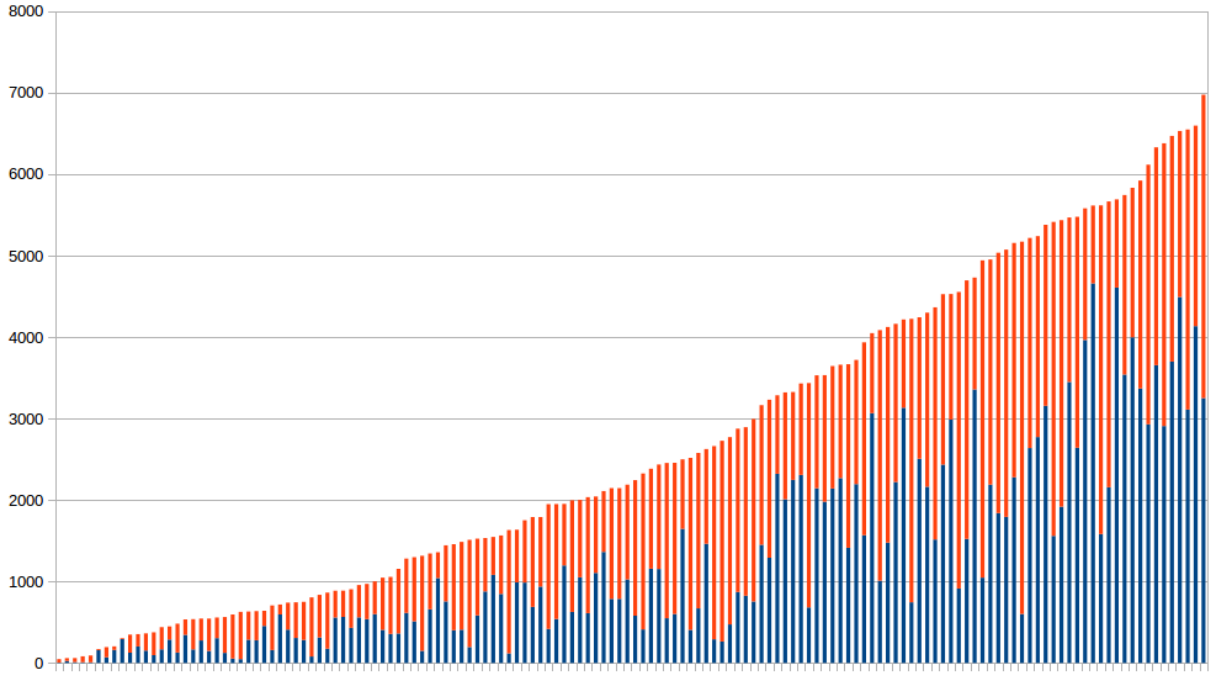


Figure 5.69: Stacked bargraphs representing the distance from the best query from all combinations to the average (bottom) and the worst(top) for jEdit. Graph is ordered by distance from best to worst.

(c) LPMBV

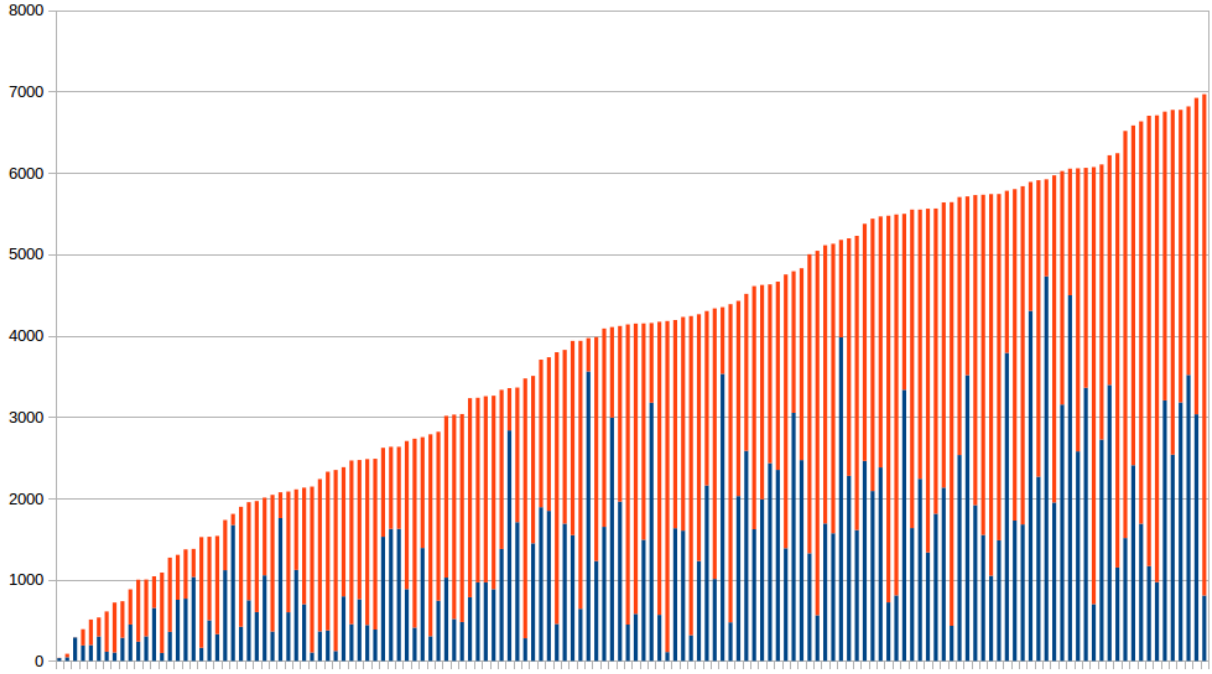


Figure 5.69: Stacked bargraphs representing the distance from the best query from all combinations to the average (bottom) and the worst(top) for jEdit. Graph is ordered by distance from best to worst.

The MRRs have been computed for each case and the results can be found in Table 5.135. Comparing the MRRs between the best cases and the results of the best configurations shows large increases from the individual corpora. The largest increase occurs for the LMPBV corpus with a difference between the best queries and the top configuration of .1454. The smallest difference occurs for the CSB corpus with a difference of .0544. Of the three corpora, the largest MRR can also be found for the LMPBV corpus while the smallest MRR can be found for the CSB corpus.

I also computed the percentage of times that the best query was a result of one of the weighting configurations. This helps to understand whether the weighted or unweighted configurations are having a bigger impact on the results for each corpus. The results of this calculation can be found in Table 5.136. For both ICL and CSB, a weighting configuration was used 100% of

Query	CSB	ICL	LMPBV
Best	0.1136	0.1412	0.1758
Average	0.0081	0.0046	0.0012
Worst	0.0011	0.0009	0.0004

Table 5.135: MRRs for choosing the best, average, and worst case for each feature for jEdit from all combinations

	CSB	ICL	LMPBV
Weighted	100	100	84

Table 5.136: The percentage of time that weighting each corpus improved the results for jEdit

B	S	SB	C	CB	CS	CSB
0	0	20	0	26	53	0

(a) CSB

L	C	CL	I	IL	IC	ICL
0	0	31	0	18	43	6

(b) ICL

V	B	BV	P	PV	PB	PBV	M	MV	MB	MBV
8	17	6	6	3	12	1	6	4	1	1
MP	MPV	MPB	MPBV	L	LV	LB	LBV	LP	LPV	
1	1	0	0	9	6	1	1	1	0	
LPB	LPBV	LM	LMV	LMB	LMBV	LMP	LMPV	LMPB	LMPBV	
0	0	3	0	0	0	0	0	0	1	

(c) LMPBV

Table 5.137: Percentage of the best queries obtained from each structural combination for jEdit

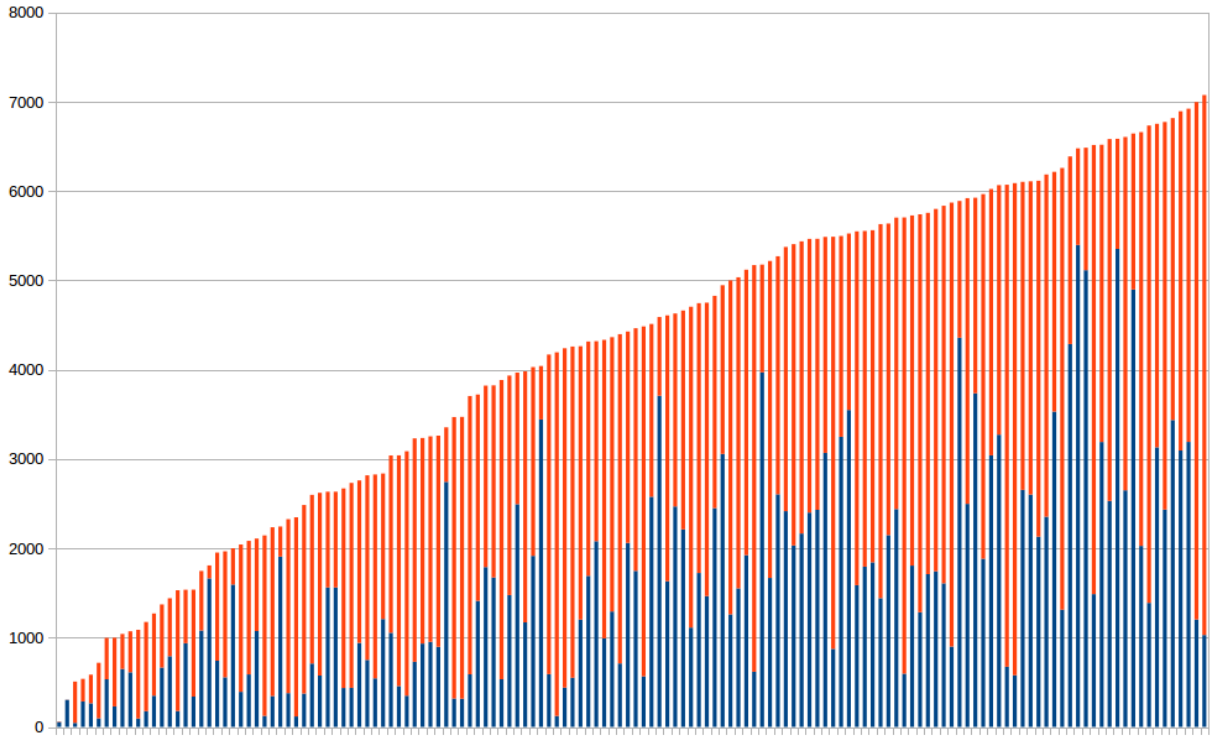


Figure 5.70: Stacked bargraphs representing the distance from the best query to the average (bottom) and the worst(top) from all combinations for jEdit. Graph is ordered by distance from best to worst.

the time, while for LMPBV a weighting configuration was used 84% of the time. This indicates that the unweighted configuration only had 16% of the best queries in the LMPBV corpus.

The percentages for the individual combinations having the best queries regardless of weighting can be found in Table 5.137. In the case of the CSB corpus, each of the two lexicon combinations contribute to the best queries with the combination of the comments and the signature (CS) contributing the highest percentage of the best queries. Neither the one lexicon combinations or the full corpus contributed to the best queries. For the ICL corpus, the full corpus and the two lexicon combinations contribute to the best queries, while none of the best queries are found in the one lexicon combinations. The highest percentage comes from

	Best	Average	Worst
MRR	0.2374	0.0012	0.0003

Table 5.138: MRRs for choosing the best, average, and worst case for each feature from all corpora and all combinations for jEdit

	CSB	ICL	LMPBV	Flat
Percentage	3	5	83	9

Table 5.139: Percentages for each corpus where the best query was found from all corpora and all combinations for jEdit

the identifiers with the comments. For the LMPBV corpus, the highest percentage comes from the body comments alone (B). The full corpus contributed 1% of the best queries in the corpus.

I performed a Friedman test with a wilcoxon post-hoc for the best, average, and worst case of each feature for each of the four corpora. The results of the analysis showed that there is a significant difference between each case.

I computed the best, worst, and average cases for each feature across all corpora using both query types. The differences between each case can be found in Figure 5.70. The greatest distance between the best query and the worst query is 7,076, while the mean distance between the best and the worst query is 4,202. The mean distance between the best query and the average case was 1,610. The smallest distance between the best and worst case was 60, while the smallest distance between the best and average cases was found to be 55. Each of these values are greater than the values for the individual corpora. The mean distance between the best and the average query is smaller than the mean between the average and the worst.

I computed the MRRs for these three cases and the results can be found in Table 5.138. The results of this computation show that there is another large increase from the individual corpora to

taking the best across all queries. To understand which corpus is contributing the highest number of best queries from the corpora, I computed the percentage for each query and recorded the results in Table 5.139. From this table it can be seen that the LMPBV corpus contributed the largest percentage of the queries for the features by far. The next closest corpus was the flat corpus with only 9% of the best queries. The combinations with the highest percentages from the three corpora include the combination of the comments and the signature from CSB with 3%, the identifiers combined with the comments (IC) with 3%, and the body comments only (B) from LMPBV with 15%. Only the full LMPBV corpus contributed to the best queries among the full corpora. A weighting configuration was used for 75% of the best queries, after adjusting for the flat corpus, the percentage of best queries that do not use a weighting configuration or the flat corpus totals 16%.

I performed a Friedman test with a wilcoxon post-hoc for the best, average, and worst case of each feature for each of the four corpora. The results of the analysis showed that there is a significant difference between each case.

5.3.4.4 muCommander

The top combinations for muCommander and the LMPBV were computed. I show boxplots of the effectiveness measures in Figure 5.71 and the MRRs in Table 5.140. To save space, I list the configurations in the boxplots with the combination and the weighting schemes separated by a '_'. There is a difference of .0021 between the top configurations. The leading comments and the method names are the two most common lexicons in the top ten configurations, with the method names and the local variables making up the top configuration. The body comments only appear in configurations combined with the leading comments and the method names, while local variables

appear with either parameters or method names. The flat corpus has the largest spread of the top configurations, while the smallest spread comes from the $C\{L = 8, M = 4, B = 1\}$ configuration.

The boxplots for the CSB corpus can be found in Figure 5.72, while the MRRs can be found in Table 5.141. There is a difference of .0042 in the top configurations. While the signature is the only lexicon present in the top configuration, the most common lexicon across all configurations is the comments. Signatures represent the second most common lexicon appearing in eight of the ten configurations, but are never weighted. The body appears in six configurations with weighting in only one case. There are three combinations with a high frequency in the top configurations, including the full corpus, the comments combined with the signature, and the comments combined with the body. Looking at the boxplots, the largest spread comes for the $C\{C = 2, S = 1\}$ configuration followed by the flat corpus. The smallest spread is seen for the $C\{C = 8, S = 1, B = 2\}$ configuration.

For the ICL corpus, the boxplots can be found in Figure 5.73, while the MRRs can be found in Table 5.142. The difference between the top configurations is .0101. The most common lexicon across the ten configurations is the identifiers, which appears in nine of the top ten configurations. However, the only lexicon that appears in the top configuration is the unweighted literals. The two most common combinations include the full corpus and the identifiers combined with the comments. While the identifiers are the most common lexicon, they are only weighted in two configurations while the comments are the most heavily weighted lexicon. The smallest spread for the ten configurations comes from the $C\{I = 4, C = 8, L = 1\}$ configurations, while the largest are from the $C\{L = 1\}$ and the $C\{I = 1, L = 2\}$ configurations.

I conducted a Friedman test with a post-hoc analysis on the top configurations and the flat

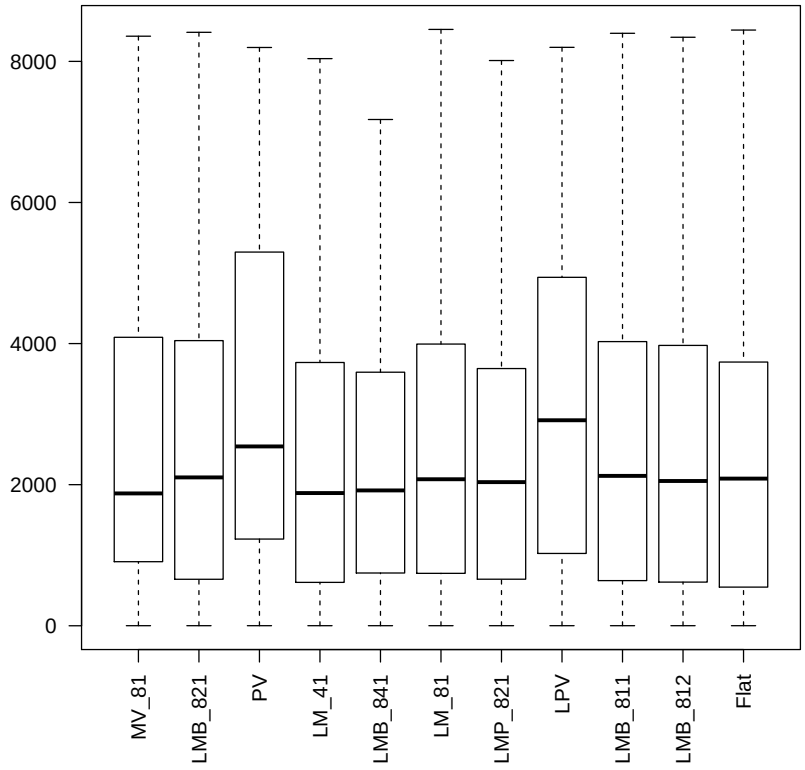


Figure 5.71: The top configurations and flat configuration for muCommander and the LMPBV corpus. The x axis is the configuration, while the y axis is the effectiveness measure.

Config	MRR
$C\{M=8,V=1\}$	0.0260
$C\{L=8,M=2,B=1\}$	0.0257
$C\{P=1,V=1\}$	0.0256
$C\{L=4,M=1\}$	0.0255
$C\{L=8,M=4,B=1\}$	0.0246
$C\{L=8,M=1\}$	0.0246
$C\{L=8,M=2,P=1\}$	0.0243
$C\{L=1,P=1,V=1\}$	0.0241
$C\{L=8,M=1,B=1\}$	0.0239
$C\{L=8,M=1,B=2\}$	0.0239

Table 5.140: Top 10 configurations from all combinations for muCommander and the LMPBV corpus

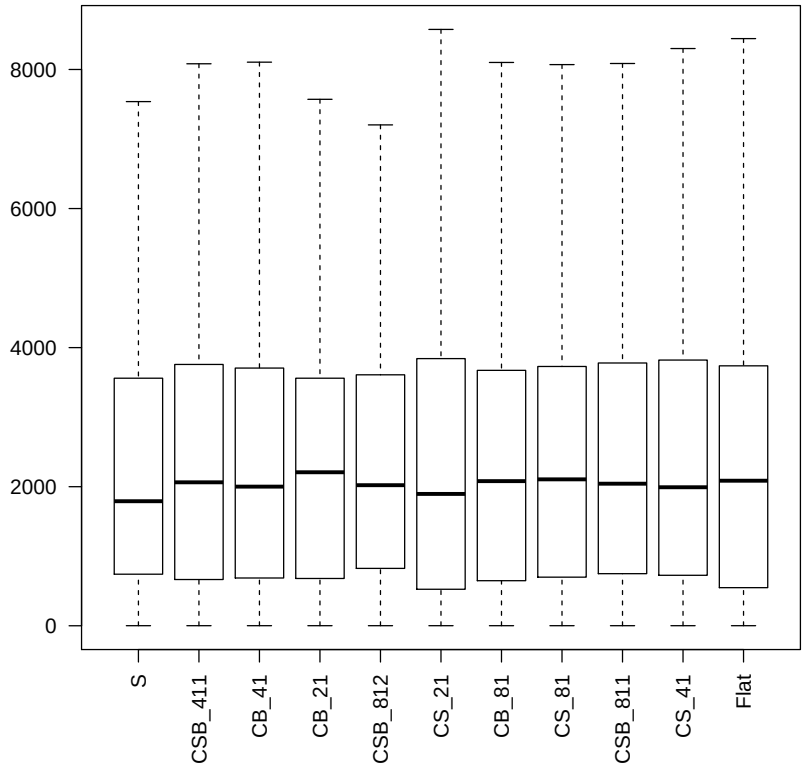


Figure 5.72: The top configurations and flat configuration for muCommander and the CSB corpus. The x axis is the configuration, while the y axis is the effectiveness measure.

Config	MRR
$C\{S=1\}$	0.0284
$C\{C=4,S=1,B=1\}$	0.0277
$C\{C=4,B=1\}$	0.0265
$C\{C=2,B=1\}$	0.0254
$C\{C=8,S=1,B=2\}$	0.0252
$C\{C=2,S=1\}$	0.0252
$C\{C=8,B=1\}$	0.0246
$C\{C=8,S=1\}$	0.0245
$C\{C=8,S=1,B=1\}$	0.0244
$C\{C=4,S=1\}$	0.0242

Table 5.141: Top 10 configurations from all combinations for muCommander and the CSB corpus

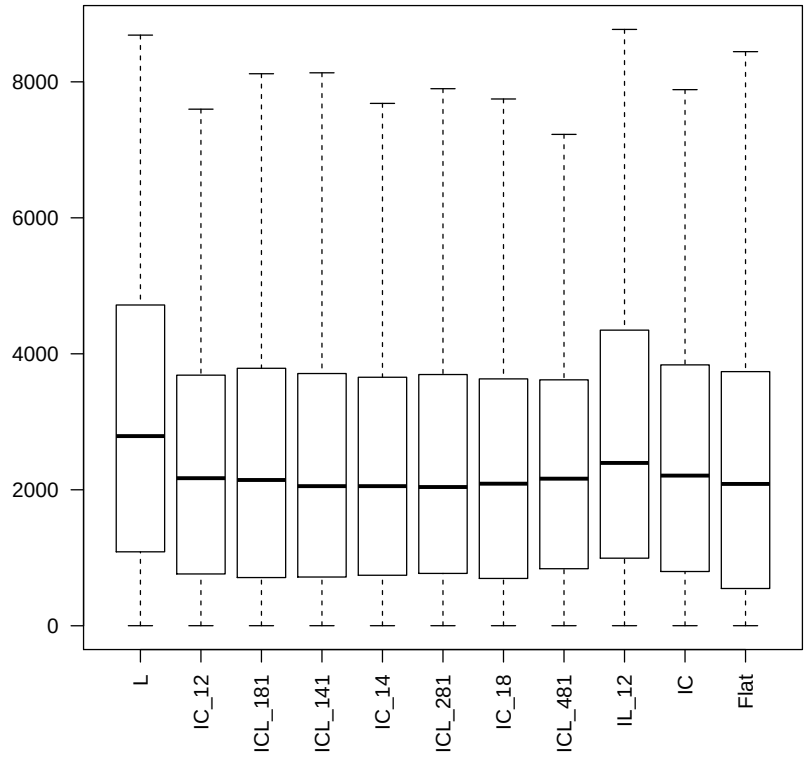


Figure 5.73: The top configurations and flat configuration for muCommander and the ICL corpus. The x axis is the configuration, while the y axis is the effectiveness measure.

Config	MRR
$C\{L=1\}$	0.0307
$C\{I=1,C=2\}$	0.0259
$C\{I=1,C=8,L=1\}$	0.0246
$C\{I=1,C=4,L=1\}$	0.0245
$C\{I=1,C=4\}$	0.0232
$C\{I=2,C=8,L=1\}$	0.0232
$C\{I=1,C=8\}$	0.0225
$C\{I=4,C=8,L=1\}$	0.0221
$C\{I=1,L=2\}$	0.0214
$C\{I=1,C=1\}$	0.0206

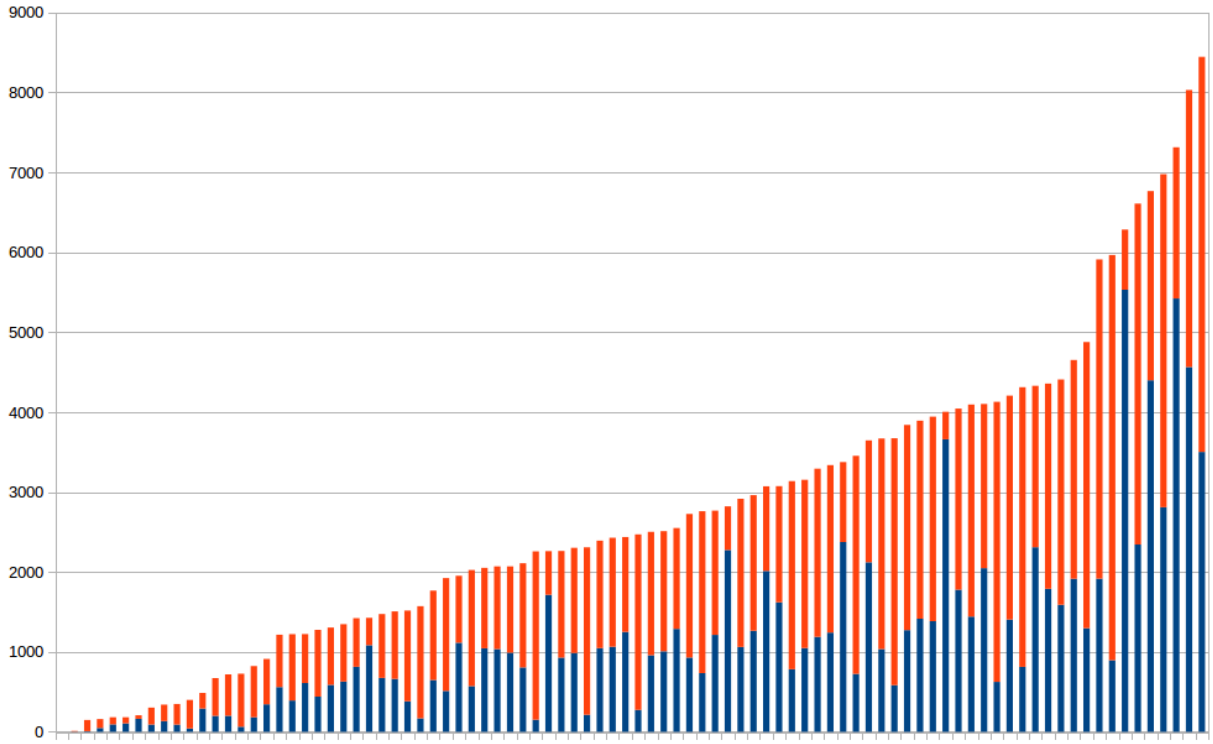
Table 5.142: Top 10 configurations from all combinations for muCommander and the ICL corpus

corpus. I chose to only look at the top configurations and the flat corpus to lower the number of possible pairings that need to be compared. No significant differences were identified for the CSB corpus. The ICL corpus had 9 significant differences identified with two of the differences being between the flat corpus and the $C\{L = 1\}$ and $C\{I = 1, L = 2\}$ configurations. For the LMPBV corpus, 22 significant differences were identified with two being between the flat corpus and the $C\{P = 1, V = 1\}$ and $C\{L = 1, P = 1, V = 1\}$ configurations.

I computed the best, worst, and average cases for each feature for each corpus. The differences between each case can be found in Figure 5.74. The greatest distance between the best query and the worst query for the three corpora is found in the CSB corpus at 8,446. The greatest mean distance between the best and the worst query is found in the LMPBV corpus at a mean distance of 4,469. LMPBV also showed the greatest mean distance between the best query and the average case at 1,789. The smallest mean distance between the best and the worst case is found for the CSB corpus with a mean distance of 2,745. This corpus also had the smallest mean distance between the best and the average case with a mean distance of 1,166. For each of the three corpora, the smallest distances between the best and the worst queries and the best query and the average case was 0. For each of the three corpora, the mean distance between the best and the average query is smaller than the mean between the average and the worst.

The MRRs have been computed for each case and the results can be found in Table 5.143. Comparing the MRRs between the best cases and the results of the best configurations shows large increases from the individual corpora. The largest increase occurs for the LMPBV corpus with a difference between the best queries and the top configuration of .1047. The smallest difference occurs for the CSB corpus with a difference of .0366. Of the three corpora, the largest MRR can also be found for the LMPBV corpus while the smallest MRR can be found for the CSB corpus.

(a) CSB



(b) ICL

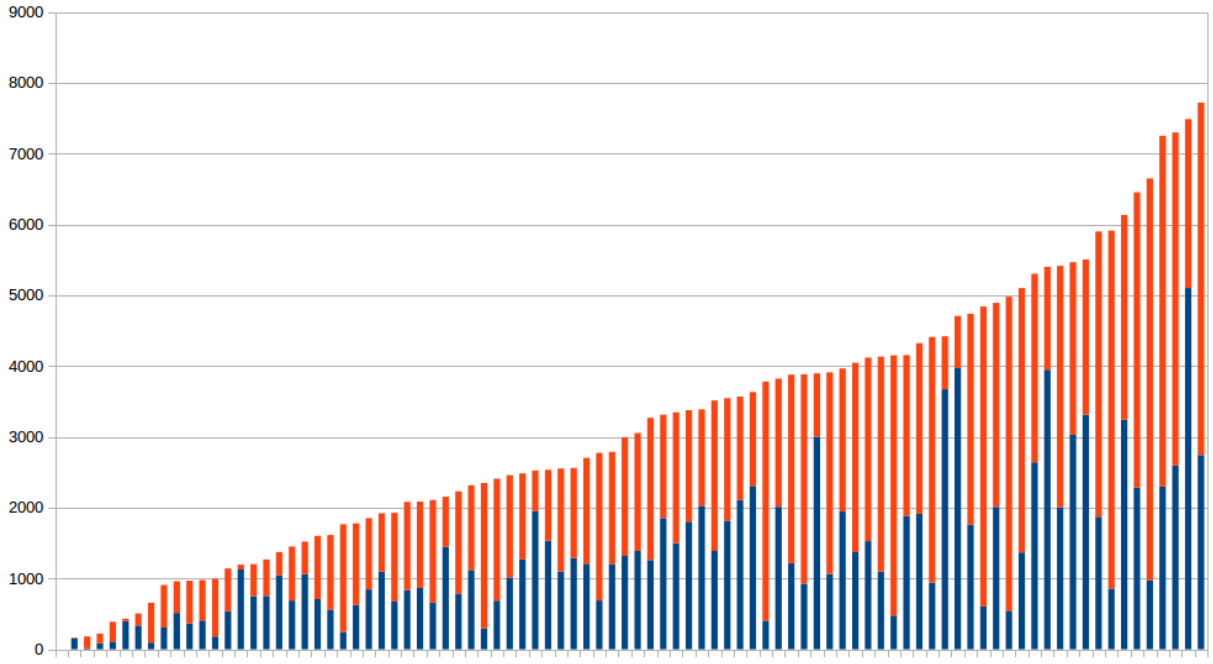


Figure 5.74: Stacked bargraphs representing the distance from the best query from all combinations to the average (bottom) and the worst(top) for muCommander. Graph is ordered by distance from best to worst.

(c) LPMBV

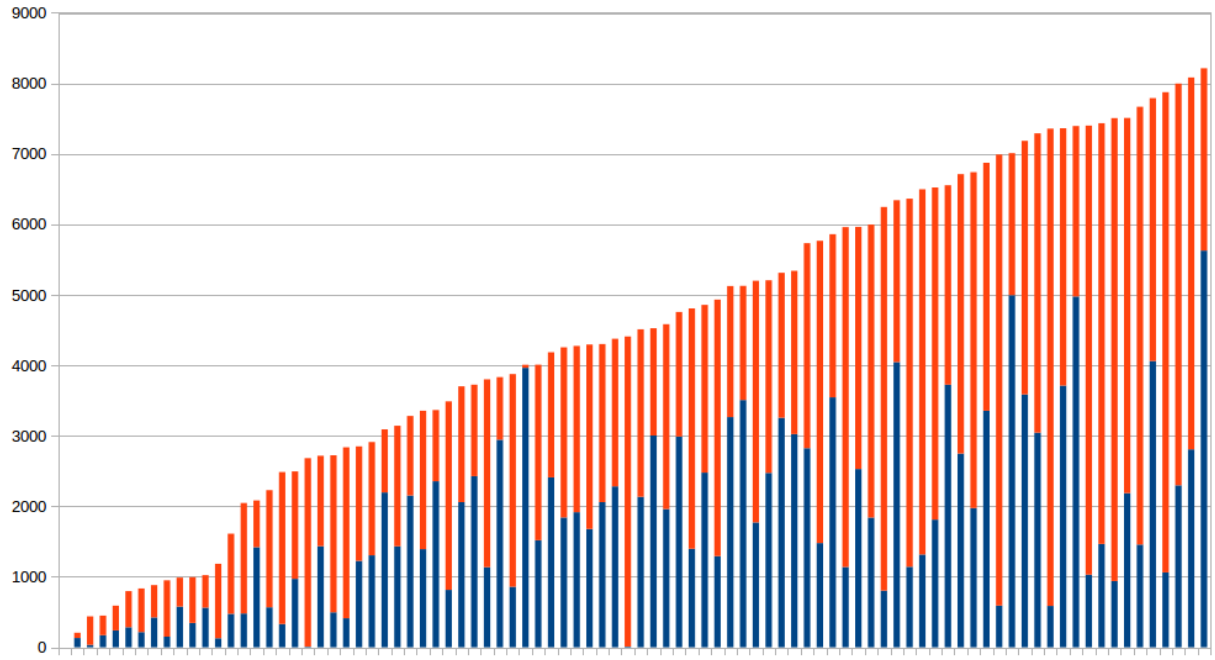


Figure 5.74: Stacked bargraphs representing the distance from the best query from all combinations to the average (bottom) and the worst(top) for muCommander. Graph is ordered by distance from best to worst.

I also computed the percentage of times that the best query was a result of one of the weighting configurations. This helps to understand whether the weighted or unweighted configurations are having a bigger impact on the results for each corpus. The results of this calculation can be found in Table 5.144. For both ICL and CSB, a weighting configuration was used 100% of the time, while for LMPBV a weighting configuration was used 91% of the time. This indicates that the unweighted configuration only had 9% of the best queries in the LMPBV corpus.

The percentages for the individual combinations having the best queries regardless of weighting can be found in Table 5.145. In the case of the CSB corpus, the highest percentage is found for the combination of the comments with the signature, while there is a tie between the comments combined with the body and the full corpus. For the ICL corpus, there is a tie for the highest percentage between the comments combined with the literals and the identifiers combined

Query	CSB	ICL	LMPBV
Best	0.0650	0.0690	0.1307
Average	0.0096	0.0072	0.0078
Worst	0.0070	0.0060	0.0059

Table 5.143: MRRs for choosing the best, average, and worst case for each feature for muCommander from all combinations

	CSB	ICL	LMPBV
Weighted	100	100	91

Table 5.144: The percentage of time that weighting each corpus improved the results for muCommander

B	S	SB	C	CB	CS	CSB
0	0	0	0	30	40	30

(a) CSB

L	C	CL	I	IL	IC	ICL
0	0	42	0	14	42	0

(b) ICL

V	B	BV	P	PV	PB	PBV	M	MV	MB	MBV
0	11	2	9	7	7	0	4	0	9	0
MP	MPV	MPB	MPBV	L	LV	LB	LBV	LP	LPV	
2	0	0	2	14	2	11	2	0	0	
LPB	LPBV	LM	LMV	LMB	LMBV	LMP	LMPV	LMPB	LMPBV	
0	2	0	2	2	0	0	2	2	0	

(c) LMPBV

Table 5.145: Percentage of the best queries obtained from each structural combination for muCommander

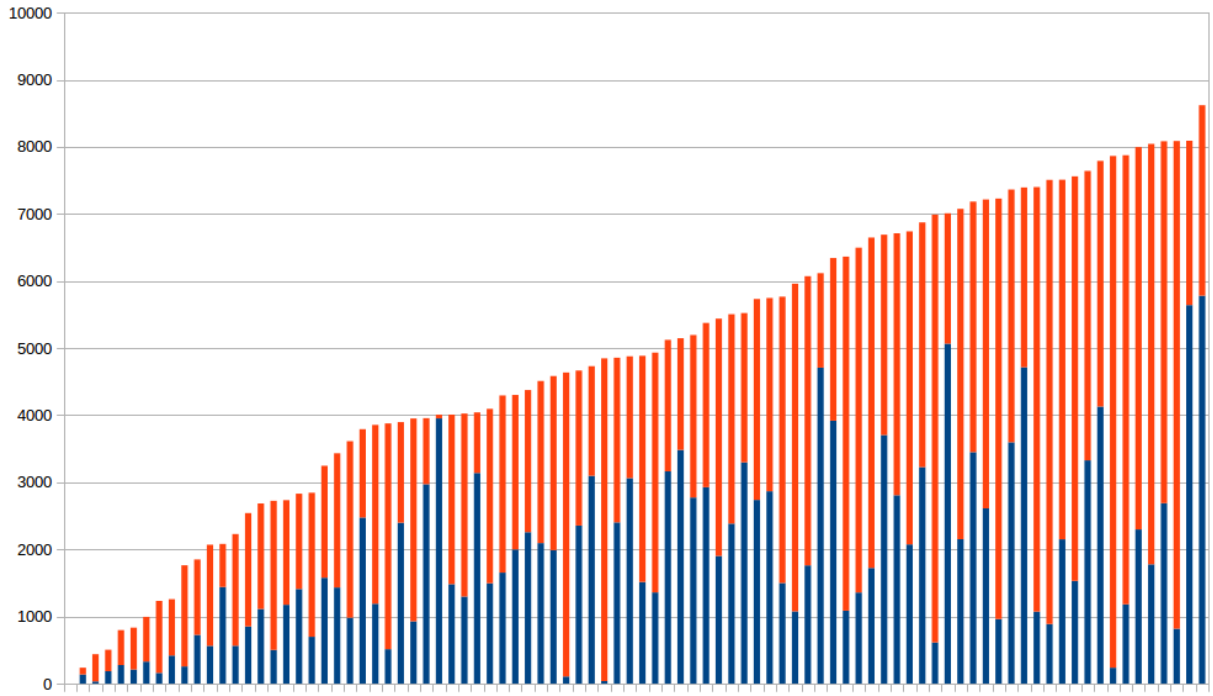


Figure 5.75: Stacked bargraphs representing the distance from the best query to the average (bottom) and the worst(top) from all combinations for muCommander. Graph is ordered by distance from best to worst.

with the comments. Only the two lexicon combinations contribute to the best queries for ICL. For LMPBV, the highest percentage comes for the leading comments followed by the body comments and the leading comments combined with the body comments. The full corpus did not contribute to the best queries.

I performed a Friedman test with a wilcoxon post-hoc for the best, average, and worst case of each feature for each of the four corpora. The results of the analysis showed that there is a significant difference between each case.

I computed the best, worst, and average cases for each feature across all corpora using both query types. The differences between each case can be found in Figure 5.75. The greatest distance between the best query and the worst query is 8,619, while the mean distance between

	Best	Average	Worst
MRR	0.1663	0.0070	0.0059

Table 5.146: MRRs for choosing the best, average, and worst case for each feature from all corpora and all combinations for muCommander

	CSB	ICL	LMPBV	Flat
Percentage	2	4	83	11

Table 5.147: Percentages for each corpus where the best query was found from all corpora and all combinations for muCommander

the best and the worst query is 4,889. The mean distance between the best query and the average case was 1,911. The smallest distances between the best and worst case and the best and the average case were both 0. Each of these values are greater than the values for the individual corpora. The mean distance between the best and the average query is smaller than the mean between the average and the worst.

I computed the MRRs for these three cases and the results can be found in Table 5.146. The results of this computation show that there is another large increase from the individual corpora to taking the best across all queries. To understand which corpus is contributing the highest number of best queries from the corpora, I computed the percentage for each query and recorded the results in Table 5.147. From this table it can be seen that the LMPBV corpus contributed the largest percentage of the queries for the features by far. The next closest percentage is the flat corpus at 11%. The combinations with the highest percentages from the three corpora include the combination of the comments and the body for CSB corpus with 2%, the leading comments from LMPBV with 13%, and the combination of the comments and literals from the ICL corpus with 4%. None of the full corpora contributed to the best queries. A weighting configurations was used for 82% of

the best queries, which means that after adjusting for the flat corpus, the percentage of best queries that do not use a weighting configuration or the flat corpus totals 7%.

I performed a Friedman test with a wilcoxon post-hoc for the best, average, and worst case of each feature for each of the four corpora. The results of the analysis showed that there is a significant difference between each case.

5.3.4.5 All Systems

The top combinations for all systems and the LMPBV were computed. I show boxplots of the effectiveness measures in Figure 5.76 and the MRRs in Table 5.148. To save space, I list the configurations in the boxplots with the combination and the weighting schemes separated by a '_'. There is a difference of .0062 between the top configurations. In each of the top ten configurations, the leading comments appear and are weighted higher than other lexicons. The most common pair of lexicons is the leading comments combined with the body comments. This pair appears with the parameters, with the method names, and alone. The boxplots do not show a spread that is clearly smaller than the others, however the $C\{L = 8, B = 1\}$ configurations appears to have the largest spread.

The boxplots for the CSB corpus can be found in Figure 5.77, while the MRRs can be found in Table 5.149. There is a difference of .0079 in the top configurations. The top four configurations are the comments combined with the body with the weighting factor for the comments being greater than or equal to the weighting factor for the body. The next most common combination includes the full corpus with the signature given a weighting factor of 1 and the comments having a weighting factor greater than or equal to the body. For the full corpus, both the comments and the body are weighted in each case. There are several top configurations with similarly small spreads, however the largest spread appears for the $C\{C = 8, B = 1\}$ configuration.

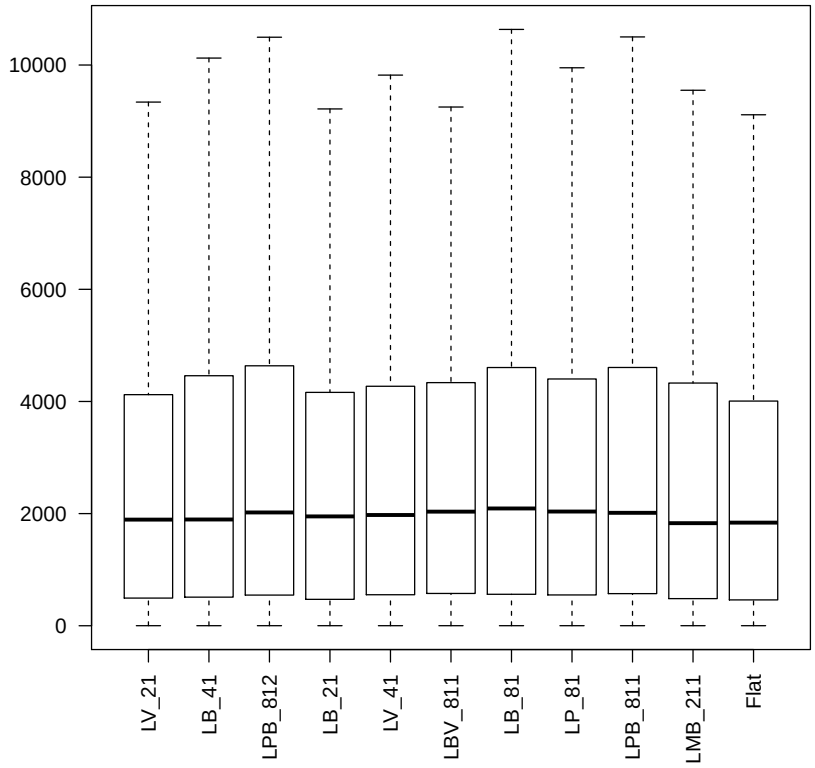


Figure 5.76: The top configurations and flat configuration for all systems and the LMPBV corpus. The x axis is the configuration, while the y axis is the effectiveness measure.

Config	MRR
$C\{L=2,V=1\}$	0.0430
$C\{L=4,B=1\}$	0.0430
$C\{L=8,P=1,B=2\}$	0.0402
$C\{L=2,B=1\}$	0.0397
$C\{L=4,V=1\}$	0.0390
$C\{L=8,B=1,V=1\}$	0.0378
$C\{L=8,B=1\}$	0.0377
$C\{L=8,P=1\}$	0.0377
$C\{L=8,P=1,B=1\}$	0.0376
$C\{L=2,M=1,B=1\}$	0.0368

Table 5.148: Top 10 configurations from all combinations for all systems and the LMPBV corpus

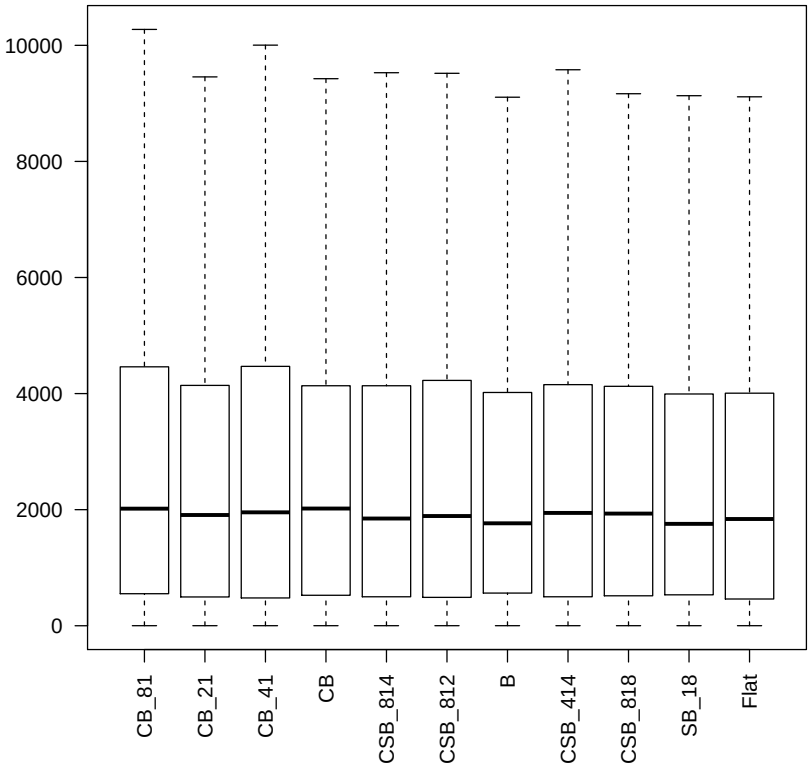


Figure 5.77: The top configurations and flat configuration for all systems and the CSB corpus. The x axis is the configuration, while the y axis is the effectiveness measure.

Config	MRR
$C\{C=8,B=1\}$	0.0557
$C\{C=2,B=1\}$	0.0539
$C\{C=4,B=1\}$	0.0513
$C\{C=1,B=1\}$	0.0497
$C\{C=8,S=1,B=4\}$	0.0487
$C\{C=8,S=1,B=2\}$	0.0486
$C\{B=1\}$	0.0482
$C\{C=4,S=1,B=4\}$	0.0482
$C\{C=8,S=1,B=8\}$	0.0480
$C\{S=1,B=8\}$	0.0478

Table 5.149: Top 10 configurations from all combinations for all systems and the CSB corpus

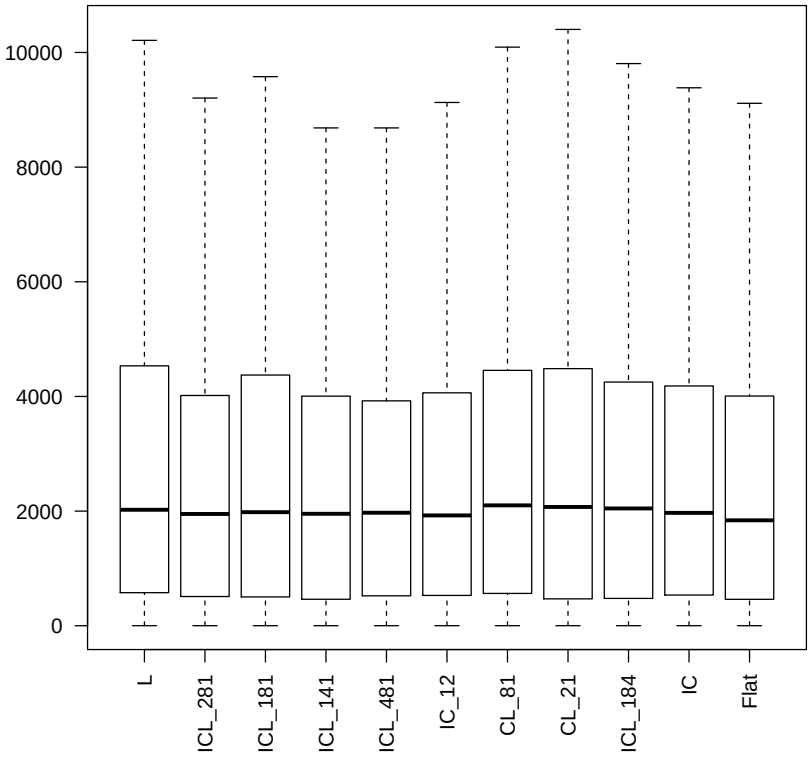


Figure 5.78: The top configurations and flat configuration for all systems and the ICL corpus. The x axis is the configuration, while the y axis is the effectiveness measure.

Config	MRR
$C\{L=1\}$	0.0552
$C\{I=2,C=8,L=1\}$	0.0459
$C\{I=1,C=8,L=1\}$	0.0457
$C\{I=1,C=4,L=1\}$	0.0449
$C\{I=4,C=8,L=1\}$	0.0444
$C\{I=1,C=2\}$	0.0443
$C\{C=8,L=1\}$	0.0437
$C\{C=2,L=1\}$	0.0435
$C\{I=1,C=8,L=4\}$	0.0423
$C\{I=1,C=1\}$	0.0421

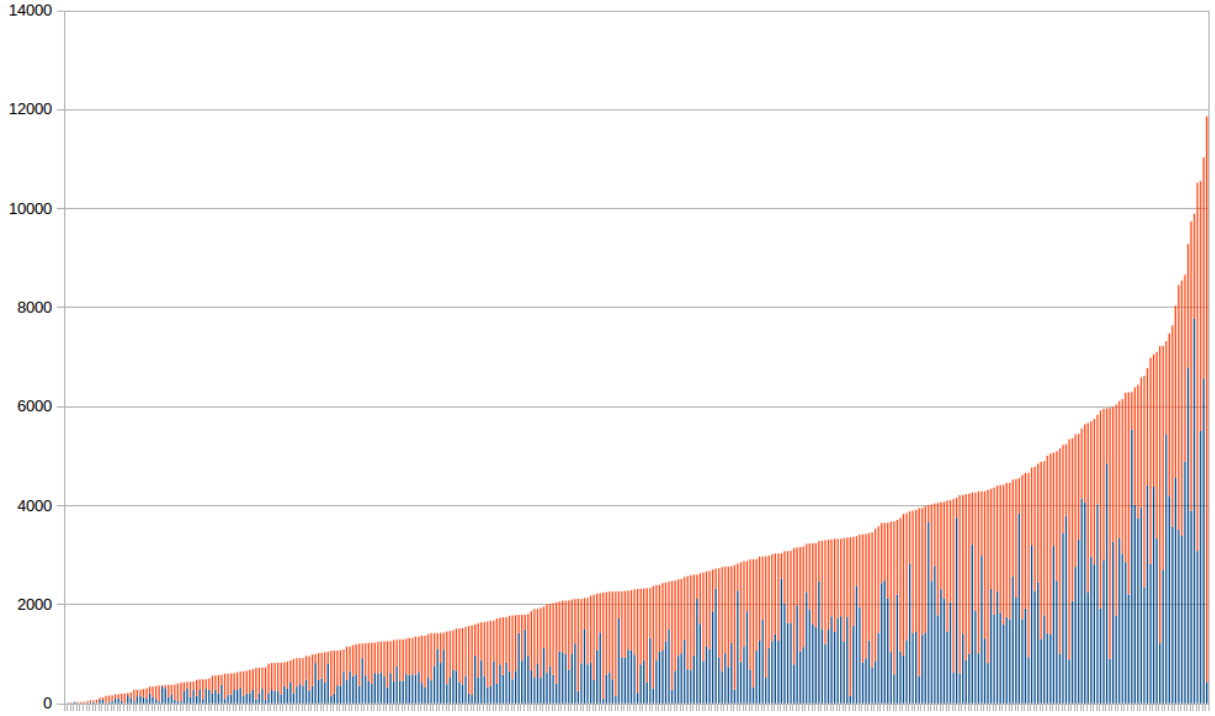
Table 5.150: Top 10 configurations from all combinations for all systems and the ICL corpus

For the ICL corpus, the boxplots can be found in Figure 5.78, while the MRRs can be found in Table 5.150. The difference between the top configurations is .0131. The top configuration is for the unweighted literals alone. However the full corpus appears in the next four positions. For the full corpus, the comments have the greatest weight with a weighting to the identifiers being preferred over a weighting of the literals. The smallest spreads are for the $C\{I = 1, C = 4, L = 1\}$ and $C\{I = 4, C = 8, L = 1\}$ which have very similar spreads.

I conducted a Friedman test with a post-hoc analysis on the top configurations and the flat corpus. I chose to only look at the top configurations and the flat corpus to lower the number of possible pairings that need to be compared. Significant differences were identfied for each of the three corpora. For the CSB corpus, 6 significant differences with 2 of the differences being between the flat corpus and the $C\{C = 8, B = 1\}$ and $C\{C = 4, B = 1\}$ configurations. The ICL corpus had 19 significant differences identified with several significant differences identified between the flat corpus and the other configurations. For the LMPBV corpus, 26 significant differences were identified with several significant differences identified between the flat corpus and the other configurations.

I computed the best, worst, and average cases for each feature for each corpus. The differences between each case can be found in Figure 5.79. The greatest distance between the best query and the worst query for the three corpora is found in the LMPBV corpus at 11,973. The greatest mean distance between the best and the worst query is found in the LMPBV corpus at a mean distance of 4,435. LMPBV also showed the greatest mean distance between the best query and the average case at 1,789. The smallest mean distance between the best and the worst case is found for the CSB corpus with a mean distance of 2,761. This corpus also had the smallest mean distance between the best and the average case with a mean distance of 1,235. For each of the three

(a) CSB



(b) ICL

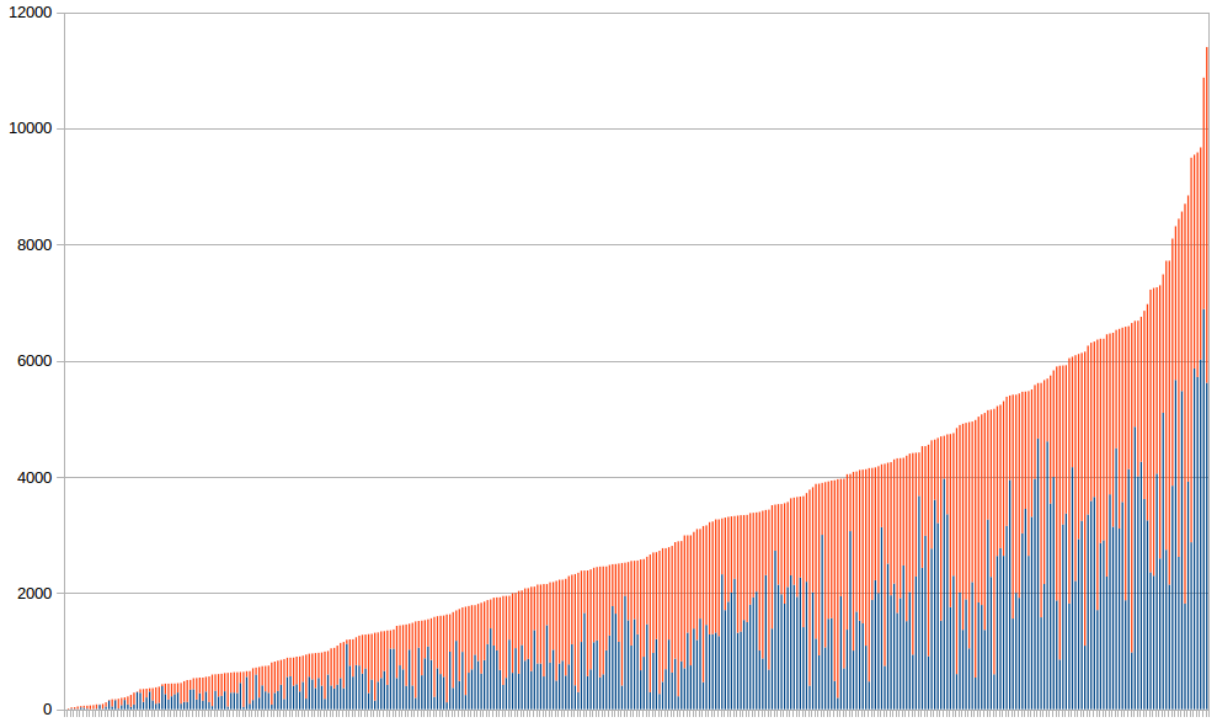


Figure 5.79: Stacked bargraphs representing the distance from the best query from all combinations to the average (bottom) and the worst(top) for all systems. Graph is ordered by distance from best to worst.

(c) LPMBV

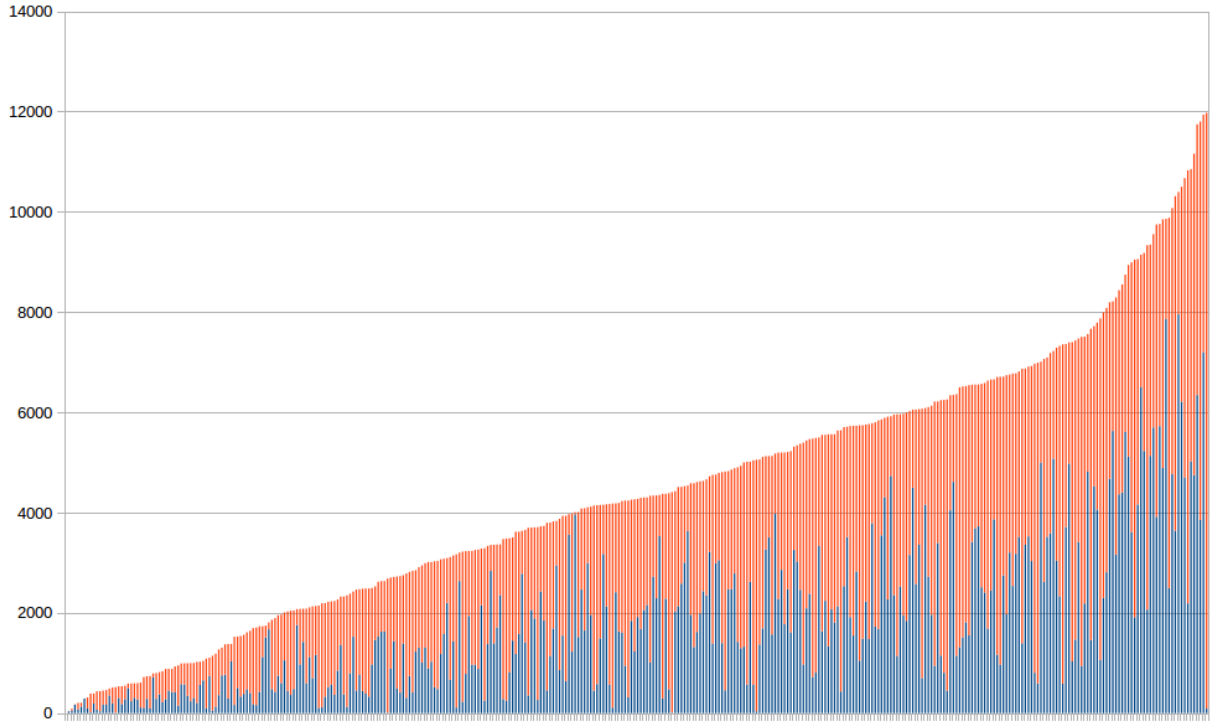


Figure 5.79: Stacked bargraphs representing the distance from the best query from all combinations to the average (bottom) and the worst(top) for all systems. Graph is ordered by distance from best to worst.

corpora, the smallest distances between the best and the worst queries and the best query and the average case was 0. For each of the three corpora, the mean distance between the best and the average query is smaller than the mean between the average and the worst.

The MRRs have been computed for each case and the results can be found in Table 5.151. Comparing the MRRs between the best cases and the results of the best configurations shows large increases from the individual corpora. The largest increase occurs for the LMPBV corpus with a difference between the best queries and the top configuration of .1324. The smallest difference occurs for the CSB corpus with a difference of .0561. Of the three corpora, the largest MRR can also be found for the LMPBV corpus while the smallest MRR can be found for the CSB corpus.

Query	CSB	ICL	LMPBV
Best	0.1118	0.1247	0.1754
Average	0.0086	0.0090	0.0035
Worst	0.0027	0.0024	0.0018

Table 5.151: MRRs for choosing the best, average, and worst case for each feature for all systems from all combinations

	CSB	ICL	LMPBV
Weighted	100	100	88

Table 5.152: The percentage of time that weighting each corpus improved the results for all systems

B	S	SB	C	CB	CS	CSB
0	0	20	0	25	35	20

(a) CSB

L	C	CL	I	IL	IC	ICL
0	0	29	0	22	27	20

(b) ICL

V	B	BV	P	PV	PB	PBV	M	MV	MB	MBV
3	12	6	9	6	8	1	6	4	4	0
MP	MPV	MPB	MPBV	L	LV	LB	LBV	LP	LPV	
1	1	0	0	9	5	6	1	1	0	
LPB	LPBV	LM	LMV	LMB	LMBV	LMP	LMPV	LMPB	LMPBV	
0	0	1	0	1	0	0	0	0	0	

(c) LMPBV

Table 5.153: Percentage of the best queries obtained from each structural combination for all systems

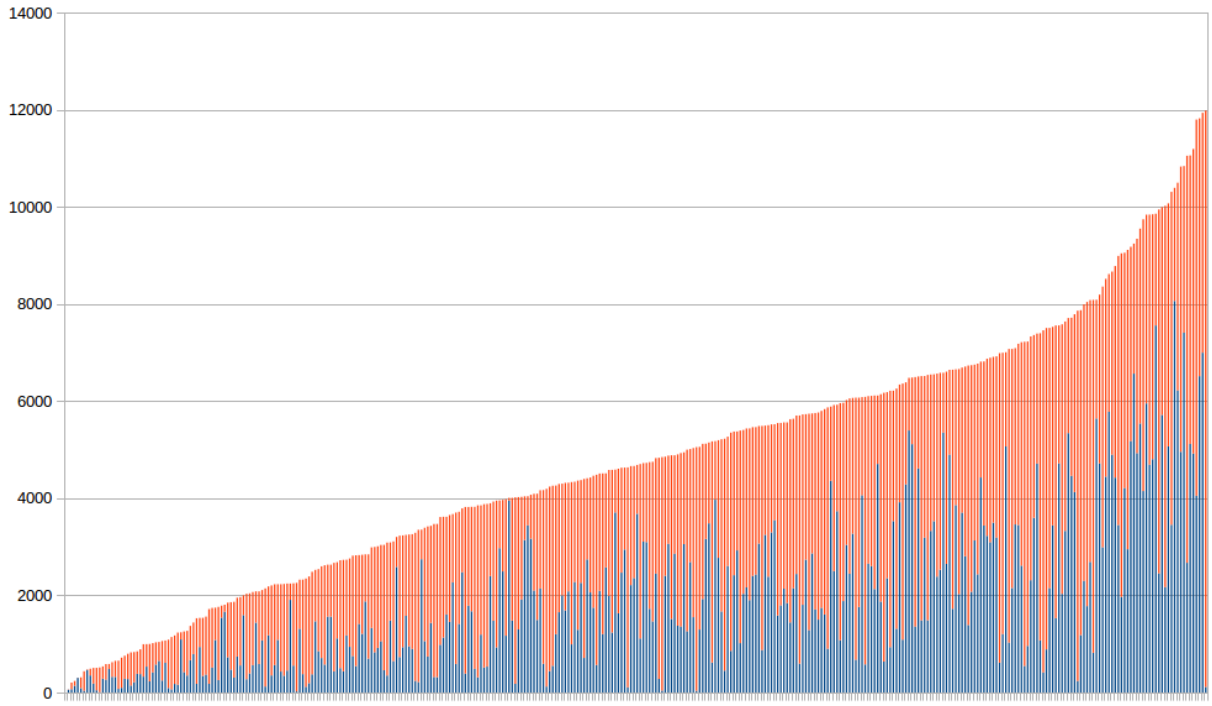


Figure 5.80: Stacked bargraphs representing the distance from the best query to the average (bottom) and the worst(top) from all combinations for all systems. Graph is ordered by distance from best to worst.

I also computed the percentage of times that the best query was a result of one of the weighting configurations. This helps to understand whether the weighted or unweighted configurations are having a bigger impact on the results for each corpus. The results of this calculation can be found in Table 5.152. For both ICL and CSB, a weighting configuration was used 100% of the time, while for LMPBV a weighting configuration was used 88% of the time. This indicates that the unweighted configuration only had 12% of the best queries in the LMPBV corpus.

The percentages for the individual combinations having the best queries regardless of weighting can be found in Table 5.153. In the case of the CSB corpus, the highest percentage is found for the combination of the comments with the signature. Each of the two lexicon combinations and the full corpus contributed to the best queries for the corpus. For the ICL corpus,

	Best	Average	Worst
MRR	0.2196	0.0028	0.0003

Table 5.154: MRRs for choosing the best, average, and worst case for each feature from all corpora and all combinations for all systems

	CSB	ICL	LMPBV	Flat
Percentage	3	5	83	9

Table 5.155: Percentages for each corpus where the best query was found from all corpora and all combinations for all systems

the highest percentage is found for the comments combined with the literals. Again, all two lexicon combinations and the full corpus contributed to the best queries. For LMPBV, the highest percentage comes from the body comments followed by the leading comments comments and the parameters. The full corpus contributed less than 1% of the best queries.

I performed a Friedman test with a wilcoxon post-hoc for the best, average, and worst case of each feature for each of the four corpora. The results of the analysis showed that there is a significant difference between each case.

I computed the best, worst, and average cases for each feature across all corpora using both query types. The differences between each case can be found in Figure 5.80. The greatest distance between the best query and the worst query is 11,986, while the mean distance between the best and the worst query is 4,788. The mean distance between the best query and the average case was 1,923. The smallest distances between the best and worst case and the best and the average case were both 0. Each of these values are greater than the values for the individual corpora with the

exception of the smallest distances. The mean distance between the best and the average query is smaller than the mean between the average and the worst.

I computed the MRRs for these three cases and the results can be found in Table 5.154. The results of this computation show that there is another large increase from the individual corpora to taking the best across all queries. To understand which corpus is contributing the highest number of best queries from the corpora, I computed the percentage for each query and recorded the results in Table 5.155. From this table it can be seen that the LMPBV corpus contributed the largest percentage of the queries for the features by far. The next closest percentage is the flat corpus at 9%. The combinations with the highest percentages from the three corpora include the combination of the comments and the signature for CSB corpus with 1%, the body comments from LMPBV with 10%, and the combination of the comments and literals from the ICL corpus with 2%. A weighting configuration was used for 79% of the best queries, which means that after adjusting for the flat corpus, the percentage of best queries that do not use a weighting configuration or the flat corpus totals 12%.

I performed a Friedman test with a wilcoxon post-hoc for the best, average, and worst case of each feature for each of the four corpora. The results of the analysis showed that there is a significant difference between each case.

5.4 Discussion of Results

In this section, I discuss the case study and try to give additional information to explain the results.

5.4.1 Does query type affect the accuracy of a structured retrieval-based FLT?

To understand how the query type changes the results of the structured retrieval-based FLT, it is important to consider how the language models are queried. Language models use query-

likelihood prediction, which in the basic language model involves computing the probability that terms in the query generated by the language model coupled with the probability that additional terms were not. Knowing this, there are three main ways that the types of queries can affect the results of the FLT.

First, by changing the probability that the terms in the query will be generated by the language model. As an example, the title of a feature request may only be made up of terms that have a high probability of coming from the language model. This leads to a high probability for a document with that language model to be related to the query. However, when the combination of the title and description is used, this probability can be lowered. The terms from the title are still present, but terms introduced by the description can have a low probability of being generated by the language model. In this case, the title would be preferred over the combined or description query types.

Second, by introducing terms that have a higher probability of being generated by the language model. Once again, if we were to consider the title query with terms that have a high probability of being generated by the language model, there is an additional lexicon that we must consider. If the title query does not contain many terms, then there is a chance that a large number of terms that the language model would have generated are not present in the query. In this case, when we add the terms from the description, we can include additional terms from this category, increasing the rating for the combined query type. This would also hold for the description if the terms in the query were a superset of the terms found in the title.

Finally, it is possible that the likelihood of the query does not change between the different queries. In this situation, it is not the likelihood between the query and the relevant result that is important, but the likelihood of the queries and the alternative results. If changing the query

changes the likelihood that the query will be generated by a false positive, then the rank of the relevant result can change. This is an indirect way of increasing or decreasing the performance of the language modeling technique.

I will now look at the systems individually, and explain how the results changed with the different query types: Title, Description, and Combined. In a completed retrieval system, it is highly likely that these query types will not be used, but the developers will write the queries themselves. The importance of this question is in understanding how different forms of information will affect the results of the technique.

In addition to looking at different query types, for each of the research questions I looked at different types of corpora. The reason for using different corpora is due to the complexity and performance of different types of language models. For instance, when looking at the CSB corpus, the comments are together in one lexicon, the comment types (leading and body) work together to form a single language model that is composed of terms from both sources. To query this lexicon requires a single query on one language model. When breaking up the comments into leading comments and body comments, two different language models are created. In order to obtain results, both comment types must now be queried which increases the time required to get results, and since there are now two language models created from a subset of the terms, the performance can be weaker. The tradeoff comes in the form of flexibility by allowing a wider range of possible queries by the developer.

5.4.1.1 ArgoUML

In this question, I looked at four different types of corpora that represented different ways that a method could be structured. I began by looking at two different types of results and using them to compare and contrast the performance of each query type on the overall results of the

structured retrieval technique. I looked at both the spread of the effectiveness measures and the MRR for the results. The reason for looking at both is due to the fact that when computing the MRR, higher ranks get more emphasis than lower ranks. Therefore, this measure alone does not give a complete picture of how the technique is performing.

For each of the four corpora that I looked at, the Title query type resulted in a higher MRR measures than any of the other query types. The Title query type was also found to be significantly different when compared to the description in three of the four query types. Leading to the conclusion that the Title query type performed better than the Description query type.

However, the purpose of this query system is to allow developers to write the best queries for each feature. Therefore, it is not sufficient to only consider the query types individually. I then looked at what would happen if I took the best query from each query type. If there was a query type that was clearly better than the others, then there would not be a significant difference between the performance of the Title query type (e.g., the top performing query type) and the combination of the query types. Given this freedom, I charted what would happen if the developer selected the best query, the worst query, or a query that was close to average for the feature. This was plotted to show that in several queries, there is a significant difference based on the choice of query for the feature. However, when the developer chooses the best query, there is an increase in performance for the FLT. For each of the corpora under investigation, I found that the Title query type had a measure around 60% of the best queries overall. This gives the other two query types around 40% of the best queries possible in the other query types. Using one type of queries therefore results in a loss for the flexibility of this technique.

For the same reason that I looked at taking the best queries in each corpus from the three query types, I looked at taking the best queries from each corpus. While it increases the complexity

of the model, it is possible to design a corpus that allows for all types of queries. Using this model, the question is whether or not one corpus dominates the results over the others, minimizing the benefits of combining information. I found a significant increase in the results of the FLT, and while the flat corpus comprised 43% of the best queries, 57% of the best results came from the unweighted, full corpora.

5.4.1.2 JabRef

I repeated the same series of experiments for each of the four subject systems and then the combination of the systems. The goal was to see whether the results were generalizable to each system or were more system dependent. When looking at the performance of the individual query types, the Title query type had a much higher MRR than the others for all four corpora. However, when looking at the spreads, in two of the corpora the Title query type had a larger spread. Indicating that there were features with decreased performance due to this query type.

The results of combining the best queries did not lead to a significant increase in the MRR. This is explained due to some of the best queries that came from the other query types still having poor rankings. The higher the rankings, the lower the impact they will have on the computation of the MRR. Therefore, even though some features had their rankings impacted, the resulting change was not good enough to make a significant impact on the MRR. Looking at the percentages still showed that both the description and the combined query types contributed to the best queries in the corpus.

However, when looking at the overall results, there was a significant increase in the performance. This indicated that there was a significant difference in how features were performning in each of the three corpora. Each corpus contained a high performing queries that were not found in the other corpora. This is further supported by the percentages between the corpora where each of

the three structured corpora contributed between 10-20% of the best queries to the overall results. As with ArgoUML, JabRef showed Title queries comprising roughly 60% of the best queries in the overall results.

5.4.1.3 jEdit

jEdit contained the largest number of features used in this study as the only system with over 100 features. In line with the previous two systems, the Title query type outperformed the other two query types. The spreads helped to support this conclusion. While the overall spreads were similar, the Title query type was more bottom heavy, indicating that there were more query results with lower rankings.

When looking at the results of the best queries, like with JabRef, the differences were small. However, as previously explained, if the rankings improved, but the ranks were still low, they would have a small impact on the overall results. As can be seen from the bar graphs, there existed several features where the difference between choosing the best query and the worst query for the feature would result in substantially different rankings, and after looking at the percentages, over 30% of the best queries were results from a query type other than the Title.

The results from the overall MRRs showed that there was a significant increase when using queries from each of the different corpora, meaning that there were queries from each corpus that performed differently for that corpus than for others. The percentages were similar to the previous two systems, with the flat corpus represented 43% of the best results and the remaining three corpora, each contributing 10-20% of the best queries, and that the Title query type was in a similar range with the previous systems.

5.4.1.4 muCommander

The muCommander results were the lowest of the four subject systems. However, the results were similar to the previous subject systems. For each corpus, the Title query type outperformed the other query types. For three of the four corpora, the spread of the Title query type was smaller than the other two query types. The only exception to this was from the flat corpus. However, for this corpus, the MRR was still substantially higher for the Title query type. This indicated that in this corpus, while there was a wider range, there were several queries that had much lower ranks than for the other two query types. This was supported by the median value for the Title query type.

We see a similar situation to the previous systems when looking at the combination of the best queries. For each corpus, the Title query type represented the query type with the highest percentage of the three query types. Another interesting note, is that with the exception of ArgoUML, the Combined query type had the lowest percentage of any query type for each corpus.

Looking at the overall results, while muCommander has the lowest results of the four subject systems, the MRR still increased substatially, with an MRR that is over double the value for the flat corpus. The percentages for the four corpora were similar to the results of the other systems with each structured corpus representing 10-20% of the best queries. As with the previous systems that showed the Combined query type as having a lower percentage for each corpus, the percentage for the Combined query type and the overall results was higher than the Description query type. This indicates that the best ranks for the Description query type were lower than the best ranks for the Combined query type in another corpus, even if they were better than the Combined query type in their own corpus.

5.4.1.5 All Systems

By combining the four subject systems and looking at the aggregated results, some similarities between the systems become easier to see. Unsurprisingly, the Title query type outperformed the other query types with higher MRRs and smaller spreads for three of the four corpora. The only exception was a smaller spread for the flat corpus for the Combined query type. However, when comparing the two spreads, you can see a lower 1Q and median value for the Title query type.

Of interest is the bargraph for the LMPBV corpus. This bargraph shows that for a majority of the features, there is little to no difference between the results of the three query types. Therefore, despite teh Title query type having better performance in the percentages, these differences were small. This is further supported by the corpus MRRs where the smallest difference was seen for the LMPBV corpus. For each corpus, there was an increase in the MRR for the best queries.

Overall, using the best queries from each corpus resulted in a substantial increase to the MRRs.

5.4.1.6 Combining Results

After looking at the results of the four subject systems and their combination, there are a few results that are worth pointing out. The Title query type outperforms the other two query types. Since the Combined query type is a superset of the Title, it indicates that the Combined query type often introduces extra noise that makes it difficult to identify the correct feature. When the Combined query type outperforms the Title query, a common cause is that the Title query type does not adequately describe the desired feature. When the Description query type outperforms the Title query type, it is often for one of two reasons. Either, the Title query type does not give

specifics to indicate the desired feature, or the Description repeats the terms in the Title and acts similar to the Combined query type by helping to supplement the information in the title with additional terms.

In most cases, the best queries are those that are concise but adequately describe the problem or feature. Additional terms can help to improve the results, but more often increase the amount of noise in the query jeopardizing the performance.

5.4.2 Does changing the combination of included fields affect the accuracy of a structured retrieval-based FLT?

Under the structured retrieval approach presented in this chapter, each structural context in a corpus becomes a new language model. For the CSB corpus, comments, method signatures, and body comments each form their own language model. These language models are composed of terms that were discovered in those contexts. For the LMPBV corpus, all comments are present in this corpus, however they are looked at in two different contexts, as leading comments or as body comments. With this corpus, there is no way to combine the two contexts into one language model for querying. The question then becomes, how does this affect the results of the FLT. Does one corpus clearly outperform another?

Furthermore, there are additional considerations. First, it is important to consider how the FLT performs when only a subset of the contexts are queried. Without querying over every context, the entire method will not be considered, but instead a subset. There are two possibilities, one is that querying specific contexts will improve the results because a query best describes one aspect of the method. A possible example of this would be a query that contains terms from an error message. When searching the entire method, the terms may not appear in the identifiers or the comments, but they will appear in the literals. However, if in other methods the terms were to

appear in the identifiers or the comments, then a false positive can result from the search degrading the performance.

Another way of looking at this is that by only querying a subset, the developer only gets a small indicator of the relationship between the method and the query. Take the same example, but let us assume that two methods exist with similar error messages. The query may contain terms that are found in the identifiers of one method, but not the other. Therefore, the results would be improved if the developer searched over more contexts.

I looked at both the Combined and the Title query types. The reason for this is to see how the information within the query could affect the performance of the different combinations.

5.4.2.1 ArgoUML

The two combinations with the best results for the LMPBV corpus and the Combined query type were the leading comments and the leading comments combined with the method names. For the Title query type, the parameters alone outperformed the other combinations. This shows that the performance of a particular combination is affected by the information in the query. Instead of getting the same leading combinations but with different MRRs for each query type, adding the information from the description changed the best performing combination. In contrast, for the CSB corpus, both the Combined query type and the Title query type showed the leading performance from the comments combined with the body and the full corpus. For the ICL corpus, there was once again a difference in the best performing combinations with the Combined query type showing better results for the identifiers combined with the comments, and the Title query type showing the literals alone to have the leading performance.

There were larger increases when taking the best queries from the combinations than there were for the different query types. The ICL corpus was shown to have the leading performance

of the three corpora, while the LMPBV corpus was shown to have the worst performance. For the CSB corpus, the body contributed the highest percentage of best queries. For the ICL corpus, the identifiers and the literals were tied, while the comments followed closely. For the LMPBV corpus, the highest percentages were for the parameters and for the method names. These were followed by the local variables and the body comments.

By taking the results of the best queries from all corpora, there was a significant increase in the results of the FLT. Unlike for the query types, the highest percentage was contributed by the CSB corpus, followed by the ICL corpus. The LMPBV corpus contributed the lowest percentage of the best queries. The combinations with the highest percentages from each corpus included the literals from the ICL, the parameters from the LMPBV, and the body from the CSB. Notable about this is that the leading percentages did not come from combinations of multiple lexicons.

5.4.2.2 JabRef

The best combinations from the LMPBV corpus and the Combined query type come from the body comments, the leading comments, and the leading comments combined with the local variables. Again, the results were different for the Title query type. While the body comments were still included, other top combinations included the leading comments combined with the body comments, the body comments combined with the local variables, and the combination of all three (leading comments, body comments, and local variables). For the CSB corpus, there is also a difference between the Combined and Title query types with the Combined query type having the leading performance for the comments, and the Title query type having the leading performance for the comments combined with the body. For the ICL corpus, the comments once again has the best performance for the Combined query type, while for the Title query type the best performance comes from the comments combined with the body.

JabRef also showed larger increases when taking the best queries from the combinations. The ICL corpus was shown to have the best performance from the three corpora. Again, LMPBV had the lowest performance of the corpora. Each of the individual lexicons had the highest percentages for the corpora.

By taking the results of the best queries from all corpora, there was a significant increase in the results of the FLT. The CSB corpus once again contributed the highest percentage from the three corpora, while the lowest percentage came from the flat corpus. This shows that the additional flexibility in the queries leads to better performance. Interestingly, while the ICL corpus presented the highest MRR of the three corpora, the highest number of best queries were contributed by the CSB corpus. This also happened for ArgoUML, and can be explained by stating that the best queries contributed by the CSB corpus typically had higher ranks than the best queries contributed by ICL.

5.4.2.3 jEdit

For the LMPBV corpus and the Combined query type, a majority of the combinations shared the same results MRRs and spreads. The three exceptions to this were for body comments, parameters alone, and parameters combined with the local variables. For the Title query type, the best performing combination was for body comments alone. For the CSB corpus, the highest MRR for both query types came from the body alone. Similarly, for the ICL query, both query types share the literals as the combination with the highest MRR.

There were large increases again for the best queries from the different combinations. The highest MRR was again found for the ICL corpus, and the lowest MRR was found for the LMPBV corpus, and once again the best combinations were the combinations composed of a single lexicon.

In the case of jEdit, we do not see the CSB once again as having the highest percentage

of the corpora, however it is close to ICL. The corpus with the lowest percentage was once again the flat corpus. The combinations that contributed the highest percentages included the body comments from the LMPBV corpus, the signature from the CSB corpus, and a tie between the literals and comments for the ICL corpus.

5.4.2.4 muCommander

The LMPBV corpus and Combined query type for muCommander found that parameters had the highest MRR, while for the Title query type, the parameters combined with the local variables had the highest performance. Parameters combined with local variables also appeared in jEdit as a notable combination. This is because there is significant overlap in the terms between the local variables and the parameters. For the CSB corpus, the comments had the best performance for the Combined query type, while the signature had the best performance for the Title query type. The literals had the leading performance for both the Combined and Title query types in ICL. In addition, the comments were also identified for the Combined query type.

As with the other subject systems, the ICL corpus had the highest MRR of the three corpora, and we once again see the highest percentages for the individual lexicons.

While there is another large increase in the MRR for looking at the best queries across corpora, the MRR is the lowest of the four subject systems. The highest percentage once again comes for the CSB corpus, while the lowest is found for the LMPBV corpus. The combinations with the highest percentages included the comments from ICL and CSB and the parameters from LMPBV.

5.4.2.5 All Systems

For the LMPBV corpus and the Combined query type, the leading comments, body comments, and leading comments combined with the local variables were shown to have the best

performance. For the Title query type, the body comments outperformed the other combinations. For the Combined query type in CSB, the comments, the comments combined with the body, and the full corpus had the best performance. The comments combined with the body had the best performance for the Title query type. For both query types, the literals had the leading performance of the combinations, while the comments were also a top performer for ICL.

As with the results for the individual systems, the ICL corpus had the highest MRR while LMPBV had the lowest. Furthermore, the individual lexicons were shown to contribute the highest percentages amongst the corpora.

Overall, the CSB corpus was shown to contribute the highest percentage of the best queries. The flat corpus was shown to contribute the least. The highest contributing combinations included the comments alone, the literals alone, and the body comments alone.

5.4.2.6 Combining the Results

Looking at the overall results, we can see that the query type can impact which combination has the leading performance. When looking at the results, the Title query type still outperforms the Combined query type. Calculating the best queries coming from each query type shows that the Title query type contributes approximately 75% of the best queries between the two query types. The changes in the combinations are highly attributed to noise from the Combined query type that makes it harder to find the relevant lexicon. Therefore, it will be important for the developer creating the queries to have an understanding of which terms to use when querying each lexicon.

Individual lexicons were shown to outperform composite combinations. Of the lexicons, the most important ones were found to be the comments and the literals. Identifiers were found to have a weaker relationship with the queries, however the top performing lexicon from the identifiers included the parameters. With these results in mind, it is important to have a corpus that

allows literals and comments to be queried independently from the identifiers. However, the highest contributions came from the CSB corpus and not the ICL corpus. This was due to the high contributions that came from the signature and the body, indicating that grouping all identifiers together will lead to a reduction in performance. After I have discussed the results of *Research Question 4*, I will introduce a new corpus model that will take into account the results from these two questions.

5.4.3 Does structural weighting affect the accuracy of a structured retrieval-based FLT?

As the previous question controlled for weighting, this question controls for the combinations by looking at the performance of weighting on the full corpus. Weighting works to improve results by placing emphasis on the lexicons that are most important to the results. Once again, if we use the example of querying with an error message, the literals may be considered the most important part of the returned result. However, a developer may realize that there could be multiple methods with similar strings and that the query might also have something to do with the identifiers in the relevant method. In this case, they can query over both lexicons but place greater emphasis on the literals.

Again, in this question I use both the Combined and Title query types.

5.4.3.1 ArgoUML

Similar to the previous question, I look at both spreads of the effectiveness measures and the computed MRRs. For the LMPBV corpus and the Combined query type, the leading comments and parameters were given weighting factors higher than the other lexicons. In both, the weighting factors for leading comments were higher than those for parameters. The Title query type however found weighted body comments with the highest factor, while leading comments and parameters were given lower weights. In the CSB corpus, signature went unweighted in most configurations

while comments and the body received higher weights. In the Title query type, a similar pattern emerged, but the signature was weighted in the top two configurations. The Combined query type for ICL emphasized identifiers and comments, while the Title query type emphasized comments and literals.

As can be observed in the bargraphs, there were wider differences between the best and worst weighting configurations versus the structural combinations. Each of the corpora also resulted in higher differences in the MRR. I computed the percentage for each corpus of the amount of instances where a weighting configuration outperformed the unweighted configuration. As expected, the percentages were close to 100% in each case. This is because the full corpus itself does not perform very well, and by emphasizing more relevant lexicons, the amount of noise caused by querying each lexicon in the corpus is reduced.

The increase in the overall MRR was smaller than for the combinations. Despite this, the change in MRR from *Research Question 1* is large. This is because while the weighting configurations perform differently when compared to the unweighted configuration, the best weightings between the corpora are similar in their rankings. The largest contribution for the corpora is from the CSB corpus, while the flat corpus comes in second.

5.4.3.2 JabRef

In JabRef, the Combined query type place the most emphasis on the method names and a lesser emphasis on leading comments. In the Title query type, the method names were emphasized again, but the leading comments were emphasized to a greater extent than in the Combined query type. For the CSB corpus, the comments and the body had the greatest emphasis in botht eh Combined and the Title query types. Finally, for the ICL corpus, the main emphasis was placed on the comments with lesser emphasis placed on the identifiers and literals.

Again, there was a large increase for the MRRs of the individual corpora when compared to the results of the query types. This is reflected in the percentages where over 90% of the best queries came from weighted configurations.

A larger increase was seen overall than for ArgoUML, however the resulting MRR was still lower than the same MRR for the combinations. Of the corpora, the largest contribution came from the CSB corpus, however the LMPBV corpus closely followed with a less than 4% difference. The flat corpus contributed an equal percentage with the CSB corpus.

5.4.3.3 jEdit

The highest weighting for the LMPBV corpus and the Combined query type was placed on the method names followed by the parameters. For the Title query type, there was still a slight emphasis on the method names, but the highest weighting factors went to the leading comments and parameters. The Combined query type for the CSB corpus resulted in higher weighting factors for the body, while the emphasis from the best of the Title query type fluctuated in importance between the comments and the body. Then, for the ICL corpus and the Combined query type, strong emphasis was placed on the identifiers while for the Title query type, strong emphasis was placed on the comments.

There were moderate increases in the results of the corpora due to the weighting configuration. Despite this, there was still a large increase in the MRRs when compared to the results of *Research Question 1*, and the weighted configurations contributed over 96% of the best queries in each corpus.

Overall, the increase for the best was large when compared to the individual corpora. However, the MRR is significantly lower than the MRR from the combinations. The CSB corpus once

again contributed the highest percentage of the best queries, while the flat corpus contributed the least.

5.4.3.4 muCommander

For each of the top configurations in the LMPBV corpus with the Combined query type, the method names were given the highest weighting factor with parameters, body comments, and leading comments given small weighting factors in a small number of configurations. However, when looking at the Title query type, the weighting factors almost invert with the leading comments, parameters, body comments, and local variables receiving heavy weightings factors. The method names reduce for this query type. In both the Combined and Title query types for CSB, the comments receive the heaviest weighting, while for the Combined query type more emphasis is placed on the signature and for the Title query type more emphasis is placed on the body. The Combined and Title query types for ICL both place heavy emphasis on the comments, however for the Title query type, heavy emphasis is also placed on the identifiers.

The differences for muCommander for each corpus are substantially higher than the results for *Research Question 1*. Again, weighting configurations contribute the majority of the best queries for each corpus with a contribution of 95% for each corpus.

The MRR for muCommander is still small compared to the other subject systems, however there is an increase for the overall results. As with the other systems, the CSB corpus contributes the highest percentage of the best queries, while the ICL corpus contributes the least. The flat corpus is outperformed by the LMPBV corpus.

5.4.3.5 All Systems

For all systems, both Combined and Title query types provide the greatest emphasis to the leading comments in the LMPBV corpus, while secondary emphasis is placed on the method

name and the parameters. In the Title query type, secondary emphasis (over the method names and parameters) is also placed on the body comments. In the CSB corpus, both the Combined and the Title query types place heavy emphasis on the comments with secondary emphasis on the body. Finally for the ICL, the Combined query type places equal emphasis on identifiers and comments, while the Title query type places the heaviest emphasis on the comments.

There is a significant increase in the MRRs from the results of *Research Question 1* in each corpus with the MRRs being over double for each corpus. This is a result of the weighting configurations contributing the most to the best queries. In both CSB and ICL, weighting configurations result in 96% of the best queries, while for LMPBV, weighting configurations contribute 95%.

The results of the overall MRR is significantly smaller than the results of the combinations, however they are significantly higher than the results of query types. Overall, the CSB corpus contributes the highest percentage of the best queries, while ICL contributes the least and there is a tie between LMPBV and the flat corpus.

5.4.3.6 Combining the Results

The results of this question should be looked at in conjunction with the results from *Research Question 2*. There is a relationship between the combinations and the results of the weighting. This is easiest to see for the CSB and ICL corpora for all systems combined. Since the weighting configurations for the structured retrieval technique are ratio based, only two lexicons will be weighted at maximum. For CSB, the best two lexicon combination is comments combined with the body, while for ICL, the best two lexicon combination is IC. When looking at the weightings, it is easy to identify these as being the lexicons that are weighted together in the top configurations for each corpus. This is an indicator that the weightings help to emphasize the

same structural combinations that lead to better results, but providing increased flexibility for each lexicon to be given a relative importance with the other member of the combination.

Some of the results emphasize different lexicons than the LDA study, one possibility for this is that LDA is topic-based while the structured retrieval approach is term-based. A possiblity is that for the topic-based approach, terms that are descriptive of the responsibilities and topics in each method are the most important, while for the term-based approach, terms that act as distinguishers (i.e., set the method apart from others) are more important. Additional studies would be required to determine whether this is really the case.

It is important to note however, that some of the combinations will see a decrease in performance if the entire corpus is used. This is due to the noise that is added by the extra lexicons and contexts. Therefore, while weighting can increase the performance of a given combination, choosing the correct combination first is an important factor.

5.4.4 How does the best configuration of structural field combination and weighting affect the accuracy of a structured retrieval-based FLT?

The previous two questions looked at structural combinations and weighting independently. In this question, I wish to show how working together will increase the results over either dimension alone. For this question, I only used the Title query type. This was done for feasibility purposes, however this does not jeopardize the results, as we will still see increases in the performance.

5.4.4.1 ArgoUML

For the LMPBV corpus, popular combinations included leading comments combined with body comments, leading comments combined with parameters, and leading comments combined with method names. This extends to combinations that combine all four of these types. In each

of the configurations, the leading comments have the highest or are tied for the highest weighting factors of the lexicons. Only two combinations were found for the CSB corpus, the full corpus and the comments combined with the body. For each of these configurations, the comments were weighted heavier than the other lexicons. Three combinations comprise the top configurations for ICL. The three combinations consist of the literals alone as the top configuration, followed by the comments combined with the literals, and the full corpus.

The results for the individual corpora are significantly higher than the previous questions. This is especially true of the LMPBV corpus. This is not difficult to understand. The LMPBV corpus has the highest level of flexibility of any of the corpora. This allows for a far wider number of possible queries for the corpus. For both CSB and ICL, a weighting configuration was used 100% of the time. For both the CSB and ICL corpora, the full corpus contributed the greatest percentage of best queries. For LMPBV however, the greatest percentage came from the parameters alone.

The performance of the overall results are significantly higher than the corpus values. Interestingly, when looking at all combinations, the LMPBV corpus is the highest contributor of best queries, indicating the importance of a structured corpus to be flexible. Of the best queries, 80% of the best queries were the results of a weighting configuration. The full corpus did not contribute significantly to the best queries for the overall results.

5.4.4.2 JabRef

For LMPBV, each of the top configurations consists of both the leading comments and the body comments with the leading comments being weighted heavier than other lexicons. Parameters and local variables are included throughout the configurations with each having minimum weighting. In all but one configuration for CSB, the comments and the body are included. Furthermore, the most common combination is the full corpus with the comments being given the heaviest

weighting. In all but one configuration for the ICL corpus, the comments and the literals are included with heaving weighting for the comments. The two most common combinations include the comments and literals alone and the full corpus.

Again there is a substantial increase in the MRR for the three corpora, but the LMPBV is once again much higher than the others. For the three corpora, both CSB and ICL have 100% of their best queries from weighting configurations, while LMPBV is lower at 87%. The full corpus contributes the highest percentages in both CSB and ICL, while the parameters is once again the highest contributor from LMPBV with body comments as the second top contributor.

The overall results of the system showed a significant increase in MRR. Only the LMPBV corpus and the CSB corpus contributed to the overall results for the system. The full CSB corpus made up the entirety of the CSB corpus's contribution to the best results. It is important to note that a small percentage such as the 6% from CSB was enough to raise the results of the LMPBV corpus by 18%.

5.4.4.3 jEdit

The most common lexicons for the top configurations include the leading comments and the parameters. Leading comments are given the highest weighting factors or are tied with other lexicons for the highest. For the CSB corpus, the most common combination included the full corpus follow by the comments combined with the body. For the ICL corpus, the top two configurations are composed of the literals alone and the comments alone. The most common configurations include the combination of the identifiers with the literals.

Again, each corpus showed a significant increase in performance with the LMPBV corpus showing the largest increase. The best queries in CSB and ICL used weighting configurations 100% of the time, while the best queries in the LMPBV corpus only used a weighting configuration

84% of the time. Interestingly, unlike the previous two systems, the comments combined with the signature contributed the highest percentage for the CSB corpus, while the identifiers combined with the comments contributed the highest percentage for the ICL corpus. The highest percentage for the LMPBV corpus came from the body comments alone.

There was a large increase for the overall results and the LMPBV corpus contributed the highest percentage of the four corpora. This system is the perfect example of why the highest contributing combinations in the overall results are different from the expectation from the individual combinations and the corpora. The highest contributor for the LMPBV corpus is the body comments alone. The comments in the CSB and ICL corpora are supersets of this model. A query that is most relevant to the body comments can perform much worse when compared to the entire comments. In these cases, the results for a query that queries only the body comments will outperform either the CSB or the ICL corpus. Lowering contributions and changing the highest contributing combinations from these corpora.

5.4.4.4 muCommander

The two most common lexicons for the LMPBV corpus consists of the method names and the leading comments. In configurations containing the leading comments and the method names, the leading comments have higher weightings factors. There is not a clear combination that is most frequent within the CSB corpus, however the top configuration includes the signature alone while every other combination is composed of at least two lexicons. The top configuration for the ICL corpus are the literals alone, while the most common combinations include both the identifiers and the comments.

The MRRs for muCommanderare the lowest of any system in each question. However, the MRR for LMPBV is substantially higher than the other corpora. Again, weighting configurations

contribute 100% of the best queries in both CSB and ICL. The comments and the signature make the best combination for the CSB corpus, while there is a tie between the identifiers combined with the comments and the comments combined with the literals for the ICL corpus. The three query types for LMPBV (leading comments, body comments, and leading comments with body comments) comprise the highest percentages for LMPBV.

The overall results once again increase for muCommander, while the percentage contributed by LMPBV is substantially higher than the other corpora.

5.4.4.5 All Systems

Leading comments are present in each of the top configurations. Body comments also appear with higher frequency than other lexicons. In each configuration, leading comments are given higher weighting factors than other lexicons. The two most common lexicons for the CSB corpus includes the comments and the body with the commments receiving higher weighting. The full corpus is the most frequent combination for the ICL corpus, while literals are the top configuration. In each configuration, comments receive higher weightings than other lexicons.

As with the subject systems, the LMPBV corpus was shown to have the highest increase in performance, and the CSB and ICL corpora were shown to use a weighting configuration for each of the best queries. The best queries for the LMPBV corpus used a weighting configuration in 88% of the queries. In each corpus, the highest percentages came from combinations that used the comments.

Overall, the LMPBV corpus contributed 83% of the best queries. The highest contributing combination for the LMPBV corpus came from the body comments alone, while the top contributing combinations for each of the other corpora included comments.

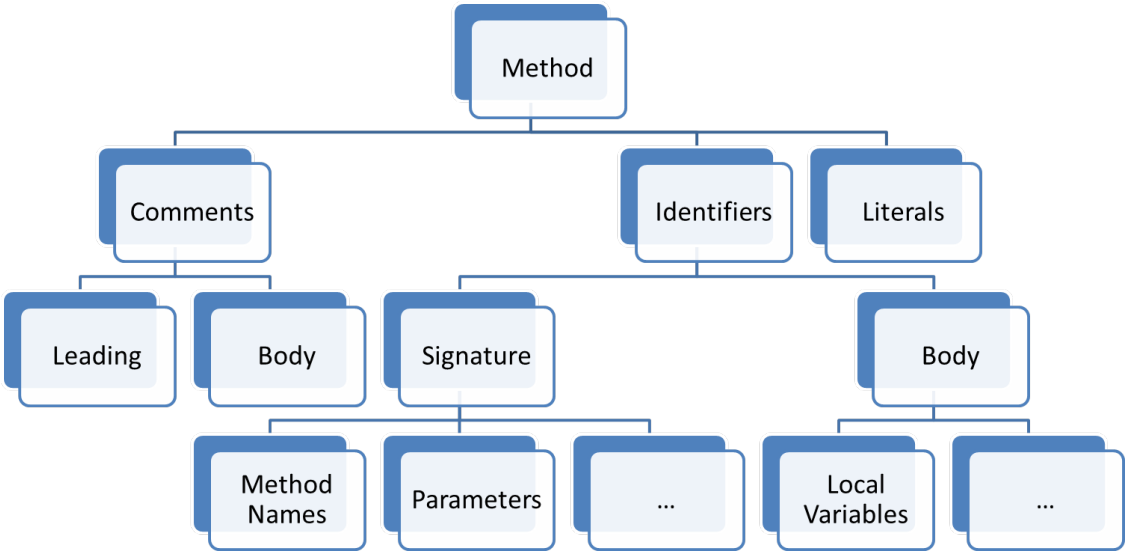


Figure 5.81: Recommended Corpus Structure

5.4.4.6 Combining the Results

The results of this problem and *Research Question 2* and *Research Question 3* help to understand why LMPBV outperformed the other corpora when all combinations were considered. From the results of *Research Question 2*, being specific when targeting a lexicon will help to improve the results, while the results from *Research Question 3* show that being able to supply relative importance to each lexicon in a combination improves results. With this in mind, a corpus should allow for sufficient flexibility in both creating the combination and in weighting the lexicons.

While being specific helps to improve the results, in some cases using supersets lead to better performance than the individual lexicon (e.g, the comment combination can lead to better results than the leading comments or the body comments). With this information, the best results will occur when given a corpus that allows the best features of the CSB, ICL, and LMPBV corpora to be used. For this reason, I recommend a new corpus for structured retrieval. The resulting structure can be found in Figure 5.81.

5.5 Summary

I conducted a large study to understand how CAS queries may be used to effect the results of a structured retrieval approach to feature location. I analyzed queries on three different dimensions. The first was how the inclusion of different pieces of information in the query may affect the results. To this end, I extracted queries from the title, description, and combination of the two from features requests. I found that the performance of the structured retrieval can be significantly impacted by the information included in the query. Second, I looked at how including different structure and structure combinations in the query affect the results. For this question, I took each combination of structure lexicons from three different corpora and analyzed how the results changed. I found that it is not always best to query over all parts of a source code element, and that searching over the wrong structural lexicons can actually lead to a degradation of results. The third dimension I looked at was the weighting of the lexicons in a query. This was similar to the structural weighting for LDA, however instead of the weighting improving the overall retrieval model, the weighting used in this study weights the results of parts of queries issued on the model. I found that in most cases using a proper weighting will lead to better results than unweighted queries by giving emphasis to the parts of the query that are most important. Finally, I looked at all of the dimensions together and found that using the proper structural combination combined with a proper weighting will work together to lead to a significant improvement in the results of the retrieval technique.

Chapter 6

LESSONS LEARNED AND FUTURE WORK

Throughout this dissertation, two techniques have been presented that use the structural location of terms to improve TR-based FLT. Each of the techniques have benefits and consequences, and determining which technique is best is not within the scope of this dissertation. In order to do so would require having a fully optimized configuration for each technique, and due to the large number of configurations and the required time to train and query these models, such a comparison is infeasible at this time. However, the studies conducted in this dissertation can provide insights into each technique and help to guide future research. In this chapter, the benefits and consequences of both techniques will be discussed. This chapter will answer the following two questions:

1. What are the benefits and consequences of using structural weighting in LDA?

2. What are the benefits and consequences of using structured retrieval for feature location?

Furthermore, future work will be presented that can extend both techniques.

6.1 What are the benefits and consequences of using structural weighting in LDA?

The main advantage of using a topic modeling approach is that it is less sensitive to the usage of terms. Instead of querying based on the terms, and therefore requiring identical terms between the query and the source code elements, querying is performed across probability distributions that represent topics. This means that topic models should be less sensitive to tasks such as

stop word removal and stemming. It also means that while synonyms can still impact the results, the topic model should be able to handle them better than a traditional algebraic model. This leads to the idea that a properly trained topic model should perform well when compared to other text retrieval techniques on general queries.

This is supported by existing research [Biggers, 2012] into the performance of LDA for software engineering, and by the results of the study presented in Chapter 4. For techniques that focus on using queries that are generated directly for feature requests, LDA has been shown to have results that are as good or better to the state of the art [Binkley et al., 2015]. These results are improved even further by structural weighting.

The limitation of this technique comes in the form of training. LDA is sensitive to proper training and configuration [Biggers et al., 2012]. Improper training can significantly impact the results of the LDA-based technique and can also lead to unreliable and unpredictable results. This issue led to the work in the preliminary study.

Training also imposes limitations with the structural weighting approach for LDA. While this research has presented recommendations for identifying a structural weighting scheme that will outperform the unweighted configuration, identifying the optimum weighting requires additional searching or learning techniques. Such techniques require training and verifying new LDA models which can take a considerable amount of time. The time required to train and verify new LDA models increases with the amount of structural locations and weighting factors that are considered. For a large system, this may be prohibitive in adopting the technique without sufficient computational resources.

6.2 What are the benefits and consequences of using structured retrieval for feature location?

As opposed to structural weighting for LDA, the purpose of the structured retrieval approach is to provide more flexibility on a query by query basis. This technique allows a developer the flexibility to modify their query based on structural context and to use the additional power of an advanced query language to obtain better results. As shown in this research, when developers are using the query language to its full potential, the results of the technique are improved significantly.

For developers to write the best queries, they need a strong mental model of how the system is structured, knowledge of the usage of terms in the system, and an understanding and desire to use the full querying capabilities. Many of these problems are areas for future research listed in Chapter 5. Future research will need to look into methods of recommending queries to developers, making the query language easier to use for developers, and providing developers with information that helps support their building of a mental model for the system.

The study presented in Chapter 5 focused on Indri and the use of language models. However, these types of queries can be incorporated and combined with other models. The only requirement is for the model to provide the ability to search specific structural contexts.

6.3 Future Work

This dissertation leaves ample opportunity for future studies. This section discusses future work that can be performed in this area.

6.3.1 Improved Learning Algorithms

A preliminary algorithm was presented for learning structural weighting for LDA. To be the most effective, this technique requires a history of changes and feature requests. Another

limitation of this technique is the time required to identify the optimum weighting configuration. This is due to the amount of time required to train a weighted LDA model for larger systems. This time requirement imposes a major limitation on the technique when trying to find an optimal weighting structure.

6.3.2 Empirical Studies

The structured retrieval approach used queries that were extracted from the feature requests and queries that were written in a modified version of Indri's querying language. However, these results do not say anything about how developers will actually use the structured retrieval system in practice. For this reason, two major studies would help support the usage of structured retrieval in software engineering tasks.

The first study would design multiple querying languages and interfaces and have the developers use the resulting tools. From this study, the hope is to identify an interface that is easy to use for the developer and allows them to make the greatest use of the structured queries. The second study would identify ways of integrating the search techniques into the developer's normal routine and encourage them to adopt the tool. While the results of the queries may have better performance than existing techniques, this performance will be unimportant if the developer does not make use of it.

6.3.3 Modeling Structural Information

In order for the structured retrieval system to be the most effective, it requires developers to have a clear mental model of the terms and their usage within the software system. This makes such a system more difficult for new developers to use without first studying the system. Future research would need to focus on reducing this hurdle. There are two methods that may help solve this problem.

The first is in machine learning algorithms that work to recommend queries to new developers by modeling the queries and the source code, while learning from a history of previously performed queries. Such a technique would require three parts: features describing the terms in each structural context, features describing the queries, and a method of determining the effectiveness of a query. Possible features for describing the structural context include unique term density, term contribution, lexicon density, part-of-speech usage, frequent n-grams, or other information of this type. Features describing the query include n-grams, part-of-speech usage, or various similarity measures with terms in each structural context. Determining effectiveness could be created by mining changes from a software repository or using relevance feedback from the developer.

The second area would be to create an interface that provides the developer with additional information about the terms and their usage in the software systems. This interface should provide information that allows a developer to build a mental model of the system quickly.

6.3.4 Additional Software Tasks

Another straightforward area for future research is to look at how these approaches perform in different software engineering tasks (e.g., traceability, triage, software search, categorization). Furthermore, this study only focuses on software systems written in the Java programming language, and while I assume that performance will be similar for other object-oriented languages, this would need to be verified.

Chapter 7

CONCLUSION

Structural context is a key factor in the performance of TR-based techniques in software engineering. Different types of terms in source code will play many different roles, and can convey different information in various contexts. For example, an identifier *test* may appear in the name of one method and in the method call in the body of another method. In the first case, the term *test* may help to describe the main responsibility of the method, while in the second case, the term may be part of a single step in a larger responsibility. However, little research prior to this dissertation has been performed on how to use context to improve the results of TR-based techniques, and less has been performed on using it to improve feature location. The research described in this dissertation resulted in the following contributions:

- A large empirical study on the effects of structural weighting on LDA-based feature location
- Recommendations and insights on how to choose a proper weighting configuration with little additional training
- A learning algorithm for determining the optimum structural weighting over time
- The introduction of content and structure (CAS)-based queries to search a corpus and provide additional flexibility to developers
- A large empirical study on the effects of CAS queries to a FLT

- Insights on how to structure a corpus for structured retrieval and for writing queries to return the best results

7.1 Structural Weighting of LDA

Chapter 4 introduced weighting configurations of LDA for four subject software systems and showed that weighting lexicons identified by their structure increases the performance of an LDA-based FLT. My results also identified characteristics of the lexicons that showed the best increases in performance when weighted highly and made recommendations for identifying which lexicons should receive higher weights. While the recommendations may not identify the configuration with the highest performance, I outlined a search process that can identify better weighting configurations over time.

The limitations of this approach are in the need to identify the lexicons that should receive higher weighting factors and the time required to identify the best weighting configuration.

7.2 Structured Retrieval

Chapter 5 showed how to use content and structure to search a structured document retrieval (SDR) [Lalmas and Baeza-Yates, 2009] model. Unlike traditional TR models, SDR models support powerful query languages in which a user may specify several constraints, including the scope of the query and the weight or probability assigned to each term or structural entity.

I conducted a large study to understand how CAS queries may be used to affect the results of a structured retrieval approach to feature location. I analyzed queries on three different dimensions. The first was how the inclusion of different pieces of information in the query may affect the results. I extracted queries from the title, description, and combination of the description and

the titile from feature requests. I found that the performance of the structured retrieval technique can be significantly impacted by the information included in the query.

Second, I looked at how including different structure and structure combinations in the query affect the results. For this question, I took each combination of structure components from three different corpora and analyzed how the results changed. I found that it is not always best to query over all parts of a source code element, and that searching over the wrong structural components can actually lead to a degradation of results.

The third dimension considered was the weighting of the components in a query. This was similar to the structural weighting for LDA. However, instead of the weighting improving the overall retrieval model, the weighting used in the study weighted the results of parts of queries issued on the model. I found that in most cases using a proper weighting will lead to better results than unweighted queries by giving emphasis to the parts of the query that are most important.

Finally, I looked at all of the dimensions together and found that using the proper structural combination combined with a proper weighting will work together to lead to a significant improvement in the results of the retrieval technique.

The results of my study identified the following insights about constructing CAS queries:

- The developer should be concise in the terms they use to describe the query, while providing sufficient information to identify the proper method
- Queries should be targeted at the components or combination of components that are most likely to be related to the query. This requires a clear mental model of the system for the developer
- Weighted configurations will often result in higher rankings than the structural combination

alone. In order to form the query, the developer should first select the combination and then determine the relative importance of each component

In addition, I gave a representation for a corpus based off of the findings in my results. Instead of using any individual corpus from those studied in Chapter 5, the best corpus combines the three corpora while still allowing for the same queries. This allows for the same performance that was shown when taking the best results from each of the individual corpora.

7.3 Final Remarks

The research presented in this dissertation shows how the structural location of terms in source code can have a significant effect on the results of TR-based FLTs. The studies in this dissertation used open source Java systems that have been studied previously by the research community. This dissertation is not a comprehensive list of the ways that structural location can be incorporated into TR techniques. Chapter 6 presented multiple areas for future research, and further research exists in incorporating structural location into other models. Source code elements are structured entities, and researchers should be aware of how this affects the TR process.

REFERENCES

Abebe, S., S. Haiduc, A. Marcus, P. Tonella, and G. Antoniol (2009). Analyzing the evolution of the source code vocabulary. In *Proceedings of the 13th European Conference on Software Maintenance and Reengineering*, pp. 189–198.

Alhindawi, N., N. Dragan, M. Collard, and J. Maletic (2013). Improving feature location by enhancing source code with stereotypes. In *Proceedings of the 2013 International Conference on Software Engineering*, pp. 762–771.

Alkhatib, G. (1992). The maintenance problem of application software: an empirical analysis. *Journal of Software Maintenance: Research and Practice 4*(2), 83–104.

Antoniol, G., G. Canfora, A. Casazza, D. Lucia, and E. Merlo (2002). Recovering traceability links between code and documentation. *28*(10), 970–983.

Antoniol, G., G. Canfora, G. Casazza, and A. De Lucia (2000). Information retrieval models for recovering traceability links between code and documentation. In *Proceedings of the International Conference on Software Maintenance*, pp. 40–49.

Antoniol, G., Y.-G. Gueheneuc, E. Merlo, and P. Tonella (2007). Mining the lexicon used by programmers during sofware evolution. In *Proceedings of the IEEE International Conference on Software Maintenance*, pp. 14–23.

Bäck, T. (1996). *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford university press.

Bajracharya, S., T. Ngo, E. Linstead, Y. Dou, P. Rigor, P. Baldi, and C. Lopes (2006). Sourcerer: A search engine for open source code supporting structure-based search. In *Companion to the 21st ACM SIGPLAN Symposium on Object-oriented Programming Systems, Languages, and Applications*, pp. 681–682.

Baldi, P., C. Lopes, E. Linstead, and S. Bajracharya (2008). A theory of aspects as latent topics. *SIGPLAN Notifications 43*(10), 543–562.

Bassett, B. and N. A. Kraft (2013). Structural information based term weighting in text retrieval for feature location. In *Proceedings of the IEEE 21st International Conference on Program Comprehension*, pp. 133–141.

Biggers, L., C. Bocovich, R. Capshaw, B. Eddy, L. Etzkorn, and N. Kraft (2012). Configuring latent Dirichlet allocation based feature location. *Empirical Software Engineering 19*(3), 465–500.

Biggers, L., B. Eddy, N. Kraft, and L. Etzkorn (2011). Toward a metrics suite for source code lexicons. In *Proceedings of the IEEE International Conference on Software Maintenance - Early Research Achievements Track*, pp. 492–495.

Biggers, L. R. (2012). *Investigating the effect of corpus construction on latent dirichlet allocation based feature location*. [Tuscaloosa, Ala.] : [University of Alabama Libraries], 2012.

Binkley, D., D. Lawrie, C. Uehlingera, and D. Heinzb (2015). Enabling improved ir-based feature location. *Journal of Systems and Software 101*, 30–42.

Blei, D., A. Ng, and M. Jordan (2003). Latent Dirichlet allocation. *Journal of Machine Learning Research 3*(0), 993–1022.

Boehm, B. (1981). *Software Engineering Economics*. Prentice Hall.

Brin, S. and L. Page (1998). The anatomy of a large-scale hypertextual web search engine. *Computer networks and ISDN systems 30*(1), 107–117.

Casella, G. and E. I. George (1992). Explaining the gibbs sampler. *The American Statistician 46*(3), 167–174.

Cleary, B. and C. Exton (2006). The cognitive assignment eclipse plug-in. In *Proceedings of the 14th IEEE International Conference on Program Comprehension*, pp. 241–244.

Cleary, B. and C. Exton (2007). Assisting concept location in software comprehension. In *Proceedings of the 19th Psychology of Programming Workshop*, pp. 42–55.

Cleary, B., C. Exton, J. Buckley, and M. English (2009). An empirical analysis of information retrieval based concept location techniques in software comprehension. *Empirical Software Engineering 14*, 93–130.

Dallmeier, V. and T. Zimmermann (2007). Extraction of bug localization benchmarks from history. In *Proceedings of the 22nd IEEE/ACM international conference on Automated software engineering*, pp. 433–436.

David, J. (2008). Recommending software artifacts from repository transactions. In *New Frontiers in Applied Artificial Intelligence*, pp. 189–198. Springer.

De Lucia, A., R. Oliveto, and G. Tortora (2008). Adams re-trace: traceability link recovery via latent semantic indexing. In *Proceedings of the 30th international conference on Software engineering*, pp. 839–842.

De Lucia, A., M. Risi, L. Rizzi, and G. Scanniello (2008). A visual framework for the definition and execution of reverse engineering processes. In *Proceedings of the 10th International Conference on Visual Information Systems: Web-Based Visual Information Search and Management Visual Information Systems*, pp. 235–246.

Deerwester, S., S. Dumais, G. Furnas, T. Landauer, and R. Harshman (1990). Indexing by latent semantic analysis. *Journal of the American Society of Information Science 41*(6), 391–407.

desRivieres, J. and J. Wiegand (2004). Eclipse: A platform for integrating development tools. *IBM Systems Journal 43*(2), 371–383.

Dit, B., L. Guerrouj, D. Poshyvanyk, and G. Antoniol (2011). Can better identifier splitting techniques help feature location? In *Proceedings of the IEEE 19th International Conference on Program Comprehension*, pp. 11–20.

Dit, B., M. Revelle, M. Gethers, and D. Poshyvanyk (2012). Feature location in source code: A taxonomy and survey. *Journal of Software Maintenance and Evolution: Research and Practice 25*(1), 53–95.

Eddy, B., J. Robinson, N. Kraft, and J. Carver (2013). Evaluating source code summarization techniques: Replication and expansion. In *Proceedings of the IEEE 21st International Conference on Program Comprehension*, pp. 13–22.

Erlikh, L. (2000). Leveraging legacy system dollars for e-business. *IEEE IT Pro 2*(3), 17–23.

Fluri, B., M. Wursch, and H. Gall (2007). Do code and comments co-evolve? on the relation between source code and comment changes. In *Proceedings of the 14th Working Conference on Reverse Engineering*, pp. 70–79.

Fox, C. (1992). Information retrieval data structures and algorithms. *Lexical Analysis and Stoplists*, 102–130.

Gao, J., J. Nie, G. Wu, and G. Cao (2004). Dependence language model for information retrieval. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 170–177.

Gospodnetic, O. and E. Hatcher (2005). *Lucene*. Manning.

Griffiths, T. and M. Steyvers (2004). Finding scientific topics. *Proceedings of the National Academy of Sciences 101*(Supplement 1), 5228–5235.

Haiduc, S., J. Aponte, and A. Marcus (2010). Supporting program comprehension with source code summarization. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering*, pp. 223–226.

Haiduc, S. and A. Marcus (2008). On the use of domain terms in source code. In *Proceedings of the 16th IEEE International Conference on Program Comprehension*, pp. 113–122.

Heinrich, G. (2009, September). Parameter estimation for text analysis. Technical report, Fraunhofer IGD, Darmstadt, Germany. Version 2.9.

Hill, E., L. Pollock, and K. Vijay-Shanker (2007). Exploring the neighborhood with Dora to expedite software maintenance. In *Proceedings of the 22nd International Conference on Automated Software Engineering*, pp. 14–23.

Hill, E., S. Rao, and A. Kak (2012). On the use of stemming for concern location and bug localization in java. In *Proceedings of the Source Code Analysis and Manipulation*, pp. 184–193.

Hindle, A., M. Godfrey, and R. Holt (2009). What's hot and what's not: Windowed developer topic analysis. In *Proceedings of the IEEE International Conference on Software Maintenance*, pp. 339–348.

Hofmann, T. (1999). Probabilistic latent semantic indexing. In *Proceedings of the 22Nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 50–57.

Jiang, H., T. Nguyen, X. Chen, H. Jaygarl, and C. Chang (2008). Incremental latent semantic indexing for automatic traceability link evolution management. In *Proceedings of the 2008 23rd IEEE/ACM International Conference on Automated Software Engineering*, pp. 59–68.

Kiczales, G., E. Hilsdale, J. Hugunin, M. Kersten, J. Palm, and W. Griswold (2001). Getting started with aspectj. *Communications of the ACM 44*(10), 59–65.

Kleinberg, J. M. (1999). Authoritative sources in a hyperlinked environment. *Journal of the ACM 46*(5), 604–632.

Kuhn, A., S. Ducasse, and T. Gírba (2007). Semantic clustering: Identifying topics in source code. *Information and Software Technology 49*(3), 230–243.

Kunz, P. (2001). The hippodraw application and the hippoplot c++ toolkit upon which it is built. In *Proceedings of the CHEP*.

Lalmas, M. and R. Baeza-Yates (2009). *Structured Document Retrieval*. Springer US.

Lawrie, D., C. Morrell, H. Feild, and D. Binkley (2006). What's in a name? a study of identifiers. In *Proceedings of the 14th IEEE International Conference on Program Comprehension*, pp. 3–12.

Li, W., D. Blei, and A. McCallum (2012). Nonparametric bayes pachinko allocation. *CoRR abs/1206.5270*.

Li, W. and A. McCallum (2006). Pachinko allocation: Dag-structured mixture models of topic correlations. In *Proceedings of the 23rd International Conference on Machine Learning*, pp. 577–584.

Liu, D., A. Marcus, D. Poshyvanyk, and V. Rajlich (2007). Feature location via information retrieval based filtering of a single scenario execution trace. In *Proceedings of the 22nd International Conference on Automated Software Engineering*, pp. 234–243.

Lormans, M. and A. van Deursen (2005). Reconstructing requirements coverage views from design and test using traceability recovery via lsi. In *Proceedings of the 3rd international workshop on Traceability in emerging forms of software engineering*, pp. 37–42.

Lu, W. and M. Kan (2007). Supervised categorization of javascripttm using program analysis features. *Information Processing and Management 43*(2), 431–444.

Lukins, S., N. Kraft, and L. Etzkorn (2008). Source code retrieval for bug localization using latent dirichlet allocation. In *Proceedings of the 15th Working Conference on Reverse Engineering*, pp. 155–164.

Lukins, S., N. Kraft, and L. Etzkorn (2010). Bug localization using latent Dirichlet allocation. *Information and Software Technology 52*(9), 972–990.

Manly, K. F. and J. M. Olson (1999). Overview of qtl mapping software and introduction to map manager qt. *Mammalian Genome 10*(4), 327–334.

Manning, C., P. Raghavan, and H. Schutze (2008). *An Introduction to Information Retrieval*. Cambridge University Press.

Marcus, A. and T. Menzies (2010). Software is data too. In *Proceedings of the FSE/SDP workshop on Future of Software Engineering research*, pp. 229–232.

Marcus, A., V. Rajlich, J. Buchta, M. Petrenko, and A. Sergeyev (2005). Static techniques for concept location in object-oriented code. In *Proceedings of the 13th International Workshop on Program Comprehension*, pp. 33–42.

Marcus, A., A. Sergeyev, V. Rajlich, and J. Maletic (2004). An information retrieval approach to concept location in source code. In *Proceedings of the 11th Working Conference on Reverse Engineering*, pp. 214–223.

Maskeri, G., S. Sarkar, and K. Heafield (2008a). Mining business topics in source code using latent Dirichlet allocation. In *Proceedings of the 1st conference on India software engineering conference*, pp. 113–120.

Maskeri, G., S. Sarkar, and K. Heafield (2008b). Mining business topics in source code using latent Dirichlet allocation. In *Proceedings of the 1st India Software Engineering Conference*, pp. 113–120.

Mimno, D., W. Li, and A. McCallum (2007). Mixtures of hierarchical topics with pachinko allocation. In *Proceedings of the 24th International Conference on Machine Learning*, pp. 633–640.

Moreno, L., L. Treadway, A. Marcus, and W. Shen (2014). On the use of stack traces to improve text retrieval-based bug localization. In *Proceedings of the 30th IEEE International Conference on Software Maintenance and Evolution*, pp. 151–160.

Müller, H., J. Jahnke, D. Smith, M.-A. Storey, S. Tilley, and K. Wong (2000). Reverse engineering: a roadmap. In *Proceedings of the Future of Software Engineering*, pp. 47–60.

Newby, G. (2000). The science of large scale information retrieval. *Internet archives*.

Nikulin, M. S. (2001). Hellinger distance. *Encyclopedia of Mathematics*.

Northover, S. and M. Wilson (2004). *Swt: the standard widget toolkit, volume 1*. Addison-Wesley Professional.

Ogilvie, P. and J. Callan (2002). Language models and structured document retrieval. In *Proceedings of the Initiative for the Evaluation of XML Retrieval Workshop*, pp. 18–23.

Panichella, A., B. Dit, R. Oliveto, M. Di Penta, D. Poshynanyk, and A. De Lucia (2013). How to effectively use topic models for software engineering tasks? An approach based on genetic algorithms. In *Proceedings of the International Conference on Software Engineering*, pp. 522–531.

Parr, T. J. and R. W. Quong (1995). Antlr: A predicated-ll (k) parser generator. *Software: Practice and Experience 25*(7), 789–810.

Pinheiro, F. and J. Goguen (1996). An object-oriented tool for tracing requirements. In *Proceedings of the Second International Conference on Requirements Engineering*, pp. 219.

Poshyvanyk, D., Y. Gueheneuc, A. Marcus, G. Antoniol, and V. Rajlich (2007). Feature location using probabilistic ranking of methods based on execution scenarios and information retrieval. *IEEE Transactions on Software Engineering 33*(6), 420–432.

Poshyvanyk, D. and A. Marcus (2007). Combining formal concept analysis with information retrieval for concept location in source code. In *Proceedings of the International Conference on Program Comprehension*, pp. 37–48.

Poshyvanyk, D., A. Marcus, R. Ferenc, and T. Gyimóthy (2009). Using information retrieval based coupling measures for impact analysis. *Empirical software engineering 14*(1), 5–32.

Rajlich, V. (2011). *Software Engineering: The Current Practice*. CRC Press.

Rajlich, V. and N. Wilde (2002). The role of concepts in program comprehension. In *Proceedings of the 10th International Workshop on Program Comprehension*, pp. 271–278.

Ramesh, B. and V. Dhar (1992). Supporting systems development by capturing deliberations during requirements engineering. *Software Engineering, IEEE Transactions on 18*(6), 498–510.

Reiss, S. (2009). Semantics-based code search. In *Proceedings of the IEEE 31st International Conference on Software Engineering*, pp. 243–253.

Revelle, M., B. Dit, and D. Poshyvanyk (2010). Using data fusion and web mining to support feature location in software. In *Proceedings of the 18th IEEE International Conference on Program Comprehension*, pp. 14–23.

Rousseeuw, P. (1986). Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics 20*(0), 53–65.

Runeson, P., M. Alexandersson, and O. Nyholm (2007). Detection of duplicate defect reports using natural language processing. In *Proceedings of the 29th International Conference on Software Engineering*, pp. 499–510.

Saha, R., M. Lease, S. Khurshid, and D. Perry (2013). Improving bug localization using structured information retrieval. In *Proceedings of the 28th IEEE/ACM International Conference on Automated Software Engineering*, pp. 345–355.

Salton, G. and C. Buckley (1988). Term-weighting approaches in automatic text retrieval. *Information Processing & Management 24*(5), 513–523.

Salton, G. and M. McGill (1986). *Introduction to Modern Information Retrieval*. McGraw-Hill.

Scanniello, G. and A. Marcus (2011). Clustering support for static concept location in source code. In *Proceedings of the 19th IEEE International Conference on Program Comprehension*, pp. 1–10.

Scheuermann, C., M. Werner, M. Kessel, C. Linnhoff-Popien, and S. Verclas (2012). Evaluation of barcode decoding performance using zxing library. In *Proceedings of the Second Workshop on Smart Mobile Applications*, pp. 1–6.

Shao, P. and R. Smith (2009). Feature location by ir modules and call graph. In *Proceedings of the 47th Annual Southeast Regional Conference*, pp. 70:1–70:4.

Sisman, B. and A. Kak (2012). Incorporating version histories in information retrieval based bug localization. In *Proceedings of the 9th IEEE Working Conference on Mining Software Repositories*, pp. 50–59.

Song, D. and P. Bruza (2001). Discovering information flow using high dimensional conceptual space. In *Proceedings of the 24th International ACM SIGIR conference on Research and development in information retrieval*, pp. 327–333.

Standish, T. (1984). Information retrieval data structures and algorithms. *IEEE Transactions on Software Engineering 10*(5), 494–497.

Strohman, T., D. Metzler, H. Turtle, and W. B. Croft (2005). Indri: A language model-based search engine for complex queries. In *Proceedings of the International Conference on Intelligent Analysis*, pp. 2–6.

Tairas, R. and J. Gray (2009). An information retrieval process to aid in the analysis of code clones. *Empirical Software Engineering 14*(1), 33–56.

Teh, Y., M. Jordan, M. Beal, and D. Blei (2006). Hierarchical dirichlet processes. *Journal of the American Statistical Association 101*(476), 1566–1581.

Turtle, H. and W. B. Croft (1990). Inference networks for document retrieval. In *Proceedings of the 13th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 1–24.

Voorhees, E. (1999). The trec-8 question answering track report. In *Proceedings of the TREC-8*, pp. 77–82.

Wang, S., D. Lo, and J. Lawall (2014). Compositional vector space models for improved bug localization. In *Proceedings of the International Conference on Software Maintenance and Evolution*, pp. 171–180.

Wille, R. (2005). Formal concept analysis as mathematical theory of concepts and concept hierarchies. In *Formal Concept Analysis*, pp. 1–33. Springer.

Zamania, S., S. P. Lee, R. Shokripoura, and J. Anvikb (2014). A noun-based approach to feature location using time-aware term-weighting. *Information and Software Technology 56*(8), 991–1011.

Zhai, C. and J. Lafferty (2004). A study of smoothing methods for language models applied to information retrieval. *ACM Transactions on Information Systems 22*(2), 179–214.

Zhao, W., L. Zhang, Y. Liu, J. Sun, and F. Yang (2004). SNIAFL: Towards a static noninteractive approach to feature location. In *Proceedings of the 26th International Conference on Software Engineering.*, pp. 293–303.

Zhao, W., L. Zhang, Y. Liu, J. Sun, and F. Yang (2006). SNIAFL: Towards a static noninteractive approach to feature location. *ACM Transactions of Software Engineering Methodologies 15*(2), 195–226.

Zhou, J., H. Zhang, and D. Lo (2012). Where should the bugs be fixed? more accurate information retrieval-based bug localization based on bug reports. In *Proceedings of the 34th International Conference on Software Engineering*, pp. 14–24.

Appendix A
EXAMPLE QUERIES

Feature	Query
549	Enumeration datatypes should be represented on the class diagram

Table A.1: Example Title Query for ArgoUML

Number	Query
0	enumer datatyp repres class diagram
1	[identifiers](enumer datatyp repres class diagram)
2	[comments](enumer datatyp repres class diagram)
3	[literals](enumer datatyp repres class diagram)
4–19	weight({1,2,3,4} [identifiers](enumer datatyp repres class diagram) {1,2,3,4} [comments](enumer datatyp repres class diagram))
20–35	weight({1,2,3,4} [identifiers](enumer datatyp repres class diagram) {1,2,3,4} [literals](enumer datatyp repres class diagram))
36–51	weight({1,2,3,4} [comments](enumer datatyp repres class diagram) {1,2,3,4} [literals](enumer datatyp repres class diagram))
52–115	weight({1,2,3,4} [identifiers](enumer datatyp repres class diagram) {1,2,3,4} [comments](enumer datatyp repres class diagram) {1,2,3,4} [literals](enumer datatyp repres class diagram))

Table A.2: Example Output Title Queries for the ICL corpus for ArgoUML and Feature 549. The {} are used to indicate that every combination of the contained values are used.