Abstract

RAMACHANDRAN, LAKSHMI. Automated Assessment of Reviews. (Under the direction of Dr. Edward F. Gehringer.)

Reviews are text-based feedback provided by a reviewer to the author of a submission. Reviews are used not only in education to assess student work, but also in e-commerce applications, to assess the quality of products on sites like Amazon, e-bay etc. Since reviews play a crucial role in providing feedback to people who make assessment decisions (e.g. deciding a student’s grade, choosing to purchase a product) it is important to ensure that reviews are of a good quality. In our work we focus on the study of academic reviews.

A review is considered to be of a good quality if it can help authors identify mistakes in their work and help them learn possible ways of fixing them. Reviews can be evaluated by metareviewing. Metareviewing is the process of reviewing reviews. An automated metareviewing process provides quick and reliable feedback to reviewers on their assessment of authors’ submissions. Timely feedback on reviews may help reviewers correct their assessments and provide more useful and effective feedback to authors.

Our work investigates the use of metrics such as relevance of a review to the submission, content types of a review, a review’s coverage of a submission, tone, quantity and plagiarism to determine the quality of a review. We use natural language processing and machine learning techniques to calculate these metrics.

Relevance helps identify to what extent a review’s content pertains to that of the submission. Relevance metric helps distinguish generic or vague reviews from the useful ones. Relevance of a review to a submission can be determined by identifying semantic and syntactic similarities between them. Our work introduces the use of a word-order graph representation, where vertices, edges and double edges (two contiguous edges) help capture sentence-structure information. Our matching technique exploits contextual similarities to determine relevance across texts. We use a WordNet-based relations metric to identify relatedness. During graph matching single and contiguous edges are compared in same and different orders to identify possible paraphrases involving word order shuffling.

Review content helps identify what type of content a review contains. In this work we focus on three types of review content namely: summative (containing a summary or praise), problem detection (identifying problems in the author’s work) and advisory (providing suggestions for improvement). A review may contain each of these content types at varying degrees. A graph-based pattern identification technique is used to determine the types of content a review contains. Patterns are extracted from reviews that represent each type of content using a cohesion detection technique. Edges that are most semantically similar to other edges in a graph are selected as patterns. These edge patterns may be used to identify the extent to which reviews contain each type of content.
Reviews must be thorough in discussing a submission’s content. At times a review may be based on just one section in a document, say the Introduction. Review coverage is the extent to which a review covers the “important topics” in a document. We study the coverage of a submission by a review using an agglomerative clustering technique to group the submission’s sentences into topic clusters. Topic sentences from these clusters are used to calculate review coverage in terms of the degree of overlap between a review and the submission’s topic sentences.

Review tone helps identify whether a reviewer has used positive or negative words in the review, or has provided an objective assessment of the author’s work. While a positive or an objective assessment may be well received by the author, the use of harsh or offensive words or phrases may disincline the author from using the feedback to fix their work. A review’s tone is determined in terms of its semantic orientation, i.e., the presence or absence of positively or negatively oriented words. Review quantity is the number of unique tokens a review contains. The purpose of this metric is to encourage reviewers to write more feedback, since feedback with specific examples and additional explanation may be more useful to the author.

Plagiarism is an important metric because reviewers who are evaluated on the quality of their reviews may tend to game the automated system to get higher ratings. We look for plagiarism by comparing a review’s text with text from the submission and with text from the Internet to make sure that the reviewer has not copy-pasted text to make the review seem relevant.

Relevance, content and coverage identification approaches have been evaluated on data from Expertiza, a collaborative web-based learning application. Our experimental results indicate that our word order based relevance identification technique succeeds in achieving an $f$-measure of 0.67. In another experiment we found that the pattern-based content type identification approach has an $f$-measure of 0.74, which is higher than the performance of text categorization techniques such as multiclass support vector machines and logistic regression. Our experiment on coverage analysis indicates a high correlation of 0.51 between system-generated and human-provided coverage values. We also report our results from a user study conducted to evaluate the usefulness of an automated review quality assessment system. Participants in the study found relevance to be most important metric in assessing review quality. Participants found the system’s output from metrics such as content type and plagiarism to be most useful in helping them learn about their reviews.
Automated Assessment of Reviews

by

Lakshmi Ramachandran

A dissertation submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Computer Science

Raleigh, North Carolina
2013

APPROVED BY:

_________________________  __________________________
Dr. James Lester            Dr. Kristy Boyer

_________________________  __________________________
Dr. Robin Dodsworth         Dr. Edward F. Gehringer
Chair of Advisory Committee
Acknowledgments

I would like to thank my family and friends for their unending support and encouragement. I would like to express my sincere gratitude and thanks to my advisor Dr. Edward F. Gehringer, who has provided me guidance, valuable advice and encouragement during this research study. I would like to mention a special thanks to the Computer Science department at North Carolina State University for assisting me in my research activities. I would also like to thank the entire staff of the Graduate School at North Carolina State University for guiding me during this dissertation. Finally, I would like to thank my colleagues for providing me with sincere feedback on my research and helping me with the surveys.
# Table of Contents

List of Tables .......................................................................................................................... vi

List of Figures .......................................................................................................................... viii

Chapter 1 Automating Review Assessment ........................................................................... 1
  1.1 Problem .............................................................................................................................. 2
  1.2 Background ......................................................................................................................... 3
  1.3 Approach Overview ........................................................................................................... 4
    1.3.1 Review relevance ......................................................................................................... 4
    1.3.2 Review content ............................................................................................................ 4
    1.3.3 Review tone ................................................................................................................ 5
    1.3.4 Review coverage .......................................................................................................... 7
    1.3.5 Review quantity .......................................................................................................... 7
    1.3.6 Plagiarism .................................................................................................................. 7
  1.4 The Expertiza Project ......................................................................................................... 8
  1.5 Roadmap ............................................................................................................................ 8

Chapter 2 Text Representation and Semantic Relatedness .................................................. 10
  2.1 Graph Representation ....................................................................................................... 10
    2.1.1 Related work .............................................................................................................. 11
    2.1.2 Word-order graph generation .................................................................................... 12
  2.2 Semantic Relatedness ....................................................................................................... 14
    2.2.1 Existing models .......................................................................................................... 15
    2.2.2 WordNet relations-based semantic metric ................................................................. 18
    2.2.3 Experiments .............................................................................................................. 19
  2.3 Contributions ..................................................................................................................... 21
  2.4 Conclusion ........................................................................................................................ 21

Chapter 3 Review Relevance ................................................................................................ 22
  3.1 Related Work .................................................................................................................... 24
  3.2 Lexico-Semantic Graph-based Matching ........................................................................ 25
    3.2.1 Phrase or token matching ......................................................................................... 26
    3.2.2 Context matching ...................................................................................................... 26
    3.2.3 Sentence structure matching .................................................................................... 28
  3.3 Experiment 1: Performance of Our Relevance Identification Approach ....................... 29
    3.3.1 Comparison with a text overlap-based approach ....................................................... 30
  3.4 Experiment 2: Comparison of Word Order Graphs and Dependency Trees .................. 31
  3.5 Experiment 3: Comparison of Different Semantic Relatedness Metrics ....................... 32
    3.5.1 Academic reviews dataset ...................................................................................... 33
    3.5.2 Product reviews from Amazon ............................................................................... 33
  3.6 Experiment 4: Generalizability of Relevance Identification Problem and Approach ........ 34
  3.7 Feedback to Reviewers—Relevance .................................................................................. 35
  3.8 Contributions .................................................................................................................... 36
# Chapter 4 Review Content

4.1 Related Work

4.2 Graph Generation and Review State

4.3 Identification of Semantic Patterns

4.4 Experiments

   4.4.1 Review content type identification

   4.4.2 Goodness of semantic patterns

4.5 Feedback to Reviewers—Content Type

4.6 Contributions

4.7 Conclusion

# Chapter 5 Review Coverage

5.1 Related Work

5.2 Cluster-based Topic Sentences Identification and Coverage Calculation

   5.2.1 Calculating semantic relatedness between sentences

   5.2.2 Grouping sentences into clusters

   5.2.3 Identifying topic sentences

   5.2.4 Measuring a review’s coverage

5.3 Experiments

   5.3.1 Identification of Review coverage

   5.3.2 Evaluation of generated topic representative sentences

5.4 Contributions

5.5 Conclusion

# Chapter 6 A User Study on the Automated Assessment of Reviews

6.1 The Study

   6.1.1 Participants

   6.1.2 Data collection

6.2 User Experience Questionnaire

   6.2.1 Participant background

   6.2.2 Importance of reviews and metareviews

   6.2.3 Importance of metareview metrics

   6.2.4 Usefulness of system’s metareview output

   6.2.5 Other metrics

6.3 Analysis of Data

   6.3.1 Importance of reviews and metareviews

   6.3.2 Importance of the review quality metrics

   6.3.3 Usefulness of system output

   6.3.4 Other metrics

   6.3.5 Usefulness of the overall automated assessment feature

6.4 Threats to Validity

6.5 Contributions

6.6 Conclusion
### Chapter 7: Assessment of Project Reviews

- **7.1 Peer Reviews** .................................................. 79
- **7.2 Project Review Metrics** .................................. 80
- **7.3 Approach to Calculate Project Review Metrics** ..... 81
  - **7.3.1 Determining metareview scores** .................. 82
- **7.4 Evaluation** ..................................................... 82
  - **7.4.1 Data and method** ...................................... 82
  - **7.4.2 Results and analysis** ................................. 83
- **7.5 Contribution** .................................................. 83
- **7.6 Conclusion** .................................................... 84

### Chapter 8: Summary and Future Work

- **8.1 Future Directions** .......................................... 85

### References ......................................................... 88
List of Tables

Table 1.1 Some examples of reviews. ................................................................. 2
Table 2.1 Evaluating the relative ordering of the WordNet match types with different sets of weight values assigned to each relation. ................................. 20
Table 2.2 Comparing WordNet-based semantic relatedness measures. This table contains correlations of each of the measures’ values with human relatedness ratings on the WS353 and RG65 datasets. .............................................. 21
Table 3.1 Comparing accuracy and $f$-measure values of identifying relevance produced by our word order graph and a dependency-tree representation. .......................... 30
Table 3.2 Using different semantic relatedness measures to determine relevance of reviews, with data from Expertiza. ............................................................... 33
Table 3.3 Using different semantic relatedness measures to determine relevance of product reviews from the Amazon dataset. ................................................... 34
Table 3.4 Accuracy of identifying entailment for some tasks of the RTE data set. PP: paraphrase recognition, CD: comparable documents, RC: reading comprehension, IR: information retrieval and QA: question-answer. ......................... 35
Table 3.5 Summary of findings—Relevance ...................................................... 37
Table 4.1 Some examples of reviews belonging to the different review content classes. 39
Table 4.2 Sample edge patterns from each of the content classes. ....................... 44
Table 4.3 Average recall, precision and $f$-measure for each of the different classifiers. 47
Table 4.4 Comparing match produced by our patterns and by Opinosis with human-provided topic representative sentences. ...................................................... 48
Table 4.5 Sample edge patterns from documents in the Opinosis dataset. .......... 50
Table 4.6 Summary of findings—Content ....................................................... 51
Table 5.1 Identifying the correlation between system-generated and human-provided coverage values on review data from Expertiza. ................................. 63
Table 5.2 Comparing ROUGE-1, ROUGE-2 and ROUGE-SU4 results from our system with those from the Opinosis paper. ...................................................... 65
Table 5.3 Summary of findings—Coverage ...................................................... 66
Table 6.1 Detailed set of instructions to help complete the survey ..................... 70
Table 7.1 This rubric assesses the quality of the code based on its implementation, comments provided and ease of understanding. ................................. 78
Table 7.2 The following rubric focuses on the design aspects of the project. ........ 79
Table 7.3 Rubric used for submissions of technical articles or papers. ................ 80
Table 7.4 Frequent and semantically important patterns captured from the set of project reviews and text reviews. ................................................................. 81
Table 7.5 Metrics that suitably represent project reviews. ............................... 82
Table 7.6 Accuracies of predicting metareview scores for project reviews using different sets of metrics.

Table 7.7 Sample project reviews
List of Figures

Figure 2.1 Charniak parser’s output for the text “The author writes like a professional.” 11
Figure 2.2 Dependency tree representations of the text “The study guide does not discuss much of the basics of ethics.” 12
Figure 2.3 Word-order representation of the text “The study guide does not discuss much of the basics of ethics.” 14
Figure 2.4 WordNet tree showing hypernym relations across synsets. 15
Figure 3.1 The figure contains a sample submission, a relevant review with explicit text overlaps, a relevant review which has lexico-semantic matches with the submission’s text and a non-relevant review. 23
Figure 3.2 Displaying the ordering difference captured by the edges of a dependency tree representation and a word-order representation for the text “The paper presented the important concepts.” 25
Figure 3.3 Context matching across two text graphs. Similar dashed lines denote the pairs of edges that are compared across for each type of context match. 26
Figure 3.4 Matching sentence segments across two text graphs. In the voice change match the dashed lines denote the pairs of vertices that are compared when the sentence segments are compared—order of comparison is flipped. 28
Figure 3.5 Example of phrase or token matching and sentence structure match between a review and a submission. 31
Figure 3.6 Comparison between word-order graph and dependency tree representations. 32
Figure 3.7 Output from our review assessment system displaying relevance of reviews. 36
Figure 4.1 Illustration of our approach with examples of summative reviews. 43
Figure 4.2 Performance of differing number of patterns in identifying review content. 47
Figure 4.3 Output from our review assessment system identifying content types for a review. 49
Figure 5.1 Submission with its topic sentences, and three reviews with high, medium and no coverage of the topic sentences. 53
Figure 5.2 Comparing sentences in a sample submission. The edges display the semantic relatedness values between the sentences. 56
Figure 5.3 Two clusters formed by grouping the submission’s sentences. 57
Figure 5.4 Topic sentences are highlighted in red in each of the generated clusters. 61
Figure 5.5 Comparing sample reviews with the selected topic sentences. Review A has more in common with the topic sentences, and so it has a higher coverage than reviews B and C. Some overt similarities have been highlighted. 62
Figure 5.6 Topic-representative sentences generated by our approach and by MEAD for a sample submission. 64
Figure 6.1 Output from the automated feedback assessment feature on Expertiza [1]. 68
Figure 6.2 Participants’ rating of importance of reviews and metareviews. 72
Figure 6.3  Participants’ rating of motivation to use reviews and metareviews to improve the quality of their submission or review respectively. The chart also shows the participants’ estimation of usefulness of the automated metareview metrics’ output.  73
Figure 6.4  Participants’ ratings of the importance of each review quality metric.  74
Figure 6.5  Participants’ ratings of the usefulness of the system’s output for each review quality metric.  75
Figure 6.6  Screenshot of description of the review relevance metric provided to the reviewer.  76
Chapter 1

Automating Review Assessment

In recent years a considerable amount of research has been directed towards developing educational systems that foster collaborative learning. Collaborative learning systems provide an environment for students to interact with other students, exchange ideas, provide feedback and use the feedback to improve their own work. Systems such as SWoRD [2] and Expertiza [3, 1] are web-based collaborative, peer-review systems, which promote team work by getting students to work together to build shared knowledge with an exchange of ideas. The past few years have witnessed a growth in Massive Open Online Courses (MOOCs) such as Coursera and Udacity, as a platform for web-based collaborative learning. These systems also provide an environment for students to give feedback to peers on their work. MOOCs require a scalable means of assessment, and for material that cannot be assessed by multiple-choice tests, peer-review fills the bill. Feedback in the form of text-based reviews helps authors identify mistakes in their work, and learn possible ways of improving it.

The process of providing feedback to peers on their work helps the student learn more about the subject and develop their critical thinking. Rada et al. found that students who evaluated their peers’ work were more likely to improve the quality of their own work than those students who did not provide peer reviews [4]. The peer-review process is also likely to help the student learn to be more responsible.

The classroom peer review process is very much similar to the process of reviewing scientific articles for journals, where students (reviewers) provide reviews and the instructor (editor) identifies the final grade (decision to accept or reject the submitted paper) based on the reviews. Scientific reviewers are likely to have prior experience reviewing articles and a considerable knowledge in the area of the author’s submission (the text under review). Students on the other hand are less likely to have had any prior reviewing experience. They have to be guided to provide high-quality reviews that may be useful to their peers.

*Metareviewing* can be defined as the process of reviewing reviews, i.e., the process of identifying the quality of reviews. Metareviewing is a manual process [3, 5, 6] and just as with any process that is manual, metareviewing too is (a) slow, (b) prone to errors and (c) likely to be inconsistent. Feedback quality
Table 1.1: Some examples of reviews.

<table>
<thead>
<tr>
<th>S No.</th>
<th>Review</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>“The example needs work.”</td>
</tr>
<tr>
<td>2</td>
<td>“The organization is poor.”</td>
</tr>
<tr>
<td>3</td>
<td>“The example code for delegation is taken from one of the references listed at the bottom of the page.”</td>
</tr>
<tr>
<td>4</td>
<td>“I would like to see a better definition/explanation of each technique before getting into the advantages and disadvantages.”</td>
</tr>
</tbody>
</table>

Reviews aid in the decision-making process, whether it is a student’s grade or the decision to accept or reject a paper. It is therefore important to ensure that the reviews are of a good quality. Review comments may be vague or unjustified. The first two comments in Table 1.1 are generic and do not refer to a specific object in the author’s submission. For instance, what type of “work” does the “example” need? Or, why is the “organization” poor? These reviews are ambiguous, and need to be supported with more information. Reviews must provide detailed information, point out problems in the authors’ work or provide suggestions for improvement (similar to the last two comments in Table 1.1). Such a review would help authors understand where their work is lacking.

Since review comments contain unstructured text, it is important to identify metrics that suitably represent the features of a review. Some of the textual features of a review include its relevance to the submission, content, coverage, tone, quantity of feedback provided and plagiarism [9].

The assessment of reviews is an important problem in education, as well as science and human resources, and is therefore worthy of serious attention.
1.2 Background

An earlier approach to manually assessing the quality of peer reviews involved the creation and use of a Review Quality Instrument (RQI) [10]. Van Rooyen et al. use the RQI to check whether the reviewer discusses the following: (1) importance of the research question, (2) originality, (3) strengths and weaknesses, (4) presentation and interpretation of results. In addition, the RQI also checks whether a review was constructive, and if the reviewer had substantiated their claims. We incorporate some of these metrics in our approach, e.g. detecting constructiveness in reviews (based on its content), checking whether reviewers substantiated their claims (by identifying relevance to the author’s submission), to automatically assess review quality.

Nelson and Schunn studied feedback features that help authors understand and use reviews [11]. They found that features such as problem localization and solution suggestion helped authors understand feedback. These are some of the types of content we look for during review content identification.

Kuhne et al. use authors’ ratings of reviews to measure the quality of peer reviews [5]. They found that authors are contented with reviewers who appear to have made an effort to understand their work. This finding is useful to our automatic review quality assessment system, which assesses reviews based on the usefulness of its content. Our system also detects the relevance of reviews, which may be indicative of the effort made by a reviewer to understand and provide specific feedback.

Xiong et al. look for problems identified by reviewers in the author’s work, in peer reviews from the SWoRD system [12]. They use a bag-of-words exact-match approach to detect problem localization features. They use a shallow semantic match approach, which uses counts of nouns, verbs etc. in the text as features. Their approach does not incorporate relevance identification nor does it identify content type. Cho uses machine-classification techniques such as naïve Bayes, support vector machines (SVM) and decision trees to classify review comments [13]. Cho manually breaks down every peer comment into idea units, which are then coded as praise, criticism, problem detection, solution suggestion, summary or off-task comment.

Review quality identification has been applied to study e-commerce reviews from Amazon, eBay etc. Product reviews’ helpfulness is determined based on how useful other users of a system think the reviews are. Zhang et al. determine review helpfulness based on ratings given by voters to a review [14]. Moghaddam et al.’s work on review helpfulness only takes raters’ information into consideration [15]. The authors suggest that the quality of a review depends on the type of reviewer, and hence they factor reviewer information into review helpfulness identification. They do not consider review content information while determining helpfulness. We do not take into consideration how other reviewers rank a certain review. Our approach aims to identify review quality based purely on the textual content of the review.

Some other approaches used to study the usefulness of reviews are those proposed by Turney [16], Dalvi [17] and Titov [18]. Turney uses semantic orientation (positive or negative) to determine whether
a review can be classified as recommended or not recommended. Turney’s approach to differentiate positive from negative reviews involves identifying similarity between phrases containing adverbs and adjectives and terms “excellent” and “poor” respectively. Turney uses semantic orientation to recommend products or movies. We also use semantic orientation, referred to as tone, to identify the degree of sensitivity (in terms of positive or negative words) with which reviewers conveyed their criticism.

Lim et al. identify reviewers who target e-commerce products and applications and generate spam reviews [19]. The problem of spamming may be analogous to the problem of copy-pasting text in order to game an automated assessment system into giving reviewers high scores on their reviews. Therefore, we introduce a metric to detect plagiarized reviews.

Some research works discuss metrics that are important in review quality identification, and some that apply shallow approaches to determine quality. However, there is no work that takes factors such as relevance, content type, coverage, tone, quantity and plagiarism into consideration while determining review quality. Our aim is to provide a suitable review assessment model that can be used to assess peer reviews as well as reviews in other application domains.

1.3 Approach Overview

Several factors help determine the quality of formative feedback. For example, the quality of feedback may depend on whether the reviewer was successful in identifying a problem in the author’s submission, or whether the reviewer provides the author suggestions or pointers to fix a problem. In order to assess quality, reviews have to be represented using metrics that capture their most important features. In general a good review contains: (1) coherent and well-formed sentences, which can be easily comprehended by the author, and (2) a sufficient amount of feedback.

In this section we discuss the metrics we use to assess reviews.

1.3.1 Review relevance

A review could be of a superior quality both semantically and syntactically, when compared to other high-quality reviews. However reviewers may tend to provide generic feedback. Reviews must be checked to ensure that they are written for the right submission. Thus a review’s relevance to a submission is identified to ensure that the reviewer has correctly paraphrased concepts of the submission, and has provided justifications for any criticisms.

1.3.2 Review content

Depending on the purpose for which a review is written, it could be classified as being either indicative or evaluative or a combination of both [20]. *Indicative* reviews contain brief summaries of the work
under evaluation, while evaluative reviews criticize the author’s work and may include suggestions for improvement.

Content of a review identifies the type of feedback provided by the reviewer. We identify content type of reviews based on whether they contain praise or a summary of the author’s work, are identifying problems or are suggesting possible improvements to the work. We look for the following types of content in a review:

- **Summation:** Summative reviews contain either a positive or a neutral assessment of the author’s submission. These reviews tend to be summaries of the author’s work. *Example:* “I guess a good study has been done on the tools as the content looks very good in terms of understanding and originality. Posting reads well and appears to be largely original with appropriate citation of other sources.” The reviewer praises the author’s work, and does not point out any problems in the work nor offer suggestions for improvement.

- **Problem detection:** Reviews in this category are critical of the author’s submission and point out problems in the submission. *Example:* “There are few references used and there are sections of text quoted that appear to come from a multitude of web sites.” The reviewer identifies the lack of references and also a possible case of plagiarism in the author’s work. Problem detection reviews only find problematic instances in the author’s work and do not offer any suggestions to improve the work.

- **Advisory:** Reviews that offer the author suggestions on ways of improving their work fall into this category. *Example:* “Although the article makes use of inline citations which is a plus, there are only a few references. Additional references could help support the content and potentially provide the examples needed.” Advisory reviews display an understanding of the author’s work, and explain the need for adding “more reference”. Also, advisory reviewers indicate that the reviewer has taken the effort to provide the author with constructive feedback.

Different types of review content have different degrees of usefulness. For instance summative reviews provide only summaries of the author’s work and are less useful to the author, whereas reviews that identify problems in the author’s work or provide suggestions can be used by authors to improve their work, and are hence considered more important.

### 1.3.3 Review tone

Tone refers to the semantic orientation of a text. Semantic orientation depends on the reviewer’s choice of words and the presentation of the review. Tone of a review is important because while providing negative criticism reviewers might unknowingly use words or text that might offend the authors. Therefore we use tone information to help guide reviewers while writing reviews. A review can have one
of three types of tones—positive, negative or neutral. We use positive and negative indicators from an opinion lexicon provided by Liu et al. [21] to determine the semantic orientation of a review. Semantic orientation or tone of the text can be classified as follows:

- **Positive**: A review is said to have a positive tone if it predominantly contains positive feedback, i.e., it uses words or phrases that have a positive semantic orientation. *Example*: “The page is very well-organized and the information under corresponding titles is complete and accurate.” Adjectives such as “well organized”, “complete” and “accurate” are indicators of a positive semantic orientation.

- **Negative**: This category contains reviews that predominantly contain words or phrases that have a negative semantic orientation. Reviews that provide negative criticism to the author’s work fall under this category, since while providing negative remarks reviewers tend to use language or words that are likely to offend the authors. Such reviews could be morphed or written in a way that is less offensive to the author of a submission. *Example*: “The approach is trivial, and the paper has been formatted very poorly.” The given example contains negatively oriented phrases “trivial” and “very poorly”. The author could consider such a review to be rude. One of the ways in which this review could be re-phrased to convey the message politely is—“The approach needs improvement. In its present form it does not appear to be conveying any new information. The paper could have been formatted better.”

- **Neutral**: Reviews that do not contain either positively or negatively oriented words or phrases, or contain an equal amount of both are considered to be neutral. *Example*: “The organization looks good overall. But lots of IDEs are mentioned in the first part and only a few of them are compared with each other. I did not understand the reason for that.” This review contains both positively and negatively oriented segments, i.e., “The organization looks good overall” is positively oriented, while “I did not understand the reason for that.” is negatively oriented. The positive and negatively oriented words when taken together give this review a neutral orientation.

In case of metrics content and tone, a single review is likely to belong to multiple categories, which are not easily distinguishable due to the nature of the text. For instance consider the review, “Examples provided are good; a few other block structured languages could have been talked about with some examples as that would have been pretty useful to give a broader pool of languages that are block structured.” While identifying content type, we see that the first part of the review, “Examples provided are good” praises the submission, while the remaining part of the review provides advice to the author. Our technique identifies the amount of each type of content (on a scale of 0–1) a review contains. Similarly in the case of tone, we identify the degree of positive, negative or neutral orientation of each review.
1.3.4 Review coverage

Reviews of submissions such as technical articles, documents, must be thorough in discussing its content. At times a review may be based on just one section in a document, say the Introduction, and may provide no feedback on any of the other sections in the document. We would like reviewers to provide feedback on as many (if not all) sections of a submission. Review coverage identifies the extent to which a review covers the “important topics” in a document. Coverage of a review helps ensure the completeness of a review with respect to the submission.

1.3.5 Review quantity

Text quantity is important in determining review quality since a good review provides the author with sufficient feedback. We plan on using this metric to indicate to reviewers how much feedback they have provided in comparison to the average review quantity (from other reviewers of the system). This may motivate reviewers to provide more feedback to the authors. We identify quantity by taking a count of all the unique tokens in a piece of review. For instance, consider the following review, “The article clearly describes its intentions. I felt that section 3 could have been elaborated a little more.” The number of unique tokens in this review is 15 (excluding articles and pronouns).

1.3.6 Plagiarism

In an automated assessment system we might encounter reviewers who may copy-paste review responses or copy text from the submission or the Internet to make their reviews appear relevant and lengthy. Therefore we include an additional metric that detects plagiarism.

Reviewers do tend to refer to content in the author’s submission in their reviews. Content taken from the author’s submission or from some external source (Internet) should be placed within quotes in the review. If reviewers copy text from the author’s submission and fail to place it within quotes (knowingly or unknowingly) it is considered to be plagiarism.

Each of the review quality metrics is calculated independently, and then integrated into a complete review-quality assessment system. Reviewers are given feedback on each of the listed dimensions/metrics, so that they get a complete picture of the quality of their review. Our aim is to develop a review-quality assessment system that would have a significant impact on students, instructors and teaching assistants—peer reviewers.
1.4 The Expertiza Project

Expertiza is a collaborative web-based learning application that helps students work together on projects and critique each other’s work using peer reviews [3, 22, 1]. Expertiza frequently hosts courses related to software engineering. Reviews completed using Expertiza provide us with the data for analyzing the performance of our review quality metrics.

When students submit their work (a link to a page they created or edited online, or a document), other students are assigned to review their work. The review process is double-blinded, i.e., the author and reviewer information is anonymized to avoid recognition and possible collusion. Rubrics are provided to guide reviewers in the reviewing process. Student reviewers provide text and numeric feedback to authors. Authors use these reviews to update and improve their work.

Reviews submitted by students are manually metareviewed to ensure that the reviews are useful. Our aim is the automation of the metareview process—to provide instantaneous feedback to reviewers on the quality of their reviews.

1.5 Roadmap

In this thesis we address the problem of automatically assessing reviews. The following is the structure of the thesis:

Chapter 2 discusses the text-representation and semantic-matching technique that aid solving the sub-problems of the review assessment task. The chapter describes our graph-based text representation, and includes the approach of creating word-order graphs. The chapter also discusses the semantic-relatedness metric our approach uses for graph matching.

Chapter 3 discusses the sub-problem of identifying the degree of relevance between a review and the author’s submission. In this chapter we describe the use of a graph-based text matching approach to determine relevance. This chapter discusses related work in the area of paraphrase identification and text summarization. It also discusses experiments that evaluate our review relevance identification approach.

Chapter 4 discusses the sub-problem of identifying the type of content a review contains. This chapter describes the use of graphs to identify semantic patterns that suitably represent the different types of review content. We use a cohesion-based pattern-identification technique to capture the meaning of a class of reviews. The chapter also discusses evaluations of our content identification approach on reviews from peer-reviewing systems such as Expertiza and Scaffolded Writing and Rewriting in the Discipline (SWoRD).

Chapter 5 discusses the problem of review-coverage identification. The chapter describes the use of a novel agglomerative clustering technique to group a submission’s sentences into topic clusters. We identify topic sentences from these clusters, and calculate review coverage in terms of the overlaps between the review and the submission’s topic sentences. We describe the evaluation of our coverage
identification approach on peer-review data from Expertiza.

**Chapter 6** discusses the user study we conducted to evaluate our automated review quality assessment system.

**Chapter 7** discusses the evaluation of project reviews using review quality metrics.

**Chapter 8** concludes the thesis with a summary of the work and directions for future research.
Chapter 2

Text Representation and Semantic Relatedness

Several applications in natural language processing and related areas involve comparison of similar-meaninged texts. An approach that relies only on a simple lexical match across compared texts may not be effective at classifying or clustering semantically related texts [23].

Identifying a review quality metric such as relevance may involve looking for lexico-semantic matches between review sentences and submission sentences. A review’s content type may be identified with the help of the most representative patterns for each content type. Our aim is to use lexical and syntactic features to identify relations within and across texts. We therefore need a representation that captures the syntax or order of tokens in a text. Hence we use a word order graph. Word order graphs are suited for identifying lexical and voice changes, which are common among paraphrased text. In this chapter we discuss the use of a meaning-based similarity metric to determine relatedness across two documents. We identify semantic relatedness between texts using a WordNet-based metric [24].

The word-order graph representation and the relations-based semantic metric we introduce in this chapter are used to determine the value of review quality metrics such as relevance, content type and coverage, discussed in Chapters 3, 4 and 5 respectively.

2.1 Graph Representation

Conventional text representation techniques include the vector-space model\(^1\), where every value in the vector represents a token in the text. Vectors capture information such as presence or absence of a token, or the number of occurrences of a token in a document. This is a bag-of-words representation, which does not capture ordering or syntax information.

\(^1\)Vector space model: http://en.wikipedia.org/wiki/Vector_space_model
Output from parsers such as the Charniak parser [25], Stanford NLP parser [26] contain syntactic parse trees, which contain labels such as S, NP or PP\(^2\). Figure 2.1 depicts the output from the Charniak parser for a piece of text. Output from a parser does not contain additional grammatical information [27] and may not be suitable for a lexico-semantic text matching.

The output from parsers may be used to generate dependency trees. Vertices in a dependency tree represent words and edges capture the asymmetric dependency relationships between a head and its modifier (modifier \(\rightarrow\) head). Figure 2.2 contains a dependency tree representation [28] for the text “The study guide does not discuss much of the basics of ethics.” We see that every token in the text is a vertex in the tree and edges depict governance relations. For example, “does” is the root of this sentence, and the edge between “guide” and “does” signifies a subject relationship represented by SBJ.

Dependency trees succeed in capturing only governance information. They do not capture ordering information. In this chapter we propose the use of a graph representation for texts that extends the dependency-tree based representation to capture word-ordering information.

### 2.1.1 Related work

In this section we discuss some of the commonly used forms of graph-based text representations such as dependency trees, sentence graphs, concept graphs and lexical chains, used for tasks such as textual entailment, pattern identification and summarization.

Haghighi et al. [29] use a dependency tree representation to determine text entailment. They use node and edge subsumption metrics to determine relationships between texts. Lin and Pantel [30] use dependency trees to identify inference rules in a piece of text. These rules represent associations between a pair of paths in a dependency tree. Dependency trees that they use capture relationships between words and their modifiers. The trees do not maintain word-order information.

Mihalcea [31] use sentence graphs to perform text extraction and summarization. In sentence graphs each sentence in a document is represented as a vertex in the graph, and the weighted edges represent the degree of overlap across content of the sentences. This representation was found to be useful in extracting the most similar sentences, which form the document’s summary.

Sinha and Mihalcea [32] use concept graphs in their work to carry out word-sense disambiguation. Coursey and Mihalcea [33] identify the topic of an input document using concept graphs, and Erkan

---

and Radev [34] perform text summarization using a sentence ranking approach. Barzilay and Elhadad [35] use lexical chains to perform text summarization. Lexical chains establish links across tokens that are semantically related. Concept graphs and lexical chains represent relationships between categories extracted from a knowledge resource such as Wikipedia. These representations do not maintain the original words or ordering information. In the next section we discuss our approach to generate word-order graphs.

2.1.2 Word-order graph generation

Vertices represent noun, verb, adjective or adverbial words or phrases in a text, and edges represent relationships between vertices. Our graph generation includes the following steps:

**Dividing text into segments:** A piece of text may contain multiple text segments. A text segment is a complete grammatical unit. Each segment of a sentence is separated by period (.), semicolon (;), exclamation point (!) or question mark (?). We use the listed set of punctuation to break the text into multiple segments.

**Part-of-speech (POS) tagging:** The text is then tagged with part-of-speech information (NN, DT, VB, RB\(^2\) etc.) We use the Stanford NLP POS tagger to generate the tagged text [26]. This information is useful in determining how to group words into phrases while still maintaining the order. For example, the text “The study guide does not discuss much of the basics of ethics” after POS tagging looks as follows—“The/DT study/NN guide/NN does/VBZ not/RB discuss/VB much/JJ of/IN the/DT basics/NNS of/IN ethics/NNS”\(^2\).

**Vertex and Edge creation:** The vertex and edge creation steps are explained in Algorithm 1. From each sentence segment consecutive subject or noun components (also referred to as a substantive\(^3\)), which include nouns, prepositions, conjunctions and Wh-pronouns are combined to form a subject vertex (Lines 4–8 of Algorithm 1). Consecutive verbs (or modals) are combined to form a verb vertex (Lines 18–23); similarly with adjectives (Lines 33–38) and adverbs. In Figure 2.3 tokens “study” and “guide” are combined to form the vertex “study guide”.

When a subject vertex is created the algorithm looks for the last created verb vertex to form an edge

---

\(^3\)http://dictionary.reference.com/browse/substantive
Algorithm 1: Vertex and Edge Creation Algorithm

Input: Text segment \( s \)
Output: Graph representation of \( s \)

1. for each token \( w \) in segment \( s \) do

   switch \( w \) do
   
   case substantive
   
   if previous vertex was a substantive then
   
   Concatenate \( w \) with previous vertex /* updating vertex */
   
   else
   
   Create a new substantive vertex \( w \) /* creating a new vertex */
   
   end
   
   if an adjective \( j \) was the previous vertex then
   
   if a predicative adjective edge exists between \( j \) and previous substantive \( n' \) then
   
   Delete edge between \( j \) and \( n' \)
   
   end
   
   Add an attributive adjective edge between \( j \) and \( w \)
   
   end
   
   if verb \( v \) exists and verb-substantive edge \((v, w)\) does not exist then
   
   Add an edge between \( v \) and \( w \) /* predicate creation */
   
   end
   
   case verb or modal
   
   if previous vertex was a verb then
   
   Concatenate \( w \) with previous vertex /* updating verb */
   
   else
   
   Create a new verb vertex \( w \)
   
   end
   
   if an adverb \( r \) was the previous vertex then
   
   if edge exists between \( r \) and previous verb \( v' \) then
   
   Delete edge between \( r \) and \( v' \);
   
   end
   
   Add an attributive adverb edge between \( r \) and \( w \)
   
   end
   
   if substantive \( n \) exists and substantive-verb edge \((n, w)\) does not exist then
   
   Add an edge between \( n \) and \( w \) /* substantive–verb edge creation */
   
   end
   
   case adjective
   
   if previous vertex was an adjective then
   
   Concatenate \( w \) with previous vertex
   
   else
   
   Create a new adjective vertex \( w \)
   
   end
   
   if substantive \( n \) exists and predicative adjective edge \((n, w)\) does not exist then
   
   Add a predicative adjective edge between \( n \) and \( w \)
   
   end
   
   case adverb
   
   /* Similar to adjective creation. */
   
   endsw
   
   endsw
   
end
between the two (Lines 15–17), i.e., create the predicate. When a verb vertex is found, the algorithm looks for the latest subject vertex to create a subject–verb edge (Lines 30–32). Ordering is maintained when an edge is created i.e., if a verb vertex was formed before a subject vertex a verb–subject edge is created, else a subject–verb edge is created.

An adjective or adverb may be used in the attributive (precedes the noun or verb) or predicative (linked to the subject usually through a verb) position. Initially the property is attached to a former substantive or verb vertex i.e. a predicative adjective is created (Lines 39–41). If a substantive or verb vertex is found to immediately follow the adjective, the property is removed from the former vertex and is attached to the new vertex i.e., an attributive adjective or adverb is created (Lines 9–14).

**Labeling graph edges:** Graph edges that are created in the previous step are labeled with dependency (word-modifier) information. After edge creation, we iterate through all the graph edges to determine whether a dependency relation exists between the edge’s vertices. If a dependency relation exists the relation information is added as the edge’s label as shown in Figure 2.3.

Word-order graphs are used to determine the values of review quality metrics—relevance, content type and coverage. The different graph structures—vertices, edges and double edges (two contiguous edges), capture context and ordering information, and aid graph matching. Our graph-based lexico-semantic matching has been explained in greater detail in Chapter 3.

### 2.2 Semantic Relatedness

Current approaches to identifying semantic relatedness use knowledge resources such as Wikipedia [36] and its web-counterpart DBPedia [37]. When comparing two terms, Wikipedia articles or concepts containing the terms to be compared are queried from over 3 million articles (∼3GB in size) [38]. Current techniques are also burdened by time-consuming preprocessing techniques [39]. Although Wikipedia has been shown to perform well at identifying semantic relatedness, its large size and querying time makes it difficult for smaller applications that do not have access to large clusters of parallel computers to adopt it as a knowledge resource.

Expensive preprocessing may also have scalability issues, especially when comparing texts con-
taining hundreds of tokens. Consider the task of review relevance identification (described in detail in Chapter 3). We calculate relevance by identifying lexico-syntactic matches between the review and submission texts. This matching is across documents, which contain more than hundreds of words. Apart from being time-consuming, Wikipedia-based approaches seem to work well when noun entities (topics) are compared.

Relevance identification involves checking for paraphrases [40], i.e., identifying lexical or word-order changes, use of synonyms, definitions or examples of tokens etc. Such a comparison between reviews and submissions would involve checking for relatedness across verb, adjective or adverbial-forms, checking for cases of nominalizations (noun form of adjectives) etc. Using vectors of concepts (as done by Gabrilovich et al. [37]) to represent review and submission texts may not be suitable for the task of relevance identification.

WordNet is a widely used resource for measuring similarity. WordNet is a network of nouns, verbs, adjectives and adverbs, which are grouped into synsets (synonymous words), and linked by lexical relations [24]. WordNet does not perform as well as Wikipedia due to its limitations in terms of the domains it covers, and its lack of real-world knowledge. However WordNet is faster to query and involves no additional preprocessing. WordNet also allows comparison across different word forms.

In this paper we introduce a unique WordNet relations-based matching approach to determine relevance, as an improvement over conventionally used path and distance-based measures [41, 42].

### 2.2.1 Existing models

Semantic relatedness measures can be classified as follows: (1) Information content and path-based measures, (2) Wikipedia (concepts) based relatedness measures and (3) Word-sense disambiguation measures. This section discusses some existing work in each of these categories.
Information content and path-based measures

Semantic relatedness using WordNet taxonomy has been explored by content and path-based measures, which make use of distances between synsets to match words. We introduce a few content and path-based measures in this section, and describe their approach to calculate relatedness.

Information content of a word is calculated in terms of its probability of occurrence in a corpus. Path-based measures determine relatedness between words $w_1$ and $w_2$ by identifying the distance between two nodes $s_1$ and $s_2$ in a taxonomy that are representative of the words. In a taxonomy such as WordNet, $s_1$ and $s_2$ represent the synsets (or synonym sets) of two words $w_1$ and $w_2$ respectively.

Jiang and Conrath [43] use Information Content (IC) to determine relatedness between terms. $lcs$ is the lowest common subsumer of the nodes $s_1$ and $s_2$. The lowest common subsumer is the node in the taxonomy that is general enough to subsume the meanings of both words $w_1$ and $w_2$. For instance in the WordNet taxonomy in Figure 2.4, “activity” is the lowest common subsumer of the word “presentation” and the set “survey, study”. Relatedness is calculated using the formula:

$$Jiang(s_1, s_2) = IC(s_1) + IC(s_2) - 2 \times IC(lcs)$$

Jiang’s formula captures the cumulative information content of the two synsets $s_1$ and $s_2$. Since both synsets include the information content of their lowest common subsumer, it is deducted from the formula in order to only consider each synset’s information content in the relatedness measure.

Leacock and Chodorow [42] determine relatedness in terms of the shortest distance ($length$) between two nodes in a taxonomy. In the following formula\(^5\) $D$ is the maximum depth of the taxonomy. According to Leacock and Chodorow’s formula relatedness is inversely proportional to the length between the two nodes or WordNet synsets. The greater the length, the lower the relatedness value. The length has been normalized using the maximum depth of the taxonomy.

$$Leacock(s_1, s_2) = -\log \left( \frac{length}{2 \times D} \right)$$

Wu and Palmer [41] determine relatedness between tokens in terms of the depth of their least common subsumer and the depth of their nodes in the taxonomy.

$$Wu(s_1, s_2) = \frac{2 \times depth(lcs)}{(depth(s_1) + depth(s_2))}$$

Later on in this chapter we compare our relations-based metric with each of these content and path-based measures.

Wikipedia-based measures

Wikipedia is a widely used knowledge resource that helps determine relatedness across concept tokens. Strube and Ponzetto [36] compare the usefulness of both WordNet (path and information-content based

\(^5\)The value produced by the faction is in the range 0–1, and since log of a value < 1 is –ve, the formula has a –log.
measures) and Wikipedia in identifying semantic similarity across tokens or phrases. They found that WordNet performs well on Miller & Charles [44] and Rubenstein & Goodenough (RG65) [45] datasets, while Wikipedia performed well on the WordSim353 (WS353) [46] dataset. Strube and Ponzetto found that WordNet did not suffer from limited coverage, since only 2 out of the 353 words in the WS353 dataset were absent in WordNet as opposed to 13 absent words in Wikipedia.

Gabrilovich et al. [37] represent text as vectors of concepts extracted from Wikipedia (which are predominantly nouns and named entities). This type of approach may be suited for tasks such as identifying topics among texts that are abounding in named entities (e.g. news articles). Their approach involves the extraction and processing of 3 million URLs containing 70GB of data.

Agirre et al.’s [39] approach to identify relatedness involves the generation of a personalized PageRank graph for every pair of words, which are compared (different edge weights). Generating a graph for every pair of words may not be feasible when identifying similarities between two large documents. Their technique takes about 15 minutes on 2000 cores to generate the corpus used for matching.

**Word-sense disambiguation measures**

Another category of measures makes use of the compared words’ senses to determine relatedness between them. Sense of a word pertains to its meaning. For example, the word “bank” may be used to mean a river’s bank or a financial institution, i.e., these are two different senses of the word “bank”. Sense identification or disambiguation techniques use context-based measures [47]. Lesk identifies overlaps between the definition of a word whose sense is to be determined, and a word whose sense is already known. Greater the overlap, higher the likelihood of both words having the same sense or meaning.

Abhishek [48] proposed a word-sense similarity metric, which utilizes the frequency of terms’ senses in a document to identify its similarity with another document. Such an approach ensures that the terms are used in the same sense in both documents. Similarity between two tokens $t_1$ and $t_2$ is calculated using the following equation.

$$WordSense(t_1, t_2) = \cosine(sense\_vector(t_1), sense\_vector(t_2))$$  \hspace{1cm} (2.1)

**Limitations of existing models**

Wikipedia or DBPedia-based approaches seem to work well in cases where only concepts or topics (nouns) are compared. Datasets on which Wikipedia-based approaches are evaluated: WordSim353 (WS353) [39], Rubenstein and Goodenough (RG65) [45] and Miller and Charles (MC30) [44] contain mainly synonymous nouns. When comparing text documents we encounter words that are of type verbs, adjectives, adverbs etc. For instance while searching DBPedia for information on the term *run*\(^7\),

\(^6\)Biased PageRank, where the transition to from a node $u$ to $v$ depends on the weight on the edges.

\(^7\)DBPedia page for *run*: [http://dbpedia.org/page/Run](http://dbpedia.org/page/Run)
we find instances referring to a baseball run, movies or songs that contain the term run in it. The meaning of run we are looking for “move at a speed faster than a walk, never having both or all the feet on the ground at the same time” is available when we search for the term running\(^8\) (a noun). Therefore, querying using knowledge resources such as Wikipedia involves:

1. Time-consuming preprocessing and data transformation steps.
2. Ambiguity in determining the search term (e.g. use of noun “running” to extract the verb “run”).

In the next section we discuss our relations-based metric to determine semantic match between tokens.

### 2.2.2 WordNet relations-based semantic metric

In order to identify similarity, we use a relations-based metric. Relatedness between two terms \(v\) and \(w\), known as \(\text{match}(v, w)\) is one of those listed below. Each of these types of matches is given a weight value depending on the importance of the match. Similarity matches are assigned values in the range of 0–6, where 6 represents the best match (an exact match) and 0 represents a distinct or non-match.

- \(v\) and \(w\) are **exactly** the same. This match is given a weight value of 6.
- \(v\) and \(w\) are **synonymous**. This match is given a weight of 5.
- \(v\) is a **hypernym** of \(w\) (i.e., \(v\) is more generic than the token \(w\)) or vice versa. Or \(v\) is a **hyponym** of \(w\) (i.e., \(v\) is a more specific form of \(w\)) or vice-versa. This match is given a weight of 4.
- \(v\) is a **meronym** of \(w\) (i.e., \(v\) is a sub-part of \(w\)) or vice versa. Or \(v\) is a **holonym** of \(w\) (i.e., \(v\) contains \(w\) as a sub-part) or vice-versa. For example, “arm” is a meronym of the token “body” and “body” is the holonym of the term “arm”. This type of match is given a weight of 3.
- If \(v\) and \(w\) have **common parents** (excluding generic parents such as “object”, “entity”, “organism” etc.), the normalized distance\(^9\) between the two tokens’ synsets is identified. \((1−\text{distance})\) gives the similarity between the two tokens, which is then scaled in the range of 0 to 6 (the lowest and highest values of our semantic metrics). If the scaled value is > 0, then a common parents match exists between \(v\) and \(w\). This match is given a weight of 2.
- Lesk [47] identifies matches between the definition of a word, whose sense is to be determined and the word, whose sense is already known. A high degree of overlap indicates that the two tokens share the same sense. We use **overlapping definitions** as a metric to determine possible

\(^8\)DBPedia page for running: [http://dbpedia.org/page/Running](http://dbpedia.org/page/Running)

context-based similarity across tokens. For instance, tokens “quantity” and “enough” have overlapping definitions. The word “adequate” can be found in both tokens’ definitions, i.e., quantity’s definition—*an adequate or large amount* and enough’s definition—*an adequate quantity*. We also find overlaps across examples of the words. If two tokens have overlapping definitions or examples, then the match gets a weight value of 1.

- *v* and *w* contain distinct tokens or phrases. Distinct match is given a weight of 0.

We use a combination of the token and its POS information for comparison with another token. This matching ensures that the right type of words is being compared. POS information helps extract the appropriate set of synonyms, hypernyms etc. for a token. For instance, the synonym set generated for verb “run” is [runner, running, operation, operator, operative, operant, functioning, campaign, campaigner, hunt, hunter, hunting, race, racer, racing], and for noun “run” is [tally, test, stream, campaign, ravel, ladder, runny, discharge]. Thus, the meaning of the generated synonyms differs with the POS information. Therefore, our metric determines relatedness using the POS values of the compared tokens.

### 2.2.3 Experiments

In this section we conduct some preliminary evaluations to study the ordering of the relations (e.g. exact match, synonym match etc.) in our semantic metric. We demonstrate that the relative ordering of the weight values plays an important role in identifying matches, i.e., exact matches are more important than synonyms, which are more important than hypernyms or hyponyms and so on. We also evaluate the performance of our metric by comparing it with existing WordNet-based content and path measures (listed in Section 2.2.1). We use two different datasets WS353 and RG65 for our evaluation.

#### Dataset

The WS353 similarity and relatedness collection contains 353 pairs of words. 13 annotators evaluated 153 word pairs, and 16 annotators evaluated 200 word pairs on a scale of 0 to 10. We use the mean similarity scores for our evaluation purposes. The RG65 dataset consists of 65 pairs of synonymous nouns. The WS353 and RG65 datasets contain the average relatedness ratings provided by human annotators. We compare the similarity values produced by our WordNet metric with the scores from the annotators. We have not calculated the mean values or made any other changes to the existing datasets.

#### Evaluation of relative ordering

In the case of our WordNet metric, the weight values that are chosen represent relative order of the WordNet relations and do not follow an interval or ratio scale. We conduct experiments with different weight values, while keeping the relative order intact. We do this to demonstrate that the order of the relations helps capture the relatedness between tokens, and this is independent of the weight values.
assigned to the matches. We use exponential values (1, 2, 4, 8, 16, 32, 64) and random values (2, 10, 23, 40, 47, 50, 52) as weights for the WordNet metrics.

System-generated values just like the human relatedness ratings are ordinal i.e., they exhibit a relative ranking and cannot be interpreted on an interval or ratio scale. Since Spearman\(^{10}\) correlation measure is more suited to determining correlations between ordinal data, we use it to correlate the system-generated values with human-provided relatedness values. Results listed in Table 2.1 indicate that the correlations achieved by the exponential and random weights are comparable to those achieved by the [0–6] weight scale. Thus we see that irrespective of the values, the relative ordering of the WordNet metrics used by our approach produces high correlations. Therefore the values of the weights are not as important as their relative order.

**Comparison with other relatedness metrics**

In order to ensure that our WordNet-based relations metric is a suitable semantic relatedness metric, we compare our approach’s results with other measures, which use only WordNet and no external knowledge resources or corpuses. We use WordNet to implement the information content, path-based and word-sense disambiguation measures.

Table 2.2 lists the Spearman correlation values achieved by each of the similarity metrics with the human annotations, for the above-mentioned datasets. Values 0.43, 0.47 and 0.83 achieved by our metric on the WS353-Full, -Test and RG65 datasets indicate that our metric has a positive correlation with human annotations.

It must be noted here that our approach produces a correlation of 0.83 on the RG65 dataset, which is greater than the correlation of 0.82 achieved by a Wikipedia-based approach proposed by Gabrilovich and Markovitch [37]. The words used in the WS353 and RG65 datasets include commonly used nouns, and we have shown that our metric has a good correlation with human-provided relatedness values on other types of datasets.

Table 2.2: Comparing WordNet-based semantic relatedness measures. This table contains correlations of each of the measures’ values with human relatedness ratings on the WS353 and RG65 datasets.

<table>
<thead>
<tr>
<th>Similarity Metrics</th>
<th>WS353-Full</th>
<th>WS353-Test</th>
<th>RG65</th>
</tr>
</thead>
<tbody>
<tr>
<td>Our WordNet measure</td>
<td>0.43</td>
<td>0.47</td>
<td>0.83</td>
</tr>
<tr>
<td>Wu [36]</td>
<td>0.30</td>
<td>0.28</td>
<td>0.82</td>
</tr>
<tr>
<td>Leacock [36]</td>
<td>0.34</td>
<td>0.35</td>
<td>0.86</td>
</tr>
<tr>
<td>Jiang</td>
<td>0.31</td>
<td>0.36</td>
<td>0.54</td>
</tr>
<tr>
<td>WordSense</td>
<td>0.28</td>
<td>0.31</td>
<td>0.56</td>
</tr>
</tbody>
</table>

### 2.3 Contributions

Some of the chief contributions of this work are:

1. A unique graph representation, which captures word order, syntax and role information.
2. A WordNet order-based relatedness metric to capture similarity across tokens.
3. We compare our metric with several WordNet-based relatedness metrics on WS353 and RG65 datasets and show that our relatedness metric produces higher correlations than the other measures.

### 2.4 Conclusion

In this chapter we introduce a graph-based text representation that captures word-ordering information. We use this word-order graph representation to calculate some of the review quality metrics such as relevance, review content type and review coverage, which involve a lexico-semantic matching approach (explained in detail in the following chapters). We introduce a unique relations-based semantic relatedness measure. We use features such as exact matches, synonymy, hypernymy, hyponymy, holonymy, meronymy, common parents and overlapping definitions to determine matches between texts.

In the next chapter we introduce the review relevance metric.
Chapter 3

Review Relevance

Reviews are text-based feedback that help people make assessment decisions, e.g., for grading students, accepting manuscripts for publication, funding grants or deciding whether to use the reviews to improve the submission. Therefore, the review’s content must be useful to the decision-making party. Kuhne et al. [5] found that authors are contented with reviewers who make an effort to understand their work. Nelson and Schunn [11] found that reviews help authors understand and use feedback effectively, if they locate specific problematic instances in the author’s work, or provide suggestions for improvement.

We manually investigated peer reviews from Expertiza and found that reviewers often tend to provide vague or generic comments. Comments such as, “Yes, it is good” and “It is very well organized” are frequent in the data set. These review comments appear to be generic and they do not reflect the reviewer’s understanding of the author’s work. Authors are less likely to be pleased with generic reviews, especially because these reviews cannot be used to improve the quality of their work.

Consider the comment, “I felt that some of the examples were clichéd.” The reviewer is criticizing the “examples” in the author’s work but does not provide any justification. The review does not reference specific examples that the reviewer finds “clichéd”. Unjustified review comments, which do not identify specific problematic instances in the author’s work, may not be useful to the author.

A relevant review paraphrases the concepts described in a submission, with possible descriptions of problems identified in the author’s work. Our aim is to decide whether a review is relevant to the work it was written for.

**Definition** Let $S$ be the set of sentences in the text under review (the submission) and $R$ be the set of review sentences. Let $s$ and $r$ represent a sentence in the submission and review respectively.

$$\text{relevance}(S, R) = \frac{1}{|R|} \sum_{r \in R} \{\arg\max_{s \in S} (\text{lexicoSemSim}(s, r))\}$$  \hspace{1cm} (3.1)

$\text{lexicoSemSim}(s, r)$ represents the lexico-semantic match between $s$ and $r$. Relevance is the average of the best lexico-semantic matches of review sentences with corresponding submission sentences. Our
The debate on internet radio centers around an initiated fee imposed upon the internet radio stations. Proponents of the fee claim that it is necessary because artists are losing money since their music can be listened to without purchase. This fee, the internet radio stations contend, will drive them out of business. Though many artists see the internet stations as a welcome marketing tool to get their music heard. The radio stations began a campaign that involved a day of silence to raise awareness and get users to write to congress in their support.

**1. Relevant review (text overlaps with the submission):** Start with a general article on Internet radio rather than a listing of stations. The list of stations can follow that.

**2. Relevant review (lexico-semantically related to the submission):** The article should include a discussion featuring the artists’ interests. Arguments imply that the fee would bankrupt the stations.

**3. Non-relevant:** The article does seem to treat differing viewpoints fairly exploring both the negative consequences of hate speech and the negative consequences of censoring it.

Figure 3.1: The figure contains a sample submission, a relevant review with explicit text overlaps, a relevant review which has lexico-semantic matches with the submission’s text and a non-relevant review.

Conventional text-similarity matching approaches look for frequent $n$-gram matches across texts. Papineni et al. [49] compare $n$-grams to identify the degree of similarity between a machine-translated candidate text and a human-provided reference text. Such similarity matching techniques may not be good at identifying the relevance of the above-mentioned types of reviews. Therefore, more sophisticated text-matching techniques and metrics are required to identify relevance.

We propose the use of a graph-based text representation and matching technique to identify relevance between review and submission texts. We use the word-order graph representation and semantic-relatedness metric described in Chapter 2 to identify relevance.

The rest of this chapter is organized as follows: Section 3.1 discusses related work in the areas of graph-based text analysis, text-similarity and paraphrase identification. Section 3.2 discusses our lexico-
semantic graph-based matching approach to identify relevance. Section 3.3 discusses the experiment we conducted to evaluate our approach on review-submission data from Expertiza. Section 3.4 discusses the performance of our word-order graph (discussed in Chapter 2) in comparison to that of a dependency tree representation. Section 3.5 demonstrates the performance of our semantic-relatedness metric (also discussed in Chapter 2). Section 3.6 contains a discussion on the generalizability of our approach for the textual entailment identification task. Section 3.7 describes the format of relevance feedback presented to reviewers. Section 3.8 lists the contributions of this work, and Section 3.9 concludes the chapter with a summary.

3.1 Related Work

There is little previous work in the area of identifying relevance between a review and a submission. Our approach is one of the first to use relevance as a metric to study review helpfulness. Due to the lack of sufficient work in relevance identification, we discuss existing research in the related application domains of text paraphrasing and summarization. Work on paraphrasing is close to our problem, since paraphrased texts may be lexico-semantically similar in meaning to the text being paraphrased. We also discuss some related work in the area of graph-based text matching.

Text paraphrasing is close to our approach since we aim to identify paraphrases in reviews, and since paraphrases and summaries tend to involve changes to word order while maintaining the original meaning of the text. Boonthum [50] lists six patterns followed commonly while paraphrasing a piece of text. Some of the patterns include the use of synonyms, change in voice, change in sentence structure and providing definitions of words. According to Liu et al. [51] a paraphrase preserves the original meaning of the text but may also contain some syntactic changes.

Kauchak et al. [52] suggest an automated technique to create paraphrases for human and machine-translated text pairs by substituting words in machine translated texts by their synonyms. They define paraphrases primarily in terms of synonyms of individual tokens. Although their token-substitution technique takes into consideration some context information they do not consider other paraphrasing techniques, which involve word or phrase shuffling across the length of the text. Qiu et al. [53] use a dissimilarity-based paraphrase identification technique. They use the significance or insignificance of unpaired sentence tuples to identify paraphrases between a pair of texts.

As noted earlier text matching with possible changes in word order is essential for a task such as relevance identification. However, existing graph representations and matching techniques are not suited to performing word-order based comparisons across texts. Mani and Bloedorn suggest the use of a graph search and matching approach for multi-document summarization [54]. The graph matching approach used by Mani and Bloedorn focuses on concept or topics-based matching. The graph captures adjacency relations between concepts or topics. Their graph representation does not capture ordering information—required for tasks involving identification of lexical-order changes.
The document index graph (DIG) used by Hammouda and Kamel, capture phrases of a document [55]. Although the DIG captures order of words within a phrase, it does not capture the order of phrases within a document. As a result this representation does not capture complete sentence-structure information, which may be necessary to decide whether a review contains sentence-structure changes.

Haghighi et al. [29] use dependency trees to determine text entailment. They determine text entailment with the help of node and path substitutions across directed graph representations of the hypothesis and the text. They determine the cost of substituting vertices and paths in the hypothesis with those found in the text. Match across the graphs is determined by comparing vertices as well as edges. They do not perform comparisons across edges in different orders to account for word-order shuffling during paraphrasing. Dinu and Lapata [56] use dependency trees to identify similarity between patterns using context information coded as latent variables in the words’ vector representations.

Ham et al. construct phrase nets using regular expressions [57]. Phrase nets are constructed for specific relations between tokens e.g. “X at Y” may indicate location of object X. Phrase nets are used as a tool to determine relations between objects in literary texts.

Although there exist independent research works that discuss graph-based summarization and paraphrasing techniques, they use only content overlap or synonym matches to determine paraphrases. They do not consider context during text comparison. Our approach uses both semantic and syntactic features during relevance identification, and this helps determine any syntactic similarities or dissimilarities that exist between texts.

3.2 Lexico-Semantic Graph-based Matching

Paraphrasing is the process by which an idea is restated by the paraphraser, thus displaying an understanding of the text [50]. While paraphrasing it is common for paraphrasers to restructure their sentences. Therefore we perform paraphrase detection by matching graph vertices and edges (edges maintain word order information).

Graph matching between the review and submission texts helps us identify whether the review ref-
Figure 3.3: Context matching across two text graphs. Similar dashed lines denote the pairs of edges that are compared across for each type of context match.

errors specific concepts in the submission (i.e., provides justifications). The degree of match between two graphs depends on the degree of match between their vertices and edges.

3.2.1 Phrase or token matching

In phrase or token matching, vertices containing phrases or tokens are compared across graphs. This matching succeeds in capturing semantic relatedness between single or compound words. When vertices “concepts” and “points” (in Figure 3.3a) are compared using WordNet, a common parents match is identified. This match would have been missed when using only an exact or synonym match.

\[
Phrase(S, R) = \frac{1}{|V_r|} \sum_{\forall r(v) \in V_r, \forall s(v) \in V_s} \text{argmax}\{\text{match}(s(v), r(v))\}
\] (3.2)

An overall phrase match is determined by taking the average of the best match that every review phrase has with a corresponding submission phrase. Similarity between two vertices is calculated as the average of matches between their constituent words or phrases. Match could be one of the WordNet relations metrics listed in Chapter 2 Section 2.2.2. In Equation 3.2, \(r(v)\) and \(s(v)\) refer to review and submission vertices respectively, and \(V_r\) and \(V_s\) are the set of vertices in a review and a submission respectively.

3.2.2 Context matching

Context matching compares edges with same and different syntax, and edges of different types across two text graphs. We refer to the match as context matching since contiguous phrases, which capture additional context information, are chosen from a graph for comparison with those in another. Edge labels capture grammatical relations, and play an important role in matching. When edges of the same syntax are compared, their labels are compared too. Some of the context-based matches include:
• **Ordered match**: Ordered match preserves the order of phrases in a text. We compare same-type edges\(^1\) with the same vertex order. Relatedness between edges is the average of the vertex matches. Edge labels are compared in ordered matching, and the match value is halved if the edge labels are different. Edge labels have a high weight in the comparison, and so the average match value decreases when no edge match is identified.

Figure 3.3a shows the comparison of single edges from two review graphs. A match is identified between edges “important–concepts” and “necessary–points”, because they capture the noun-modifier relationship (NMOD), and because a common parents relation exists between tokens “concepts” and “points”.

• **Lexical change**: Lexical match flips the order of comparison, e.g., we compare subject–verb with verb–object edges or vice versa. The match identifies paraphrases, which contain lexical changes. Figure 3.3b depicts lexical-change match. When comparing edge “paper–presented” with edge “included–points”, we compare vertex “paper” with “points” and “presented” with “included”. A match is found between tokens “paper” and “points” causing the edge pair to get a match value > 0.

• **Nominalization match**: The match identifies noun nominalizations—nouns formed from verbs or adjectives (e.g. abstract → abstraction, ambiguous → ambiguity). We compare vertices of different types, e.g., the subject and verb vertices or the subject and adjective vertices. This match also captures relations between nouns and their adjective forms (e.g. ethics → ethical), and nouns and their verb forms (e.g. confusion → to confuse). When we compare the edge “paper–presented” with edge “presentation–included”, we compare “paper” with “included” and “presented” with “presentation”. Token “presentation” is the nominalization of “presented”, as a result of which a match is identified between the two edges.

\[
Context(S, R) = \frac{1}{3|E_r|} \left( \sum_{r(e) \in E_r} \sum_{s(e) \in E_s} \arg\max \{\text{match}_{ord}(s(e), r(e))\} + \sum_{r(e) \in E_r} \sum_{s(e) \in E_s} \arg\max \{\text{match}_{lex}(s(e), r(e))\} + \sum_{r(e) \in E_r} \sum_{s(e) \in E_s} \arg\max \{\text{match}_{nom}(s(e), r(e))\} \right) 
\]

In Equation 3.3, \(r(e)\) and \(s(e)\) refer to review and submission edges. The formula calculates the average for each of the above three types of matches \(\text{match}_{ord}, \text{match}_{lex}\) and \(\text{match}_{nom}\). \(E_r\) and \(E_s\) represent the sets of review and submission edges. \(\text{match}_{ord}, \text{match}_{lex}\) and \(\text{match}_{nom}\) are calculated as the average of the best ordered, lexical or nominalization matches that each of the review edges have with corresponding submission edges.

---

\(^1\) Same-type edges are edges with same types of vertices.
3.2.3 Sentence structure matching

Sentence structure matching compares double edges (two contiguous edges\(^2\)), which constitute a complete segment (e.g. subject–verb–object), across graphs. In this work we consider only single and double edges, and not more contiguous edges (triple edges etc.), for text matching. The matching captures similarity across segments and it captures voice changes. Relatedness between double edges is the average of the vertex matches. Some sentence structure matches are:

- **Ordered match**: Double edges capture more word order than single edges, hence this matching captures more context. In Figure 3.4a double edges “paper–presented–concepts” and “presentation–included–points” are compared. Vertices “paper”, “presented” and “concepts” are compared with vertices “presentation”, “included” and “points” respectively.

- **Voice change**: Voice match captures word or phrase shuffling. Change of voice from active to passive, or vice versa is common with paraphrased text. Vertices of the same type are compared across double edges. However, the order of comparison is flipped. Consider the comparison between active and passive texts “The author presented the important concepts.” and “Necessary points were explained by the author.” in Figure 3.4b. We compare “author” and “author” (exact match), “presented” and “were explained” (synonym match), and “concepts” and “points” (common parents match). This results in a cumulative voice match value of 4\(^3\). Only a voice-change match succeeds in capturing such a relationship across the length of a sentence segment.

---

\(^2\)Two consecutive edges sharing a common vertex.

\(^3\)Average of the vertex match values—6 for exact match, 5 for synonym match, 2 for common parents match. Edge labels are not compared since the order of comparison of the vertices is flipped.
\[
SentStruct(S, R) = \frac{1}{2|T_r|} \left( \sum_{r(t) \in T_r} \arg\max_{s(t) \in T_s} \{match_{ord}(s(t), r(t))\} + \sum_{r(t) \in T_r} \arg\max_{s(t) \in T_s} \{match_{voice}(s(t), r(t))\} \right)
\] (3.4)

The cumulative sentence structure match in Equation 3.4 calculates the average of \(match_{ord}\) and \(match_{voice}\) matches. \(r(t)\) and \(s(t)\) refer to double edges, and \(T_r\) and \(T_s\) are the number of double edges in the review and submission texts respectively. \(match_{ord}\) and \(match_{voice}\) are the averages of the best ordered and voice change matches that a review’s double edges have with corresponding double edges from the submission.

The formula for relevance in Equation 3.1 can be re-written using the lexico-semantic relatedness values calculated for phrase, context and sentence structure matches as follows:

\[
\text{relevance}(S, R) = \frac{1}{3} (\text{Phrase}(S, R) + \text{Context}(S, R) + \text{SentStruct}(S, R))
\] (3.5)

### 3.3 Experiment 1: Performance of Our Relevance Identification Approach

We evaluate the performance of our graph-matching approach in identifying the relevance of a review. We also study the performance of each match: \(\text{Phrase}\), \(\text{Context}\) and \(\text{SentStruct}\) to determine whether the matches add value and help improve the overall performance of our approach.

We select review-submission pairs from assignments completed using Expertiza [1]. For the purpose of evaluation we identify whether a review is relevant or not relevant to a submission. We choose 986 review-submission pairs containing an equal number of relevant and non-relevant reviews for our study from a set of 2211 pairs. The original dataset contained more non-relevant pairs, and the baseline \% of the dataset is 59.79\%. Two annotators labeled 19\% of randomly selected data as \text{relevant} or \text{non-relevant}. We found an 80\% agreement, Kappa value of 0.38, and a Spearman correlation of 0.44 (significance \(p < .0001\)) between the two annotators’ ratings. We use labels from the first annotator for testing due to the high percentage agreement. Relevance thresholds for the different matches are determined based on the averages. We use human annotations to determine the precision, recall and \(f\)-measure of our approach.

Table 3.1 contains the accuracy and \(f\)-measure values of our approach in identifying relevance. A phrase or token matching contains no context. Consider the sample review “I would retitle ‘Teaching, Using and Implementing Ethics’ to ‘Teaching and Using Codes of Ethics’.” This review gets a high phrase match value of 3.3 with a submission discussing different codes of ethics (in Figure 3.1). However, this review is not fully relevant to the content of the submission, since it is suggesting a change in title, and does not discuss the submission’s content. Thus a simple phrase match tends to magnify the degree of relatedness between two texts. Thus although a phrase match is important, the lack of context may inflate relevance.
In the case of context matching, we found that lexical and nominalization matches produce lower match values than an ordered match. This happens because not all reviews contain word-order changes or nominalizations, and flipping the order of matching results in a lower match when compared to that from an ordered match. The lower values decrease the average context matching, thus rendering a review non-relevant to a submission. This phenomenon explains the dip in context matching’s accuracy and \(f\)-measure. We observed a similar trend with sentence-structure matches, where voice match produced a lower value than the ordered match in some of the cases.

The average sentence structure match from Equation 3.4 has an accuracy of 65\%, which is an increase over both phrase and context matches (Table 3.1). Relevance is identified with an accuracy of 66\% and \(f\)-measure of 0.67. The average of the phrase, context and sentence structure matches is higher than that of each of the individual matches. This indicates that the addition of context (ordering) from edges and double edges contributes to an improvement in accuracy and \(f\)-measure values.

Figure 3.5 contains two sample reviews displaying phrase and sentence-structure matching with sentences from a sample submission. The first review has some instances of exact match with the submission and its relevance may be easy to identify. However, a text-overlap match may not be suited to determine the second review’s relevance. Our order-based matching and semantic-relatedness metric help capture the relevance between the second review and the submission.

### 3.3.1 Comparison with a text overlap-based approach

We compare our approach with a simple, overlap-based relevance identification approach. For the overlap-based measure we calculate the average of 1-, 2-, 3- and 4-gram overlaps between review and submission texts to determine relevance. This is a precision-based metric, similar to the one used by Papineni et al. [49].

\[
    relevance_{\text{overlap}} = \frac{\text{overlap}(R, S)}{|R|}
\]  

\(3.6\)

Where \(\text{overlap}\) calculates the number of tokens in review \(R\) that overlap with tokens in submission
Figure 3.5: Example of phrase or token matching and sentence structure match between a review and a submission.

$|R|$ indicates the number of tokens in the review. Stopwords and frequent words are excluded from the numerator and denominator during overlap calculation.

The overlap-based approach classifies a majority 62% of the reviews as non-relevant, and has an $f$-measure value of 0.59. The overlap approach has a high false negative rate i.e., several relevant reviews were predicted as non-relevant (recall of 0.52). Thus, a simple text overlap does not identify relevance as accurately as our approach.

3.4 Experiment 2: Comparison of Word Order Graphs and Dependency Trees

We compare two types of text representations (1) our word-order graph representation explained in Chapter 2 Section 2.1 and (2) a dependency tree based representation [28] on the relevance identification task. We perform relevance identification by only comparing the vertices of the review and submission texts. We show that the use of our graph representation results in faster text matching for relevance identification, due to the presence of fewer vertices.

The difference between the $f$-measure values of relevance calculated for word-order graphs (0.67) and dependency trees (0.62) is statistically significant, with a $p$-value of 0.02 ($p < 0.05$, 95% confidence
interval) for a one-tailed $z$-test.

The dataset used is the same as the one used in Experiment 1, Section 3.3. The dataset contains 986 review-submission pairs extracted from Expertiza with an equal distribution of relevant and non-relevant pairs. Dependency trees perform better for phrase matching (Table 3.1) than for context or sentence-structure matching. Accuracy and $f$-measure of dependency trees are lower for context, sentence structure and the overall relevance matches possibly because edges in dependency trees capture only governance (word→modifier relation) information and not word order.

Charts in Figures 3.6a and 3.6b illustrate the difference in the number of vertices and edges between a word-order graph and a dependency tree representation for a set of 41 randomly selected reviews. Dependency trees contain a larger number of vertices and edges when compared to our word-order graph representation. A similar trend was observed for submission texts. Fewer vertices imply fewer vertex comparisons in the case of word-order graphs, which results in a decrease in the time taken to carry out pairwise comparison between review and submission texts.

We compared the time taken by the different representations to determine relevance and found that in most cases dependency trees take more time than our graph (Figure 3.6c). Our graph produces higher accuracy and $f$-measure and is also faster than a dependency-tree representation.

### 3.5 Experiment 3: Comparison of Different Semantic Relatedness Metrics

In this section we compare the performance of the different types of semantic relatedness metrics in determining review relevance. Our aim with this evaluation is to show the usefulness of our relations-based metric (explained in detail in Chapter 2) in identifying relevance. We compare our approach with Wu & Palmer’s [41] and Leacock & Chodorow’s [42] path-based measures, Jiang & Conrath’s [43] information-content based measure, and a word-sense disambiguation measure [48]. Each of these WordNet-based measures has been described in Chapter 2, Section 2.2.1). We use precision, recall and
Table 3.2: Using different semantic relatedness measures to determine relevance of reviews, with data from Expertiza.

<table>
<thead>
<tr>
<th>Similarity Metrics</th>
<th>Precision</th>
<th>Recall</th>
<th>f-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>WordNet metric [0-6]</td>
<td>0.55</td>
<td>0.65</td>
<td>0.60</td>
</tr>
<tr>
<td>Word-sense metric</td>
<td>0.59</td>
<td>0.54</td>
<td>0.56</td>
</tr>
<tr>
<td>Wu &amp; Palmer</td>
<td>0.50</td>
<td>0.71</td>
<td>0.58</td>
</tr>
<tr>
<td>Leacock &amp; Chodorow</td>
<td>0.40</td>
<td>0.39</td>
<td>0.39</td>
</tr>
<tr>
<td>Jiang &amp; Conrath</td>
<td>0.52</td>
<td>0.52</td>
<td>0.52</td>
</tr>
</tbody>
</table>

f-measure to evaluate the measures’ performance.

3.5.1 Academic reviews dataset

Our experiments are performed on review-submission pairs from two assignments completed with Expertiza [1]. 292 review-submission pairs with an equal distribution of relevant and non-relevant pairs are selected for evaluation. Around 50% of the review-submission data was randomly selected and annotated by two different annotators. The two annotators’ ratings had a 78.3% agreement and a Kappa value of 0.56. Due to the agreement between the annotators, labels from the first annotator are used for evaluation.

The results of using the different relatedness metrics in determining relevance are listed in Table 3.2. We see that our semantic relatedness metric produces the highest f-measure values. In this evaluation we are using only relatedness measures that use WordNet as a knowledge resource. We plan on studying the impact of using additional knowledge resources (e.g. ontologies in the educational domain) in improving the process of relevance identification in the future.

3.5.2 Product reviews from Amazon

In order to study the generalizability of our semantic relatedness metric on other datasets, we study its performance in determining relevance of product reviews. Relevance may be determined by comparing product reviews with a description of the product. In order to test the generalizability of our metric, we evaluate it on product reviews from the Amazon dataset [58].

The Amazon review data contain tags of product features described in the reviews. We use these tags to determine whether a review is relevant to a product’s description. Reviews that discuss product features are tagged as relevant, and those without feature tags are marked as non-relevant. We randomly selected 398 product reviews containing an equal number of relevant and non-relevant cases. We extracted product descriptions from Amazon.

We notice that our metric produces higher f-measure than each of the other semantic relatedness
Table 3.3: Using different semantic relatedness measures to determine relevance of product reviews from the Amazon dataset.

<table>
<thead>
<tr>
<th>Similarity Metrics</th>
<th>Precision</th>
<th>Recall</th>
<th>f-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>WordNet metric [0-6]</td>
<td>0.54</td>
<td>0.62</td>
<td>0.58</td>
</tr>
<tr>
<td>Word-sense metric</td>
<td>0.68</td>
<td>0.38</td>
<td>0.49</td>
</tr>
<tr>
<td>Wu &amp; Palmer</td>
<td>0.53</td>
<td>0.53</td>
<td>0.53</td>
</tr>
<tr>
<td>Leacock &amp; Chodorow</td>
<td>0.50</td>
<td>0.43</td>
<td>0.46</td>
</tr>
<tr>
<td>Jiang &amp; Conrath</td>
<td>0.57</td>
<td>0.58</td>
<td>0.57</td>
</tr>
</tbody>
</table>

metrics. Some other measures come close to the performance of our measure e.g., Jiang & Conrath, which produces an f-measure of 0.57. Although Jiang produces a high f-measure the approach is time consuming. Jiang’s metric takes on average 138 seconds to calculate matches and hence relevance, while approach takes on an average 28 seconds to perform matching. Our approach produces the highest recall value, indicating a high degree of agreement with human-provided relevance values. This experiment shows that a relations-based semantic metric may be suited to determining matches in other types of reviews as well.

3.6 Experiment 4: Generalizability of Relevance Identification Problem and Approach

In order to demonstrate the ability of our approach to identify paraphrases and possibly some other forms of text entailment, we test it on data from the RTE challenge [59]. The RTE data consists of paraphrase recognition (PP), comparable documents (CD), reading comprehension (RC), information retrieval (IR) and question-answer (QA) matches. The text pairs are annotated as true, to indicate entailment and false to indicate the absence of entailment. While determining textual entailment the aim is to check whether the hypothesis successfully captures the meaning of the text. Haghighi et al.’s graph representation, matching technique and similarity metrics are different from ours. They use dependency trees to construct graphs. They do not perform comparisons across edges in different orders to account for word-order shuffling.

Our relevance identification problem checks whether the review’s content is semantically related to the content of the submission. Thus we draw parallels between the hypotheses and the reviews and between the texts and submissions for our evaluation. We compare our approach with Haghighi et al.’s [29] entailment identification approach, and we choose the RTE 2005 dataset to evaluate our approach.

We determine entailment for 800 text-hypothesis pairs in the data set, and compare it with human
Table 3.4: Accuracy of identifying entailment for some tasks of the RTE data set. PP: paraphrase recognition, CD: comparable documents, RC: reading comprehension, IR: information retrieval and QA: question-answer

<table>
<thead>
<tr>
<th>Structure</th>
<th>PP</th>
<th>CD</th>
<th>RC</th>
<th>IR</th>
<th>QA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phrase or token matching</td>
<td>62%</td>
<td>77%</td>
<td>56%</td>
<td>47%</td>
<td>48%</td>
</tr>
<tr>
<td>Context matching</td>
<td>66%</td>
<td>65%</td>
<td>46%</td>
<td>51%</td>
<td>51%</td>
</tr>
<tr>
<td>Sentence structure matching</td>
<td>62%</td>
<td>59%</td>
<td>44%</td>
<td>49%</td>
<td>49%</td>
</tr>
<tr>
<td>Haghighi et al. (dependency tree)</td>
<td>58%</td>
<td>74%</td>
<td>53%</td>
<td>52%</td>
<td>55%</td>
</tr>
</tbody>
</table>

true/false\textsuperscript{4} annotations of the RTE 2005 dataset to determine accuracy. Our accuracies for the individual graph structures’ matches are listed in Table 3.4. Although the aim of our approach is not to recognize entailment we see that it does well for tasks such as paraphrase recognition (PP), producing an average accuracy of 63%. Our accuracy is higher than what Haghighi et al. achieved—58%. We notice that edges perform better than vertices for task PP.

Tasks CD and RC contained several cases of $n$-gram ($n = 5$) overlaps for the text-hypothesis pairs (20.4% and 26% respectively). Since phrase matches do not rely on word-order information, they produce higher accuracies for CD and RC tasks than the other graph structures. Text-hypothesis pairs in the IR and QA tasks have very few $n$-gram overlaps (4.3% and 3.7% respectively). Accuracy for IR and QA tasks are higher for context and sentence structure comparisons (as seen in Table 3.4). For tasks IR and QA, syntax information from edges and double edges may play an important role in improving accuracy.

### 3.7 Feedback to Reviewers—Relevance

Screenshots of the output from our review assessment system can be seen in Figure 3.7. In this example we have two sets of reviews written for an article on *software extensibility*\textsuperscript{5}. The sample review in Figure 3.7a has a relevance of 0.13 (on a scale of 0–1). However, the review in Figure 3.7b contains no information that is relevant to the article on software extensibility, and so has a relevance of 0.

Our aim with such a review assessment system is to motivate reviewers to make their review more relevant to the submission. This would help authors to better understand details of the review, and use the review to fix and improve their work. In the future we are planning to improve the format of this output by providing textual feedback in addition to the numeric feedback. The feedback will point to specific instances of the review that need improvement. This may make it easy for reviewers to interpret the numeric score, and may motivate them to use the metareview feedback to improve their reviews.

\textsuperscript{4}Data was annotated by 3 human judges who had an agreement in the range 91–96%.

\textsuperscript{5}Software Extensibility: https://en.wikipedia.org/wiki/Extensibility
3.8 Contributions

Some important contributions of this work are listed below. A list of our findings have been provided in Table 3.5.

1. A graph-based text matching technique that uses syntactic and semantic features to identify relevance.

2. A paraphrase-recognition technique to compare same and different types of edges.

We have shown that:

1. The use of additional context information from edges and double edges helps improve the accuracy of identifying relevance.

2. Word order graph representation produces higher accuracy and $f$-measure than a dependency tree representation for the review relevance identification task.

3. Our semantic relatedness metric produces a higher $f$-measure than the other relatedness metrics while identifying relevance.
Table 3.5: Summary of findings—Relevance

<table>
<thead>
<tr>
<th>Finding</th>
<th>Most relevant previous strategy</th>
<th>Dataset</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>§3.3, 3.3.1 High accuracy achieved by the lexico-semantic relevance identification approach</td>
<td>Dependency trees, text overlap-based measure</td>
<td>Expertiza</td>
<td>66% accuracy and 0.67 f-measure</td>
</tr>
<tr>
<td>§3.4 Faster graph-matching</td>
<td>Dependency trees</td>
<td>Expertiza</td>
<td></td>
</tr>
<tr>
<td>§3.5.1 High f-measure produced by the semantic-relatedness metric</td>
<td>Word-sense metric, Wu &amp; Palmer, Leacock &amp; Chodorow and Jiang &amp; Conrath</td>
<td>Expertiza</td>
<td>0.6 f-measure</td>
</tr>
<tr>
<td>§3.5.2 High f-measure produced by the semantic-relatedness metric</td>
<td>Word-sense metric, Wu &amp; Palmer, Leacock &amp; Chodorow and Jiang &amp; Conrath</td>
<td>Amazon</td>
<td>0.58 f-measure</td>
</tr>
<tr>
<td>§3.6 High accuracy on the paraphrase recognition task</td>
<td>Haghighi et al. [29]</td>
<td>RTE 2005</td>
<td>63% accuracy</td>
</tr>
</tbody>
</table>

4. Our approach produces a higher average accuracy on the paraphrase recognition task in the RTE dataset.

3.9 Conclusion

In this chapter we use a graph-based approach to determine whether a review is relevant to a piece of submission. Relevance helps us identify whether a review has been written for the correct submission. A word-order graph representation is used, since it helps capture ordering information. Ordering information is suited to identify instances of paraphrases in the review text. Graph edges and double-edges (two contiguous edges) incorporate additional context and structure information. Relevance is identified by comparing review and submission graphs. We use a WordNet-based relations metric to determine the degree of semantic relatedness between review and submission graphs. Matching involves comparing vertices and edges and double-edges in same and different orders to identify possible shuffling of words or phrases. An average of the different structural matches gives us the degree of relevance of a review. The reviewer is presented with a relevance score in the range of 0–1. A lower relevance score implies that the review does not contain relevant text, and that the reviewer should discuss more of the submission’s content in the review.

The next chapter discusses our graph-based pattern-identification approach to identify content type.
Chapter 4

Review Content

Reviews play a crucial role in providing feedback to people who make assessment decisions. Nelson et al. found that reviews that locate problems in the author’s work or provide suggestions for improvement, helped authors understand and use feedback effectively [11]. Review content information could be used by reviewers to understand where their review is lacking, and provide improved reviews to the authors of a paper. One of the ways in which review content can be classified is:

**Summative** Provide positive feedback or a summary of the author’s work. E.g. “The page is organized logically, and gives an example code next. The page has proposed to solve the problem with the concept of currying.”

**Problem detection** Identify problems in the author’s submission. E.g. “The page lacks a qualitative approach. It also lacks an overview.”

**Advisory** Provide suggestions to authors on ways of improving their work. E.g. “There could be more ethics related links on the page and more in-depth analysis of ethical issues on the study guide.”

The task of review content identification is important because it gives us information on the type of content a review contains. Reviews that contain only praises are not as useful as those that contain instances of problems, which in turn are not as useful as reviews that provide suggestions for improvement [11]. Identifying a review’s content type would help reviewers learn where their reviews are lacking, and thus write more effective reviews.

From the above examples and a set of sample reviews in Table 4.1 we see that summative, problem detection and advisory reviews discuss similar points (e.g. page organization, presence of links), but differ in the way these points are discussed. For instance, summative reviews make observations (e.g. “organized logically”), problem detection reviews identify problems (e.g. “lacks a qualitative approach”) and advisory reviews provide suggestions for improvement (e.g. “more in-depth analysis of ethical issues”). Current approaches to automatic review classification use machine-learning techniques with
Table 4.1: Some examples of reviews belonging to the different review content classes.

<table>
<thead>
<tr>
<th>Category</th>
<th>Review</th>
</tr>
</thead>
</table>
| Summative     | “Yes, the page provides general info on the discussion questions and then jumps right into the ethical considerations for each issue.”  
|               | “Author has used his own knowledge along with the source on web and the text book to write the page. It is clearly evident from the links the author has given in the page.”                                  |
| Problem detection | “No, capitalization does not follow convention. Only the first word and proper nouns are supposed to be capitalized.”  
|               | “Some sentences on this page are a bit crammed up and complex. The organization may confuse the reader.”                                                                                             |
| Advisory      | “I would like to see a better definition of each technique, with more links to content, before getting into the advantages and disadvantages.”       
|               | “The page should elaborate on all the caches and should give a good overview of all techniques. More links could have been provided.”                                                                      |

shallow text features such as counts of nouns and verbs [13]. Due to the presence of overlapping text, techniques that rely only on token frequencies may not succeed in distinguishing between the different types of content.

Ours is a pioneering effort in applying a semantic pattern-identification technique to the little-explored field of review content identification. The aims of this work are:

1. To identify patterns that represent each review content class by exploring semantic relatedness across tokens and phrases of reviews, and

2. To use patterns to identify content type of new reviews.

We use a word-order graph representation, in which edges capture order and context information. We use the semantic relatedness metric described in Chapter 2.2 to determine similarity between texts.

The rest of this chapter is organized as follows. Section 4.1 contains related research in the area of pattern and topic-identification. Section 4.2 describes the process of identifying a review’s state during graph generation. Section 4.3 describes the steps involved in the semantic pattern-identification process—graph generation, pruning edges that are either rare or too frequent, pattern identification and determining content types of new reviews. Section 4.4 contains a discussion of the experiments we conducted to evaluate our approach. Section 4.5 describes the format of content type feedback presented to reviewers. Section 4.6 describes the contributions of our content-type identification approach. Finally, Section 4.7 concludes the chapter with a summary of the work.
4.1 Related Work

Ours is a pioneering effort in applying a semantic pattern-identification technique to the little-explored field of review content identification. Cho [13] employs a bag-of-words based approach to classify peer reviews. Xiong et al. [12] use shallow features such as counts of nouns and verbs in the review to locate problematic instances identified by the reviewer.

Our aim is to identify phrases or clauses that capture the central meaning of each of the review content types. The problem of identifying patterns using lexical cohesion\(^1\) techniques has been explored in other areas such as text summarization and topic identification, which involve identifying the topics that represent a piece of text. Barzilay and Elhadad [35] introduce the use of lexical chains to perform text summarization. They use word-sense information to form lexical chains. Lexical chains establish links across tokens that are semantically related. They use a cohesion-based approach to identify strong chains i.e., the more representative sentences that are included in the summaries.

Radev et al.’s approach, MEAD, also uses a centroid-based summarization technique to identify the best sentences to be included in a summary [60]. Erkan et al. [34] use a centrality-based summarization technique to determine the central ideas of text graphs. In these approaches, sentences of a document are represented as vertices of a graph, and cosine similarity between adjacent sentences identifies the degree of similarity between them. Sentences, which are most similar to the other sentences in a document are considered to be representative of the meaning of a document.

Hovy et al. [61], in their paper on text summarization, perform topic identification in order to generate useful summaries. They use frequently used cue-words or phrases such as “in summary” or “in conclusion”, and other indicator words and phrases to identify topics in a text.

Our approach uses a cohesion-based technique to identify patterns in a class of texts.

4.2 Graph Generation and Review State

A graph representation is created for existing reviews containing either summative, problem detection or advisory content.

**Review state:** Consider the review sentence, “The paper is not clear.” An approach that does not handle negation is likely to misidentify this review as a summative review. State of a sentence may be **positive**, **negative** or **advisory**. Words such as “none”, “never”, “failed” give the text a negative orientation. Words such as “would”, “should”, “perhaps” are indicators of suggestion. The state of a review may not be evident from looking at the tokens independently. We therefore propose a heuristic approach to identify state, based on tokens and their contexts.

---

\(^1\)Lexical cohesion is the semantic relatedness between different parts of a text.
Algorithm 2: ReviewState(segment seg)

Input: Sentence segment whose state is to be identified
Output: State of the segment.

1. CurrentState indicates the present state of a segment;
2. CurrentState = no-state;
3. for every token in the segment do
4.     Identify token’s state—negative or advisory;
5.       if CurrentState is no-state then
6.           if token is negative or advisory then
7.               CurrentState = token’s state;
8.               /* replace default state */
9.           end
10.       end
11. else if CurrentState is negative then
12.       if token is negative then
13.           if nouns, pronouns or prepositions in between negative words or if previous word is
14.             “no”, “none” or “never” then
15.               continue;
16.               /* CurrentState remains negative */
17.           end
18.           CurrentState = positive;
19.           /* double negation => positive */
20.       end
21. else if token is advisory then
22.       CurrentState = advisory;
23. end
24. else if CurrentState is advisory then
25.       if token is negative and no nouns or verbs in between the advisory and negative words
26.           then
27.           CurrentState = negative;
28.       end
29. end
30. else if CurrentState is no-state then
31.       CurrentState = positive;
32. end
33. return CurrentState
Graph generation follows the steps in Chapter 2, Section 2.1. The state of each review segment is identified during graph generation. State is updated for each sentence segment, and updated throughout graph generation. Tokens from the same segment will have the same state, but tokens from different segments may not have the same state.

The steps involved in identifying the state of a segment have been described in Algorithm 2. Reviews are broken down into segments at coordinating conjunctions. A review segment is assigned a default no-state (Line 2 in Algorithm 2) until a token or phrase of negative or advisory state is identified (Lines 5–7). CurrentState maintains the present state of a segment, which changes based on the state of its tokens. Algorithm 2 identifies state of a segment based on a token’s state and the state of the segment that has been covered so far—CurrentState.

Our approach also takes into consideration cases of double negatives [62]. We use context information such as the presence or absence of nouns in between tokens to identify whether double negations can be resolved to a positive. For instance, in the text “It is hardly understandable, and the text is incomplete.”, the presence of the noun “text” in between “hardly” and “incomplete” causes the state to remain negative. Negative words, separated by tokens, embellish\(^2\) the negative orientation of the text (Lines 10–13). The presence of negations such as “no”, “none” and “never” in front of other negative words also strengthen the negation, e.g. “No the explanation does not help!” (conditional statement on Line 12).

Consider the segment “It is hardly incomplete.” There are no nouns or verbs between the negative descriptors “hardly” and “incomplete”. The two negative words cancel each other out, resulting in a positive state (Lines 10–11 and Line 15).

Similarly in the case of advisory indicators, context plays an important role in determining state change. If an advisory token is followed by a negative token, then the state changes from advisory to negative (Lines 20–22). In the example “…could not understand…”, when the advisory token “could” is followed by the negation “not”, the segment gets a negative orientation. However, presence of nouns or verbs between the advisory and negative tokens would cause the state to remain advisory e.g. “I would suggest the author to not include…”.

The algorithm concludes by returning the value of CurrentState. If no negative or advisory token is identified, the algorithm returns a positive state (Lines 26–29).

We use a set of negative and advisory words and phrases to help identify a review’s state. We manually collected a set of indicator words and phrases from 100 reviews from Expertiza [1]. We use additional negative indicators from an opinion lexicon provided by Liu et al. [21]. Some examples of negative and advisory indicators are \{not, won’t, don’t, didn’t, barely, hardly\} and \{could, should, maybe, perhaps\} respectively.

Figure 4.1 shows a graph generated for summative reviews “Covers all the information to make an

\(^2\)Negative concord: http://en.wikipedia.org/wiki/Negative_concord
Figure 4.1: Illustration of our approach with examples of summative reviews

ethical decision.” and “Covers some of the pros and cons that go into an ethical decision.” Notice that the vertices are tagged with type noun, verb or adjective as well as with state information (P for positive and N for negative). Edges are tagged with role information.

4.3 Identification of Semantic Patterns

We use a cohesion-based approach to determine patterns that represent the different types of review content. Table 4.2 contains some sample edge patterns. In our approach cohesion is determined in terms of the semantic relatedness between edges, and edges with the highest similarities are chosen as patterns. Our pattern-identification approach involves the following steps:

1. Generating graphs: A graph representation is created for existing reviews (training data) containing either summative, problem detection or advisory content. Figure 4.1 shows a graph representation of two summative reviews from which summative patterns are to be extracted.
Table 4.2: Sample edge patterns from each of the content classes.

<table>
<thead>
<tr>
<th>Summative</th>
<th>Problem Detection</th>
<th>Advisory</th>
</tr>
</thead>
<tbody>
<tr>
<td>issues—are covered</td>
<td>too–cluttered</td>
<td>should be–more</td>
</tr>
<tr>
<td>author’s prose—easy</td>
<td>not—been</td>
<td>would benefit–more</td>
</tr>
<tr>
<td>sticks—topic</td>
<td>ambiguous—about what</td>
<td>more—detail</td>
</tr>
<tr>
<td>page—discussed</td>
<td>not—covered</td>
<td>could be–bit</td>
</tr>
<tr>
<td>parts—original</td>
<td>is typing—mistake for</td>
<td>would benefit–more</td>
</tr>
<tr>
<td>good—examples</td>
<td>grammatical—problems</td>
<td>more—depth analysis</td>
</tr>
</tbody>
</table>

2. **Pruning by frequency:** Not all generated edges may be suitable representatives of the content of a class of texts. For instance, some edges may be very rare in the corpus, while others may be very frequent (e.g., “is–very”). Frequency of an edge is the number of times it is referred to in the corpus. Pruning out the rare edges is a form of regularization, since we would be avoiding over-fitting the model to infrequent edges (outliers) in the class. Edges with frequencies $\geq 1$ and $\leq 10$ are selected. We select a broad range so as not to exclude too many edges.

3. **Calculating edge importance:** Edges are compared with each other (as shown in Figure 4.1) to identify those with the highest semantic value. The importance of an edge $e$ is calculated by taking the average of the matches that $e$ has with each of the other edges in the set. Importance is given by Equation 4.3, where $E$ is the set of all edges.

$$
\text{Importance of } e = \frac{1}{|E| - 1} \left( \sum_{\forall f \in E, f \neq e} \text{Similarity}(e, f) \right)
$$

$\text{Similarity}(A, B)$ between edges $A$ and $B$, with vertices $(A_1, A_2)$ and $(B_1, B_2)$ respectively, is calculated as follows:

$$
\text{Similarity}(A, B) = \pm \frac{1}{2} \left( \text{match}(A_1, B_1) + \text{match}(A_2, B_2) \right)
$$

WordNet is used to identify match between tokens. The value of match is one of those described in Chapter 2, Section 2.2. When edges are compared, their respective states are compared. State’s impact on similarity is given by the $\pm$ of Equation 4.2. If two edges have the same state then similarity is $+$-value, but if the edges have different states, then it is $-$-value. For example, if two tokens have an exact match but have different states then they get a match value of $-6$.

4. **Selecting edge patterns:** Edges that have a high similarity with other edges in the class of texts are selected as patterns. Edges whose importance values are greater than the average importance of all generated edges, are selected as patterns. In Figure 4.1 the selected edge patterns are shown.
with a thick border. Some of the patterns selected from the summative reviews are “covers–pros cons”, “make–decision”, “go–decision”.

Table 4.2 lists some sample edge patterns that represent each type of review content. We can see that summative patterns are positive e.g. “author’s prose–easy”. The problem detection patterns, on the other hand, identify cases of problems, e.g. “too–cluttered”. In the case of advisory reviews, patterns such as “more depth–analysis” offer suggestions to the author.

5. **Identifying review content**: The content type of a new review is identified by generating a graph representation for the new review, and comparing each of its edges with each content type’s patterns. Relatedness between the review’s edges and each content type’s edges is calculated using Equation 4.2.

\[
content_C = \frac{1}{|E|} \sum_{\forall e \in E} \left( \frac{\sum_{\forall p \in P_C} Similarity(e,p)}{|P_C|} \right)
\]

\(content_C\) is the amount of content type \(C\) that the new review contains, where \(C\) could be summative, problem detection or advisory. \(content_C\) is the average of the matches between a new review’s edges \((E)\) and \(C\)’s patterns \((P_C)\).

Patterns are given state values in order to aid matching across edges. Summative patterns are assigned a positive state, while problem detection patterns are assigned a negative state, and advisory patterns are assigned an advisory state.

In Figure 4.1 summative patterns are compared with two new reviews. Review “The discussion covers a lot of different ethical issues.” is summative. Due to the absence of any negative or advisory tokens, its vertices are tagged with a positive state. The review “Ethical issues are not sufficiently covered.” is of type problem detection. It contains a negation (not), and so its vertices are tagged with a negative state. The reviews’ similarities with the summative patterns are 1.19 and \(-0.94\) respectively. As expected the former review has a positive similarity and contains summative content while the latter does not.

4.4 **Experiments**

4.4.1 **Review content type identification**

We evaluate our content type identification approach on peer-review data from Expertiza [1] and the Scaffolded Writing and Rewriting in the Discipline project (SWoRD)\(^3\) [63]. Expertiza and SWoRD are web-based collaborative learning applications, which allow for peer-reviewing.

---

\(^3\)SWoRD: http://www.lrdc.pitt.edu/projects/pdetail.asp?Proj_ID=2
We demonstrate that word-order graphs together with semantic relatedness metrics produce patterns that are better at identifying content type of a review than *bag-of-words* based classifiers. Text-classification techniques such as logistic regression and support vectors, when used with unigrams or bigrams from the text, use exact matches to perform classification. We also use edges (generated by word-order graphs) as features (i.e., with additional context), to construct the models.

**Dataset and method**

We evaluate our technique on 953 academic reviews collected from Expertiza and 1048 reviews from SWoRD—a total of 2001 reviews. The data contains 796 summative reviews, 609 problem detection reviews and 596 advisory reviews. The baseline% of this dataset is 39.7%. For the purpose of evaluation, reviews are classified based on their most predominant content type. For instance, consider the review “I did not see any reference. I think that there are more issues related to this topic that could have been identified.” The review is classified as an advisory review, since it contains suggestions for improvement.

We randomly selected 10% of the reviews from Expertiza and got four annotators to classify them as summative, problem detection or advisory. We found an average inter-rater agreement of 82% and a Kappa value of 0.74\(^4\) between the four annotators’ ratings. Because of a good degree of agreement we use a single annotator’s labels for the 953 Expertiza reviews. Two human judges coded reviews from the SWoRD dataset. The reviews were annotated as summary, explicit problem and explicit solution. The judges had Kappa in the range of 0.51–0.92 [63].

We use a hold-out (splitting) based validation, in which the data set is divided into two disjoint sets—training and testing. The training data set is used to identify the semantic patterns. Patterns are used to predict the classes for the reviews in the test set. We use 1459 reviews for training (≈ 70% of the data) and the remaining 542 reviews for testing. The training and testing sets have approximately the same number of data points from each of the classes, which prevents the model from being biased towards any one content class. We calculate our results based on a 3-fold cross-validation.

**Results and analysis**

Results in Table 4.3 show that our pattern-matching approach produces high precision, recall and *f*-measure values for the task of review content identification.

Consider the problem detection review “There are quite a few grammatical errors making the webpage more difficult to understand than it should be.” Tokens “more” and “should be” appear often among advisory reviews (see Table 4.2), which cause the problem detection review to be misclassified as an advisory review. There are times when an advisory review may contain a brief description of the problem. The presence of overlapping problem detection content causes some of the advisory reviews to be misclassified.

\(^4\)Cohen’s Kappa: http://explorable.com/cohens-kappa
Effect of selecting patterns: We evaluated the performance of different numbers of generated patterns in identifying review content. We compare the performance of three sets of patterns: (1) all edges used as patterns (all edges), (2) edges selected by our approach (selected patterns), and (3) a set of patterns smaller than that selected by our approach (pruned patterns). These sets contained on average 26, 16 and 5 patterns respectively. A comparison of the recall and precision values produced by these patterns is provided in Figure 4.2.

Patterns produced by our approach have a higher precision and recall than when all edges or a smaller set of edges are used as patterns. This shows that using too many or too few edges for content identification may not be as efficient as selecting the most important patterns by our approach.

Comparison with bag-of-words approach

We compare our pattern-based content identification approach with two classification techniques: (1) logistic regression, and (2) multi-class support-vector categorization. Logistic regression is increasingly being used for text categorization, [64] and its performance has been found to be comparable

Table 4.3: Average recall, precision and $f$-measure for each of the different classifiers.

<table>
<thead>
<tr>
<th>Approach</th>
<th>Recall</th>
<th>Precision</th>
<th>$f$-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Patterns</td>
<td>0.72</td>
<td>0.76</td>
<td>0.74</td>
</tr>
<tr>
<td>Support Vectors, Unigram</td>
<td>0.70</td>
<td>0.73</td>
<td>0.71</td>
</tr>
<tr>
<td>Support Vectors, Bigram</td>
<td>0.67</td>
<td>0.71</td>
<td>0.69</td>
</tr>
<tr>
<td>Support Vectors, Edges</td>
<td>0.34</td>
<td>0.62</td>
<td>0.44</td>
</tr>
<tr>
<td>Logistic Regression, Unigram</td>
<td>0.71</td>
<td>0.74</td>
<td>0.72</td>
</tr>
<tr>
<td>Logistic Regression, Bigram</td>
<td>0.67</td>
<td>0.72</td>
<td>0.70</td>
</tr>
<tr>
<td>Logistic Regression, Edges</td>
<td>0.45</td>
<td>0.59</td>
<td>0.51</td>
</tr>
</tbody>
</table>
Table 4.4: Comparing match produced by our patterns and by Opinosis with human-provided topic representative sentences.

<table>
<thead>
<tr>
<th>Approach</th>
<th>ROUGE-1</th>
<th>ROUGE-2</th>
<th>ROUGE-SU4</th>
<th>Avg. # of tokens</th>
</tr>
</thead>
<tbody>
<tr>
<td>Opinosis [65]</td>
<td>0.28</td>
<td>0.09</td>
<td>0.09</td>
<td>15</td>
</tr>
<tr>
<td>Patterns</td>
<td>0.54</td>
<td>0.17</td>
<td>0.21</td>
<td>16</td>
</tr>
</tbody>
</table>

...to other linear classification approaches such as support vectors. We use the statistical analysis tool R’s\(^5\) LiblineaR package to perform logistic regression. We set the cost (regularization) parameter to 1. We also use the (linear) multiclass support-vector classification algorithm available in the LiblineaR package. We use unigrams, bigrams and graph edges as three sets of text features for each of the above classification techniques.

In the case of logistic regression a number of summative reviews were misclassified as problem detection reviews, and several problem detection reviews were misclassified as advisory reviews. This caused the precision to drop. Similar trends were observed in the case of support-vector classifiers.

Unigrams perform well for both support vectors and logistic regression classifiers, and produce higher precision, recall and \(f\)-measure values than a bigram-based model. The models perform poorly when graph edges are used as features. Edges contain on average 22 tokens per review, and the increased number of tokens could have resulted in a decrease in the recall, precision and \(f\)-measure values. Thus, our approach produces better precision, recall and \(f\)-measure values than support vectors and logistic regression classifiers.

### 4.4.2 Goodness of semantic patterns

Identifying patterns is similar to identifying important topics or concepts in a text. We therefore compare our approach with Opinosis (Ganesan et al., [65]), which identifies the “important points” or “topics” in a document in order to generate document summaries. Opinosis uses a graph-based approach to extract the main points in a document. Opinosis represents the state of the art in identifying topic sentences among product reviews. We therefore evaluate the quality of patterns generated by our approach on the Opinosis dataset. We demonstrate that our approach succeeds in capturing patterns that have a good degree of overlap with human provided topic sentences.

#### Dataset and method

Opinosis’ data consists of 51 documents, each containing approximately 100 reviews. The human references in the Opinosis dataset contain 2-3 sentences each.

---

\(^5\)R project: http://www.r-project.org/
Ganesan et al. use Recall-Oriented Understudy for Gisting Evaluation (ROUGE) to evaluate their system-generated summaries. ROUGE identifies $n$-gram overlaps across compared texts. We use ROUGE to calculate 1-gram, 2-gram and SU4-gram (skip-bigram match, i.e., bigrams with a gap of 4 words between them) matches.

\[
\text{ROUGE}_n = \frac{n\text{-gram overlaps between patterns and human summaries}}{\text{number of tokens in human summaries}}
\] (4.4)

**Results and analysis**

Table 4.4 contains the results from our experiments. Match is calculated on the length of human summaries as shown in Equation 4.4. Results indicate a good degree of agreement between the patterns and human-provided summaries for 1-, 2- and SU4-gram overlaps. This indicates that our approach succeeds in capturing the main points or patterns in a document.

Opinosis is an abstractive approach, and contains shorter, precise sentences. Our patterns capture the key topics using short, pithy phrases (sample patterns are in Table 4.5). Despite using patterns to capture the key topics we see that our approach has a good match with human values. The average number of tokens in our patterns is 16, which is comparable to the average number of tokens Opinosis’ outputs
Table 4.5: Sample edge patterns from documents in the Opinosis dataset.

<table>
<thead>
<tr>
<th>Accuracy of Garmin GPS</th>
<th>Interior of a Toyota Camry</th>
</tr>
</thead>
<tbody>
<tr>
<td>accurate–directions</td>
<td>roomy–interior</td>
</tr>
<tr>
<td>times info–accurate</td>
<td>interior–nice</td>
</tr>
<tr>
<td>after that it–easy</td>
<td>comfortable quiet–interior</td>
</tr>
<tr>
<td>adjunct travel trip</td>
<td>interior–luxurious comfortable</td>
</tr>
<tr>
<td>directions–accurate</td>
<td>stylish–roomy</td>
</tr>
</tbody>
</table>

contain—15. Thus, we see that with nearly the same number of tokens, our approach captures more relevant information than Opinosis.

4.5 Feedback to Reviewers—Content Type

A screenshot of the output from our review assessment system can be seen in Figure 4.3. In this example the review is written for an article on software extensibility.

The sample review is rated on a scale of 0–1 and has a value of 0.27 for summative content (e.g. “...simple and easy...”, “...good examples...”), 0.36 for problem detection (e.g. lack of “...examples”) and 0.36 for advisory content (e.g. “...would have been better...arguments for and against using extensibility...”). This gives the reviewer information on the different types of content a review contains. Our aim is to motivate reviewers to improve the review in the areas where it is lacking.

4.6 Contributions

Some important contributions of this work are listed below. A list of our findings have been provided in Table 4.6.

1. A graph-based text-matching approach that uses semantically important graph edges to form patterns.
2. Using semantic patterns to identify the content type of new reviews.

We show that:

1. Our semantic pattern-based content-identification technique has an $f$-measure of 0.74.
2. Our approach produces a higher $f$-measure than support vectors and logistic regression-based classifiers in learning the content type of reviews.
3. Our approach succeeds in capturing the most representative patterns from topic documents in the Opinosis dataset, and has a good degree of agreement with human summarizers.
Table 4.6: Summary of findings—Content

<table>
<thead>
<tr>
<th>Finding</th>
<th>Most relevant previous strategy</th>
<th>Dataset</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>§4.4.1 High f-measure produced by the semantic pattern-identification approach</td>
<td>Multiclass support vector machine and logistic regression with features: unigrams, bigrams and edges</td>
<td>Expertiza</td>
<td>0.74 f-measure</td>
</tr>
<tr>
<td>§4.4.2 High correlation of system-generated patterns with human references</td>
<td>Ganesan et al. [65]</td>
<td>Opinosis</td>
<td>0.54 for ROUGE-1, 0.17 for ROUGE-2, 0.21 for ROUGE-SU4</td>
</tr>
</tbody>
</table>

4.7 Conclusion

In this chapter we have discussed the process of automatically identifying the content type of reviews. Content type of a review may be—summative, problem detection and advisory. Content identification is important because it gives us information on the types of content a review contains. This chapter discusses how the system learns semantic patterns from past reviews and uses them to identify the content of new reviews. We use a graph-based cohesion identification approach to identify semantically important edges as patterns for each type of review content. The pattern-identification approach incorporates the identification of state of a review. State of a review may be positive, negative or advisory. A review is compared with patterns for each content type and the amount of each content type a review contains is displayed to the user on a scale of 0–1. This information helps the user identify whether their review is lacking in summative, problem detection or advisory content. This information may help reviewers write better quality reviews, which might be more effective in helping authors improve the quality of their submissions.

**Future work:** Our pattern-identification approach could be extended to other applications such as distinguishing the problem and solution sections in incident reports, logs, medical transcripts or diagnosis files. The problem and solution descriptions may use the same set of words or phrases, in which case it may be hard to distinguish between these sections. A semantic-patterns based approach may be required to gain a deeper understanding of the difference between the segments.
Chapter 5

Review Coverage

Since reviews play a crucial role in helping authors, it is important to ensure that they are complete, and their content is useful to authors. At times reviews may cover just one section in the author’s submission (the text under review), say the “Introduction”, and provide no feedback on any of the other sections in the document.

Kuhne et al. [5] found that authors are contented with reviewers who have made an effort to read and understand their complete work. Reviews that cover the important sections of the author’s work are likely to be more useful, since they are more complete than reviews discussing a single section. A complete review also reflects positively on a reviewer’s understanding of the author’s work.

Existing approaches use shallow text features such as word count to analyze a review’s usefulness. Xiong et al. use a bag-of-words, exact match approach to identify instances of problems (in the author’s work) caught by peer reviews [12]. Cho uses machine-classification techniques such as naïve Bayes, SVM (support vector machines) and decision trees to classify feedback [13]. Until now none of the automatic review-analysis approaches look for the degree of coverage of a submission by a review.

Our aim with this work is to measure the coverage of a submission’s “main points” or “topic sentences” by a reviewer’s feedback.

**Definition of Topic Sentence** Let $S = \{s_1, s_2 \cdots s_n\}$ be the set of sentences in a submission. A set $T = \{t_1, t_2 \cdots t_n\}$, where $T \subset S$ is a set of topic sentences for a submission such that $T$ succeeds in capturing the topic or the central meaning of $S$.

**Definition of Coverage** Let $S$ and $R$ be the set of sentences in a submission and review respectively. Let $T$ be the set of topic sentences in a submission. Coverage is defined as the fraction of tokens in the review that are also present in the topic sentences.

To calculate coverage we need to know the topic sentences in a submission. Hence in this work we describe our approach to identifying topic sentences followed by the calculation of a review’s coverage.
To illustrate the problem we use real-world submission and review data from assignments completed using Expertiza [1]. Figure 5.1 contains a sample submission with its topic-representative sentences in bold, and three sample reviews with high, medium and no coverage of the submission’s topic sentences. The first review covers the submission because it mentions *ethical principles* and *ethics*. However, the review with *medium* coverage mentions just *ethics*, and the review with *no* coverage does not contain any relevant information.

One of the chief contributions of this work is the focus on the important but often ignored problem of identifying review coverage. There is little or no previous work in this area. The rest of the chapter is organized as follows: Section 5.1 discusses related work, Section 5.2 contains a description of our coverage identification approach. Section 5.3 describes our experiments and results. Section 6.5 describes the contributions of our coverage identification approach, and finally Section 5.5 concludes the chapter.

### 5.1 Related Work

The problem of coverage identification may be broken down into two steps: (1) the identification of topic representative sentences (in the submission), and (2) determining the degree of overlap between a review and the topic sentences. The problem of identifying topic representative sentences, using clustering and pattern identification techniques, has been explored in text summarization applications [66].
Qazvinian et al. [67] produce a document’s summary using a cluster-based approach. They use a hierarchical agglomerative algorithm to cluster sentences. They use LexRank [34] with cosine as a measure of similarity to identify the important sentences. ClusterRank algorithm proposed by Garg et al. [68] applies a clustering technique to identify sentences belonging to the same topic. They use token frequencies to represent sentences in a document. Steinbach et al. use the bisecting k-means clustering technique, with a simple cosine metric as the similarity measure to group documents [69].

Radev et al.’s approach (MEAD) uses a clustering technique with cosine similarity metric (with frequency of tokens as features) to identify the most appropriate sentences (centroids) to be included in a summary [60].

The above topic identification approaches use a shallow measure to group sentences or documents. Other topic-identification approaches proposed by Michalcea [31] and Coursey et al. [33] use the Google PageRank and Hyperlink-Induced Topic Search (HITS) algorithms to identify topic-representative sentences. We employ a modified agglomerative clustering technique (described in Section 5.2) to group submission sentences into clusters representing topics. We then identify the most representative sentences from across the different clusters.

Sentences discussing the same topic, but containing different terms may not be effectively grouped by a clustering approach relying only on the frequency of words. Steinbach et al. found that agglomerative clustering with a word-frequency based matching is prone to making mistakes early in the run of the algorithm, and mistakes made initially cannot be fixed later on [69]. The algorithm tends to group documents containing the same words but belonging to different classes into the same cluster; thus, word-frequencies may not be a good indicator of relatedness between documents.

We employ a lexico-semantic matching technique, since it captures context information. We use a word-order graph to represent text, since it captures syntax or order of tokens in a text. Word-order graphs are suited for identifying lexical and voice changes, which are common among paraphrased text. Similarity should capture the degree of relatedness between texts. Hence we use a WordNet-based metric [24]. Topic-representative sentences are selected from the clusters. The topic sentences are compared with sentences in a review to identify coverage.

5.2 Cluster-based Topic Sentences Identification and Coverage Calculation

We define the problem of clustering as follows – Given a set of sentences $S_1, S_2 \cdots S_n$, we group these sentences into a set of clusters $C_1, C_2 \cdots C_k$ such that the sentences in each cluster are semantically more similar to each other than those in the other clusters.

We use an agglomerative clustering technique to group sentences into clusters. Sentences belonging to the same cluster discuss the same topic. We use the UPGMA (Unweighted Pair Group Method with
Arithmetic Mean) scheme for agglomerative clustering [69] i.e., average of the pairwise similarities between a new sentence \( S \) and every sentence \( S_C \) in a cluster \( C \) to determine whether a sentence belongs to a cluster.

### 5.2.1 Calculating semantic relatedness between sentences

Before clustering begins, similarities between all pairs of sentences in a submission are calculated. Similarity is measured as the average of the vertex and edge matches between the compared sentences. Graph vertices contain phrases or tokens. Vertex match, referred to as \( \text{PhraseSem} \), succeeds in capturing semantic relatedness between single or compound words. Edge match, referred to as \( \text{ContextSem} \), compares contiguous phrases (vertices) across graphs, and they capture more context than vertices.

\[
\text{Similarity}(A, B) = \frac{1}{2}(\text{PhraseSem}(A, B) + \text{ContextSem}(A, B)) \tag{5.1}
\]

\[
\text{PhraseSem}(A, B) = \frac{1}{|V_A| + |V_B|} \left( \sum_{V_A} \max_{V_B} \{ \text{sem}_v(V_A, V_B) \} + \sum_{V_B} \max_{V_A} \{ \text{sem}_v(V_B, V_A) \} \right) \tag{5.2}
\]

\[
\text{ContextSem}(A, B) = \frac{1}{|E_A| + |E_B|} \left( \sum_{E_A} \max_{E_B} \{ \text{sem}_e(E_A, E_B) \} + \sum_{E_B} \max_{E_A} \{ \text{sem}_e(E_B, E_A) \} \right) \tag{5.3}
\]

\( \text{Similarity}(A, B) \) gives the similarity between sentences \( A \) and \( B \). \( V_A, V_B \) are the vertices, and \( E_A, E_B \) are the edges in sentences \( A \) and \( B \) respectively. We identify the best (\( \max \)) semantic match for every vertex or edge in sentence \( A \) with a vertex or edge in sentence \( B \) (and vice versa). The average of the best vertex and edge matches gives us \( \text{PhraseSem} \) and \( \text{ContextSem} \) matches between sentences \( A \) and \( B \) respectively.

Match \( \text{sem}_v \) between two tokens could be one of the matches described in Chapter 2, Section 2.2—\{exact, synonym, hypernym or hyponym (more generic or specific), meronym or holonym (sub-part or whole), presence of common parents (excluding generic parents such as “object”, “entity”), overlaps across definitions or examples of compared tokens, distinct or non-match\}. Match between two edges—\( \text{sem}_e \) is calculated as the average of the matches between their vertices. Each match is calculated using WordNet.

### 5.2.2 Grouping sentences into clusters

The clustering algorithm described in Algorithm 3, starts by initializing every sentence in the text to its own cluster. Every cluster has a similarity value, which we try to maximize (inverse of a cluster’s diameter [70]). A cluster’s similarity is the average of the similarity between all pairs of sentences it
contains. Initially every cluster’s similarity is set to 0.

We rank sentence pairs based on their similarity (highest to lowest) using the merge-sort algorithm. We select sentence pairs in the order of their ranks. For a sentence pair, we choose the target cluster based on the cluster’s similarity values, i.e., the cluster with a higher similarity is chosen as the target (Lines 5–8 in Algorithm 3). If both sentences’ clusters have the same similarity, then we select the target based on the number of sentences in each cluster (Lines 10–13). A cluster with more “similar” sentences is chosen over one with fewer sentences. If both the cluster similarity and the number of sentences are the same, then we randomly select a target cluster (Line 15). In a sentence pair $S_1 - S_2$, if $S_2$’s cluster is chosen as the target, then $S_1$ is added to $S_2$’s cluster if it satisfies the condition in Equation 5.4 (Lines 18–21).

\[
(C.clusterSimilarity - \sum_{\forall S_C \in C} \frac{\text{Similarity}(S, S_C)}{|C|}) \leq \alpha
\]  (5.4)

According to the above condition, the difference between the cluster’s similarity and $S$’s average semantic similarity with $C$’s sentences must not be more than $\alpha$. Since different types of texts have different similarities, it may not be suitable to set a constant threshold for clustering. Therefore one of the ways in which $\alpha$ may be set is by calculating it as the average of the difference between all sentences’ similarities. We choose $\alpha$ as the average of the difference between sentence similarities because (a) it gives us the degree of variance between the sentences’ similarities, and (b) it prevents sentences that are much too dissimilar from being grouped into the same cluster.

As explained earlier, in our clustering approach we select sentence pairs ordered by their similarities (from most similar to least similar). This ensures that the most similar sentences are grouped together.
earlier on in the process. This may help avoid mistakes that arise in the earlier rounds of agglomerative clustering [69]. A cluster’s similarity is re-calculated each time a new sentence is added to it.

The condition in 5.4 ensures that sentences that are added to the cluster have high similarity with a cluster’s sentences. Thus, the process ensures that the clusters contain only semantically related sentences, i.e., they contain sentences that are similar in meaning and context. Figure 5.3 depicts the two clusters generated for the sentences in the sample submission.

**Running time:** Our clustering algorithm depends on the time taken to (1) compute similarities, (2) rank sentence pairs and (3) group sentences into clusters. The time taken to calculate sentence similarities is \( \Theta(n^2) \), where \( n \) is the number of sentences in a submission. We use merge sort to rank sentence pairs based on their similarities. The number of sentence pairs produced by \( n \) sentences is approximately \( n^2 \). Therefore the ranking takes \( \Theta(n^2 \log(n^2)) \). The time taken to cluster sentences is \( \Theta(n^3) \)—since we iterate through each sentence pair \( (n^2) \) to determine the cluster to which the sentences would belong, and then recalculate the cluster’s similarity, which entails comparison with \( n \) sentences in a cluster. Thus the total running time is \( \Theta(n^2) + \Theta(n^2 \log(n^2)) + \Theta(n^3) \). Therefore the running time is \( \Theta(n^3) \).
Algorithm 3: Clustering Algorithm

**Input:** Sentences $S_1, S_2 \cdots S_n$ that are to be clustered.

**Output:** Clusters $C_1, C_2 \cdots C_k$.

1. Initially each sentence belongs to its own cluster.
2. Initialize every cluster’s similarity i.e. $C_i$’s similarity = 0.
3. Let $R$ be the ranked list of sentence pairs, and $R = \{(S_1, S_2), (S_3, S_4) \cdots (S_{n-1}, S_n)\}$ such that $\text{Similarity}(S_1, S_2) \geq \text{Similarity}(S_3, S_4)$ and so on.
4. **for each sentence pair $(S_i, S_j)$ in $R$ do**
   
   /* Identifying destination cluster. */
   
   5. **if** $S_i$’s cluster’s similarity > $S_j$’s cluster’s similarity **then**
   6. ** Set** $S = S_j$ and destination cluster $C = S_i$’s cluster
   7. **else if** $S_i$’s cluster’s similarity < $S_j$’s cluster’s similarity **then**
   8. ** Set** $S = S_i$ and destination cluster $C = S_j$’s cluster
   9. **else if** $S_i$’s cluster’s similarity == $S_j$’s cluster’s similarity **then**
   10. ** if** $S_i$’s cluster’s sentence count > $S_j$ cluster’s sentence count **then**
   11. ** Set** $S = S_j$ and destination cluster $C = S_i$’s cluster
   12. **else if** $S_i$’s cluster’s sentence count < $S_j$ cluster’s sentence count **then**
   13. ** Set** $S = S_i$ and destination cluster $C = S_j$’s cluster
   14. **else**
   15. Randomly select destination cluster $C$
   16. ** end**
   17. ** end**
   18. **if** $S$ satisfies Equation 5.4 and maximizes $C$’s similarity **then**
   19. $C = \{C \cup S\}$
   20. RecalculateClusterSimilarity($C$)
   21. ** end**
   22. ** end**

Each cluster represents a separate topic or concept discussed by the author. However not all topics in a submission need to be discussed in a review. Since authors tend to write more about significant topics than the insignificant ones, we rank clusters based on the number of sentences they contain. We use the average of the number of sentences across clusters as a threshold for selecting the important clusters.

### 5.2.3 Identifying topic sentences

In this step the most representative sentences are identified from each of the selected clusters. For our approach we take inspiration from cohesion-based methods to identify the most important topic sentences in a submission. In a cohesion-based method only the most well connected vertices are taken
to form the summary [35]. In our approach cohesion is determined in terms of the semantic similarity between sentences, and only those with the highest similarities are chosen as topic sentences.

The problem involves identifying the smallest set of topic-representative sentences $T$ that cover (or are most similar to) every other sentence in the cluster $C$. Sentences in a cluster are only connected to other sentences with whom their similarity $\geq C.clusterSimilarity$. For each cluster, the set $T$ that satisfies the condition, “Maximize pairwise similarities between the topic and cluster’s sentences and minimize size of $T$ (or maximize $\frac{1}{|T|}$)”, is selected as the set of topic sentences.

$$\forall C \max \left( \sum_{\forall S_t \in T, \forall S_i \in C, S_t \neq S_i} \frac{\text{Similarity}(S_t, S_i)}{|T|} \right)$$

The topic-sentence identification problem can be thought of as being similar to identifying a minimum set of vertices that cover all the edges in a graph. However, identifying a minimum vertex cover is a well-known NP-complete problem. Avis et al. propose a list heuristic in which the vertices of a graph are scanned in a certain order, and for every scanned vertex the algorithm makes a decision on whether the vertex should be included in the cover [71].

We also use a list heuristic to handle the topic representative selection problem. Our heuristic approach to identifying the smallest number of sentences that successfully capture the meaning of a cluster is described in Algorithm 4. We statically order sentences based on (a) decreasing order of their average similarity values, and (b) decreasing order of the number of sentences they are adjacent to (i.e., degree of a sentence) (Line 4 in Algorithm 4). Our approach ensures that topic sentences with the highest semantic similarity, that cover previously uncovered sentences are added to the cover set (Lines 6–9). The cover set suitably represents all sentences in the topic cluster. Cover sets from across all clusters taken together form the set of topic-representative sentences for a submission. The topic-representative sentences for the sample submission are highlighted in Figure 5.4.

5.2.4 Measuring a review’s coverage

The coverage of a review is calculated in terms of the number of overlapping matches it has with a submission’s topic sentences. The set of tokens exclude stopwords and frequent words. Definition 5 is used to calculate coverage, where $S$, $R$ and $T$ refer to the set of submission, review and the submission’s topic sentences respectively.

$$\text{coverage}_{\text{recall}}(S, R) = \frac{1}{|T|} \sum_{\forall r \in R, \forall t \in T} \text{overlaps}(t, r)$$

59
**Algorithm 4: Topic Sentence Identification**

1. Let $S$ be a sentence in cluster $C$;
2. Let $U$ be the set of uncovered sentences;
3. Initialize $U = C$;
4. Let $R$ be the ranked list of sentences i.e., $R = \{S_1, S_2 \ldots S_n\}$ such that $S_i$’s average similarity with other sentences in the cluster $\geq S_j$’s average similarity;
5. Sentences in $R$ are ordered by degree (number of sentences adjacent to a sentence)—largest to smallest;
6. for each sentence $S \in R$ do
7.   if $S$ is adjacent to one or more uncovered sentences $S_u$ in $U$ then
8.     $T = \{T \cup S\}$;
9.     $U = \{U - S_u\}$
10. end
11. if $U$ is empty then
12.   Break /* No sentence remains uncovered. */
13. end
14. end
15. return $T$ /* The set of topic representative sentences */

Equation 5.5 calculates the coverage as the overlaps between topic sentences and reviews with respect to the number of words in the generated topic sentences ($|T|$). Therefore this is a recall-based measure viz., the number of tokens in the topic sentences covered by the review. However, we also evaluate coverage using a cumulative measure, which uses both review and topic sentences’ lengths to calculate coverage (Equation 5.6).

$$coverage_{cumulative}(S, R) = \frac{1}{|T|+|R|} \sum_{r \in R, t \in T} 2 \times overlaps(t, r) \quad (5.6)$$

The recall-based measure is not normalized for the length of reviews. The cumulative measure penalizes longer reviews that discuss points, which may not be directly related to the submission’s topic sentences. While we do expect reviews to discuss the main points in the submission, we do not expect every sentence in the review to refer to content in the submission, since reviews may contain praise or criticism, which may (or not) overlap with any content in the submission. Therefore, although we do measure cumulative coverage, we treat the recall measure to be a more suitable estimate of review coverage.

In Figure 5.5 the submission’s topic sentences are compared with three sample reviews. We see that the first review “I would consider the ethical hacking...” has more in common with the submission’s topic sentences, and therefore has a higher coverage value than the other reviews.
5.3 Experiments

5.3.1 Identification of Review coverage

We have explained the need for a review to cover the topic sentences in a submission. We have proposed a cluster-based approach to identify the important sentences in a submission. In this section we study the usefulness of our approach in determining a review’s coverage.

We compare our approach with MEAD, a centroid-based summarization approach [60]. MEAD is an extractive summarization approach, and in our approach too we extract the most representative sentences from a submission. Thus we find MEAD to be an ideal system with which to compare our approach. The ease of availability of the well-documented MEAD platform strengthened our decision to select this tool as a baseline for comparison.

Dataset and method

We use peer-review data from computer science classes to evaluate our approach. A dataset containing 577 reviews and submissions is selected from Expertiza [1]. These are reviews provided by students during peer reviews, and hence not all of the data available may have a good coverage of the author’s submission. The dataset contained submissions on a variety of topics including “Integrated Development Environments for Ruby languages”, “Programming paradigms”, “Ruby Closures”.

Review data is annotated on a scale of 0–5, where 0 indicates no coverage and 5 indicates maximum coverage. The values 0 through 5 indicate relative ordering of the degrees of coverage of a submission’s
content by a review. The small size of the dataset is due to the availability of few reviews that have a moderate to high coverage of a submission.

We randomly selected 39 data points ($\simeq 7\%$) from the data, and got six annotators to label them. We use weighted Kappa to determine inter-rater agreement, since the compared coverage values are ordinal. The average inter-rater agreement among the six annotators is 0.603.

We found a high average Spearman correlation\(^1\) of 0.63 among the six annotators, and an average correlation of 0.6 between an annotator $A$ and the remaining five annotators. For a correlation of 0.6 and $N = 39$, the probability that this correlation is a chance occurrence is very small $p = .0001$ (one-tailed, $t$-test for correlation significance with $\alpha = 0.05^2$). Therefore $A$’s labels for the 577 records are used to evaluate system-generated coverage values.

The baseline\(^3\) of this dataset is 48%, since review-submission pairs rated with a coverage value of 1 constituted the largest set, containing 279 out of a total 577 pairs.

We calculate the degree of coverage of reviews using (1) topic sentences generated by our system and (2) MEAD’s summaries. We determine the correlation between the coverage values generated by our approach and by MEAD’s approach with human-provided coverage values. MEAD summaries are generated for each of the submissions, and reviews’ coverage of the summaries are calculated using the recall and cumulative metrics listed in Section 5.2.4. This experiment will show us how effective our

---


\(^2\)Significance of correlation co-efficient: [http://janda.org/c10/Lectures/topic06/L24-significanceR.htm](http://janda.org/c10/Lectures/topic06/L24-significanceR.htm)

---

62
approach is at determining coverage.

**Results**

Results of the correlation between human-provided coverage values and values generated by our approach are listed in Table 5.1. Our approach produced topic sentences containing on average 108 words per submission. The average number of words the submissions contained is 403. Therefore our approach produces topic sentences containing approximately 26% of the number of words in the submissions.

We identify correlations achieved by system-generated coverage with human-provided coverage values. System-generated values, just like the human annotations, are ordinal, i.e., they exhibit a relative ranking but do not follow an interval. Therefore the Spearman correlation is suited to determining the relationship between the system-generated and human-provided coverage values. The value of the correlation coefficient is indicative of the strength of the relationship that exists between two variables. A positive correlation of 0.51 indicates that the system has a good degree of agreement with human-provided coverage values i.e., the system indicates higher coverage when the human identifies a high coverage, or predicts lower coverage when the human identifies a lower coverage. For $N = 577$ and correlation coefficient of 0.51, the null hypothesis that this correlation is a chance occurrence may be rejected, since $p = .0001$ (one-tailed, $t$-test with $\alpha = 0.05$).

The cumulative coverage value has a lower correlation than the recall coverage. This happens because apart from discussing material in the submission, reviews tend to contain praise or criticism of the work (i.e., text which does not overlap with topic sentences), which might lower the cumulative coverage values. Consider the review, “The page is well organized. The examples are original. However some minor parts need more clarity like the definition of OO languages. Glue languages could be defined.” This review contains additional praise, i.e., an increased number of unique tokens in the review. Thus, when compared with a submission discussing OO languages, this review receives a lower cumulative coverage value than a recall coverage.

**Comparison with MEAD:** We select the first 100 words of MEAD summaries for our evaluation, since our topic sentences contained 108 words on average. MEAD’s summaries produce lower correlations (recall = 0.46 and cumulative = 0.35) with human-provided coverage than that produced by the topic sentences generated by our approach. Most of the summaries generated by MEAD contained the top

<table>
<thead>
<tr>
<th>Approach</th>
<th>$coverage_{recall}$</th>
<th>$coverage_{cumulative}$</th>
<th>Avg. # words</th>
</tr>
</thead>
<tbody>
<tr>
<td>Our system</td>
<td>0.51</td>
<td>0.41</td>
<td>108</td>
</tr>
<tr>
<td>MEAD summarizer</td>
<td>0.46</td>
<td>0.35</td>
<td>100</td>
</tr>
</tbody>
</table>

Table 5.1: Identifying the correlation between system-generated and human-provided coverage values on review data from Expertiza.
Figure 5.6: Topic-representative sentences generated by our approach and by MEAD for a sample submission.

$k$ sentences from the submission text, which may not be fully representative of the most “important” sentences. Figure 5.6 contains topic sentences from our approach and a summary from MEAD for a submission on “software development methodologies”.

With the help of these experiments we have demonstrated the ability of our approach to effectively estimate a review’s coverage of these topic sentences.

5.3.2 Evaluation of generated topic representative sentences

We now evaluate our topic-identification approach to ensure that it produces sentences that represent the most significant concepts discussed in a submission. We evaluate our approach with the dataset used in the Opinosis paper [65]. We use Opinosis for our comparison because: (1) Ganesan et al. use a graph-based summarization approach to create abstractive summaries. Their approach looks for redundant paths in a text graph to create summaries. Their approach uses a graph-based representation and matching approach to determine the topic representative sentences. (2) The Opinosis dataset contains reviews, which is the type of data our work focuses on. (3) During summary creation for Opinosis, the human summarizers were requested to focus on the “major opinions” expressed in the documents; and this aligns with our goal of identifying the most important points in a submission text. Thus, although the

---

3Their data contains product reviews while ours uses academic reviews. However they are similar in that both types of reviews are assessing or evaluating something.
Table 5.2: Comparing ROUGE-1, ROUGE-2 and ROUGE-SU4 results from our system with those from the Opinosis paper.

<table>
<thead>
<tr>
<th>Approach</th>
<th>ROUGE-1</th>
<th>ROUGE-2</th>
<th>ROUGE-SU4</th>
<th>Avg. # of tokens</th>
</tr>
</thead>
<tbody>
<tr>
<td>Our system</td>
<td>0.49</td>
<td>0.12</td>
<td>0.13</td>
<td>19</td>
</tr>
<tr>
<td>Opinosis [65]</td>
<td>0.28</td>
<td>0.09</td>
<td>0.09</td>
<td>15</td>
</tr>
</tbody>
</table>

Opinosis paper deals with the task of summarization their approach and data is suited for our evaluation. The Opinosis data consists of 51 topics, each containing approximately 100 sentences. The reference summaries contain 2-3 sentences each.

We use ROUGE (Recall-Oriented Understudy for Gisting Evaluation) to evaluate the approach [72]. ROUGE uses an $n$-gram based co-occurrence identification approach to compare texts. Ganesan et al. found that their summaries yielded high precision values, i.e., the summaries managed to capture the essence of the article and contained no additional information. We use ROUGE with 1-gram, 2-gram and SU4-grams to determine match.

$$\text{ROUGE}_n = \frac{n\text{-gram overlaps between patterns and human summaries}}{\text{number of tokens in human summaries}} \quad (5.7)$$

We ran our algorithm and generated the most representative sentences for each topic document. We restrict our analysis to the top 2 representative sentences generated by our approach. The match values (calculated using Equation 5.7) for ROUGE-1, ROUGE-2 and ROUGE-SU4 are listed in Table 5.2. As seen from the table our approach has a good degree of agreement with human-identified topic sentences. Our approach generated, on an average 19 tokens per topic, which is not much higher than the average 15 tokens/topic generated by Opinosis. Results indicate that our approach produces topic sentences that are closer to human-provided references.

### 5.4 Contributions

To the best of our knowledge our approach is the first of its kind in applying clustering and topic-identification techniques to calculate review coverage. Some important contributions of this work are listed below. A list of our findings have been provided in Table 5.3.

1. A word-order graph representation with an agglomerative clustering approach to identify topic clusters in a submission.

2. A heuristic to identify topic-representative sentences from each cluster.
Table 5.3: Summary of findings—Coverage

<table>
<thead>
<tr>
<th>Finding</th>
<th>Most relevant previous strategy</th>
<th>Dataset</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>§5.3.1</td>
<td>Good correlation between system-generated and human coverage values</td>
<td>MEAD [60]</td>
<td>Expertiza</td>
</tr>
<tr>
<td>§5.3.2</td>
<td>Good correlation of system-generated topic sentences with human references</td>
<td>Ganesan et al. [65]</td>
<td>Opinosis</td>
</tr>
</tbody>
</table>

5.5 Conclusion

Since reviews are central to the process of assessment it is important to ensure that they cover the main points of a submission. Reviews of technical articles or documents must be thorough in discussing their content. At times a review may be based on just one section in a document, say the Introduction. Review coverage is the extent to which a review covers the “important topics” in a document. In this chapter we have explained our approach to solving the problem of automatically determining a review’s coverage of a submission. We use an agglomerative clustering technique to group the submission’s sentences into topic clusters. We identify topic sentences from these clusters, and calculate review coverage in terms of the overlaps between the review and the submission’s topic sentences.

We evaluate our coverage identification approach on peer-review data from Expertiza, a collaborative, web-based learning application. Our approach produces a good correlation of 0.51 with human-provided coverage values. We also evaluated the goodness of our approach by generating topic sentences for data from the Opinosis dataset. A good correlation was observed between the topic sentences generated by our approach and the human references.

Future work: Coverage analysis could be extended to the domain of e-commerce reviews (e.g., Amazon, TripAdvisor). Coverage analysis of a product’s description by a review may be useful in determining whether the product review discusses relevant content. Coverage analysis may be useful in identifying reviews that provide a complete description of a product’s features. Users may benefit from reading complete reviews, rather than having to spend time reading and identifying which reviews are useful.

In the next chapter we discuss the user experience survey conducted to evaluate the usefulness of the automated metareview feature.
Chapter 6

A User Study on the Automated Assessment of Reviews

Metareviewing, as defined earlier, is the process of reviewing reviews. In this work we use a specific set of metrics to carry out the review of reviews. Some of these metrics (discussed in earlier chapters) include a review’s relevance to the work under review (or the submission), the type of content a review contains, coverage of the important points discussed in the review, tone of the review and a quantity of feedback provided. We have integrated the automated metareview feature (with the listed set of metrics) into Expertiza. The purpose of this feature is to provide instantaneous feedback to reviewers. A screenshot of the output is shown in Figure 6.1.

We decided to study the experience of using an automated metareview system, since different types of reviewers, students, teaching assistants and faculty, may use this feature. We study the extent to which users of an automated quality assessment system would perceive it to be useful. The study is important because it will help us understand whether reviewers learn and benefit from such an automated metareview system. This study would also help us learn what aspects of the feature can be improved, by identifying what the surveyed reviewers liked or disliked about the feature. A positive experience from using this feature may mean that reviewers would be more inclined to use it to improve their reviews.

According to Kuniavsky [73], user experience is “the totality of end-users” perceptions as they interact with a product or service. These perceptions include effectiveness (how good is the result?), efficiency (how fast or cheap is it?), emotional satisfaction (how good does it feel?), and the quality of the relationship with the entity that created the product or service (what expectations does it create for subsequent interactions?).” There exist several other definitions for the term user experience (abbreviated as UX) [74]. UXMatters 1 defines user experience as that which “Encompasses all aspects of a digital product that users experience directly—and perceive, learn, and use—including its form, behavior, and content.” They also state that “Learnability, usability, usefulness, and aesthetic appeal are key factors

---

1UXMatters: User experience definition http://www.uxmatters.com/glossary/
in users’ experience of a product.” Therefore, apart from a study of factors such as user’s perceptions, feelings or responses to a system, a user experience survey should also involve a study of the learning gained from a system and the usefulness of a system.

The aim of this study is to identify the degree of importance participants attach to each of the metareview metrics. This study will help us identify how effective the system is at helping reviewers learn about characteristics of their reviews. The rest of this chapter is organized as follows: Section 6.1 discusses the design of our user study including steps involved in recruiting participants and collecting data, Section 6.2 discusses our user experience questionnaire. Section 6.3 provides an analysis of the data. Section 6.4 discusses some factors that may pose threats to our study’s validity, and finally Section 6.6 concludes the chapter.

### 6.1 The Study

To study the usefulness of our review quality assessment system we investigate the following broad research questions:

Figure 6.1: Output from the automated feedback assessment feature on Expertiza [1].
RQ1: Do automated metareviews provide useful feedback?
RQ2: Which of the review quality metrics are more or less important than the others?
RQ3: Which of the review quality metrics’ output did the reviewers find more or less useful when compared to the others?

6.1.1 Participants

In order to identify how useful users of the automated metareview feature find it to be, we recruited 24 participants to (1) use the feature on Expertiza and (2) provide us with information on their experience by filling out a survey. Participants were recruited with an email message, which explained to them the purpose of the study. The set of participants included 15 doctoral students, 3 masters’ students and 1 undergraduate student, all of whom were from the computer science department at North Carolina State University, and 5 research scientists from academia and industry.

6.1.2 Data collection

Our data-collection process involved two steps. In the first step, participants were asked to use the automated metareview feature on Expertiza. They use the system to write a review for an article. For our study, we chose a wiki article on “Software Extensibility”\(^2\). We chose this article since we were recruiting subjects from the field of computer science, and Software Extensibility is a topic most computer science students or researchers are familiar with. A detailed set of instructions was provided to each of the participants to help them complete the study (Table 6.1).

A review rubric is provided to the participants to help them write the review. The rubric contains questions on the organization, originality, clarity and coverage of the article under review. The rubric also evokes information on quality of the definitions, examples and links found in the article.

When participants submit their reviews, they are presented with automated feedback from our system. This feedback gives them information on different aspects of their review such as (1) content type, (2) relevance of the review to the article, (3) tone, (4) quantity of text and (5) presence of plagiarism. A screenshot of the output can be seen in Figure 6.1. The participant reviewer reads and understands the metareview feedback.

In the second step of data collection, the participant reviewer is asked to fill out a user-experience questionnaire (Step 8 in Table 6.1). The user-experience questionnaire is an important part of this study, and has been explained in detail in Section 6.2.

\(^2\)Software Extensibility: https://en.wikipedia.org/wiki/Extensibility
Table 6.1: Detailed set of instructions to help complete the survey

1. Use username/password to log into Expertiza.
2. Click on assignment “User Study”
3. Click on “Others’ Work” (Since you will be reviewing someone else’s work.)
4. Click on “Begin” to start the review.
5. Click the url under the “Hyperlinks” section. Read the article on Software Extensibility. Please keep in mind that you are reviewing this article.
6. Answer questions on the review rubric describing the quality of the article you read. After answering all the review questions, click on the “Save Review” button.
7. Wait for a few minutes for the system to generate the automated feedback.
8. Fill out the user-experience questionnaire.

6.2 User Experience Questionnaire

The user-experience questionnaire consists of four sections: participant background, importance of reviews, importance of metrics, usefulness of system’s output. The questions we use in our user-experience survey are discussed in the following sections. Answers to each of these questions are given on a scale of 1 (lowest) to 5 (highest).

6.2.1 Participant background

In the background section, participants were questioned about their experience in writing reviews, and in their experience with using peer-review systems such as Expertiza. The exact questions were:

Q1: Do you have prior reviewing experience?
Q2: Do you have prior experience using the Expertiza system?
Q3: Have you used a peer-review system before?
Q4: Are you a(n): Undergraduate, Masters or PhD student, or Other?

6.2.2 Importance of reviews and metareviews

In the importance section, we questioned participants on the importance of reviews and metareviews to a system.

Q5: How important do you think reviews are in a decision-making process?
Q6: How important do you think metareviews (review of a review) are in a decision-making process?

Answers are given on a 5-point scale: unimportant, somewhat important, neutral, important and extremely important. This section also includes an open question to gather text feedback from participants. All these questions are optional, i.e., the participant may choose not to respond to any of them.

We also gauge whether participants would be motivated to use reviews to improve the quality of their submission (as an author), and metareviews to improve the quality of their reviews (as a reviewer). We therefore included the following questions in the questionnaire:

Q7: Would better reviews inspire you to use the feedback in your revisions?
Q8: Would automated metareviews motivate you to update your reviews?
Q9: Do the automated metareviews provide useful feedback?

6.2.3 Importance of metareview metrics

In the importance of metrics section we identify how important participants think the different metareview metrics are in gauging the quality of a review.

Q10: How important do you think each of the review quality metrics is in learning about the quality of your review? 1. Review relevance, 2. Review content 3. Tone 4. Quantity 5. Plagiarism

The answers are given on a 5-point scale. This question helps us identify the metrics to which users of the system attach most importance, or to which ones they attach the least importance. This section also allows participants to provide any additional comments—which helps us learn about the participants' opinions of the different metrics and other related information.

6.2.4 Usefulness of system’s metareview output

This section helps us study the usefulness of the system’s outputs. These questions gauge whether reviewers learned something about their review’s quality from the automated feedback.

Q11: How useful do you think the output from each of the review quality metrics is (from what you saw on Expertiza)? 1. Relevance, 2. Review content 3. Tone 4. Quantity 5. Plagiarism

Answers are given on a 5-point scale with values: not useful, somewhat useful, neutral, useful or extremely useful. The ratings indicate usefulness of the chosen design for the system’s output. These questions help us learn whether participants are able to successfully comprehend the meaning of the system’s output. This information coupled with the information from the previous question on importance of metrics will help us identify the set of metrics that need improving. This section also includes
an open question to gather any other comments participants may have on the system’s output.

6.2.5 Other metrics

We included an open question on the survey to identify other review quality metrics the participants think would be useful in an automated metareview system.

Q12: What other information do you think might help you improve your review quality? Are there any specific review features you would like to get feedback on? E.g. language of the review, grammar, vocabulary etc.

The next section discusses our analyses on the collected data.

6.3 Analysis of Data

In this section we discuss some of the findings from our data. Out of the 24 participants, 19 had prior reviewing experience. Only 7 of the participants had prior experience with the Expertiza system.

6.3.1 Importance of reviews and metareviews

All of the participants agreed that reviews play an important role in the decision-making process (Figure 6.2). A majority of the participants also agreed on the importance of metareviews (review of reviews). One participant did not respond to these questions.

We asked participants whether high-quality reviews would motivate them to fix their submission. All participants agreed (7 agreed strongly) that they would incorporate suggestions from the feedback in their work (Figure 6.3). We asked participants whether automated feedback on their reviews would
inspire them to improve their reviews. Out of the 24 participants 13 agreed that they would use the automated feedback. However 8 participants displayed doubt in the use of automated metareview feedback answering “neither agree nor disagree” to this question. A small number said that they would not be inclined to use the automated metareview feedback to improve their reviews.

As authors, participants agree that high-quality feedback would motivate them to fix their work, but as reviewers they may not be inclined to use metareview feedback to update their reviews (and help other authors improve their work). The concept of automated assessment of reviews is new, and a lack of understanding of the purpose of these metrics could be one of the reasons why reviewers may not feel motivated to use metareview feedback to fix their reviews.

6.3.2 Importance of the review quality metrics

We analyze how participants judge each of the automated metrics’ importance. The results are displayed in Figure 6.4. The metric that participants rated as the most important is relevance. Out of the 24 participants 23 agree that relevance is important in assessing the quality of a review (3 thought it was extremely important). The next most important metric was found to be review content, with 21 of the participants agreeing on its importance (3 thought it was extremely important).

Participants found quantity to be the least important metric, with 9 of them expressing doubts on its usefulness (neither important nor unimportant) and 4 of them describing it as somewhat unimportant. The Wilcoxon rank-sum test is used to determine whether two metrics’ ratings have identical distributions (null hypothesis) [75]. We use this test to compare metric quantity with metrics relevance and content (which have been identified as the most important metrics) at the 0.05 significance level. The p value for the test on metrics quantity and relevance is 0.0003, and for metrics quantity and content
Figure 6.4: Participants’ ratings of the importance of each review quality metric.

...is 0.002. Since these \( p \) values are < 0.05, we conclude that the ratings of quantity are significantly different from those of the most important metrics—relevance and content.

Quantity is the number of unique tokens in a review text, and is meant to motivate reviewers to write more feedback. Quantity may be obvious to reviewers, since they are aware of the amount of feedback they have provided. Hence quantity may turn out to be the least effective, when compared with the other metrics, in conveying any new information to the reviewer. This could be why quantity is ranked as the least important quality metric.

6.3.3 Usefulness of system output

We questioned participants on the usefulness of the system’s metareview output, to study how informative or understandable they find it. The results of studying usefulness of metrics are displayed in Figure 6.5. The metrics participants rated as most useful are plagiarism and review content, with 17 of participants (9 found plagiarism extremely useful, and 2 found content extremely useful) agreeing that these metrics were useful in helping them understand where their reviews are lacking.

Tone is the second most useful metric with 16 of the participants agreeing on its usefulness, despite 8 participants judging it to be neither important nor unimportant (from previous section). Similarly in the case of quantity, 13 of the participants found the system’s output for quantity to be useful (2 of them thought it was extremely useful), although 9 of the participants said that they thought it to be neither important nor unimportant (Figure 6.4).

We use the Wilcoxon test (at a significance level of 0.05) to determine whether there is a significant difference (increase) in the distribution of the importance and usefulness ratings of quantity. We selected pairs whose ratings for usefulness were higher than their corresponding importance ratings. The ratings have a \( p \) value of 0.03 < 0.05, which indicates that the usefulness ratings of quantity are significantly
higher than their importance ratings. Therefore although participants thought quantity might not be important to a metareview assessment system they found the system’s output for this metric to be useful.

Despite being judged as the most important review assessment metric, the output of the relevance metric was considered useful by only 12 of the participants. One of the participants expressed difficulty in interpreting the meaning of the relevance score. Our metareview feedback contains only real-valued scores in the range 0–1, which may not have been very useful to the reviewer in understanding the degree of relevance. This could have caused the relevance’s usefulness ratings to be lower when compared to the ratings of metrics such as plagiarism, which has a Boolean (true or false) output.

In the future we are planning to improve the format of the output by providing textual feedback in addition to the numeric feedback. The feedback will point to specific instances of a review that need improvement. This may make it easy for reviewers to interpret the numeric score, and maybe further motivate reviewers to use the information to improve their reviews.

6.3.4 Other metrics

Some other metrics in which participants expressed interest are the grammar and syntax of reviews. One of the participants suggested the use of sentence structure variability across sentences as a means of assessing a review. The participant suggested that though short phrases may succeed in communicating the idea, they may not succeed in conveying the idea completely. The presence of well-structured sentences in a review may help the author comprehend the content of a review with ease. Well-structured sentences also indicate to authors that the reviewer put in a lot of thought and effort into writing the review. This is similar to the case of another suggested metric—word complexity.
Another metric suggested by a participant is text cohesion. Reviews sometimes contain a set of sentences that may appear to be disconnected, i.e., lack a meaningful flow from one sentence to the next. Cohesive text helps make reading and understanding reviews easier.

### 6.3.5 Usefulness of the overall automated assessment feature

We surveyed participants on the usefulness of the overall automated feedback system. Out of 24 participants 15 agreed that the feedback was useful (Figure 6.3), and 8 neither agreed nor disagreed.

One of the participants expressed concern with the use of plagiarism as a metric to assess reviews. This is likely because the participant did not see the motivation for a reviewer to plagiarize while writing reviews. Students using Expertiza are evaluated (given scores) based on the quality of the reviews they write. Hence they do have a motivation to copy either good quality reviews (available online) or chunks of text from the submission and submit them as a good quality review. Plagiarism could be caught by manual metareviewers, but may be missed by an automated system. Hence we have this additional feature to ensure that reviewers do not try to game the system by copying reviews.

### 6.4 Threats to Validity

During the evaluation we noticed that a majority of the participants did not have prior experience in using Expertiza, which could have affected their overall performance.

We also learned, from the comments section of the questionnaire, that a few of the participants did not fully understand the meaning of some of the metrics. An understanding of the purpose of the metareview metrics is essential to assessing their importance and the output’s usefulness. Hence, a lack of complete understanding of the metrics may pose as a threat to our results.

Four of the participants failed to provide any textual reviews, which caused the system to output a value of 0 for each of the metareview metrics. Reviewers may not be able to judge the usefulness of metrics’ outputs for which they have received a score of 0. These are some of the threats to the validity
of our results.

### 6.5 Contributions

Some important findings of this user study are listed below.

1. Participants thought review relevance to be the most important metric when compared with the others (§6.3.2).

2. Participants found the system’s output for metrics content type and plagiarism to be most informative when compared to the others (§6.3.3).

### 6.6 Conclusion

Assessment of reviews is an important problem in education, science and human resources, and so it is worthy of serious attention. In this chapter we investigate the usefulness of metrics such as review relevance, content type, tone, quantity and plagiarism in determining the quality of reviews. We surveyed 24 participants, who used the metareview feature on Expertiza, to determine the importance of metrics and usefulness of the review quality assessment’s output. The aim of the study was to identify reviewers’ perception of the usefulness of the metareview feature and its different metrics. Results suggest that participants find relevance to be the most important and quantity to be the least important metric in determining a review’s quality. Participants also found the system’s feedback from metrics such as content type and plagiarism to be most useful and informative.
Chapter 7

Assessment of Project Reviews

A good understanding of a review’s content is necessary in order to provide feedback on the quality of a review. For reviews written for an article or paper, the feedback is more likely to provide a summary of the author’s work [76], or offer praise or criticism. An example of a review written for a paper is “The example for delegation is taken from one of the references listed at the bottom of the page.” This review is critical of the author’s work and implies that an example on the topic “delegation” had been copied from another source. We see that the content of this review is directly linked to the content of the submission.

In the earlier chapters we discussed metrics that help determine the quality of reviews of text documents (referred to as text reviews in this chapter to differentiate them from reviews of projects). In this chapter we focus on the study of project reviews. Project reviews point out issues in the design, implementation or in the testing of a project. For example, consider the following review, “The system controller has the bulk of the functionality. It contains functions related to users, posts, replies and votes. Could have segregated these into separate sub-controllers.” This review talks about the way the project has been implemented and the distribution of functionality across modules.

An important difference between text and project reviews is that while text reviews tend to directly reference content in the author’s submission, project reviews tend to discuss the way in which the project was implemented (e.g. code, syntax, design, architecture). Project reviews may include more references

Table 7.1: This rubric assesses the quality of the code based on its implementation, comments provided and ease of understanding.

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Has this team avoided duplication of code?</td>
</tr>
<tr>
<td>2</td>
<td>Has this team incorporated design patterns into its code?</td>
</tr>
<tr>
<td>3</td>
<td>Has this team provided adequate comments in their code?</td>
</tr>
<tr>
<td>4</td>
<td>On a scale of 1 (worst) to 5 (best), how easy is it to understand the code?</td>
</tr>
</tbody>
</table>
Table 7.2: The following rubric focuses on the design aspects of the project.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Have the authors adequately explained the changes to be made to the system?</td>
</tr>
<tr>
<td>2</td>
<td>Does the design appear to be sound, following appropriate principles and using appropriate patterns?</td>
</tr>
<tr>
<td>3</td>
<td>Does the design appear to be as simple as possible, given the requirements?</td>
</tr>
<tr>
<td>4</td>
<td>Do the class diagram and/or other figures or text that clearly describe the changes to be made to the system?</td>
</tr>
</tbody>
</table>

to the code or design of the project than to the functionality of the project. The content of project reviews is not restricted by the content in software documents (requirements, design etc.). As a result, a metric such as review relevance (discussed in Chapter 3) may not be suited to studying the quality of project reviews, since relevance cannot be inferred from a comparison of the project review and a description of the project.

However, there are project reviews that are evaluative in nature i.e., they provide positive or negative criticism of the implementation or design of the project. For instance consider the review “Yes, good Object Oriented style has been made use of, and the right MVC architecture has been employed.” This review is praising the author’s choice of object-oriented patterns. Hence the tone metric may apply to project reviews. Thus although there appear to be differences in the content and the way they are written, project reviews and text reviews may have some similarities.

In this chapter we focus on project reviews and try to find metrics that suitably capture their quality. We study the similarities and differences between the two types of reviews, and try to identify what metrics are more likely to work for project reviews.

This chapter is organized as follows. Section 7.1 discusses the data used to perform our analyses. Section 7.2 describes the characteristics of project reviews. Section 7.3 contains a description of our approach, which involves studying similarities and differences between the two types of reviews and ultimately identifying the set of metrics that uniquely represent project reviews. In section 7.4 we study the performance of each of the identified metrics by conducting experiments to predict metareview scores. Section 7.6 concludes the chapter with a summary of our work.

### 7.1 Peer Reviews

Peer-review data for our analyses is collected from Expertiza [1]. Different types of assignments including coding projects are hosted on Expertiza. Expertiza uses review rubrics to collect feedback from students and instructors. The feedback contains textual comments and numeric scores.

Some examples of project-review rubrics used in Expertiza are listed in Tables 7.1 and 7.2. As can be seen from these tables, the questions focus more on how the project was implemented (e.g.
Table 7.3: Rubric used for submissions of technical articles or papers.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Do the page(s) stick to the topic?</td>
</tr>
<tr>
<td>2</td>
<td>Are there an appropriate number of links to outside sources?</td>
</tr>
<tr>
<td>3</td>
<td>Does the analysis clearly identify the ethical issues?</td>
</tr>
<tr>
<td>4</td>
<td>Do the page(s) treat differing viewpoints fairly?</td>
</tr>
<tr>
<td>5</td>
<td>Is the organization of page(s) logical?</td>
</tr>
<tr>
<td>6</td>
<td>Do the page(s) identify several issues that are important in learning about the topic?</td>
</tr>
</tbody>
</table>

code duplication, use of design patterns, presence of comments, class diagrams) rather than the specific functionalities of the project. Table 7.3 lists a text review rubric used to gather feedback for a technical article. Table 7.3 focuses on content (what) of the technical article (e.g. analysis of topics, discussion of differing viewpoints). Although one question in this rubric seeks information on the page’s organization (how), the responses are more likely to contain references to content in the article itself.

### 7.2 Project Review Metrics

To study factors that are likely to differentiate project reviews from text reviews, we utilized a graph-based similarity matching technique to extract the most important patterns from each of the review sets (approach discussed in Chapter 4). Patterns are word phrases that capture the most frequent and semantically important subject–verb, verb–object, subject/object–adjective and verb–adverb relationships in reviews.

We use 662 text reviews written for technical articles to identify text patterns. We collected project review data from two software projects. Reviewers are presented with links to authors’ projects, which include code and design documents. These software artifacts are reviewed with the help of the review rubrics (Tables 7.1 and 7.2). The two development projects when taken together had 1427 reviews.

Some patterns for text and project reviews are listed in Table 7.4. In the case of patterns from text reviews we see that the focus is directly on the objects in the article such as ethics, utilitarianism, sentence definition and basic architectures. We can also see that these reviews contain explicit praise and criticism e.g. “explanatory”, “distinct”, “is copied”. However, we can see that project reviews include discussion of code duplication, references to specific helper functions, the test suite and test cases and even some language specific tools such as capybara and rspec.

For project reviews we identify metrics that are similar to those used to study text reviews. Project reviews are less likely to contain summaries of authors’ work i.e., a description of the project’s functionality. However, project reviews may identify problems or offer possible solutions. Hence content categories problem detection and advisory may be useful in assessing project reviews. In addition to
Table 7.4: Frequent and semantically important patterns captured from the set of project reviews and text reviews.

<table>
<thead>
<tr>
<th>Patterns from project reviews</th>
<th>Patterns from text reviews</th>
</tr>
</thead>
<tbody>
<tr>
<td>seem have avoided–lot code duplication through use helper functions</td>
<td>ethics links–are</td>
</tr>
<tr>
<td>seem have been added rounding–test suite bit</td>
<td>sentence definition callback–is copied</td>
</tr>
<tr>
<td>functionality test cases–have been added</td>
<td>basics architectures–could have been provided</td>
</tr>
<tr>
<td>use Capybara rspec–user admin username password</td>
<td>diagrams–explanatory</td>
</tr>
</tbody>
</table>

that we introduce a project-review specific metric that helps identify the degree to which a review discusses the project or code. We refer to this metric as project review content.

From Table 7.4, we see that the prominent project review patterns contain references to code and functionality. They do not appear to explicitly contain words with a positive or negative semantic orientation i.e., the patterns seem to indicate that project reviews may contain a neutral tone. Therefore tone may not play an important role in evaluating project reviews. However, quantity might be necessary to determine whether a reviewer has provided sufficient feedback to the author. Table 7.5 lists some of the metrics that help represent project reviews.

### 7.3 Approach to Calculate Project Review Metrics

We use a supervised text-classification technique called latent semantic analysis (LSA) to determine the content of project reviews [77]. LSA produces a succinct representation of a term-document matrix in a space of reduced dimensions. We then apply the cosine metric to identify the document that is closest to a new review. The closest document’s class is used to identify the content category of the new review. LSA is used to calculate metrics problem detection and advisory content, as well as project review content.

Content is identified using a supervised approach, by training a model on a set of reviews annotated as problem detection or advisory (categories used for annotation–none, problem detection and advisory).

Project review content is identified by comparing the reviews with patterns identified from a set of existing project reviews. The degree of match with patterns indicates the extent to which these reviews discuss the project.

Quantity is a count of the unique tokens in a review.
Table 7.5: Metrics that suitably represent project reviews.

<table>
<thead>
<tr>
<th>Metric</th>
<th>Description</th>
</tr>
</thead>
</table>
| Content             | Identifies the type of content a review contains.  
– problem detection reviews identify problems in the author’s work and  
– advisory reviews offer suggestions to the author on ways of improving the submission.|
| Project review content | Identifies project specific content e.g. references to design, code, software tools.                                                   |
| Quantity            | Identifies the number of tokens a review contains and helps identify the quantity of feedback provided.                                    |

7.3.1 Determining metareview scores

We use project and text quality metrics to determine metareview scores for new reviews. Review vectors are formed with the project or text review metrics. Metareview scores of new reviews are identified by comparing (using cosine similarity) review vectors of new reviews with those of existing metareviewed reviews. The closest review’s metareview score is used to assign the metareview score of a new review.

7.4 Evaluation

We carried out the following experiment to identify the effectiveness of the metrics we identified for project reviews. We compare the use of a new set of metrics with text reviews’ metrics (content, tone and quantity) to predict metareview scores.

7.4.1 Data and method

Reviews on Expertiza are manually metareviewed and are given metareview scores. Metareview scores are typically given on a Likert scale with values from 1 to 5, where 1 is the lowest and 5 is the highest. The metareview score of a complete review is determined by taking the average of metareview scores awarded to each (metareview) rubric question\(^1\).

From a set of 1427 reviews we selected 636 annotated reviews for training, to identify content type. We use a set of 82 semantic patterns mined from the project reviews to identify the project review content. We use LSA and cosine to identify problem detection and advisory content and the degree to which reviews discuss project-related content.

A complete review response includes all responses provided by a reviewer to an author i.e., one completed review rubric. 1427 reviews constitute around 213 complete review responses (for a rubric

\(^1\)The reviews were segmented as part of a pre-processing step and hence the large number of reviews.
Table 7.6: Accuracies of predicting metareview scores for project reviews using different sets of metrics.

<table>
<thead>
<tr>
<th>Metrics Used</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content, tone and quantity</td>
<td>57.8%</td>
</tr>
<tr>
<td>Problem detection and advisory content, project review content and quantity</td>
<td>84.3%</td>
</tr>
</tbody>
</table>

with 6 questions). This set is divided into 149 training and 64 testing. Metareview scores of test reviews are predicted using the metareview scores of the training reviews.

### 7.4.2 Results and analysis

We predict accuracy by determining whether the predicted metareview scores are within 1 unit of the actual metareview scores, i.e., \( |\text{predicted metareview score} - \text{actual metareview score}| \leq 1 \). Since metareview scores are on a scale of 1–5 looking for an exact match between the predicted and actual values might be too constraining. A difference of 1 between the predicted and actual metareview scores is a permissible difference. If the compared metareview scores are 4 and 5, they are treated equal, since both 4 and 5 are high metareview scores with a difference of 1. However if the actual and predicted metareview scores are 2 (low) and 4 (high), they are not treated as being the same. Therefore we use the above condition to study the extent to which predicted metareview scores agree with those given by metareviewers.

Table 7.6 lists the accuracy values of predicting metareview scores for project reviews using the identified set of metrics, as well as the text reviews. The accuracy we get when using the new set of metrics is 84.3%. The accuracy of predicting metareview scores using the text-review metrics (content, tone and quantity) is 57.8%. Thus we see that for the current training and testing data, the system is able to predict metareview scores with a higher accuracy with the new set of metrics. Some examples of correctly classified project reviews are listed in Table 7.7.

Any review with no feedback should have been given a metareview score of 0. However, we noticed that for 28.1% of the test reviews, which contained no textual feedback, metareviewers had given scores >> 1. The metareviewers in this example tend to be quite generous in awarding scores. Metareviewers must be provided with a suitable rubric that will guide them better during the metareview process. This could ensure more accurate metareview scores.

### 7.5 Contribution

The chief contribution of this work is the introduction of a new set of metrics: problem detection and advisory content, project review content and quantity to predict metareview scores of project reviews.
Table 7.7: Sample project reviews

<table>
<thead>
<tr>
<th>S.No.</th>
<th>Reviews</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>No design patterns involved in this code. I will however require clarification from author regarding the same. The project mainly dealt with performance analysis. So no design patterns were involved. Some more comments could have been provided. Comments are added as requested.</td>
</tr>
<tr>
<td>2</td>
<td>The test cases cover the functionality required. There are different test cases. However, I still would require a clarification from the authors on which tests to look into and a readme file would be great. Strongly advice you to write a README file indicating what is done and where to find the required code and also how to run the tests. I did a bit of digging myself still could not figure out which tests were performed by you and how to review the same.</td>
</tr>
<tr>
<td>3</td>
<td>Have u deployed the code on VCL to run the tests? I currently have my project hosted on VCL and hence can’t take another session. Can u provide me a VCL session to verify the test cases?</td>
</tr>
<tr>
<td>4</td>
<td>It seems like the design patterns that were already followed by expertiza has been retained and also introduced some. Yes. Modules have been made. Some functionality moved to other models. Testing done.</td>
</tr>
</tbody>
</table>

7.6 Conclusion

In this chapter we have identified a new set of metrics to determine the quality of project reviews. We found that project reviews discuss content (e.g. test cases, specific functions, software tools) that may not refer directly to the project description or requirements, in contrast to text reviews, whose content is often directly related to the technical article or paper’s text. We identified problem detection and advisory content, project review content and quantity to be the metrics that best capture the quality of project reviews. We found that project reviews produce higher accuracy values with the new metrics than with metrics content, tone and quantity common to plain text reviews.

In the next chapter we provide a summary of our work and directions for the future.
Chapter 8

Summary and Future Work

Reviews are central to the process of assessment—whether it be assessment of students, employees, computer code, or scientific research. In education, student projects are reviewed and graded by the instructor or teaching assistant. In business, annual or quarterly performance reviews are performed for all employees. In the software industry, code reviews are carried out to gauge the reliability of code before it is released. Journal articles are accepted, and grant proposals funded based on peer reviews. Therefore, review quality assessment is an important problem that needs to be addressed.

While there exist several systems that help students learn course-related material, there are not many systems that help them write good reviews. Computers have helped the peer-review process by providing a convenient way to store and retrieve review information. But they have rarely been used to assess the quality of the review information itself. Since reviewing is a skill that might be of use later on in their career, it is important to help students become better reviewers. Our aim with this work is therefore to develop a robust metareviewing system that provides instantaneous feedback to reviewers on the quality of their reviews. This will help them learn their mistakes, and motivate them to write better reviews.

8.1 Future Directions

In the future we plan on investigating some of the following areas:

- **Improving metareview output:** In order to improve the system’s metareview output we plan to highlight snippets of the review that need to be updated. During the user study, two participants suggested the need for additional information on review content types such as problem detection and solution suggestion. We plan to provide information on specific instances (of the author’s work), which the reviewer needs to read and assess to identify problems or provides suggestions. Also, providing feedback to reviewers with samples of high-quality reviews may help them learn how to write better reviews.
• **Study of improvement in reviewing skills:** We plan to study whether reviewers who get feedback from the system show signs of improvement, i.e., whether their reviewing skill improves with time. This would indicate that reviewers learn from the system’s feedback to provide more specific and more useful reviews to authors.

• **Study of improvement in the quality of submissions:** We would also like to investigate the impact a review-quality assessment system has on the overall quality of the authors’ submissions.

• **Summarizing multiple reviews:** In a peer-review system a single document may receive multiple reviews. Reviews discussing the same items may not provide any new information to the author. In order to avoid overwhelming the author with multiple similar reviews, we could eliminate reviews that appear to be of a poor quality or are redundant. Reviews discussing unique sections of the author’s work could be combined in a way that is useful to the author. Coverage-identification techniques could be applied to identify sections in the author’s paper covered by the different reviews. An abstractive summarization technique may be applied to combine the unique reviews in a meaningful way. A visualization of the parts of the submission that are covered by different reviews may also be useful to the author.

• **Suggesting relevant knowledge resources:** If a review is lacking in content, documents related to the author’s topic domain could be presented to the reviewer to help them learn more about the topic. The problem involves studying relevant knowledge resources (e.g. scientific papers, technical blogs, Wikipedia) that would help a reviewer make meaningful suggestions to the author on how the submission could be improved. This problem may involve identifying the main points in the submission and mapping it to concepts discussed in other papers, blogs or articles, in order to determine the most relevant references to suggest, as useful reading, to the author. Citation network analysis techniques may also be used to identify the most similar and useful references.

• **Other metareview metrics:** We plan on investigating the use of other metrics such as sentence structure, cohesion and word complexity to study a review’s quality. At present our graph-based representations capture sentence structure (e.g. subject-verb-object), but we do not study cohesion across sentences in a review. A study of cohesion may involve exploring other areas of natural language processing such as anaphora resolution [78].

In this report we have explained our approach to solving the problem of automatic review-quality assessment. We use text mining and natural language processing techniques to identify metrics that help determine the quality of a review. We have provided a description of the problem and each of its sub-problems, explained our approach to solving them, and analyzed the results from our different experiments. We have also explained our work in the context of related research in the areas of text quality...
assessment, paraphrase identification, text summarization, topic identification and text classification. Our thesis is that identifying quality of human-authored reviews through computational techniques can yield accurate and timely feedback (on reviews) comparable to or even better than those provided by humans.
References


