Int J Artif Intell Educ (2017) 27:534–581
DOI 10.1007/s40593-016-0132-x

ARTICLE

Automated Assessment of the Quality of Peer Reviews using Natural Language Processing Techniques

Lakshmi Ramachandran[1] · Edward F. Gehringer[2] · Ravi K. Yadav[2]

Published online: 11 January 2017

Abstract A *review* is textual feedback provided by a reviewer to the author of a submitted version. Peer reviews are used in academic publishing and in education to assess student work. While reviews are important to e-commerce sites like Amazon and e-bay, which use them to assess the quality of products and services, our work focuses on academic reviewing. We seek to help reviewers improve the quality of their reviews. One way to measure review quality is through *metareview* or review of reviews. We develop an automated metareview software that provides rapid feedback to reviewers on their assessment of authors' submissions. To measure review quality, we employ metrics such as: review content type, review relevance, review's coverage of a submission, review tone, review volume and review plagiarism (from the submission or from other reviews). We use natural language processing and machine-learning techniques to calculate these metrics. We summarize results from experiments to evaluate our review quality metrics: review content, relevance and coverage, and a study to analyze user perceptions of importance and usefulness of these metrics. Our approaches were evaluated on data from Expertiza and the Scaffolded Writing and Rewriting in the Discipline (SWoRD) project, which are two collaborative web-based learning applications.

Lakshmi Ramachandran is currently working for A9.com, Palo Alto, California.

✉ Lakshmi Ramachandran
imaksha@gmail.com

Edward F. Gehringer
efg@ncsu.edu

Ravi K. Yadav
rkyadav@ncsu.edu

[1] Pearson, Boulder, CO, USA

[2] North Carolina State University, Raleigh, NC, USA

Keywords Intelligent tutoring systems · Collaborative learning · Peer reviews

Introduction

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 Scaffolded Writing and Rewriting in the Discipline (SWoRD—now called Peerceptiv) (Cho and Schunn 2007) and Expertiza (Gehringer 2010) are web-based, peer-review systems, that allows students to exchange ideas and to build shared knowledge. The past few years have witnessed a growth in Massive Open Online Courses (MOOCs) such as Coursera and Udacity, which serve as platforms for web-based collaborative learning. MOOCs require a scalable means of assessment, and for material that cannot be assessed by multiple-choice tests, peer-review fills the bill. Text-based feedback helps authors identify mistakes in their work, and learn how to improve it.

Students learn from giving feedback as well as from receiving it. Rada et al. (1994) 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.

The classroom peer review process is very similar to reviewing articles for scientific journals, where students (reviewers) provide reviews and the instructor (editor) decides on a 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. 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.

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 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

Table 1 Some examples of reviews

Review
1. "The example needs work".
2. "The organization is poor".
3. "The example code for delegation is taken from one of the references listed at the bottom of the page".
4. "I would like to see a better definition/explanation of each technique before getting into the advantages and disadvantages".

out problems in the author's work or provide suggestions for improvement (similar to the last two comments in Table 1). Such a review would help authors understand where their work is lacking.

Metareviewing can be defined as the process of reviewing reviews, i.e., the process of assessing the quality of reviews. Metareviewing is currently a manual process (Gehringer 2010; Kuhne et al. 2010; Wessa and De Rycker 2010) and just as with other manual processes, metareviewing is (a) slow, (b) prone to errors and (c) likely to be inconsistent. Feedback quality can be poor, because of a lack of training or review skills—the same problem that makes metareviewing necessary (Ramachandran and Gehringer 2010).

The assessment of reviews is an important problem in education, as well as science and human resources, and is therefore worthy of serious attention. Figure 1 describes the workflow involved in the peer review assessment process. Student submissions (from Step 1) are reviewed by peers (Step 2), and the reviews are metareviewed by other students (Step 3) to verify the correctness of the reviews. The metareview feedback is provided to reviewers (Step 4), who may fix their reviews and provide the updated feedback to students (Step 5). Our aim is to automate the process of metareviewing (in the dashed box) in order to provide instantaneous feedback to reviewers.

Automatic essay-scoring systems and other intelligent tutoring systems provide automated evaluation of students' work (Burstein et al. 2003; Foltz et al. 2000). However, few (if any) systems automatically give feedback on the quality of reviews written by students. This work aims to develop a system that automatically evaluates student reviews and provides reviewers with metareview feedback to help them write better reviews. This feedback is likely to motivate them to improve the feedback they give to authors. Being automated, it also supplies consistent, bias-free feedback to all reviewers.

Since review comments contain unstructured text, it is important to identify metrics that suitably represent the features of a review. Important attributes of a review include its relevance to the submission, content, coverage, tone, volume of feedback provided and plagiarism (Ramachandran and Gehringer 2011). We use reviews submitted to Expertiza and SWoRD (Nelson and Schunn 2009) to test our metrics. Both Expertiza and SWoRD provide double-blind reviewing for artifacts submitted by students. The review process is double-blind, i.e., the author and reviewer information is anonymized to avoid recognition and possible collusion. Rubrics are provided to

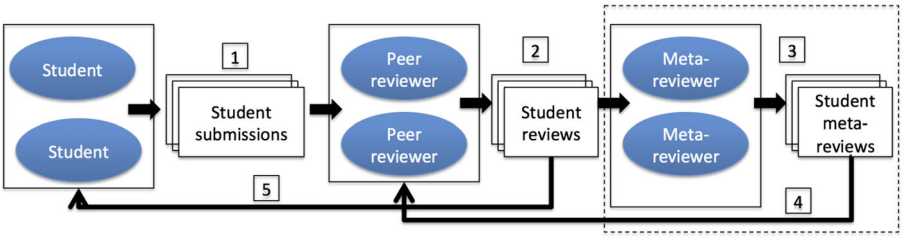


Fig. 1 Workflow of the peer-review assessment process

guide reviewers. Student reviewers provide textual feedback and numeric scores to authors.

Our approach uses word-order graphs to represent review and submission texts. Graph vertices, edges and double edges (two contiguous edges) help capture sentence-structure information. We use a matching technique that exploits contextual similarities to determine relatedness between texts. We use the following set of metrics to identify the quality of student reviews.

- *Review content type* helps identify what type of content a review contains. We focus on three types of review content namely, summary or praise, problem detection (identifying problems in the author's work) and advisory reviews (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.
- *Relevance* helps identify the extent to which a review's content pertains to that of the submission. It helps distinguish generic or vague reviews from useful ones. Relevance is computed using graph matching. Edges and double edges are compared in same and different orders to look for possible paraphrases involving word order shuffling.
- *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.

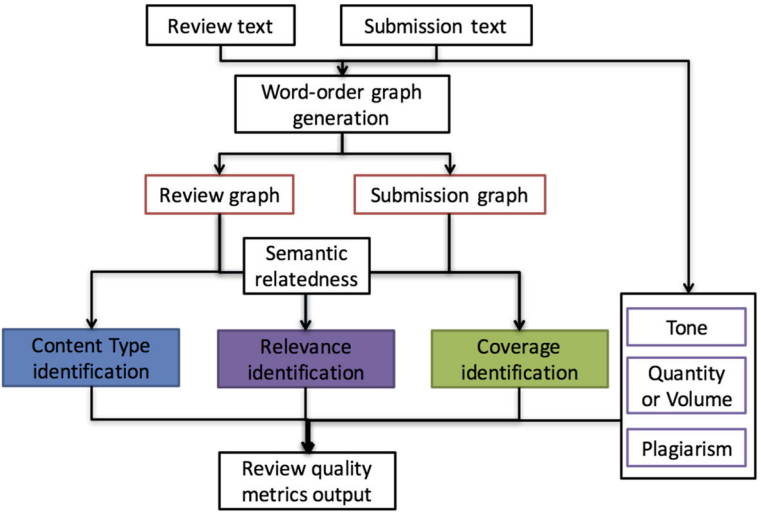


Fig. 2 System architecture–depicts the tools and workflow involved in computing the different review quality metrics

- Some other review quality metrics include: *tone*–presence of positive or negative words in the review, or has provided an objective assessment of the author's work, *volume*–the number of unique tokens a review contains, and *plagiarism*–identifies whether a reviewer copy-pasted text feedback in order to get high ratings.

Figure 2 illustrates the architecture of the system. Inputs to the system include the review, whose quality is to be determined on a set of metrics and the submission for which the review was written. Graphs are generated for the review and submission texts. Graphs and the relatedness metric are used to compute each of the review quality metrics. The computed results are collated and presented to the student (see Fig. 3). Semantic relatedness tool is used to compute matches between graph structures for each of the metrics in the system.

Several of the experimental results have been published piecemeal by the authors in other venues (Ramachandran and Gehringer 2015; 2013a; Yadav 2016; Ramachandran and Gehringer 2013b). This paper pulls together these results and a new study to present and evaluate the complete review-quality feedback system.

Feedback to Reviewers

Figure 3 shows the information that our review-assessment system presents to the reviewer. Our aim is to motivate reviewers to make their review more relevant to the submission, and thus to help authors improve their work. This paper focuses not on the user interface for review feedback, but on the review quality metrics and how they are computed. There is ample room to improve how the feedback is presented (with better charts, for instance). But here we do provide a foundation for reviewers to write reviews that are more consistent and easier for their peers to understand and use.

In this example the review is written for an article on *software extensibility*. The sample review in the figure has a relevance of 0.14 (all metrics are reported on a scale of 0–1). The review has a value of 0.27 for summary or praise content (e.g. "...simple and easy...", "...good examples..."), 0.36 for problem detection (e.g. "...little ambiguity ...") and 0.36 for advisory content (e.g. "...would have been better..."). This gives the reviewer information on the different types of content a review contains.

The sample review has a coverage value of 0.4 for the topic sentences in the submission's text. Parts of the review that cover the submission's topic sentences include "software ...interface", "extensibility", "forward compatibility" and "system architecture". The highlighted phrases and numeric estimates give the authors information on the coverage of their review.

When we began the project, we intended to develop a formative assessment tool to be integrated into Expertiza. We therefore do not compute an overall quality score. However, we could derive a holistic summative score from a linear combination of these metrics. The use of these quality metrics is supported by Yadav's (2016) work showing that most metrics are highly correlated with a review's overall quality.

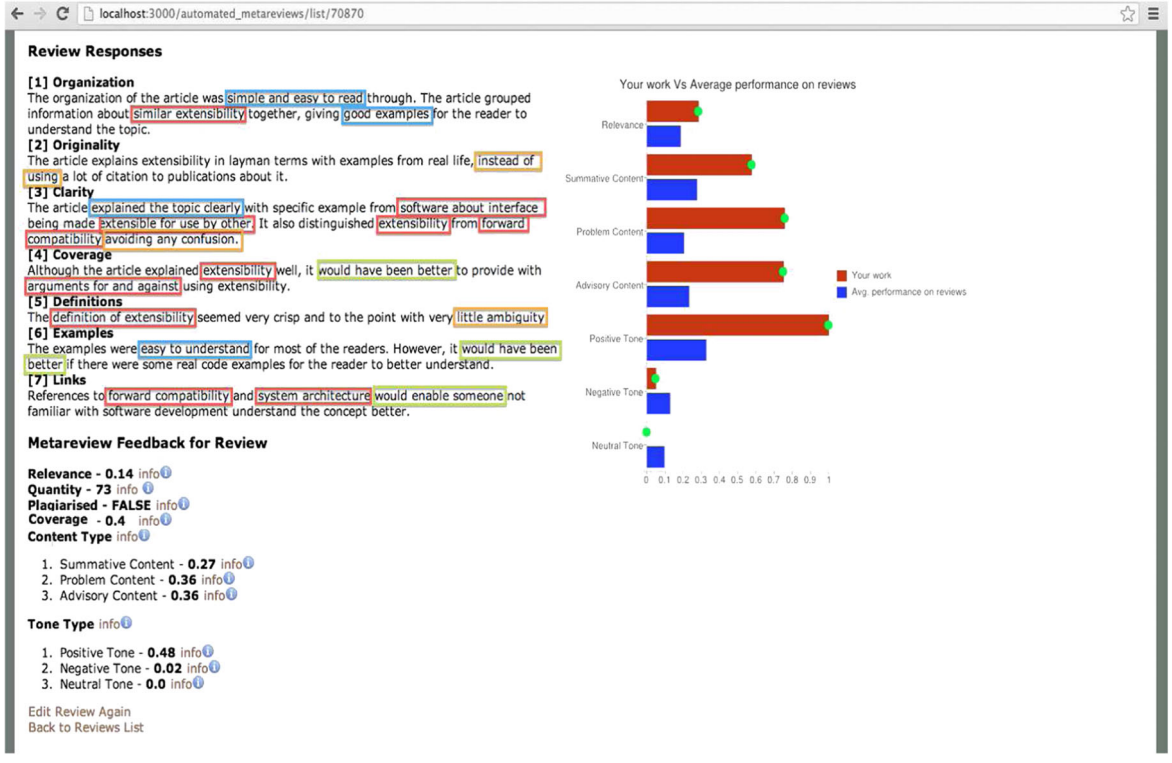


Fig. 3 Output from our review assessment system identifying: relevance to the submission text, volume (i.e. number of tokens in the review), presence of plagiarized text, coverage of the topic points in the submission, content types of the review and tone types of the review. Summary patterns are highlighted in *blue*, problem-detection patterns are highlighted in *orange* and advisory patterns are highlighted in *green*. Parts of the review that cover the submission's topic sentences are highlighted in *red*. The coverage and highlighting features depicted in the figure are yet to be implemented for the system. The bar chart shows how the student's metareview scores compare with average performance across all reviewers in the system

The rest of this paper is organized as follows. The "Related Work" section discusses related work in the domain of automated assessment of reviews. The "Text Representation and Similarity Matching" section discusses the text-representation and semantic-matching technique that aid in computing each of the review quality metrics. The "Review Content Type" section discusses the use of a cohesion-based pattern-identification technique to capture patterns among reviews. The section presents the evaluation of our approach on reviews from Expertiza and SWoRD. The "Review Relevance" section describes the use of a graph-based text-matching approach to determine relevance of a review. The "Review Coverage" section 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. The section includes the evaluation of our coverage identification approach on peer-review data from Expertiza. The "Other Quality Metrics: Tone, Volume and Plagiarism" section discusses some of the other review quality metrics. The section on "Study" presents a user study designed to evaluate our automated review quality assessment system. Section "Summary and Future Work" concludes the paper with a summary of our work and directions for the future.

Related Work

An earlier approach to manually assessing the quality of peer reviews involved the creation and use of a Review Quality Instrument (RQI). Rooyen et al. (1999) 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 whether its claims were substantiated. We incorporate some of these metrics in our approach, e.g., detecting constructiveness in a review based on its content and checking whether a review's claims are substantiated by identifying relevance to the author's submission.

Nelson and Schunn (2009) studied feedback features that help authors understand and use reviews. 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. (2010) use the author's ratings of reviews to measure the quality of peer reviews. They found that authors are content 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 their 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.

Using reviews from the SWoRD system, Xiong et al. (2010) look for problems identified by reviewers in the author's work. They use a bag-of-words exact-match approach to detect problem localization features. They employ 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. Nguyen and Litman (2013) use a patterns-based approach to detect localizations in peer reviews written for argument diagrams. Cho (2008) uses machine-classification techniques such as naïve Bayes, support vector machines (SVM) and decision trees to classify review comments. 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 e-commerce reviews from Amazon and e-Bay among others. Product reviews' helpfulness is determined based on how useful other users of a system find them. Zhang and Tran (2010) determine review helpfulness based on ratings provided by reviewers to a review. Moghaddam et al. (2011)'s work on review helpfulness takes raters' information into consideration. They do not consider review content information while determining helpfulness. By contrast, our approach does not take other reviewers' ratings into consideration. Our approach aims to identify review quality based purely on the textual content of the review.

Other approaches to study the usefulness of reviews are proposed by Turney (2002) and Dalvi et al. (2009) and Titov and McDonald (2008). 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 convey their criticism.

Lim et al. (2010) identify reviewers who target e-commerce products and applications and generate spam reviews. The problem of spamming may be analogous to the problem of copy-pasting text (plagiarism) 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 exists no review assessment system that factors in all these metrics, i.e., relevance, content type, coverage, tone, volume and plagiarism to provide automated metareview feedback. Our aim is to provide a suitable review assessment model that can be used to evaluate student-written reviews, and could potentially be used to assess reviews in other application domains.

Text Representation and Similarity Matching

This section describes the tools—graph-based text representation and the semantic matching technique we use to automatically compute review content type, relevance and coverage metrics, which have been discussed in subsequent sections.

Word Order Graphs

Word-order graphs capture the ordering of words or phrases in text, which helps capture context. Context is not available in a bag-of-words or a dependency tree representation (Bohnet 2010) (which captures only head → modifier relations). Context has been found to be useful for tasks such as sense disambiguation (Lesk 1986).

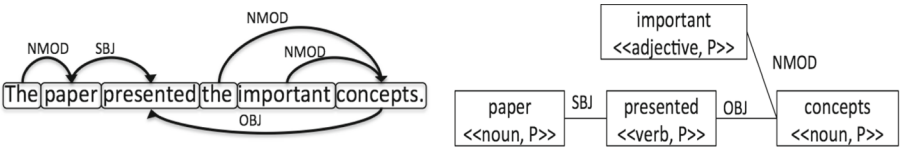
During graph generation, each review is tagged with parts-of-speech (POS) using the Stanford POS tagger (Toutanova et al. 2003). We use a heuristic phrase chunking technique to group consecutive subject components (nouns, prepositions etc.) into a subject vertex, consecutive verbs (or modals) into a verb vertex, and similarly for adverb and adjective vertices. A vertex may therefore contain a phrase or a token.

When a verb vertex is created the algorithm looks for the last created subject vertex to form an edge between the two. Ordering is maintained when an edge is created, i.e., if a subject vertex was formed before a verb vertex a subject—verb edge is created, else a verb—object edge is created. An adjective or an adverb is attached to the subject or verb vertex respectively (i.e., subject—adjective or verb—adverb edge).

Post edge creation, we iterate through all edges to determine whether a dependency exists between the tokens representing the edge's vertices. We add an edge label if a dependency exists, e.g., "concepts—important" in Fig. 4b captures the noun-modifier (NMOD) relation. Labels capture the grammatical role played by tokens in a text. Ramachandran and Gehringer (2012) provide a detailed description on the process of generating word-order graphs.

The *state* (described in detail in the section on "Identifying Semantic Patterns") of each sentence is identified during graph generation and is represented as part of a graph's vertex. State helps determine whether a word or a phrase in the review is being used in a positive, negative or advisory sense, and is useful to identify a review's content type.

Edge labels and state are used to compute semantic relatedness for different metrics. Edge labels play an important role in computing context matches while identifying review relevance. State information plays an important role in computing content type. The way in which graph properties are used by the metrics to compute similarity is described in detail in the following sections.



(a) Dependency tree (b) Word-order graph

Fig. 4 Displaying the difference between a dependency tree and a word-order graph for the text "The paper presented the important concepts". A dependency tree does not capture ordering. For instance, if we read edges of this dependency tree we will get paper → presented, concepts → presented. Word-order graph captures ordering, e.g. paper–presented, important–concepts, presented–concepts

Semantic Relatedness

Match between two tokens can be one of: (1) an exact match, (2) a synonym match, (3) a hypernym or hyponym match (more generic or specific), (4) a meronym or holonym match (sub-part or whole) (5) a common parents match, or (6) overlapping definitions or examples match, or (7) a distinct or non-match. Each match is given a weight value, which represents its *degree of importance*, e.g., exact matches are more important than synonym matches, which are in turn more important than hypernyms or hyponyms and so on. Weight values are in the [0-6] range, 0 being the lowest match (distinct) and 6 the best match (exact). Unlike other approaches, which capture just exact or synonymy matches, our approach captures semantic relatedness between tokens using a few types of matches. Each match is identified using WordNet (Fellbaum 1998). WordNet has been used successfully to measure relatedness by Agirre et al. (2009). An evaluation of this relatedness metric is available in Ramachandran and Gehringer (2013c). This approach to compute relatedness is common to a few of the quality metrics such as review content type, relevance and coverage, which are discussed in detail in the next few sections.

Review Content Type

Reviews that contain only praise are not as useful as those that only contain instances of problems caught by peer reviews, which in turn are less useful than reviews that provide suggestions (Nelson and Schunn 2009). Identifying a review's content type shows reviewers where their reviews are lacking, and thus help them write more effective reviews. A review may contain:

Summary or praise (summative) content Positive feedback or a summary of the author's work. E.g. "The page is organized logically, and gives an example code". The usage of the term summative in this context is different from the word's use in summative assessment.

Problem-detection content Identifies problems in the author's submission. E.g. "The page lacks a qualitative approach. It also lacks an overview".

Advisory content Provides suggestions to the authors on ways to improve their work. E.g. "The page could contain more ethics related links and more in-depth analysis of ethical issues".

The objective is to identify and inform reviewers on the amount of each type of content reviews contain, so that they can provide more feedback when they find their review to be lacking. All metrics (including this one) described in the paper are formative because they provide feedback to reviewers (via numbers, text and charts), but do not produce scores for these reviews that go into computing each student's grade. The aim is to have the review quality analysis feature as a continuous process so that student reviewers can *write, learn, improve* their reviews in an iterative process.

From the above examples for summary, problem-detection and advisory content, we see that they discuss *similar points* (e.g. page organization), but the difference lies in the *way* the points are discussed. For example, summary reviews make positive observations (e.g. "... organized logically..."), while problem-detection reviews identify problems (e.g. "... lacks ... approach ...") and advisory reviews provide suggestions (e.g. "... more ... analysis ...").

Current approaches to automatically identify review content use machine-learning techniques with shallow text features such as counts of nouns and verbs (Cho 2008). Techniques that rely only on token frequencies as features may not succeed in distinguishing content types containing overlapping text. Nguyen and Litman (2014) use sentence-level annotations of reviews to predict the content type of reviews written for argument diagrams. They focus on the use of a fine-grained labeled corpus to help train a model. The paper uses simple linguistic features to train the models.

We identify phrases or clauses that capture the meaning of each review content type. The problem of identifying patterns using lexical cohesion techniques has been explored in other areas such as text summarization and topic identification. Lexical cohesion refers to the semantic relatedness between different parts of a text. Barzilay and Elhadad (1997) use lexical chains to establish links across tokens that are semantically related. They use a cohesion-based approach to identify strong chains or representative sentences in a document, while summarizing it. Radev et al. (2004)'s MEAD uses a centroid-based summarization technique to identify the best sentences to be included in a summary. Erkan and Radev (2004) use a centrality-based technique to determine the main ideas in a document. Sentences of a document are represented as vertices of a graph, and cosine similarity between adjacent sentences identifies the degree of similarity between them. The most similar sentences are considered to be central to the meaning of the document. Similarly, in our patterns-based approach to identify content types of reviews we compute the semantic similarity between all pairs of edges and then select the edges with the highest semantic similarity to represent each content type's patterns (Fig. 5).

Identifying Semantic Patterns

Determining Semantic Similarity Between Graph Edges

Relatedness is measured as the average of the matches between vertices of two compared edges. Similarity between edge A (vertices (A_1, A_2)) and edge B (vertices (B_1, B_2)) is calculated as shown in (1). Match between two tokens is identified using the approach described in section "Semantic Relatedness".

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

Review state State helps identify whether tokens are used in the *negative, advisory* or *positive* sense. State helps identify cases of negation in reviews. Consider the review, "The paper is not clear". An approach that does not handle negation is likely to misclassify this review as a summary. Words such as *none, never, not, won't,*

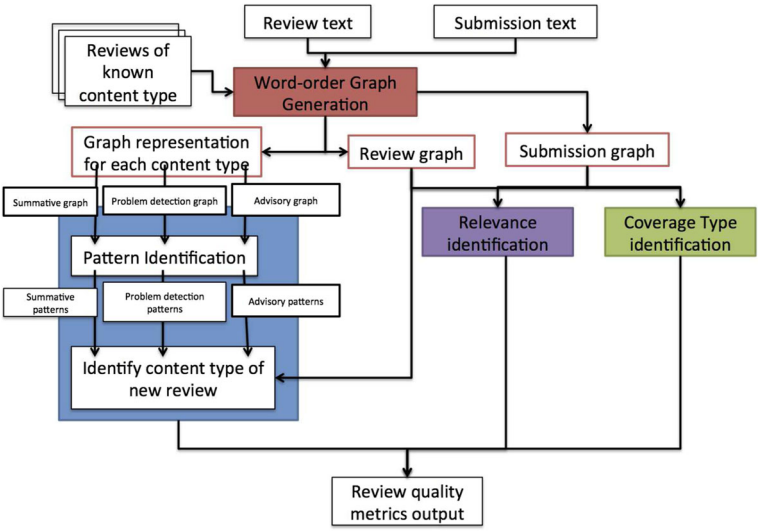


Fig. 5 Review content type identification, system overview

don't, didn't, barely, hardly give the text a negative orientation. Tokens such as *could, should, maybe, perhaps* are indicators of suggestion. Review state plays an important role in determining content type of reviews, but since it is not the main problem we are trying to solve in this paper the evaluation of this heuristic is beyond the scope of this paper.

Chapman et al. (2001) use regular expressions to determine negations of clinical terms in patients' medical records. Manshadi et al. (2013) use negations to determine the scope of ambiguous quantifiers in a text. We apply a rule-based approach to identify state based on tokens and their contexts. Our approach not only identifies negations, but also identifies advisory terms or phrases in reviews.

In order to distinguish between the state of each segment in a review, the reviews are broken down into segments at connectives such as "and" and "but". A segment is assigned a default state until a token or phrase of negative or advisory state is identified.

We use the presence or absence of nouns in between tokens (context) to determine how double negations should be resolved. 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 the negative polarity of the text (Tubau 2008). 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!"

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.

In the case of advisory indicators, context plays an important role in determining state change. Advisory tokens when followed by a negative token results in a change

of state from advisory to negative. In the example "...could not understand...", since the advisory token "could" is followed by "not", the segment gets a negative orientation. However, presence of nouns or verbs between advisory and negative tokens would cause the state to remain advisory. In the case of segment, "I *would suggest* the author to *not* include...", the presence of the noun "author" between the advisory token "would" and the negation "not" causes the sentence to remain a suggestion–advising the author against doing something.

After parsing every token in the segment, the algorithm returns the final state. If no negative or advisory token is identified, the review has a positive state. In Fig. 6 the graph vertices contain state, where *P* represents positive and *N* represents negative state. We manually collected a set of negative indicator words and phrases, found commonly among educational reviews (e.g. "grammatical errors", "off topic", "too short"), from 100 reviews completed using Expertiza. We use additional negative indicators from an opinion lexicon provided by Liu et al. (2005).

When edges are compared, their respective states are compared. If two edges have the same state, then similarity is +value. If the edges have different states, then the similarity is −value to indicate that the word or phrase was used in opposing senses. For example, if two tokens have an exact match but have different states then they get a match value of −6.

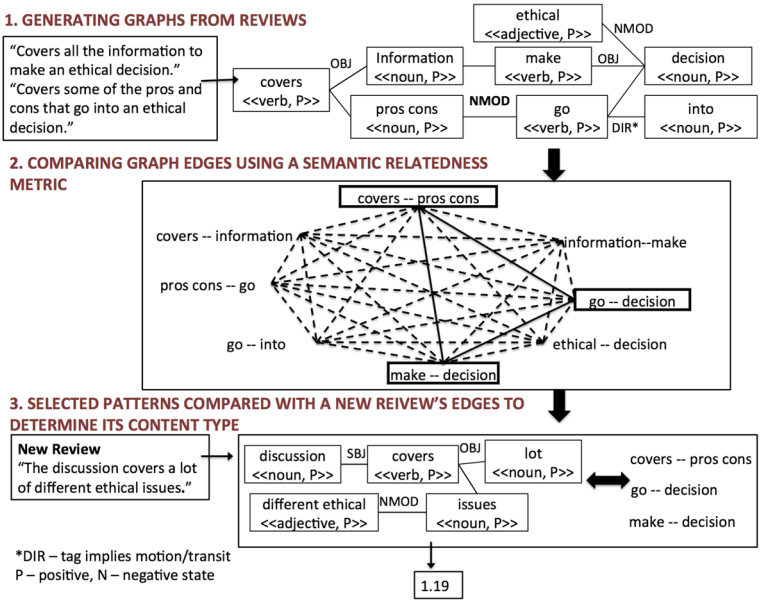


Fig. 6 Illustration of our approach–Steps **1** and **2**: Patterns are identified from sample summary reviews "Covers all the information to make an ethical decision". and "Covers some of the pros and cons that go into an ethical decision". Step **3**: A new review's semantic similarity is identified by comparing it with the generated patterns

Selecting Edge Patterns

Reviews can contain more than one type of content, so for the training process, i.e., for the selection of edge patterns that represent each type of content, we select reviews that contain predominantly one type of content. The aim is to extract strong, representative patterns for each content type.

In Fig. 6 edges from reviews containing summary or praise content are compared. 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 the formula in (2), where E is the set of all edges in the graph. Descriptive edges such as noun – adjective, verb – adverb capture properties of nouns, verbs in a text and these edges help distinguish the *way* in which objects or concepts are discussed in the different review types.

$$\textit{Importance of } e = \frac{1}{|E| - 1} \left(\sum_{\forall f \in E, f \neq e} \textit{similarity}(e, f) \right) \tag{2}$$

Edges that have a high average similarity with other edges that represent the same content type are selected as patterns. We compare patterns generated for each of the three different content types to ensure no overlap exists across the patterns. For instance, if the same pattern occurred among summary and p roblem-detection reviews, it is removed, and only patterns unique to a content type are retained. After this step we select the top 50 patterns from each content type to ensure that the same number of patterns represents every content type.

Table 2 lists some sample edge patterns selected from each of the content classes. We can see that summary or praise patterns are positive and contain a brief description of the page 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.

In Fig. 6 the selected edge patterns are depicted with a thick border (Step 2). Some of the patterns selected from summary or praise reviews are "covers–pros cons", "make–decision" and "go–decision".

Table 2 Sample edge patterns for each review content type

summary or praise	problem detection	advisory
sticks–topic	ambiguous–about what	more–detail
page–discussed	not–covered	could be–bit
parts–original	is typing–mistake for	would benefit–more
good–examples	grammatical–problems	more–depth analysis
issues–are covered	too–cluttered	should be–more
author's prose–easy	not–been	would benefit–more

Identifying Content Type of New Reviews

Content type of a new review is identified by comparing the edges of the new review's graph with the patterns for each content type. We identify the best semantic match for each review edge with a content's patterns. The average of the review edges' matches gives the semantic match between a review and the content's patterns (3).

$$content_C = \frac{1}{|E|} \sum_{\forall e \in E} \left(\underset{\forall p \in P_C}{\text{argmax}} \; similarity(e, p) \right) \tag{3}$$

In (3), $content_C$ represents the degree of match between the new review (with edges E) and patterns of content type C (P_C), where C could be *summary, problem detection* or *advisory*. Patterns are given state values in order to aid matching. Summary or praise patterns have a positive state, problem-detection patterns have a negative state, and advisory patterns are assigned an advisory state.

In Fig. 6 summary patterns are compared with a new review (Step 3). In order to illustrate our approach, we compare a new review with summary patterns. However, in reality new reviews would be compared with problem-detection and advisory patterns too to determine the degree of match with each content type's patterns. The review "The discussion covers a lot of different ethical issues". has summary or praise content-edges such as "covers – lot" and "covers – issues". These edges have a similarity value of 3 with the summary pattern "cover – pros cons", since "covers" has an exact match (of value 6) while the other vertex for both edges ("lot" and "issues") have a distinct or non-match (value of 0) with the pattern and similarity between edges is computed as the average of the vertices' matches (1). The other edges in the review have a 0 similarity match with the summary patterns. This review has a $content_{summary}$ match of 1.5 (3) with the selected patterns. A positive $content_{summary}$ score indicates the presence of praise content in the new review.

Content Type Identification Study

In this section we summarize the study conducted to evaluate the patterns-based approach. A detailed description of the study can be found in Ramachandran and Gehringer (2015).

The question we address with this study is: *Does a patterns-based approach succeed in automatically identifying the content type of reviews?*

For the purpose of evaluating this patterns-based approach, review segments are classified based on its predominant content type. The machine selects content type C, which produces argmax($content_C$) (from (3)). Our approach was evaluated on peer-review data from Expertiza and SWoRD. Both datasets had a high degree of human agreement on the content type of reviews.

Data We evaluated our technique on 1453 academic reviews selected randomly from Expertiza (Gehringer 2010) and on 1048 reviews from the SWoRD project (Patchan et al. 2009), i.e., a total of 2501 reviews. Ten percent of the reviews from Expertiza were annotated by four humans. The annotators were given, by the authors, sample

reviews and their corresponding content types. This was the only training that the annotators received. The average inter-rater agreement between the four annotators was 82 % and the average Kappa was 0.74 (Fleiss et al. 1969). The average Kappa between each of the three raters and a fourth rater was 0.75. A high Kappa indicates that humans agree on the types of content the reviews contain. We provide average numbers because an average gives a composite measure of the degree of agreement across all the raters and we believe this to be a more appropriate number to report in comparison to agreement between pairs of raters. Because of a high degree of agreement, the fourth annotator labeled all reviews, and these labels were used in the pattern learning process.

We obtained annotated SWoRD data from the project's team at the University of Pittsburgh (Patchan et al. 2009). According to the authors the judges coded the data in two steps: (1) they determine the type of feedback (summary, praise, problem/solution) and (2) they distinguish the problem and solution reviews. The Kappa for each of the coding steps was 0.91 and 0.78 respectively (Patchan et al. 2009).

In order to combine the two datasets for our evaluation, SWoRD reviews that were coded as summary or praise were treated as *summative* reviews, and reviews coded as explicit problems are treated as *problem detection* reviews, while those coded as explicit solutions are treated as *advisory* reviews.

The dataset contains a total of 1047 summative, 710 problem-detection and 744 advisory reviews. 1751 reviews were used for training ($\approx$ 70 % of the data) and the remaining 750 for testing. Patterns were extracted from the training set, which are used to identify content type of reviews in the test set.

We calculate our final results using a 5-fold cross-validation. During each run patterns are identified from 4-folds of the dataset and tested on the 5^{th} fold. The results from the five runs are averaged to get the final results listed in Table 3.

Table 3 Average recall, precision and f-measure for the different systems

Approach	Accuracy	Precision	Recall	f-measure
Patterns	67.07 %	0.68	0.66	0.67
SVM, Unigram	35.76 %	0.33	0.34	0.33
LR, Unigram	33.73 %	0.33	0.33	0.33
SVM, Bigram	31.39 %	0.32	0.33	0.32
LR, Bigram	35.07 %	0.33	0.33	0.33
SVM, edges	32.16 %	0.32	0.32	0.32
LR, edges	35.09 %	0.34	0.35	0.34
SVM, tokens+state	35.84 %	0.36	0.36	0.36
LR, tokens+state	36.08 %	0.35	0.35	0.35
SVM, topics	33.79 %	0.34	0.33	0.34
LR, topics	34.45 %	0.32	0.33	0.32

*The differences between precision, recall and f-measure values of the patterns-based approach and the classifiers' results are significant (two-tailed test, p-values < 0.05, thus the null hypothesis that this difference is a chance occurrence may be rejected). *SVM: support vectors, LR: Logistic Regression

Cross-validation ensures that data from both sources go into the training (pattern identification) and testing steps.

We use the statistical analysis tool (R 2008) and the `LiblineaR` package (Fan et al. 2008) to run multi-class SVM and logistic regression. Both SVM and Logistic regression learners are being used as multi-class classifiers. The regularization parameter (cost) is set to 1. We did not notice an improvement in performance with the use of an optimal cost value (tuned using the *heuristicC* function available in `Liblinear`) as the regularization parameter.

The aim of the different machine learning baselines is to determine whether algorithms such as LR and SVM would succeed in learning the patterns from the set of edges, or tokens with state, or unigram-bigram information. Pattern identification is a big part of this metric and we wanted to see whether the learners are as good as our edge-matching technique in identifying patterns. We show that the baseline algorithms were unable to successfully capture patterns in the text, even when presented with more context in the form of edges, tokens with state or topic information.

The optimizer function is softmax (for logistic regression), which produces the probability of predicting each of the content classes. The precision, recall and f-measure values are the macro averages over the content type classes.

Results and Discussion The results from our approach are listed in Table 3. Due to the nature of the different studies we use different metrics for each. We used (i) unigrams, (ii) bigrams, (iii) graph edges, (iv) unigrams tagged with state (positive, negative or advisory) (referred to as tokens+state in Table 3) and (v) topic words generated using Latent Dirichlet Allocation (LDA) (Blei et al. 2003) as features with learning algorithms (1) L1-regularized logistic regression and (2) multi-class support vectors (Joachims 1998; Zhang and Yang 2003; Echeverría et al. 2013; Fan et al. 2008) as our baselines. We demonstrate that word-order graphs together with semantic relatedness metrics produce patterns that are better at identifying the content type of a review than classifiers trained on non-trivial semantic features.

The patterns-based approach's overall increase in accuracy over the best performing baseline model is 30.99 %. It was good at identifying advisory content. Fewer advisory reviews were wrongly classified as summary or problematic reviews by advisory patterns. Problematic reviews, on the other hand, were often misclassified as summary or advisory.

Consider the 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 2). These tokens cause the problem-detection review to be misclassified as an advisory review.

Logistic regression and support vectors performed well for cases where there is a good overlap between the vocabularies of the train and test sets. However, in the case of content type identification the semantics and structural information of reviews play a crucial role. Words such as "easy", "great" and "well-organized" are common among the summary or praise reviews in the training dataset and are weighted highly by the logistic regression models. As a result, reviews containing segments such as

"The prose is not easy to understand", "...just clean up the few errors throughout the paper and you will have a great paper" or "...well organized, but there are several grammatical errors throughout the text that need to be addressed ..." tend to get misclassified as summative reviews. Approaches that focus on exact matches of tokens or edges, that do not take varying degrees of similarity into consideration, may not succeed in capturing patterns that distinguish the review content types.

According to the results in Table 4 our approach also performed well when patterns were extracted (trained) from one dataset and tested on another and vice-versa. Thus the generated patterns are generalizable across datasets and do not appear to be dependent on the topics the reviews discuss.

Table 4 Average recall, precision and f-measure obtained when trained on one data source and tested on a different source

Approach	Accuracy	Precision	Recall	f-measure
Train: Expertiza, Test: SWoRD				
Patterns	62 %	0.66	0.59	0.62
SVM, Unigram	31.88 %	0.47	0.35	0.40
LR, Unigram	41.08 %	0.38	0.38	0.38
SVM, Bigram	32.16 %	0.35	0.29	0.32
LR, Bigram	39.78 %	0.34	0.34	0.34
SVM, edges	31.51 %	0.32	0.32	0.32
LR, edges	38.94 %	0.34	0.34	0.34
SVM, tokens+state	30.11 %	0.32	0.33	0.33
LR, tokens+state	36.80 %	0.35	0.35	0.35
SVM, topics	43.87 %	0.44	0.33	0.38
LR, topics	42.84 %	0.32	0.38	0.34
% of largest class	43.87 %			
Train: SWoRD, Test: Expertiza				
Patterns	66 %	0.70	0.65	0.67
SVM, Unigram	41.64 %	0.42	0.41	0.42
LR, Unigram	43.43 %	0.40	0.40	0.40
SVM, Bigram	30.14 %	0.54	0.42	0.47
LR, Bigram	39.02 %	0.34	0.35	0.34
SVM, edges	31.52 %	0.31	0.32	0.32
LR, edges	39.16 %	0.37	0.35	0.36
SVM, tokens+state	36.41 %	0.36	0.35	0.35
LR, tokens+state	46.73 %	0.47	0.44	0.45
SVM, topics	42.81 %	0.40	0.40	0.40
LR, topics	37.78 %	0.31	0.34	0.32
% of largest class	41.5 %			

*The differences the patterns-based approach's results and the classifiers' results are significant (two-tailed test, p-values < 0.05). *SVM: support vectors, LR: Logistic Regression

Summary: Review Content Type

In this section we have described the importance of identifying the content type of a review, and have outlined the approach for automatically quantifying content type. We summarize results from previous work on evaluating this approach on data from Expertiza and SWoRD (Ramachandran and Gehringer 2015). According to the results:

1. The cohesion-based pattern extraction technique has an f-measure of 0.67, and
2. The approach produces a higher f-measure than support vectors and logistic regression-based classifiers in learning the content type of reviews.

Current approaches to assess review quality do not use semantic patterns to identify a review's content type. Our approach is an interesting and novel addition to the field in that respect. Our approach involves generating newer patterns that adequately capture each content type from new (fresh) review data, at regular intervals. Though this may be a time-consuming process, the patterns generated are likely to better reflect the content type of new reviews.

In the following section we discuss the relevance metric, which is used to determine the degree of relevance between the review's content and the submission.

Review Relevance

A *relevant review* discusses the concepts described in a submission, which may involve some amount of paraphrasing. Our aim is to identify whether a review is relevant to the work it was written for. While paraphrasing, an idea may be restated by the reviewer with possible lexical and syntactic changes to the text. According to Liu et al. (2009) a good paraphrase should contain some syntactic changes, while preserving the original meaning of the text. According to Boonthum (2004), paraphrasing often uses the patterns of lexical synonymy, change in voice and change in sentence structure. Thus, conventional text matching approaches, which look for exact matches, may not be good at identifying relevance.

We do not expect all reviews to contain paraphrases or summaries of the author's submission, however we do expect that the reviewers would discuss the content of the submission within the reviews. While exact matching techniques may be useful they do not capture potential re-writing of terms used by the author. Hence we use a lexico-semantic matching technique to identify relevance.

Definition 1 Let S be the set of sentences in the submission and R be the set of review sentences. Let s and r represent a sentence in the submission and review respectively.

$$relevance(S, R) = \frac{1}{|R|} \sum_{\forall r \in R} \{\underset{\forall s \in S}{\operatorname{argmax}}(lexicoSemSim(s, r))\} \tag{4}$$

lexicoSemSim(*s*,*r*) represents the *lexico-semantic* match between *s* and *r*. Relevance is the average of the best lexico-semantic matches of a review's sentences with corresponding submission sentences. We acknowledge that all review sentences may not have corresponding matches in the submission. Our aim is only to identify the proportion of review text that is lexico-semantically relevant to a submission.

Since our aim is to identify the lexico-semantic match between texts, we 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 in paraphrased text. We perform paraphrase detection by matching graph vertices and edges, which help maintain word order information (as described in the section on "Word Order Graphs"). Graph matching between the review and submission texts helps us identify whether the review references specific concepts in the submission (Fig. 7).

Figure 8 contains a sample submission and three sample reviews. The first review has some instances of exact match with the submission and is therefore relevant to the submission. However, the relevance of the second review may not be determined by a text overlaps match. The third review is not relevant to the article. The review talks about hate speech and censorship, whereas the article discusses the effects of fees imposed on radio stations. This review example is lexico-semantically distinct from the submission, and therefore an irrelevant review.

There is little previous work in the area of identifying relevance between a review and a submission. Ours is a pioneering effort in the application of relevance identification to the study of review helpfulness.

We list some related work in the area of text matching, with a focus on approaches that use graphs such as lexical chains or dependency trees to represent text. Haghighi et al. (2005) use dependency trees to determine text entailment. They use node and path substitutions to compare text graphs. A graph representation that captures ordering information would be suited for tasks involving comparison of lexical-order

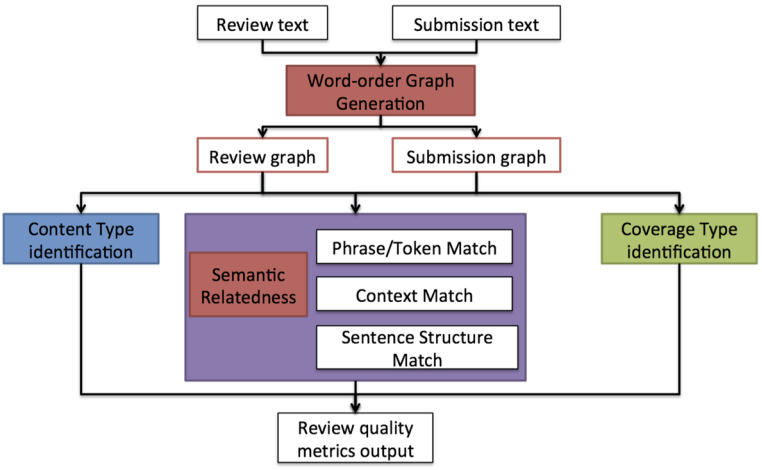


Fig. 7 Relevance identification system overview

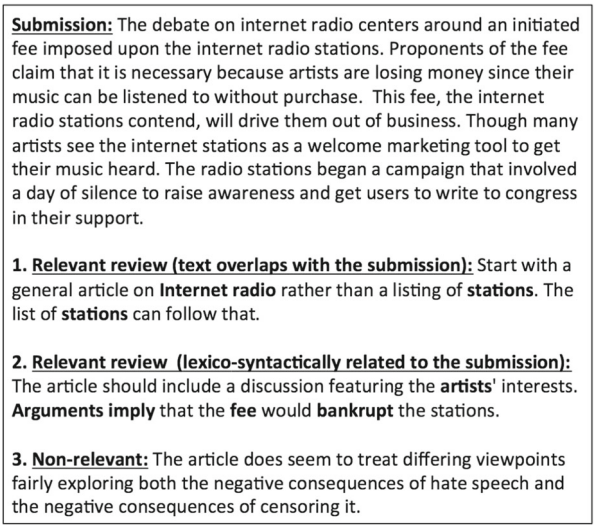


Fig. 8 The figure contains a sample submission, two relevant reviews – one with overt text matches and another that is lexico-semantically similar to the submission, and an irrelevant review

changes. As noted earlier, text matching with possible changes in word order is essential for a task like relevance identification. Existing representations and matching techniques do not capture this information.

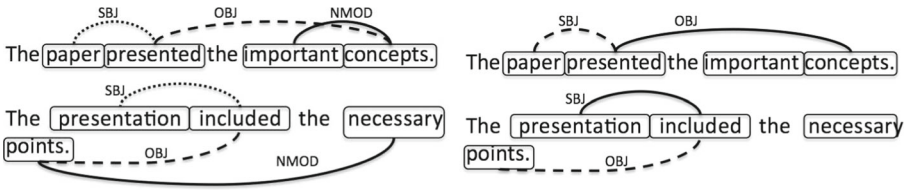
Kauchak and Barzilay (2006) suggest an automated technique to create paraphrases for human and machine-translated text pairs, by substituting words in machine translated texts with their corresponding synonyms. They define paraphrases primarily in terms of synonyms of individual tokens. Although there do exist independent research works that discuss graph-based summarization and paraphrasing techniques, they use content overlap or synonym matches to determine paraphrases. They do not consider context during text comparison. Our work is an amalgamation of existing research in the areas of text matching and paraphrase recognition.

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 Fig. 9a) are compared using WordNet, a *common parents* match is identified. This match would have been missed when using only an exact or synonym match.

$$Phrase(S, R) = \frac{1}{|V_r|} \sum_{\forall r(v) \in V_r} \underset{\forall s(v) \in V_s}{\operatorname{argmax}} \{match(s(v), r(v))\} \tag{5}$$

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. The match could be one of the WordNet relations metrics listed in the



(a) Ordered match. (b) Lexical change

Fig. 9 Context matching across two text graphs. Similar dashed lines denote the pairs of edges that are compared for each type of context match

"Semantic Relatedness" section. In (5), $r(v)$ 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.

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 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 9a 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 that contain lexical changes. Figure 9b 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.

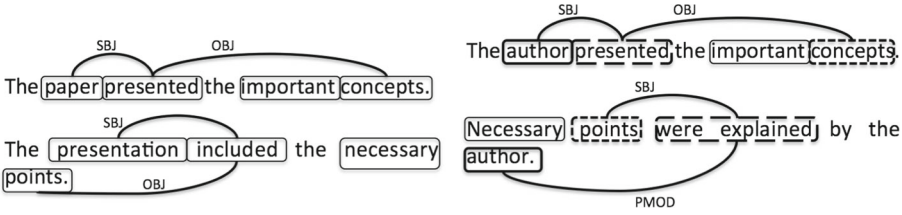
$$Context(S, R) = \frac{1}{3|E_r|}\left(\sum_{r(e)\in E_r} \underset{\forall s(e)\in E_s}{\operatorname{argmax}} \{match_{ord}(s(e), r(e))\} + \sum_{r(e)\in E_r} \underset{\forall s(e)\in E_s}{\operatorname{argmax}} \{match_{lex}(s(e), r(e))\} + \sum_{r(e)\in E_r} \underset{\forall s(e)\in E_s}{\operatorname{argmax}} \{match_{nom}(s(e), r(e))\}\right) \quad (6)$$

In (6), $r(e)$ and $s(e)$ refer to review and submission edges respectively. The formula calculates the average for each of the above three types of matches $match_{ord}$, $match_{lex}$ and $match_{nom}$. E_r and E_s represent the sets of review and submission edges. $match_{ord}$, $match_{lex}$ and $match_{nom}$ are calculated as the average of the best ordered, lexical or nominalization matches that each of the review edges have with corresponding submission edges.

Sentence Structure Matching

Sentence structure matching compares double edges (two contiguous edges or two consecutive edges sharing a common vertex), which constitute a complete segment (e.g. subject–verb–object), across graphs. In this work we consider only single and double edges 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 Fig. 10a 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.



(a) Ordered sentence structure match. (b) Voice-change match.

Fig. 10 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

- **Voice change**: Voice match captures word or phrase shuffling. Change of voice from active to passive, or vice versa is common with paraphrased text. We check for voice change because we can't be sure whether the reviewer has provided a sentence of the feedback in active of passive voice. This type of match helps capture relatedness when word order has been flipped. 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 Fig. 10b. 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. 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. Only a voice-change match succeeds in capturing such a relationship across the length of a sentence segment.

$$SentStruct(S, R) = \frac{1}{2|T_r|}\left(\sum_{r(t)\in T_r} \underset{\forall s(t)\in T_s}{\arg\max}\{match_{ord}(s(t), r(t))\} + \sum_{r(t)\in T_r} \underset{\forall s(t)\in T_s}{\arg\max}\{match_{voice}(s(t), r(t))\}\right) \quad (7)$$

The cumulative sentence structure match in (7) 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 (4) can be re-written using the lexico-semantic relatedness values calculated for phrase, context and sentence structure matches as follows:

$$relevance(S, R) = \tfrac{1}{3}(Phrase(S, R) + Context(S, R) + SentStruct(S, R)) \quad (8)$$

Relevance Identification Study

In this section we summarize the study and results of using a graph-based text matching approach to identify review relevance. A detailed description of the study can be found in Ramachandran and Gehringer (2013a).

The question we address with this study is: *Does a lexico-semantic word-order based graph matching technique help identify how relevant a review is to the content of the author's submission?*

Data We selected review-submission pairs from assignments completed using Expertiza. Each review was compared with its respective submission and other unrelated submissions, in order to include some explicit irrelevant cases.

As shown in Fig. 3, our assessment system provides students with a numeric estimate of the degree of relevance (which is the degree of graph matching between the review and submission texts). Our feedback also shows them how they perform relative to other students, which may help them gauge whether their review is "relevant enough".

However, for the purpose of evaluation, we simply classify a review as relevant or irrelevant. This allows us to explore the problem as a two-class problem. Accordingly, we asked our human annotators simply to tell us whether the review was relevant or not. The average of the match values for each match type was used as the (relevant-irrelevant) classification threshold.

986 review-submission pairs containing an equal number of relevant and irrelevant reviews were chosen for our study. Two annotators labeled 19 % of the data, and had an 80 % agreement, and a Spearman correlation of 0.44 (significance $p < .0001$). The annotators were trained with sample reviews that were relevant or not relevant to the author's submission. This served as a guide for the annotation process. Because there exists a good human agreement between annotators, labels from the first annotator were used to test our approach.

Results and Discussion A dependency tree-based matching approach is one of the baselines we compare our word-order graph based approach with. In this experiment we test how well word-order graphs perform when keeping everything else constant and swapping just the graph representation, i.e. word-order graphs for dependency trees. Since the graphs form such an integral part of the relevance identification process we tested the approach's efficacy by replacing it with a strong graph representation in the baseline.

According to the results in Table 5 there is an overall 5 % increase in accuracy with the use of the order-based matching. Our system also has better precision, recall and f-measure than a dependency-based model.

Table 5 Comparing accuracy, precision, recall and f-measure values of our word order graph with those of a dependency-tree representation

Metric	Phrase	Context	Sentence Structure	Relevance
Word order graph				
accuracy	64 %	62 %	65 %	66 %
precision	0.64	0.63	0.65	0.64
recall	0.67	0.60	0.63	0.71
f-measure	0.65	0.62	0.64	0.67
Dependency tree				
accuracy	64 %	50 %	52 %	61 %
precision	0.63	0.50	0.52	0.6
recall	0.7	0.40	0.41	0.65
f-measure	0.66	0.44	0.46	0.62

Table 5 contains results when using each graph property to compute relevance. This helps us determine how much each piece of the relevance function contributes to the overall result.

A phrase or token matching contains no context. Consider the sample review "I would retitle Internet Radio: Free Music to Merits of Internet Radio". This review gets a good phrase match value with the submission discussing Internet radio. However, this review is not fully relevant to the content of the submission, since it is only suggesting a change in title, and does not discuss the submission's content. Thus a simple non-context based phrase match tends to magnify the degree of relatedness between two texts. So, although a phrase match is important, the lack of context may inflate relevance.

Although context matching performs well, we found that not all reviews contain lexical or word order changes, or nominalizations. As a result, the average context match is likely to be low, sometimes causing a relevant review to get a poor match with a submission. This problem can be overcome by weighting the ordered match more heavily than the other types of matches.

The study also found that in most cases dependency trees take more time to perform matching than a word-order graph. Dependency trees contain more vertices and edges than our graphs do, which results in an increase in the time needed to carry out pairwise comparison between the review and submission texts.

We compare our approach with a technique based on text overlaps, normalized by review length as a baseline. To determine relevance, we use the average of 1 to 4-gram overlaps between a review and the corresponding submission. We normalize by review length since our aim is to determine the number of tokens in the review that match those in the submission text (a precision measure). The approach had a high false negative rate when compared to our graph-matching technique (Ramachandran and Gehringer 2013a). Thus, a simple text overlap measure does not succeed in capturing the relevance of a review to a submission.

Figure 11 contains two sample reviews displaying phrase and sentence structure graph matching with sentences from a sample submission (from the example in Fig. 8). The first review has some instances of exact match with the submission and its relevance may be easy to identify. However, relevance of the second review may not be determined by a text-overlap match.

Figure 12 contains screenshots of our system providing feedback on a review's relevance. Two sets of reviews written for an article on software extensibility. The sample review in Fig. 12a has a relevance of 0.13 (scale of 0–1). However, the review in Fig. 12b contains no information that is relevant to the article on software extensibility, and so has a relevance of 0.

Summary: Review Relevance

Review relevance is an important metric to determine how relevant the content of a review is to the work under review. In this section we have provided an overview of the approach to computing review relevance and have discussed results from evaluating this approach on data from Expertiza (Ramachandran and Gehringer 2013a). The

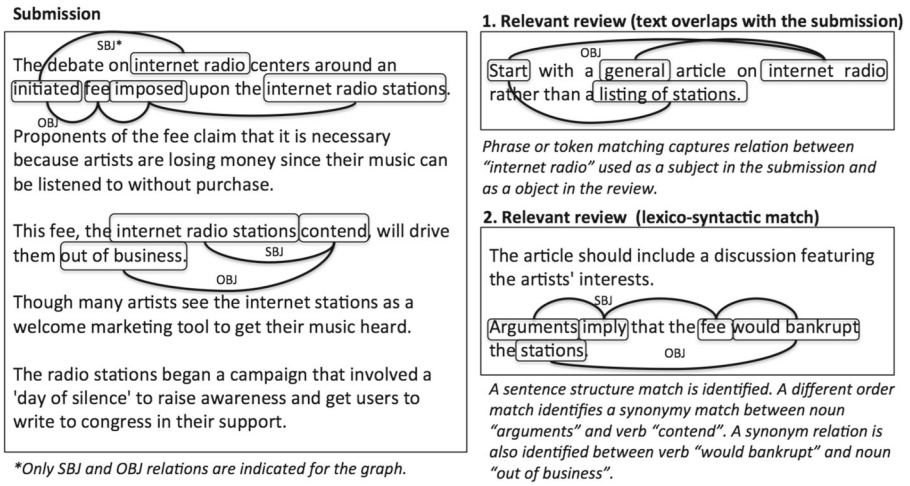
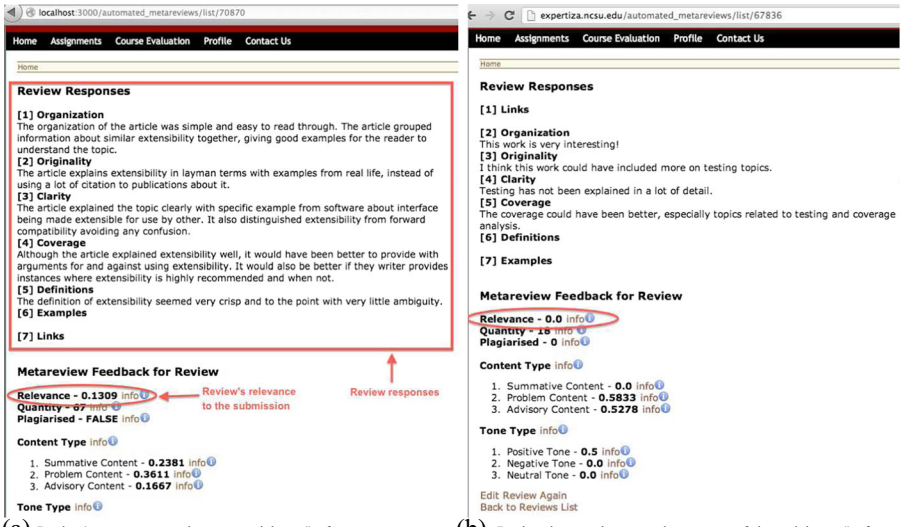


Fig. 11 Example of phrase or token matching and sentence structure match between a review and a submission

use of additional context information from edges and double edges helps us identify relevance more accurately than when using a dependency-tree representation.

Our approach is unique in that it uses sentence structure information to compute degree of relevance. This is a unique and interesting new metric to consider while identifying the quality of a review. Current approaches to graph-based paraphrasing or summarization do not include context-based text comparison.



(a) Review's contents are relevant to article on "software extensibility".

(b) Review is not relevant to the content of the article on "software extensibility".

Fig. 12 Output from our review assessment system displaying relevance of reviews. The review on the *left* discusses "software extensibility", whereas that on the *right* does not contain any content relevant to the author's submission

One of the limitations of this graph-based relevance identification approach is that the generation and comparison of graphs may be a time-consuming process. However, the graph matching can be optimized by caching similarities among vertices and edges that appear frequently in the dataset. Further if a single submission has multiple reviews, we can optimize the relevance identification process by avoiding the re-generation of the submission graph for every comparison. Although working with graphs can be expensive, there are ways in which this process can be optimized to get good results.

Review Coverage

A good review covers all parts of the reviewed document, rather than just one section, say the "Introduction". Kuhne et al. (2010) found that authors are content with reviewers who have made an effort to read and understand their work. Reviews that cover the complete work are more likely to be useful to the author.

To judge usefulness, existing approaches tend to use shallow text features such as word count. Xiong et al. (2010) use a bag-of-words, exact match approach to identify instances of problems (in the author's work) caught by peer-reviews. At present 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 identify the degree of coverage of a submission's "main points" or "topic sentences" by a reviewer's feedback. In order to calculate coverage we need to determine the topic sentences in a submission.

Coverage was introduced as a way to encourage students to read and discuss the whole submission, rather than focusing on just one part. Reviewers could discuss specific parts of a paragraph or section in the author's work, which although not the "main topic" may be semantically related to some part of the work. Our approach uses semantic relatedness to capture relations among words or phrases, which may help us identify whether the reviewer's comments are semantically related to the submission's main topics. The more topics the review covers, the higher the coverage score is likely to be. This may give the author the confidence that the reviewer has read through and paid attention to the different sections in their submission.

A topic sentence is defined as follows:

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 the set of topic sentences for a submission, if T's sentences succeed in expressing the topic or the central meaning of S.

Sentences discussing the same topic, but containing different terms may not be effectively clustered by a bag-of-words based exact match approach. Steinbach et al. (2000) found that agglomerative clustering with a word-frequency based matching made mistakes by grouping nearest documents belonging to different classes into the same cluster.

We employ an agglomerative clustering technique to group submission sentences into clusters or topics based on lexico-semantic similarity between sentences. We use word order graphs to represent text, since they capture syntax or order of tokens in a

text. We identify the most representative sentences from across the different clusters, and then calculate the coverage of the topic sentences by a review (Fig. 13). Review coverage can be defined as follows:

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-representative sentences in a submission. Coverage may then be defined as $match(T, R)$, where *match* gives the fraction of tokens in the review that overlap with the topic sentences.

We test our approach on real-world submission and review data from assignments completed using Expertiza. Figure 14 contains a sample submission with its topic-representative sentences in bold, and three sample reviews containing high, medium and no coverage of the submission's topic sentences. The first review *covers* the submission because it mentions *ethical principles* and *ethics*. The review with *medium* coverage mentions just *ethics*, and the review with *no* coverage does not contain any relevant information.

The novelty of our work lies in the utilization of topic identification techniques for the task of identifying coverage of peer-reviews. In this section we look at related work in the area of cluster-based approaches to topic identification. Cluster-based approaches have been widely applied to text and other knowledge mining applications. Clustering and topic extraction techniques have been applied to a variety of tasks such as summarization of Twitter feeds (Yang et al. 2012; Meng et al. 2012), determining the diversity of a document (Bache et al. 2013) and summarizing opinions expressed in product reviews (Zhai et al. 2011; Ganesan et al. 2010; Lappas et al. 2012).

Product reviews are different from academic reviews in that product reviews tend to focus on the sentiments pertaining to the quality of a product (positive, negative aspects etc.) (Lu et al. 2011). Peer reviews contain not just opinion information, but also contain description of the problem with the author's work and suggestions or

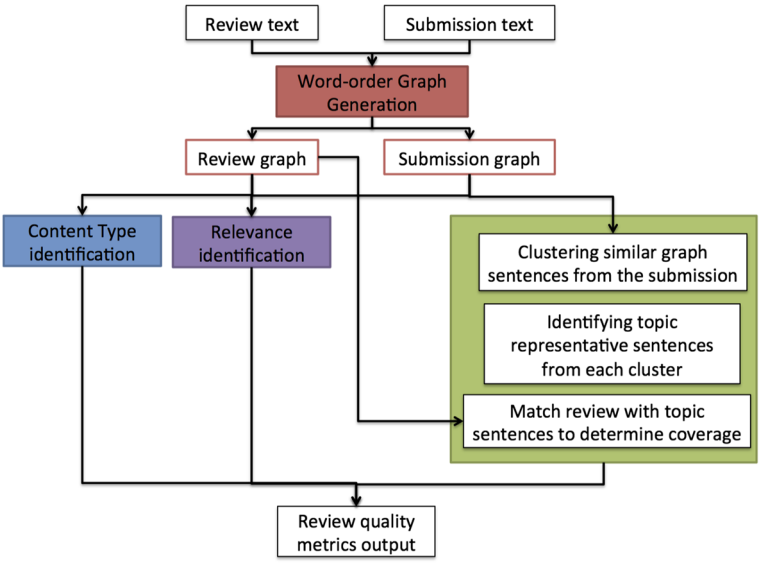


Fig. 13 Review coverage identification system overview

Submission with topic sentences:
Codes of ethics generally fall into two categories, based on their length and level of detail. However, they are also more open to personal interpretation and application which provides flexibility in applying the ethical principles in a wide variety of situations, possibly breaching the intent of the ethical principle itself. The shorter codes also generally do not provide specific examples and courses of action to take, which can make them harder to use in a potentially unethical situation, since it is up to the individual to find an appropriate course of action. Some of the codes, such as the ACM's code of ethics, contain both a short version and a long version, which provides the best of both worlds.

High Coverage: I would consider the *ethical* hacking article to be more appropriate for the hacking topic than for *ethical principles* (which is related to *principles* theoretical *ethics*).

Medium Coverage: The two pages sufficiently discuss vaporware but *ethical* points are indirectly covered.

No Coverage: I don't see a link to the old version of the pages, but I might have missed it. It does have a link to the topic description.

Fig. 14 Submission with topic sentences (in bold), and three reviews with high, medium and no coverage of the topic sentences

advice on how to improve the work. Peer reviews focus on the content of the author's submission.

Several topic identification and sentence ranking approaches utilize graph-ranking algorithms such as PageRank and HITS (Mihalcea 2004) to identify topic-representative sentences. Coursey and Mihalcea (2009) determine the topic of an input document by identifying the central vertex using the Google PageRank formula. In their SemanticRank paper, Tsatsaronis et al. (2010) use semantic graphs for keyword extraction. They use a combination of statistical (term frequency inverse document frequency) and semantic metrics to calculate relatedness between tokens in a document. Tsatsaronis et al. use weighted PageRank and HITS algorithms to rank the sentences or keywords in a graph. Wang et al. (2013) use influence analysis to identify topic hierarchies to help summarize the content of texts.

Graph-based approaches have been used for a wide variety of tasks such as text summarization and topic identification. Mihalcea (2004) uses a graph representation to perform sentence extraction. A vertex in the graph represents a sentence in a document, and the weighted edges represent the degree of overlap across content of the sentences. Erkan and Radev (2004) use graph-based centrality algorithms to rank sentences in a document. They determine similarity by taking the cosine of sentences represented as term vectors. An important distinction between their problem and ours is that they are not trying to determine the degree of coverage of the important topics in a document.

Unlike the graph representations listed here our graph contains word-ordering information in which vertices represent words or phrases, and edges represent the syntactic relations between vertices. We also notice that most of the existing

approaches that identify topics or rank sentences in a document use statistical techniques, which involve identifying only exact matches across the frequent terms in texts. Instead, we use a semantics-based similarity identification technique to group sentences into clusters.

We are not suggesting that reviews always contain summaries of a submission. A review may provide an assessment of the kind of work that was done—praising the submission's positive points, identifying problems, if any, and offering suggestions to help improve the submission. But reviews do so while discussing the main points of the submission. Our goal is to identify sentences central to the submission and to determine a review's coverage of those sentences. For this we take inspiration from research in the area of topic identification and summarization, and even compare our approach to state-of-the-art summarization techniques.

Grouping Sentences into Topic Clusters

We define the problem of clustering as follows—Given a set of sentences $S_1, S_2, \cdots, S_n$, we would like to 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. Sentences belonging to the same cluster are considered to discuss the same topic.

We use an agglomerative clustering technique to group sentences into clusters. The clustering algorithm starts by assigning 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 (Charikar and Panigrahy 2001)). 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. Every step of the clustering process is listed in Algorithm 1.

Algorithm 1: Clustering Algorithm

Input: Sentences $S_1, S_2, \cdots, S_n$ that are to be clustered.
Output: Clusters $C_1, C_2, \cdots, C_k$.

Initially each sentence belongs to its own cluster.
Initialize every cluster's similarity i.e., C_i's similarity = 0.
Let R be the ranked list of sentence pairs, and $R = \{(S_i, S_1), (S_i, S_2), \cdots (S_{n-1}, S_n)\}$ such that *Similarity*$(S_i, S_1) \geq$ *Similarity*(S_i, S_2) and so on.
for *each sentence pair* (S_i, S_j) *in* R **do**
    `/* Identifying destination cluster. */`
    if S_i*'s cluster's similarity* $>$ S_j*'s cluster's similarity* **then** Set $S = S_j$ and destination cluster $C = S_i$'s cluster ;
    else if S_i*'s cluster's similarity* $<$ S_j*'s cluster's similarity* **then**
        Set $S = S_i$ and destination cluster $C = S_j$'s cluster
    else if S_i*'s cluster's similarity* $==$ S_j*'s cluster's similarity* **then**
        if S_i *cluster's sentence count* $>$ S_j *cluster's sentence count* **then**
            Set $S = S_j$ and destination cluster $C = S_i$'s cluster
        else if S_i*'s cluster's sentence count* $<$ S_j*'s cluster's sentence count* **then**
            Set $S = S_i$ and destination cluster $C = S_j$'s cluster
        else
            Randomly select destination cluster C
        end
    end
    if S *satisfies Equation 8* **then**
        $C = \{C \cup S\}$
        RecalculateClusterSimilarity(C) `/* Average of similarities of the cluster's sentences (with the newly added sentence) is calculated. */`
    end
end

We rank sentence pairs based on their similarity (highest to lowest) using the mergesort algorithm. We select sentence pairs for clustering in the order of their ranks. For a sentence pair, we choose the target cluster depending on the cluster's similarity values, i.e., the cluster with a higher average similarity is chosen as the target. If both sentences' clusters have the same similarity, then we select the target based on the number of sentences in each cluster. 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. 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 (9). We use the UPGMA (Unweighted Pair Group Method with Arithmetic Mean) scheme for agglomerative clustering (Steinbach et al. 2000).

$$\left(C.clusterSimilarity - \sum_{\forall S_C \in C} \frac{Similarity(S, S_C)}{|C|} \right) \leq \alpha \tag{9}$$

We calculate the average of the pairwise similarities between a new sentence S and every sentence S_C in a cluster C. According to constraint 8, the difference between the cluster's similarity and S's average semantic similarity with C's sentences must not be more than α. The condition ensures that sentences that are added to the cluster have high similarity with a cluster's sentences. Thus, the process ensures that the clusters that are created contain semantically related sentences, i.e., they contain sentences that are similar in meaning and context.

Since for different texts the similarities between sentences and clusters vary, setting a constant threshold may not be suitable. We choose α as the average of the difference between sentence similarities because (a) it gives us the degree of variance of the similarities, and (b) it prevents sentences that are much too dissimilar from being grouped into the same cluster.

Selecting sentence pairs in the order of their similarities 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. A cluster's similarity is re-calculated each time a new sentence is added to it.

Each cluster represents a separate topic or concept discussed by the author. However not all topics in a submission need to be discussed in the reviews. 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 to select the important clusters. Figure 15 contains the clusters generated for the sample submission.

Identifying Salient Sentences in a Submission's Text

In this step the most representative sentences are identified from each of the selected clusters. We use cohesion-based methods to identify the most important concepts in a submission. In a cohesion-based method only the most well-connected vertices are

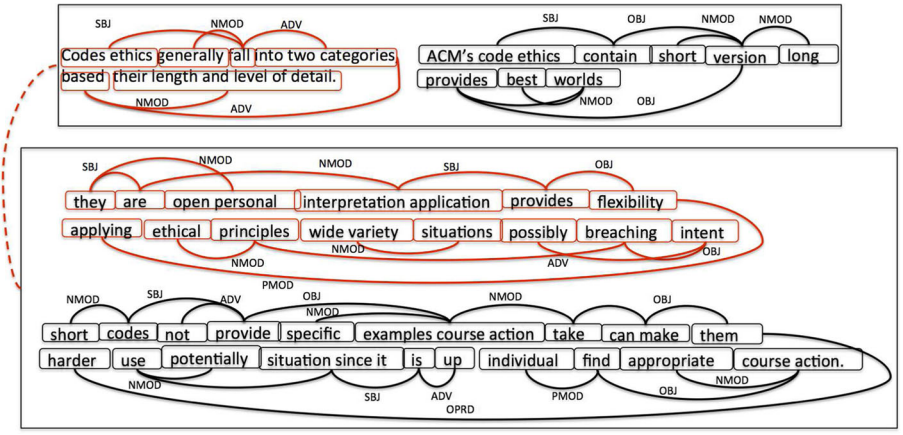


Fig. 15 In both clusters the topic sentence is highlighted in *red*

taken to form the summary. 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 identification of the smallest set of topic-representative sentences T that *cover* (are most similar to) every other sentence in the cluster C. For each cluster, the set T that satisfies the condition, i.e., maximizes pairwise similarities between the topic and cluster's sentences and minimizes size of T (or maximize $\frac{1}{|T|}$) is selected as the set of topic sentences.

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

The topic sentence identification problem can be thought of as similar to that of 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 and Imamura (2007) 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. We also use a list heuristic to handle the vertex cover problem. Our heuristic approach to identifying the smallest number of sentences that successfully capture the meaning of a cluster is described in Algorithm 2. 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). Sentences in a cluster are only connected to other sentences whose similarity to them is greater than the cluster's (average) similarity.

Algorithm 2: Topic Sentence Identification

```
Let S be a sentence in cluster C.
Let U be the set of uncovered sentences.
Initialize U = C
Let R be the ranked list of sentences i.e., R = {S_1, S_2, ···, S_n} such that S_i's average similarity with other sentences in
the cluster ≥ S_j's average similarity.
Sentences in R are ordered by degree (number of sentences adjacent to a sentence) - largest to smallest.
for each sentence S ∈ R do
    if S is adjacent to one or more uncovered sentences S_u then
        T = {T ∪ S}
        U = {U − S_u}
    end
    if U is empty then
        Break /* No sentence remains uncovered.  */
    end
end
```

Our approach ensures that topic sentences with the highest semantic similarity, that cover previously uncovered sentences are added to the cover set. The cover set suitably represents all sentences in the topic cluster. Cover sets from across all clusters together form the set of topic-representative sentences for a submission. The topic-representative sentences for the sample submission are highlighted in red in Fig. 15.

Matching Review and Topic Sentences to Determine Coverage

The coverage of a review is calculated in terms of the number of overlapping matches (excluding stopwords and frequent words) it has with the submission's topic sentences.

$$cov(S, R) = \frac{1}{|T_S|+|R|} \sum_{\forall r \in R, \forall t \in T_S} 2 \times match(t, r) \tag{10}$$

Equation (10) calculates the coverage as the overlaps between topic sentences and the reviews. The measure evaluates coverage in terms of both review ($|R|$) and topic sentences' lengths ($|T_S|$). S, R and T_S refer to the set of submission, review and the submission's topic sentences respectively. The cumulative measure penalizes reviews that are long and may potentially contain less relevant content.

In Fig. 16 the submission's topic sentences are compared with three sample reviews. We see that review A has more in common with the submission's topic sentences (colored text), and therefore has a higher coverage value than B, which has a higher coverage than C.

Review Coverage Study

In this section we describe our study to evaluate review coverage.

The question we address with this study is: *Does a graph-based clustering technique help capture representative sentences using which coverage of reviews may be evaluated?*

We compare our approach with two state-of-the-art text summarization systems–MEAD and Opinosis. We also use a simple baseline that contains the first ≈ 2

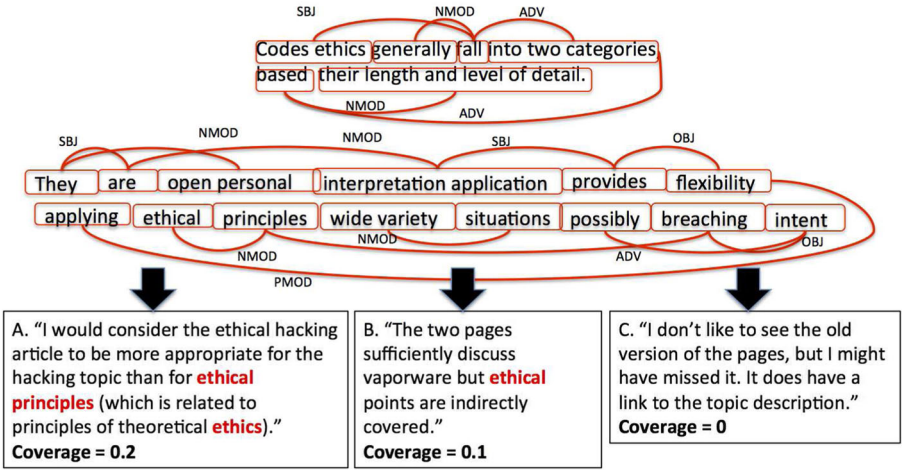


Fig. 16 Comparing sample reviews with the selected topic sentences. Review **A** has more in common with the topic sentences, and so it has a higher coverage than reviews **B** and **C**. Some overt similarities have been highlighted

sentences of the submission text. The use of this baseline is inspired by NIST's baseline summary creation technique, which involves extraction of the first n words to generate a document's summary.

We compare our approach with MEAD, a centroid-based summarization approach. Radev et al. (2004)'s approach uses the most common words in a document to identify the best sentences to be included in a summary. They use the Cosine metric (exact match between tokens) to determine similarity. MEAD is an extractive summarization approach, and since in our approach too we extract the most representative sentences from a submission, we find MEAD to be an ideal system to compare our approach with. We also compare our approach with (Ganesan et al. 2010)'s Opinosis summarizer. They use a graph-based summarization technique to create abstractive summaries. Our approach also uses a graph-based representation and matching technique to identify the topic representative sentences.

We evaluate our approach by calculating the degree of coverage of student-written reviews and comparing it with human-provided coverage values. Reviews' coverage of MEAD and Opinosis' summaries are also computed using the coverage metric. We determine the correlation between the coverage values generated by each of the approaches and the human-provided coverage values.

Data

We evaluate our approach on peer-review data from Expertiza.

We use peer-review data from computer science classes over a couple of semesters to evaluate our approach. A dataset containing 979 reviews written for 132 submissions was collected from Expertiza. These are reviews provided by students and

hence not all of the reviews 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" and "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.

Thirty-nine data points ($\simeq$ 4 % of the data) were randomly selected from the dataset and labeled for the degree of coverage by six annotators. Annotators were trained for the process with examples of reviews-submission pairs with their corresponding coverage values as a guide. We computed the correlations between the six annotators using Spearman correlation. We use Spearman correlation since it is suited for ordinal data. We found a high average correlation of 0.63 among the six annotators. This indicates that human annotators agree on the relative ordering of coverage values. The average correlation between five annotators and a sixth annotator A was 0.6. Since the annotators have a high agreement with A, A labeled all the 979 data points, and these labels are used to evaluate system-generated coverage values.

Results

Correlations between human-provided and system-generated coverage values are listed in Table 6. Our approach produced topic sentences containing on average 109 words per submission.

System-generated values just like the human annotations are ordinal, i.e., they exhibit a relative ranking. Therefore Spearman correlation may be suited to determine the relationship between system-generated and human-provided coverage values.

Correlation is indicative of the strength of the relationship that exists between two variables. System coverage has a correlation of 0.49 with human coverage. A positive

Table 6 Identifying the correlation between system-generated and human-provided coverage values on review data from Expertiza

Approach	Correlation	Avg. # words
Our system	0.49*	109
Our system (truncated)	0.43*	33
MEAD summarizer	0.32	34
Opinosis	0.34	35
Baseline top 2 sentences	0.36	38

*The differences between the correlations of our system's coverage with human coverage values and MEAD and Opinosis' correlations are significant. Using the two-tailed test the p-values < 0.05, thus the null hypothesis that this difference is a chance occurrence may be rejected

correlation of 0.49 means that the system and human agree on the directionality of the coverage (high or low).

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 object-oriented 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 object-oriented languages, this review receives a low coverage.

Comparison with MEAD and Opinosis We select the first two sentences from summaries generated by MEAD for our evaluation. MEAD's summaries produce lower correlations with human coverage values compared to the topic sentences generated by our approach (Table 6). Summaries generated by MEAD contain just the top k words from the submission text, which may not be fully representative of the most "important" sentences.

Opinosis produces abstractive summaries with an average word count of 35. With shorter topic sentences, Opinosis' coverage correlations are lower compared to our approach. We truncate the topic sentences generated by our system in order to have summaries of nearly the same size as those generated by Opinosis. Truncated version of our summaries have higher correlations with human coverage values than Opinosis' correlations.

For $N = 979$, correlations of 0.49 and 0.43 are significant. Since $p < .0001$ for a t-test the null-hypothesis that these correlations are a chance occurrence may be rejected. With the help of these experiments we have demonstrated the ability of our approach to effectively identify the topic-representative sentences from a document, and estimate a review's coverage of these topic sentences.

Summary: Review Coverage

We have described the use of graph-based clustering techniques to determine a review's coverage of the author's submission. 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. We have evaluated the approach on peer review data from Expertiza and have compared it with two state-of-the-art summarization tools MEAD and Opinosis. We use:

1. a word-order graph representation with an agglomerative clustering approach to identify topic clusters in a submission, and
2. a heuristic to identify topic-representative sentences from each cluster.

We have demonstrated that our approach is better at capturing topic sentences for coverage identification than summarizers such as MEAD and Opinosis.

One of the novel contributions of this work is the use of a review coverage metric. The use of clustering techniques to determine topic sentences whose coverage

represents one aspect of a review's quality is a new addition to the field. As in the case of review relevance, generating graphs to identify topic sentences may be an expensive step. However, this step can be optimized by avoiding re-generating topic sentences for a submission, whose coverage is to be evaluated for multiple reviews.

Other Quality Metrics: Tone, Volume and Plagiarism

Tone

Tone refers to the semantic orientation of a text. 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. We look for positively or negatively oriented words to identify the tone of a review (Turney 2002). We use positive and negative indicators from an opinion lexicon provided by Liu et al. (2005). 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 good 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 examples are not easy to understand and have been copied from other sources. Although the topic is Design Patterns in Ruby, no examples in Ruby have been provided for Singleton and Adapter Pattern".
  The given example contains negatively oriented words or phrases such as "not easy to understand", "copied", "no examples". Review segment"...have been copied from other sources ..." implies that the author has plagiarized, and could be construed as a rude accusation by the author. One of the ways in which this review could be re-phrased to convey the message, so as to get the author to acknowledge the mistake and make amends, is as follows. "The topic on Design Patterns in Ruby could be better understood with more examples, especially for the Singleton and Adapter patterns. Please try to provide original examples from your experience or from what was discussed in class".
- **Neutral:** Reviews that do not contain either positively or negatively oriented words or phrases, or contain a mixture of both are classified into this category.
  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.

Quantity or Volume of Feedback

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 the reviewer the amount of feedback they have provided in comparison to the average review quantity (from other reviewers of the system), thus motivating them to provide more feedback to the authors. We identify quantity by taking a count of all the unique tokens in a review. For instance, consider the following review, "The article clearly describes its intentions. I felt that the section could have been elaborated a little more". The number of unique tokens in this review is 15 (excluding articles and pronouns).

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 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 as plagiarism.

Each of the review quality metrics listed is determined independently, and then integrated into a complete review quality assessment system. Reviewers are given feedback on each of these metrics, so that they get a picture of the completeness and quality of their review.

Study

In this section we discuss (1) a study to evaluate whether the discussed metrics help in assessing the overall quality of a review, and (2) a user study conducted to identify the importance reviewers attach to each of the assessment metrics.

Study on Metrics and Review Quality

In a study conducted by Yadav (2016), 119 students provided metareview feedback for each of the quality metrics described in this paper. Feedback was on a Likert scale

of 1-5 where 1 indicates that the review did not meet the quality metric, to 5 where the review met the quality metric. For example in the case of quantity, a 1 indicates that the review length was too small, while 5 indicates that the review was of a good length.

The metareviewers also assigned an overall review quality to each of the reviews. The authors found that review quality metrics such as coverage, relevance, advisory content and volume or quantity of feedback have a high correlation with the quality of the reviews. This indicates that these metrics are useful in identifying the overall quality of reviews. They also found that tone, though positively correlated with quality, was not highly correlated. This seems to imply that a positive-tone review may not necessarily be a high-quality review, or a negative-tone review a poor quality one.

User Study on the Automated Assessment of Reviews

In this section we discuss the user study conducted to study the usefulness of the review quality metrics. A detailed description of the work can be found in Ramachandran and Gehringer (2013b). The importance of our work and the automated assessment system can only be gauged by its users—students, teaching assistants, and faculty. The aim of our user study was to get feedback from prospective users of the system, which would help us identify what the surveyed reviewers liked or disliked in the system.

The purpose of this study is to determine whether participants found the feedback on the review quality metrics–review relevance, content, tone, quantity and plagiarism, to be useful. We leave for future work the question of whether review quality feedback helps reviewers produce better reviews, and in turn, leads students to improve their submissions, thus enhancing learning outcomes.

UXMatters states that "Learnability, usability, usefulness, and aesthetic appeal are key factors in users' experience of a product". Therefore, a user-experience survey should include study of the learning gained from a system, i.e., its usefulness (UXMatters 2005).

Data

We explored the usefulness and presentation of automated feedback with seven users, who were students in a recent masters-level course at North Carolina State University. In that course, each student had submitted about fifteen reviews, over a total of four assignments. A total of 107 user-experience responses was collected.

Steps to complete the data collection process are listed in Table 7. Each of the 7 reviewers were asked to use the automated metareview feature on Expertiza. They used the system to write reviews for submissions they had been assigned to review (peer review in Expertiza). A review rubric is provided to guide participants while writing the review. The rubric contained 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.

Table 7 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 submission under review.
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**.

When participants submitted their reviews, they were presented with automated feedback, which gave them information on (1) content type, (2) relevance of the review to the article, (3) tone, (4) volume of text and (5) presence of plagiarism. Reviewers were then asked to fill out a user experience questionnaire (Step 8 in Table 7).

Questionnaire The user-experience questionnaire consisted of the following sections:

- 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.
- In the *helpfulness* section, we questioned participants on whether the automated metareview feedback was helping them learn what was lacking in their reviews. Reviewers rate each metareview metric as —*extremely helpful, helpful, somewhat helpful* or *not helpful*.
- In the *interpretability* section we identified whether participants found the output from the metareview metrics to be interpretable. They rated each metric as *extremely easy to interpret, easy to interpret, somewhat easy* or *not easy*.
- In the next section we gauged whether reviewers were willing to update their reviews based on the feedback they received. Reviewers rated each metric as *highly likely, likely, somewhat likely* or *unlikely*.
- Reviewers also rated usefulness of each metric on a scale of 1-5 where 1 is least useful and 5 is most useful. Reviewers were also asked to specify the overall helpfulness of a metareview feedback and how likely they were to fix their review based on the metareview feedback provided to them.

Discussion of Results from the User Experience Analysis

This section summarizes the study's findings. All the reviewers in the study have had prior reviewing experience and have had experience using Expertiza.

The current implementation of the relevance metric was too slow to be used in this analysis, since it must perform WordNet lookups for every token or phrase. The implementation could be optimized by caching lookups to WordNet. Although we don't have an analysis of the relevance metric in this study, our previous study (Ramachandran and Gehringer 2013b) found that participants rated relevance as the most important metric.

In terms of helpfulness of the metrics, for 83.1 % of responses participants found plagiarism to be very helpful. Helpfulness percentages are computed using the number of reviews classified as 'helpful' or 'extremely helpful'. Most reviewers also found metrics such as tone, volume and review content to be helpful. See Table 8 for details of the numbers. For over 90 % of the responses, they found plagiarism, volume and tone to be easily interpretable.

Participants found tone and plagiarism metrics to be useful for over 70 % of the responses. They rated 55.1 % of the responses as helpful overall (see Table 8).

For over 50 % of the responses, participants said they would be willing (i.e., highly likely or likely) to update their reviews for tone, volume and plagiarism. In addition, they were willing to fix the entire review for 44.8 % of responses. At the other end of the scale, for 25.2 % (volume) to 28.9 % (review content) of the responses the reviewers said that it was unlikely that they would make changes. We suspect that more reviewers would be induced to make changes if provided with online instructions and tips on how to improve their reviews' score on each of these metrics.

In an earlier study we surveyed the participants on some of the other metrics that may be of interest to them (Ramachandran and Gehringer 2013b). Participants were interested in getting feedback on 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 complete thought. 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. 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 makes reading and understanding reviews easier.

Summary and Future Work

Reviews are central to the process of assessment—whether it is 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.

Table 8 Response of Expertiza reviewers to the user experience questionnaire. A large percentage of reviewers rated features content, tone, volume and plagiarism to be helpful and interpretable. 55.1 % of the responses were rated to be helpful overall

Classification	Review content	tone	volume	plagiarism
Helpfulness of metrics in understanding where reviews were lacking.				
Extremely helpful	23	29	30	37
Helpful	50	54	51	52
Somewhat helpful	29	22	24	17
Not helpful	5	2	2	1
% helpful	68.2 %	77.5 %	75.7 %	83.1 %
Interpretability of metrics				
Extremely easy to interpret	31	36	36	49
Easy to interpret	54	61	62	51
Somewhat easy	20	10	9	7
Not easy	2	0	0	0
% interpretable	79.4 %	90.6 %	91.5 %	93.4 %
Usefulness of metrics				
5-extremely useful	20	38	28	42
4-useful	40	39	32	35
3-neither useful nor useless	24	12	16	11
2-useless	11	12	19	10
1-extremely useless	12	6	12	9
% useful	56 %	71.9 %	56 %	71.9 %
Overall helpfulness*				
Extremely helpful	3			
Helpful	56			
Somewhat helpful	41			
Not helpful	7			
% helpful	55.1 %			
Willingness to update				
Highly likely	18	20	19	15
Likely	31	47	36	53
Somewhat likely	27	12	25	10
Unlikely	31	28	27	29
% willing to update	45.7 %	62.6 %	51.4 %	63.5 %
Willingness to fix entire review*				
Highly likely	14			
Likely	34			
Somewhat likely	40			
Unlikely	19			
% willing	44.8 %			

*Numbers apply to all metrics

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.

In this work we use (1) a cohesion-based technique that uses semantically important graph edges to form patterns, (2) a graph-based text-matching approach to compare same and different types of edges to identify relevance of reviews, and (3) a graph-clustering technique to identify topic-sentences in author submissions that need to be covered by a good review.

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 places (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. Also, showing reviewers how their work is rated relative to other reviewers, for instance, are they above or below other reviewers in relevance or coverage, might constitute useful feedback.
- **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.

- **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 (Tognini-Bonelli 2002), which is beyond the scope of this paper.

In this paper 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 a suite of metrics that are important in assessing reviews. We have provided a description of the problem of automated review quality assessment and each of its sub-problems. We have explained our approach to solving them, and analyzed the results from our different experiments. We have shown that our approach performs better than state-of-the-art techniques. 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.

Acknowledgments We would like to thank Da Young Lee for helping us review an early draft of the paper.

References

Agirre, E., Alfonseca, E., Hall, K., Kravalova, J., Paşca, M., et al. (2009). A study on similarity and relatedness using distributional and wordnet-based approaches. In *The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics* (pp. 19–27). NAACL.

Avis, D., & Imamura, T. (2007). A list heuristic for vertex cover, (Vol. 35, Elsevier.

Bache, K., Newman, D., & Smyth, P. (2013). Text-based measures of document diversity. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. KDD* (pp. 23–31). doi:10.1145/2487575.2487672.

Barzilay, R., & Elhadad, M. (1997). Using lexical chains for text summarization. In *Proceedings of the ACL Workshop on Intelligent Scalable Text Summarization.* (pp. 10–17).

Blei, D.M., Ng, A.Y., & Jordan, M.I. (2003). Latent dirichlet allocation. *The Journal of machine Learning research*, *3*, 993–1022.

Bohnet, B. (2010). Very high accuracy and fast dependency parsing is not a contradiction. In *Proceedings of the 23rd International Conference on Computational Linguistics* (pp. 89–97). COLING.

Boonthum, C. (2004). istart: paraphrase recognition. In *Proceedings of the ACL 2004 workshop on Student research. ACLstudent*.

Burstein, J., Marcu, D., & Knight, K. (2003). Finding the write stuff: Automatic identification of discourse structure in student essays. *IEEE Intelligent Systems*, *18*, 32–39.

Chapman, W.W., Bridewell, W., Hanbury, P., Cooper, G.F., & Buchanan, B.G. (2001). A simple algorithm for identifying negated findings and diseases in discharge summaries. *Journal of biomedical informatics*, *34*, 301–310.

Charikar, M., & Panigrahy, R. (2001). Clustering to minimize the sum of cluster diameters. In *Proceedings of the thirty-third annual ACM symposium on Theory of computing* (pp. 1–10).

Cho, K., & Schunn, C.D. (2007). Scaffolded writing and rewriting in the discipline: A web-based reciprocal peer review system. *Computers and Education*, *48*, 409–426. doi:10.1016/j.compedu.2005.02.004.

Cho, K. (2008). Machine classification of peer comments in physics. In *Educational Data Mining* (pp. 192–196).

Coursey, K., & Mihalcea, R. (2009). Topic identification using wikipedia graph centrality. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers, NAACL-Short '09* (pp. 117–120).

Dalvi, N., Kumar, R., Pang, B., & Tomkins, A. (2009). *Matching reviews to objects using a language model*. EMNLP '09.

Echeverría, V., Gomez, J.C., & Moens, M.F. (2013). Automatic labeling of forums using bloom's taxonomy. In *Advanced Data Mining and Applications, Springer* (pp. 517–528).

Erkan, G., & Radev, D.R. (2004). Lexrank: graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research*, *22*, 457–479.

Fan, R.E., Chang, K.W., Hsieh, C.J., Wang, X.R., & Lin, C.J. (2008). Liblinear: a library for large linear classification. *The Journal of Machine Learning Research*, *9*, 1871–1874.

Fellbaum, C. (1998). Wordnet: an electronic lexical database, MIT Press.

Fleiss, J.L., Cohen, J., & Everitt, B. (1969). Large sample standard errors of kappa and weighted kappa. *Psychological Bulletin*, *72*, 323.

Foltz, P.W., Gilliam, S., & Kendall, S.A. (2000). Supporting content-based feedback in online writing evaluation with LSA. *Interactive Learning Environments*, *8*, 111–129.

Ganesan, K., Zhai, C., & Han, J. (2010). Opinosis: a graph-based approach to abstractive summarization of highly redundant opinions. In *Proceedings of the 23rd International Conference on Computational Linguistics. COLING '10* (pp. 340–348).

Gehringer, E.F. (2010). Expertiza: Managing feedback in collaborative learning. In *Monitoring and Assessment in Online Collaborative Environments: Emergent Computational Technologies for E-Learning Support* (pp. 75–96).

Haghighi, A.D., Ng, A.Y., & Manning, C.D. (2005). Robust textual inference via graph matching. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing. HLT* (pp. 387–394).

Joachims, T. (1998). Text categorization with support vector machines: Learning with many relevant features, Springer.

Kauchak, D., & Barzilay, R. (2006). Paraphrasing for automatic evaluation. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics. HLT-NAACL '06* (pp. 455–462).

Kuhne, C., Bohm, K., & Yue, J.Z. (2010). Reviewing the reviewers: a study of author perception on peer reviews in computer science. In *CollaborateCom* (pp. 1–8).

Lappas, T., Crovella, M., & Terzi, E. (2012). Selecting a characteristic set of reviews. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. KDD* (pp. 832–840). doi:10.1145/2339530.2339663.

Lesk, M. (1986). Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone. In *Proceedings of the 5th annual international conference on Systems documentation* (pp. 24–26). SIGDOC.

Lim, E.P., Nguyen, V.A., Jindal, N., Liu, B., & Lauw, H.W. (2010). Detecting product review spammers using rating behaviors. In *Proceedings of the 19th ACM international conference on Information and knowledge management* (pp. 939–948). CIKM '10.

Liu, B., Hu, M., & Cheng, J. (2005). Opinion observer: analyzing and comparing opinions on the web. In *Proceedings of the 14th International Conference on World Wide Web* (pp. 342–351).

Liu, B.Q., Xu, S., & Wang, B.X. (2009). A combination of rule and supervised learning approach to recognize paraphrases. In *Proceedings of the International Conference on Machine Learning and Cybernetics*, (Vol. 1 pp. 110–115).

Lu, B., Ott, M., Cardie, C., & Tsou, B.K. (2011). Multi-aspect sentiment analysis with topic models. In *IEEE 11th International Conference on Data Mining Workshops (ICDMW), 2011. IEEE* (pp. 81–88).

Manshadi, M., Gildea, D., & Allen, J. (2013). Plurality, negation, and quantification: Towards comprehensive quantifier scope disambiguation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL-13)*.

Meng, X., Wei, F., Liu, X., Zhou, M., Li, S., et al. (2012). Entity-centric topic-oriented opinion summarization in twitter. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. KDD* (pp. 379–387).

Mihalcea, R. (2004). Graph-based ranking algorithms for sentence extraction, applied to text summarization. In *Proceedings of the ACL 2004 on Interactive poster and demonstration sessions. ACLdemo*.

Moghaddam, S., Jamali, M., & Ester, M. (2011). Review recommendation: personalized prediction of the quality of online reviews. In *Proceedings of the 20th ACM international conference on Information and knowledge management* (pp. 2249–2252). CIKM '11.

Nelson, M.M., & Schunn, C.D. (2009). The nature of feedback: How different types of peer feedback affect writing performance. In *Instructional Science*, (Vol. 27 pp. 375–401).

Nguyen, H.V., & Litman, D.J. (2013). *Identifying Localization in Peer Reviews of Argument Diagrams, Berlin, Heidelberg*, (pp. 91–100). Berlin Heidelberg: Springer.

Nguyen, H.V., & Litman, D.J. (2014). Improving peer feedback prediction: The sentence level is right, 99. ACL 2014.

Patchan, M., Charney, D., & Schunn, C. (2009). A validation study of students' end comments: Comparing comments by students, a writing instructor, and a content instructor. *Journal of Writing Research*, *1*, 124–152.

R. (2008). *R: A language and environment for statistical computing*. Vienna, Austria: R Foundation for Statistical Computing. ISBN 3-900051-07-0.

Rada, R., Michailidis, A., & Wang, W. (1994). Collaborative hypermedia in a classroom setting. *Journal Education Multimedia Hypermedia*, *3*, 21–36.

Radev, D.R., Jing, H., Styś, M., & Tam, D. (2004). Centroid-based summarization of multiple documents. *Information Processing & Management*, *40*, 919–938.

Ramachandran, L., & Gehringer, E.F. (2010). Automating metareviewing. In *poster presentation at Workshop on Computer-Supported Peer Review in Education, associated with Intelligent Tutoring Systems*.

Ramachandran, L., & Gehringer, E.F. (2012). A word-order based graph representation for relevance identification [poster]. In *CIKM 2012, 21st ACM Conference on Information and Knowledge Management*.

Ramachandran, L., & Gehringer, E.F. (2013a). Graph-structures matching for review relevance identification. In *Proceedings of TextGraphs-8 Graph-based Methods for Natural Language Processing* (pp. 53–60).

Ramachandran, L., & Gehringer, E.F. (2013b). A user study on the automated assessment of reviews. In *Proceedings of the Workshops at the 16th International Conference on Artificial Intelligence in Education AIED 2013, Memphis, USA, July 9–13, 2013*.

Ramachandran, L., & Gehringer, E.F. (2013c). An ordered relatedness metric for relevance identification. In *In Semantic Computing (ICSC), 2013 IEEE Seventh International Conference on* (pp. 86–89). doi:10.1109/ICSC.2013.23.

Ramachandran, L., & Gehringer, E.F. (2011). Automated assessment of review quality using latent semantic analysis. In *ICALT 2011, 11th IEEE International Conference on Advanced Learning Technologies* (pp. 136–138).

Ramachandran, L., & Gehringer, E.F. (2015). Identifying content patterns in peer reviews using graph-based cohesion. In *Proceedings of Florida Artificial Intelligence Research Society Conference*.

Rooyen, S.V., Black, N., & Godlee, F. (1999). Development of the review quality instrument (rqi) for assessing peer reviews of manuscripts. *Journal of Clinical Epidemiology*, *52*, 625–629. doi:10.1016/S0895-4356(99)00047-5.

Steinbach, M., Karypis, G., Kumar, V., et al. (2000). A comparison of document clustering techniques. In *KDD workshop on text mining.*, (Vol. 400 pp. 525–526).

Titov, I., & McDonald, R. (2008). Modeling online reviews with multi-grain topic models. In *Proceedings of the 17th international conference on World Wide Web* (pp. 111–120). WWW '08.

Tognini-Bonelli, E. (2002). Corpus linguistics at work. *Computational Linguistics*, *28*, 583–583.

Toutanova, K., Klein, D., Manning, C.D., & Singer, Y. (2003). Feature-rich part-of-speech tagging with a cyclic dependency network. In *Inproceedings of HLT-NAACL 2003* (pp. 252–259).

Tsatsaronis, G., Varlamis, I., & Nørvåg, K. (2010). Semanticrank: ranking keywords and sentences using semantic graphs. In *Proceedings of the 23rd International Conference on Computational Linguistics. COLING* (pp. 1074–1082).

Tubau, S. (2008). Negative concord in English and Romance: Syntax-morphology interface conditions on the expression of negation. Netherlands Graduate School of Linguistics.

Turney, P.D. (2002). Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*.

UXMatters (2005). User experience definition.

Wang, C., Yu, X., Li, Y., Zhai, C., & Han, J. (2013). Content coverage maximization on word networks for hierarchical topic summarization. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management* (pp. 249–258): ACM.

Wessa, P., & De Rycker, A. (2010). Reviewing peer reviews: a rule-based approach. In *International Conference on E-Learning (ICEL)* (pp. 408–418).

Xiong, W., Litman, D.J., & Schunn, C.D. (2010). Assessing reviewer's performance based on mining problem localization in peer-review data. In *EDM* (pp. 211–220).

Yadav, R.K. (2016). Web services for automated assessment of reviews. Master's thesis, NC State University.

Yang, X., Ghoting, A., Ruan, Y., & Parthasarathy, S. (2012). A framework for summarizing and analyzing twitter feeds. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. KDD* (pp. 370–378). doi:10.1145/2339530.2339591.

Zhai, Z., Liu, B., Xu, H., & Jia, P. (2011). Clustering product features for opinion mining. In *Proceedings of the fourth ACM international conference on Web search and data mining* (pp. 347–354).

Zhang, R., & Tran, T. (2010). Review recommendation with graphical model and em algorithm. In *Proceedings of the 19th international conference on World wide web* (pp. 1219–1220). WWW '10.

Zhang, J., & Yang, Y. (2003). Robustness of regularized linear classification methods in text categorization. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval* (pp. 190–197).