Assessing MCR Discussion Usefulness using Semantic Similarity

Thai Pangsakulyanont*, Patanamon Thongtanunam†, Daniel Port‡, Hajimu Iida†

* Kasetsart University, Thailand
b5410545036@ku.ac.th
† Nara Institute of Science and Technology, Japan
patanamon-t@is.naist.jp, iida@itc.naist.jp
‡ University of Hawaii at Manoa, USA
dport@hawaii.edu

***Abstract*—Modern Code Review (MCR) is an informal practice whereby reviewers virtually discuss proposed changes by adding comments through a code review tool or mailing list. It has received much research attention due to its perceived cost-effectiveness and popularity with industrial and OSS projects. Recent studies indicate there is a positive relationship between the number of review comments and code quality. However, little research exists investigating how such discussion impacts software quality. The concern is that the informality of MCR encourages a focus on trivial, tangential, or unrelated issues. Indeed, we have observed that such comments are quite frequent and may even constitute the majority. We conjecture that an effective MCR actually depends on having a substantive quantity of comments that directly impact a proposed change (or are "useful"). To investigate this, a necessary first step requires distinguishing review comments that are useful to a proposed change from those that are not. For a large OSS projects such as our Qt case study, manual assessment of the over 72,000 comments is a daunting task. We propose to utilize semantic similarity as a practical, cost-efficient, and empirically assurable approach for assisting with the manual usefulness assessment of MCR comments. Our case-study results indicate that our approach can classify comments with an average F-measure score of 0.73 and reduce comment usefulness assessment effort by about 77%.**

***Keywords*—Modern Code Review, Software Quality, Text Mining**

I. Introduction

Software code review is a well-established software quality management practice. Fundamentally, the intent is to identify defects early and encourage quality development policy and practices (such as adherence to coding standards and improving code readability). Traditionally, code reviews were a highly structured activity, but nowadays, less formal and lightweight code reviews such as Modern Code Review (MCR) [1] are commonly used within both industry and OSS projects. One reason for this is that with MCR, unlike formal inspections [2], in-person meetings are not required. Reviewers can find problems in proposed changes and hold discussions on them by adding comments through a code review tool or a mailing list. Developers then improve the changes in response to these comments and re-submit for review until the changes are approved. The popularity of MCR and readily available data have made it an attractive research area. In particular, because performance of code review has a clear relationship to the quality of the software, many studies have investigated the factors that influence MCR [2]–[5] effectiveness. However, little research exists investigating *how* review discussion for a proposed change impacts software quality. Given that most proposed changes are triggered from reviewer comments [2], it's possible that comments can either positively contribute to the proposed changes (i.e. are "*useful*"), or be a discussion which contributes little directly to the proposed changes (i.e. relatively "*useless*"). Since MCR does not require strict guidelines, checklists, or formal processes, some reviewers tend to focus on trivial issues (e.g. coding styles), leaving deeper and possibly more quality-important issues undiscussed [1]. This is a concern because a low-degree of useful comments may degrade the overall effectiveness [6], [7] and perceived utility of MCR.

Prior research has focused primarily on the relationship between number of comments and quality. To the best of our knowledge, the connection between degree of useful and useless comments and quality has not been studied. One reason for this may be that to assess the impact of discussion in MCR, an effort-intensive manual classification is required as in the study of Beller et. al. [2]. So while the data may be easy to access, MCR can produce a massive amounts of change requests and comments [8] and it is painstaking and time consuming to manually assess their usefulness. Moreover, given that comments in MCR are unstructured natural text, open to subjective judgment, contain tacit understanding, and possible misinterpretation, assessment errors are inevitable.

In this paper, we present a text mining approach using semantic similarity to assist in classifying the usefulness of comments. As manual classification can quickly become an untenable task, our proposed approach is meant to ease classifying large amounts of MCR comments in order to conduct comment and code quality studies (such as those discussed earlier). By introducing automation support for assessment grounded with objective empirical criteria, our proposed method aims to both reduce effort and increase confidence in classification to a degree that is practical for addressing very large MCR data sets. Hence through our case study of Qt project, in this work we explore the following research questions:

RQ1: Is semantic similarity a good indicator of MCR comment usefulness?

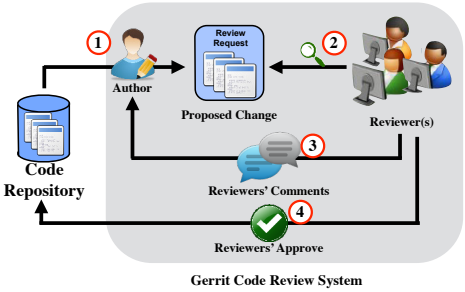
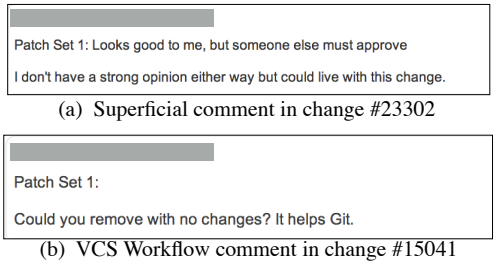


Fig. 1: A simplified version of MCR Process based on Gerrit system



(a) Superficial comment in change #23302

(b) VCS Workflow comment in change #15041

Fig. 2: Examples of comment in code reviews of Qt project.

RQ2: Is semantic similarity classification cost-efficient, assurable, and scalable for large MCR data sets?

The contributions in this paper are as follows:

- An investigation on the practicality of mining natural text comments in code reviews to classify their usefulness using semantic similarity.
- Determination and validation of a model relating usefulness of comments and semantic similarity.
- A case-study application and performance analysis of the model for a large open source project based (Qt) on actual human effort.

II. Modern Code Review Process

An overview of MCR process based on the Gerrit code review system is shown in Fig. 1. The grey area shows the review process of the system which is composed of four main steps: (1) An author creates a patch and submits a set of new or modified files as a review request to the Gerrit system. (2) Reviewers examine the proposed code changes for defects or concerns. (3) Reviewers provide comments to the author. The author creates a revised proposed change according to the comments and re-submits for review. Reviewers then examine the revised proposed change. If the revision is not approved, reviewers provide comments to fix and the process continues until (4) reviewers can determine that a proposed change can be merged into the project (approved change) or should not be merged (rejected).

According to this process, reviewers' comments are the most important factor in determining software quality. In particular, McIntosh et. al. [4] found that components which were reviewed without discussion (i.e. no comments) are likely to contain bugs. However, it has not been shown that components with comments are less likely to contain bugs. Tools supporting MCR such as Gerrit allow reviewers to freely write messages to authors (and other reviewers), and frequently, these comments are superficial or do not clearly identify defects or relevant issues for the proposed change and are thus of questionable usefulness and consequently may have little impact on quality. For example, a comment may be related to something outside the scope of the proposed change, perhaps author or reviewer personal issues or inter-personal communication e.g. "Keep up the good work everyone!". Microsoft developers reported that the reviewers often only focus on minor logic errors rather than discussing deeper design issues [1]. Our observations of the Qt project also corroborate this finding. We found that many examples of comments that are superficial and unhelpful to the proposed changes. For example, the superficial and unconfident comment shown in Fig. 2(a) is certainly within the scope of the proposed change, yet provides little useful information. While the comment in Fig. 2(b) about using the version control system (in this case Git) is clearly out of scope, and not directly useful for the proposed change.

Ultimately, our aim is to understand the impact of useful and useless comments on code quality. Specifically, what degree of useful comments are needed to effect positive impact on quality. Study of this necessitates ascertaining, with high confidence, the usefulness of a large number of comments. This had proven difficult to accomplish manually, so here we investigate the pragmatics of using semantic similarity to assist with this task.

III. Semantic Similarity Classification of Comment Usefulness

Definition: We define a *useful* comment as one that directly contributes to improving a proposed change, and a *useless* comment as one that does not[1].

Our goal here is to improve the efficiency and confidence in the task of determining the usefulness of review comments. As with many subjective assessments, usefulness is not dichotomous and hence we must accept a third qualification, *unclear*, to allow for the case where a comment does not directly make a positive contribution but is not clearly out of scope or tangential i.e. it may be useful. Our approach is largely based on the assumption that usefulness is determined by the relevance a comment has to the proposed change (as documented within a MCR system). A primary factor in relevance is similarly, and so it is natural for us to consider classification using *similarity conditions*. We speculate that the more similar a comment is to the proposed change, the more relevant it is and hence more likely useful. Conversely, the more dissimilar[2], the less relevant, and hence less useful.

A. Approach

We seek to develop a model based on similarity criteria to classify (i.e. *determine*) comments as either useful or useless.

[1] Although we call these comments *useless* in our study, it might actually have utility–or sometimes be required—in some other contexts, such as to facilitate communication or to enforce guidelines or process.

[2] This is technically not simply the opposite of similarity. Unlike similarity, dissimilarity can be arbitrarily large.

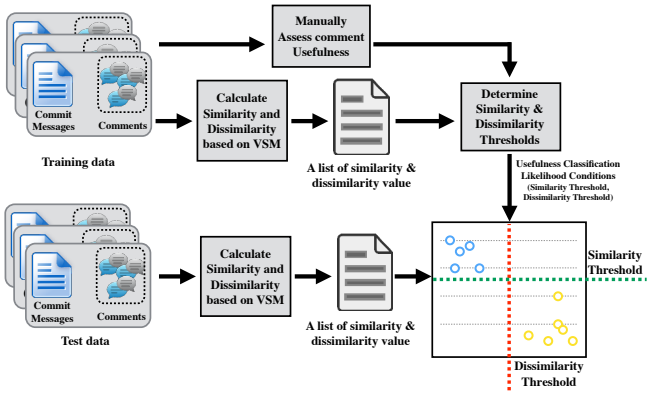


Fig. 3: An overview of classification of comment usefulness using semantic similarity

Given the subjective nature of usefulness, precise classification conditions probably do not exist and so we resort to conditions that indicate a meaningful *likelihood* of classification. That is, for some degree of similarity a comment is determined to be *most likely useful*, and for some degree of dissimilarity it is *most likely useless*. Because this is a likelihood model, it is possible that a classification cannot be determined (i.e. no classification is likely enough) or determined to be in multiple classifications (i.e. overlap in likelihood conditions). It is important to keep in mind that these last two outcomes are not actual classifications whose interpretation is difficult and perhaps unreliable. The key to our approach is to determine the classification conditions in a practical, assurable manner that also is effective to use for assessing comment usefulness.

Fig. 3 shows an overview of our proposed classification approach. Usefulness of a comment is determined by computing its semantic similarity with the commit message of its proposed change and observing if it satisfies *usefulness classification likelihood conditions*. The conditions are simply threshold values of similarity and dissimilarity empirically determined to optimize a desired likelihood objectives. While more complex conditions are possible and may perform better, we begin with this simple model and investigate if it is sufficient for our assessment task.

We calculate semantic similarity using the Vector Space Model (VSM) with the cosine similarity measure, which is a well-known technique for retrieving relevant documents written in unstructured natural language. The Euclidian distance, which is a well-established measure of dissimilarity, applies here analogously for retrieving irrelevant documents.

The model is a form of *supervised classification* where a training set is used to empirically derive similarity and dissimilarity thresholds that optimize meaningful likelihoods e.g. likelihood that a comment is useful given its similarity is greater than some threshold value. The training data is a small set of randomly-selected pairs of comments and the corresponding commit messages. These comments are manually assessed for usefulness. For statistical purposes, at least 30 useful and 30 useless comments must be collected [9]. The similarity and dissimilarity metrics are then computed for the training data set. These are used to estimate the similarity and dissimilarity thresholds S_T and D_T that best discriminate useful comments and S'_T and D'_T from useless comments. We find these thresholds by selecting s_t, d_t values that maximize the F-measure [6], [7] which is a performance measure for a binary classification that compromises trade-offs in accuracy of classification (precision) and coverage of classification (recall).

B. Usefulness Classification Model

Using thresholds derived from the training data as discussed above, we can classify the usefulness of comments for the entire review according to the classification model as follows: given a comment c and a corresponding commit message m,

- **Useful**: $\Theta(c, m, S_T, D_T) = \text{True}$ iff $\text{sim}(m, c) \geq S_T$ and $\text{dist}(m, c) \leq D_T$.
- **Useless**: $\Omega(c, m, S'_T, D'_T) = \text{True}$ iff $\text{sim}(m, c) \leq S'_T$ and $\text{dist}(m, c) \geq D'_T$.
- ***undetermined***: neither (no condition holds) or both (overlap of conditions) of the above conditions are True.

The functions $\text{sim}(m, c)$ and $\text{dist}(m, c)$ are the similarity and dissimilarity measures relative to the proposed change commit message m for the text of comment c using cosine similarity and euclidian distance, respectively. Two metrics are used because they are independently generated and thus provides better discrimination for classification. We also found experimentally that using two metrics has higher performance than just one. We note again that "undetermined" is not a meaningful classification in our model.

While our proposed approach is straightforward, empirically driven, and automatically adjusts to the quality of the data, it has a few drawbacks. First, we must accept that some comments cannot be reliably or confidently classified. We denote these as *undetermined* to differentiate them from comments whose usefulness is *unclear* (i.e. cannot be subjectively assessed as useful or useless). Secondly, owing to the subjective assessment of usefulness and our need to represent this, not just any classification, this fundamentally implies performing supervised classification requiring nontrivial training data to define representative classification sets.

IV. Case Study

Our case study uses the Qt project review history from the Gerrit code review tool provided by Hamasaki et. al. [5]. Qt project is a large open source project composed of numerous small subsystems. We selected the most active subproject, which is called `qtbase`. From this subproject, we used the review history from May 2011 to June 2012, which contains 6,605 changes and 72,484 comments.

A. Data Preparation

We use the commit messages and comments within Gerrit as our data set. Before classifying the usefulness of comments, we first processed the commit message and comments as follows:

1) Removal of automatically generated messages: To consider only reviewers discussion, we ignore all messages that were automatically generated. These messages may be generated by Gerrit, the continuous integration system, and the

sanity bot[3] to record activities. *"Change has been successfully cherry-picked to the staging branch as ..."*, and *'Sanity review passed'* are some examples. In practice, they are not considered in a code review and are not substantively relevant to the proposed changes and do not directly impact software quality [4].

We first find and remove all comments written by bots (i.e. *Qt Continuous Integration System* and *Qt Sanity Bot*). Next, we removed messages, including reviewers comments, with common useless patterns or occur frequently using regular expressions. Some example are *"Upload patch set 1."* This leaves us only human-written comments that cannot automatically be classified as useless.

2) Data preprocessing: As is customary for VSM processing, we extracted semantic words from commit messages and comment messages before converting to vector. For each message, we removed all punctuation signs (except apostrophe) and other non alphanumeric characters. We also removed common words (e.g. a, an, the) using Google stop word list[4]. We then used Porter stemming algorithm to remove the commoner morphological and inflexional endings from words in English.

Table I summarizes data set we used for this study after preparation.

TABLE I: A summary data sets and some statistics.

	Total Number	Percentage
Commit Messages	6,605	-
All Comments	72,484	-
Reviewers Comments	10,583	15%
Automated Comments	61,814	85%

B. Manual Comment Usefulness Assessment

In this case study, three participants (the first two authors and one student) independently assessed comment usefulness by addressing the question, "Is this comment directly useful for approving or modifying the change request?" Then, the participants gave a vote for `YES` if the comment is likely to be useful and `NO` otherwise. From the voting scores, we regarded comments with three `YES` votes as *useful* comments, and comments with no `YES` votes (i.e. three `NO` votes) are *useless* comments. For the comments with one and two `YES` votes, we defined them to have *unclear* usefulness.

C. Research Questions

RQ1: Is semantic similarity a good indicator of MCR comment usefulness?

To answer this question, we randomly sampled 318 comments from our data set and manually assessed their usefulness. Then, we used these comments for both training and test data for our approach to determine its effectiveness. We also determined the confidence in the effectiveness estimates using bootstrapping cross validation to estimate variability in the performance measures due to using sample data.

[3]It is a bot that automatically checks new proposed changes for trivial sanity issues, such as line endings, copyright notices, and commit messages

[4]Available at http://meta.wikimedia.org/wiki/Stop_word_list/google_stop_word_list#English

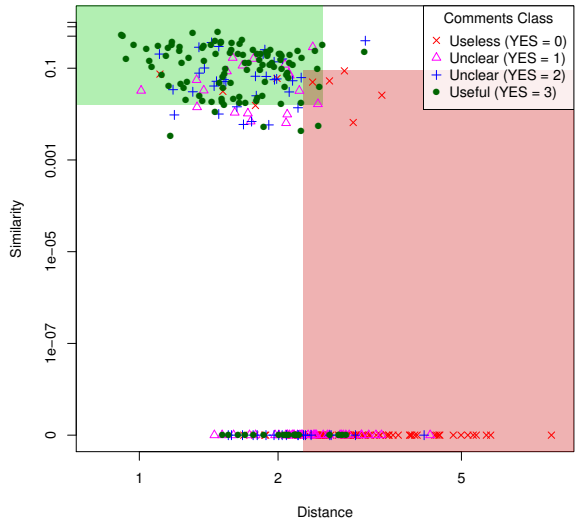


Fig. 4: The similarity and distance plot of the training data. The symbol represents the score, which ranges from 0 to 3. The green area represents the *useful* classification model and the red area represents the *useless* model.

To determine similarity and dissimilarity thresholds for our model, we iterated S_t and D_t values from every unique similarity and dissimilarity value in our data set. We selected the thresholds that best classify useful and useless comments based on maximizing the F1-measure score. The models are $\Theta(c, S_T = 0.015529, D_T = 2.494944)$ for useful comments, and $\Omega(c, S'_T = 0.087522, D'_T = 2.265679)$ for useless comments.

Model Effectiveness Results: From 318 samples, the comment sets from manual assessment were 85 were useless (`YES` = 0), 111 were unclear (`YES` = 1 or 2), and 122 were useful (`YES` = 3). The precision and recall were 0.701 and 0.787 for useful model, and 0.648 and 0.824 for useless model.

Figure 4 shows the relationship between usefulness classes from our model and comment sets from manual assessment. The green area represents the *useful* classification model and the red area represents the *useless* classification model. The comments drawn in the green area are those that satisfied the condition for useful comments, and the comments drawn in the red area are those satisfied the useless condition, while the comments drawn in the white area are those not satisfied any condition (i.e. *undetermined*) and hence remain unclassified. The total number of comments from this classification results is described in Table II.

Corresponding to the precision and recall values, the figure shows that the majority of comments drawn in green area are comments from the useful set; very few comments from useless set fall in this area. Similarly, the majority of comments drawn in the red area are comments from the useless set. It also shows that the similarity and dissimilarity values of comments in each set are not different from other comments in the same set. Most comments from the useful set are in the top left of the graph while those from the useless set adhere in the bottom right of the graph. However, the unclear set still spread out over the similarity intervals and therefore we cannot determine a

TABLE II: Number of comments classified by our approach against ground truth data

Result from Classification Models	Results from manual assessment				Precision
	Useless (YES=0)	Unclear (YES=1)	Unclear (YES=2)	Useful (YES=3)	
Useful Class	3	12	24	95	0.709
Useless Class	69	23	7	6	0.657
Overlap	1	1	0	1	-
Undetermined	12	24	20	20	-
Recall	0.812	-	-	0.778	

relationship with their similarity values. That is, we have no evidence to support that the set of undetermined comments corresponds to the unclear set, and hence, undetermined is not a useful classification.

There is an overlap section shown in Fig. 4. This is a conflict between the two classification conditions where a comment satisfies both simultaneously. This phenomenon can occur since similarity and dissimilarity threshold values of each group are "fuzzy" and both models tend to maximize coverage of classification. Also similarity and dissimilarity are discreet and subject to some degree of granularity issues However, from the results in Table II, only three comments were selected by both models (overlap). In more extended experiments, less than 2% were in the overlap set and this contradictory classification is relatively small. Excluding the overlap classification results, we can calculate precision and recall as described in Table II. Furthermore, 12 useless comments (14%) and 21 useful comments (17%) are still undetermined (drawn in white area). They were not satisfied by our model, thus suggesting that there are likely other indicators different from similarity that determine usefulness.

Validation: We validated our approach using bootstrapping cross validation. By validation, we mean to estimate the variability in the performance measure estimates from the training set to indicate how a accurate these are for the method in general. We randomly selected 90% of 318 comments for training set and determine thresholds. The constructed model is then applied on the remaining 10% comments as the validation set. The precision, recall and F-measure scores are measured and recorded. This validation was repeated 300 times to give the average and standard deviation for these performance measures. Bootstrapping is used because of concern about our relatively small sample size and possibly instability in using customary jackknifing cross-validation. For a baseline comparison, we also determined the performance of a random classifier (i.e. places items in classes with equal probability) using the same methods.

Table III describes the performance of useful and useless classification models including an average and standard deviation of precision, recall, and F-measure. The results show that both of our models can achieve 60% of precision and 75% of recall, approximately. In addition, while we do not present hypothesis test results, it is clear that our model significantly outperforms the random model on all performance measures (for example, the mean precision for Ω is notably higher than the random classifier with about the same standard deviation).

RQ2: Is semantic similarity classification cost-efficient, assurable, and scalable?

TABLE IV: Percentage of classifications results for all comments in the `qtbase` project

Result from Classification Models	Sample Comments 318 comments	All Comments 10,583 comments
Useful Class	41.8%	42.8%
Useless Class	33.1%	33.7%
undetermined	25.1%	23.5%

Automatic systems are generally known for saving human effort and also reduce errors. We analyze the results to determine the performance of our models. In addition, we also consider scalability of our models as it only validated for small numbers of comments. The concern is that the model may behave poorly at larger scale possibly leaving more manual assessment effort than is saved through using the method.

Cost Efficiency: We determined cost efficiency in terms of time use for manual assessment. We considered only the time interval between two votes that are not more than 10 minutes apart since practitioners did not make assessments under a controlled environment. The average time interval between votes is 28 seconds. Thus, for 318 comments, the assessment time should be 7.42 person hours. Using the same average interval time for all 10,583 comments, we can imply that the manual assessment time would be 82.3 person hours. When applying our approach to classify these all comments, the models left only 23.5% of comments for manual assessment (undetermined comments) as shown in Table IV. This means that only 19.34 person hours of additional manual assessment of these undermined comments are needed, thus saving 76.5% of human effort.

Assurance: In Table II, 88% of comments classified as useful received `YES` = 2 and 3 voting scores. The classification of the useless class also have similar proportion i.e. having high number of comment with `YES` = 0 and 1 voting scores. We conjecture that the unclear comments with `YES` = 2 might actually be useful comments and comments with `YES` = 1 might actually be useless. The unclear result could be from error of manual assessment caused by human. To verify this conjecture, we asked practitioners to re-examine the unclear comments. We found that 17% of these 66 comments became unclear because of human error. This shows that some unclear comments are actually useful or useless, thus giving our model more confidence.

Scalability: Applying our classification model to all comments results in the proportions shown in Table IV. As the table shows, the proportions are very similar in both the training dataset and all dataset. Indeed experiments for our case study using increasingly large random subsets of comments show the percentage of undetermined and overlap classification comments (and hence the manual effort required) is constant at about 23%. While the percentage might vary for other projects, this result suggests that our method is scalable to very large numbers of comments—that is, the percentage of unclassified comments does not increase. This enables us to assess MCR comments for very large projects that currently are impractical to do without assistance.

TABLE III: Results from bootstrapping cross validation of our classification models against random models

Classifcation Models		Precision		Recall		F-measure		Accuracy	
		Avg.	STD.	Avg.	STD.	Avg.	STD.	Avg.	STD.
Useful	$\Theta(c, S_T, D_T)$	0.654	0.116	0.759	0.123	0.693	0.089	0.752	0.067
	Random	0.421	0.114	0.376	0.116	0.496	0.144	0.496	0.089
Useless	$\Omega(c, S'_T, D'_T)$	0.636	0.144	0.755	0.148	0.681	0.118	0.815	0.064
	Random	0.336	0.131	0.269	0.115	0.478	0.182	0.500	0.089

V. Discussion

Our results have shown that our semantic similarity is practical to assist in reducing effort and error proneness of manual assessment of comment usefulness. Besides this result, we also found some noteworthy findings in our case study:

A. What kind of comments cannot be classified using semantic similarity?

From the results of RQ1, we found that some comments were not determined by our model. Specifically, we were interested in why some manually assessed useful comments (`YES` = 3) and useless comments (`YES` = 0) were undetermined. To investigate the reason behind this, we reviewed these comments and the corresponding commit messages. We found that there were few common keywords between the comments and commit messages and their lengths are relatively unequal. This makes for low similarity values and low dissimilarity values as shown in bottom left white area in Fig. 4. There simply isn't enough semantic information to determine anything.

B. Can other text mining techniques be used to classify discussions?

To find relevance between comments and the corresponding commit message, topic modeling technique can be used, as the study of [10]. This technique discovers topics for a given collection of documents. However, in MCR context, the documents generally are very short statements and discussions, which makes it unsuitable for LDA. Nevertheless, we have tried applying Latent Dirichlet Allocation (LDA, a well-known topic modeling technique) model on our data set. As expected, LDA generated ambiguous topics with many unrelated keywords. Again, as discussed above, there is simply not enough information since the content of the documents are too short, and thus is insufficient for LDA to correctly determine the topic. Others techniques could be investigated in the future to improve results.

C. Can the determined thresholds in this study be used to other projects?

In this study, we determined the similarity conditions using a training set from the Qt project and the results show that the conditions can indeed classify usefulness of comments. However, we have no basis for believing that the threshold values obtained for one project would apply equally well for another project. To apply our approach on other projects, a new training set is still required which implies additional manual assessment. This limits the practicality of application our our method to projects with a large enough number of comments, where the classification effort saved overcomes the training costs. To fully automate this approach, we have to investigate the use of similarity conditions for many other different projects. Thus, additional studies are needed to improve our approach and potentially identify invariant conditions for similarly and usefulness.

VI. Conclusions

In this paper, we propose a practical approach to classify usefulness of discussion comments in MCR based on VSM similarity. The results from our case study show that our approach can classify comments with better performance than a random model both for precision and recall. In addition, we found that 85% of comments in Gerrit are automatically generated and not likely useful (but important for record keeping). For the remaining 15%, about 43% of them are automatically classified as useful, while about 34% are classified as not useful. The remaining 23% could not be classified by model, and required manual classification. Thus the classification effort is reduced by about 77%.

More work could be done in the future to relate the amount of useful comments to the software quality. This could enable a way of assessing process quality in a quantitative way. Future work could also consider different models other than semantic similarity.

References

[1] A. Bacchelli and C. Bird, "Expectations, Outcomes, and Challenges of Modern Code Review," in *Proc. of ICSE '13*, 2013, pp. 712–721.

[2] M. Beller, A. Bacchelli, A. Zaidman, and E. Juergens, "Modern Code Reviews in Open-Source Projects : Which Problems Do They Fix ? Categories and Subject Descriptors," in *Proc. of MSR'14*, 2014, pp. 202–211.

[3] O. Baysal, O. Kononenko, R. Holmes, and M. W. Godfrey, "The Influence of Non-Technical Factors on Code Review," in *Proc. of WCRE'13*, 2013, pp. 122–131.

[4] S. Mcintosh, Y. Kamei, B. Adams, and A. E. Hassan, "The Impact of Code Review Coverage and Code Review Participation on Software Quality Categories and Subject Descriptors," in *Proc. of MSR'14*, 2014, pp. 192–201.

[5] K. Hamasaki, R. G. Kula, N. Yoshida, C. C. A. Erika, K. Fujiwara, and H. Iida, "Who does what during a Code Review ? An extraction of an OSS Peer Review Repository," in *Proc. of MSR' 13*, 2013, pp. 49–52.

[6] P. K. Prasetyo, D. Lo, P. Achananuparp, Y. Tian, and E.-P. Lim, "Automatic classification of software related microblogs," in *Proc. of ICSM'12*, 2012, pp. 596–599.

[7] S. Gottipati, D. Lo, and J. Jiang, "Finding relevant answers in software forums," in *Proc. of ASE'11*, 2011, pp. 323–332.

[8] P. Thongtanunam, R. G. Kula, A. E. C. Cruz, N. Yoshida, and H. Iida, "Improving Code Review Effectiveness through Reviewer Recommendations," in *Proc. of CHASE'14*, 2014, pp. 119–122.

[9] R. V. Hogg and E. A. Tanis, *Probability and Statistical Inference 7th Edition*, 2005.

[10] A. Barua, S. W. Thomas, and A. E. Hassan, "What are developers talking about? An analysis of topics and trends in Stack Overflow," *Jornal of EMSE'12*, vol. 19, no. 3, pp. 619–654, 2012.