Quality Questions Need Quality Code: Classifying Code Fragments on Stack Overflow

Maarten Duijn,* Adam Kučera,† Alberto Bacchelli‡
*M.J.Duijn@student.tudelft.nl, †kucerad5@fit.cvut.cz, ‡a.bacchelli@tudelft.nl
*‡Delft University of Technology, The Netherlands, †Czech Technical University in Prague, Czech Republic

***Abstract*—Stack Overflow (SO) is a question and answers (Q&A) web platform on software development that is gaining in popularity. With increasing popularity often comes a very unwelcome side effect: A decrease in the average quality of a post. To keep Q&A websites like SO useful it is vital that this side effect is countered. Previous research proved to be reasonably successful in using properties of questions to help identify low quality questions to be later reviewed and improved.**

We present an approach to improve the classification of high and low quality questions based on a novel source of information: the analysis of the code fragments in SO questions. We show that we get similar performance to classification based on a wider set of metrics thus potentially reaching a better overall classification.

I. Introduction

With over 7 million questions answered and 2 million users, Stack Overflow is the largest Q&A website on software development. Like many other Q&A websites the SO community is governed by a reputation system. Users are allowed to vote on questions and answers, providing insight into its perceived quality. This measure of quality is then reflected into the user's reputation. By storing this information, in addition to the question, its answers, and comments, Q&A websites have become a platform that allow experts to share knowledge, and help developers by providing solutions and insight into programming problems.

This knowledge gathering process starts with a developer asking a question. As with any user generated content, the quality of this question is however debatable. SO has a clear guideline on what a good question is, what it contains, and how it should be tagged [1]. The potential benefits of writing quality questions are notable for all parties involved. A well defined question is known to lower response times and thus increase user satisfaction [2]. The platform itself benefits from having better content and thus attracting more users. Creating a tool that detects low quality questions at the instant they are posted could therefore bring value to Q&A systems. The tool could point out some of the flaws to users, and inform moderators so that the question is quickly improved.

Previous research on SO questions (focusing mainly on the metrics of the question itself, the users involved, and the external references mentioned) has provided some insight into what constructs indicate quality [3], [4]. In this paper, we aim to provide additional insight into the quality of a question by investigating the code fragments in a question. SO guidelines tell us that the best questions contain a bit of code but not entire software programs, ideally just enough code for others to reproduce the problem [1]. Existing research has shown that there is a certain optimal code-to-text ratio [5], but more in depth research into code fragments on Q&A sites is lacking.

As required by this year's MSR Challenge we combine two information sources: We consider the code-to-text ratio (which also considers the natural language part), then we focus on code only information by adding existing metrics of code readability [6] and using a set of metrics of our own making. These metrics were created by manually investigating over 200 code fragments in low rated Java questions on SO. During this qualitative study we found 30 constructs indicative for a low quality code fragment. Together with the code-to-text ratio and readability metrics, metrics for these constructs were used in a classifying algorithm. This algorithm classifies questions as either 'good' or 'bad' with an accuracy of approximately 80%.

II. Related Work

The quality of questions has already been investigated from different perspectives. Yang *et al.* [7] found that the number of edits on a question is a very good indicator of question quality. Ponzanelli *et al.* [3] investigated questions' classification as 'very good', 'good', 'bad', and 'very bad' using different SO-specific, readability, and popularity metrics. They reported a precision of ∼80%. We leverage parts of their approach.

A number of studies investigated the relation between question quality and the contained code fragments. Correa and Sureka [4] investigated closed questions on SO, also finding that the occurrence of code fragments is significant. Squire & Funkhouser [5] studied the relation of such an occurrence and the score of the questions and answers; they determined an optimal ratio between text and code. Subramanian and Holmes[8] investigated the code snippets in Java Android questions and they tried to tie them to specific functions from the Android SDK they reference. Allamanis and Suton[9] tried to divide SO questions into different categories and also used several code fragment related features. Buse and Weimer[6] performed analysis on short and incomplete code snippets regarding their readability by humans. From human assessment of readability they inferred several readability metrics, which are independent of the length of the code snippet. Using these metrics they evaluated the code readability. We use some of these metrics in addition to our our own.

III. Methodology

A. Exploratory Findings

In the first step in our data analysis process we manually investigated a number of questions and analyzed their code fragments. We hypothesized that more detail in a code fragment should indicate a better question thus longer code fragments should mean more answers and a quicker response time. Our analysis however revealed that the opposite was true: Questions with more code generally receive fewer answers. This motivated us to take a closer look at the code fragments and their contents and led us to form the premise for our research.

B. Research Questions

Related work suggested that code-to-text ratio is a metric often used for classification but that more in depth analysis of code fragments is very rarely considered[5]. Our goal is to improve the classification of SO questions using analysis of code samples by providing an insight into what features in such a fragment are indicators of question quality. This lead us to the following research questions:

RQ1: Is the question quality influenced by different readability and quality metrics of code snippets?

RQ2: What code snippet metrics are most relevant for determining quality of a question?

C. Dataset Acquisition

To answer our research questions we used the MSR Challenge SO data dump [10]. We imported the data into a MySQL database for a pre-processing step, in which we extracted the code fragments from the questions. If a question contained more than one fragment, we concatenated them into a single one. By filtering all the questions that did not contain code fragments, or questions that were not about the Java programming language we got a dataset containing 521,530 questions.

D. Qualitative Analysis

We determined code metrics indicative for question quality by manually inspecting over 200 Java code fragments in SO questions. During the inspection we noted the constructs that we hypothesized were indicative for the question's quality.

We focused our efforts on finding constructs indicating *low quality* questions. We did this for the following reasons: (1) Research into readability [6] of code reports that most metrics that were used, such as the line length, and the number of keywords, are negatively related to readability; (2) when rating software quality, tools such as inFusion[1] act similarly, only indicating likely problems; and (3) pinpointing a low quality piece of code is, intuitively, easier than identifying high quality code.

We only analyzed questions with a negative score and noted the constructs we thought to be common and likely to be indicative for question quality. As an example we thought the number of comparison operators (*i.e.*, '==' in Java) to be a good candidate for indicating low quality questions. The reasoning behind this is that it is a common rookie mistake to try and check object equality with this operator. Instead the 'equals' method should be used in Java.

Apart from noticing certain constructs, we observed that most of the questions have a negative score because of general shortcomings of the questions themselves. For example, users often ask very easy questions, the questions are not descriptive enough, or a question is not asked at all. Nonetheless, even in questions where code was clearly not the sole cause for a low rating, we still found interesting constructs while investigating the embedded code fragments.

E. Metrics

The metrics used for classification questions are a combination of readability metrics [6], metrics based on the constructs found during the qualitative analysis, and the number of errors reported by a formatting style checker.[2].

Each of the constructs found by the qualitative analysis has one or two associated metrics. These metrics were calculated by first counting the number of occurrences of the construct or measuring the length of the construct. The number returned by counting or measuring is then normalized by either dividing this number by the number of lines in the fragment, or by taking the maximum number of occurrences per line in the fragment. For example, the metric for the '==' construct is calculated by counting the number of its occurrences in a fragment, and then normalized by dividing this number by the number of lines in the fragment. The complete list of metrics created by us is available at http://sback.it/codemetrics. Because of limitations to the size of this paper we only report the most significant here.

F. Question Quality Criteria

To verify the classification of questions one needs to know how good they actually are. We used two approaches. First, SO questions have a score which is close to a metric for quality. The score of a question is a combination of its popularity and its quality, thus we split the questions into two categories: (1) Low quality, *i.e.*, questions with a score < 0; and (2) high quality, *i.e.*, questions with a score > 0.

Questions with a score of zero are usually not popular enough to get a reliable verdict, thus we excluded them from the dataset. The edited questions pose another problem, the score could potentially represent an older version of the question and therefore they were excluded as well. With the above constraints, our dataset contained 123,688 questions, out of which 11,997 low quality and 111,671 high quality.

Yang *et al.* [7] reported the number of edits to be a valid indicator for question quality. We based our second approach on this finding, and divided the questions into two categories: (1) Low quality, *i.e.*, questions without edits; and (2) high quality, *i.e.*, questions with edits. This caused the dataset to be

[1] http://www.intooitus.com/products/infusion

[2] http://checkstyle.sourceforge.net/

TABLE I
CLASSIFICATION ACCURACY BY ALGORITHMS AND QUALITY CRITERIA

ML algorithm	Classification accuracy			
	Score quality criteria		Edit quality criteria	
	code/text metric	all metrics	code/text metric	all metrics
Decision tree	58.3%	60.4%	52.3%	55.0%
Logistic regression	58.0%	65.1%	52.3%	55.6%
Random forest	69.2%	79.8%	73.3%	81.2%

divided into 247,156 non-edited questions and 274,374 edited questions.

G. Classification

Classifying the questions into the previously mentioned categories was done in three steps. (1) We measured correlations between individual metrics and our quality metrics. (2) To classify questions based on the calculated metrics we used three classification algorithms: decision tree, logistic regression, and random forest. We used a balanced sample of randomly selected questions to train and test the algorithms. To evaluate the performance of the algorithms, we conducted 10-fold cross validation. (3) To determine the most important metrics for the question quality, we used the feature importance of the random forests algorithm, metrics correlations, and created decision tree analysis.

IV. RESULTS

A. Question Classification

To answer RQ1, we used the aforementioned quality criteria and classification algorithms. Table I summarizes the results, which show that the algorithms reach a reasonable accuracy. Providing evidence that the quality of a post is influenced by readability and quality of code snippets. All metrics combined improve results of code-to-text ratio alone, thus showing the relevance of adding this new source of information. The improvement is especially evident on the best classifier, *i.e.*, random forest.

B. Metrics Importance

To answer our second research question, we analyzed the results more in-depth and focused on combined metrics. First we evaluate the metrics with a Pearson's correlation coefficient. Most of the metrics show a small negative correlation with question quality, suggesting that the chosen code metrics indicate bad code. The metrics with the most significant correlation with score are summarized in figure 1. Most of the metrics also have a small positive correlation with the number of edits, which is shown in figure 2.

However, the correlation itself shows only the dependence of quality on metrics and some of these metrics might not be useful for classification algorithms. Therefore we also extracted metric importance from the random forest algorithm of which the most significant are summarized in figure 3. Lastly the created decision tree was used to gain insight not only into which values were most important but also their thresholds.

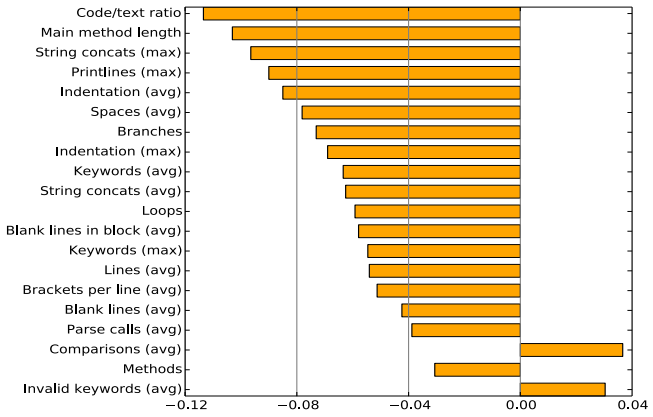

Fig. 1. Pearson's correlation between metrics and score

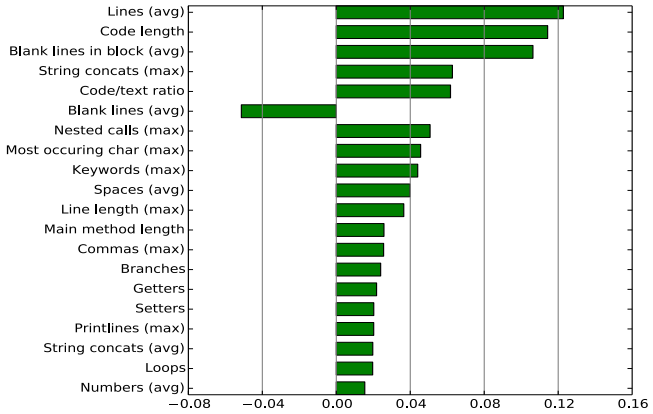

Fig. 2. Pearson's correlation between metrics and number of edits

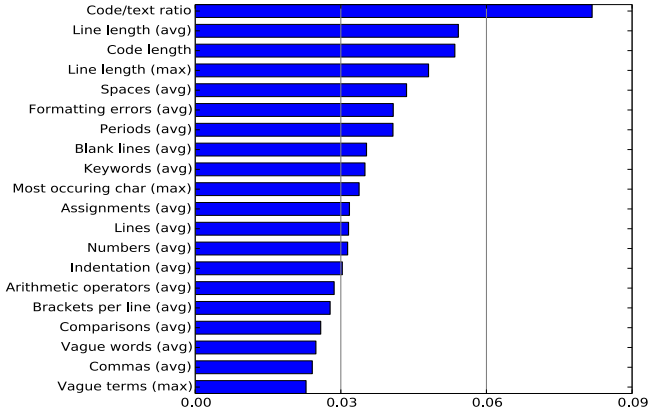

Fig. 3. Random forests metric importance

V. DISCUSSION

The performance of the Random forest classification algorithm is very good and this algorithm could be used for

detecting low quality questions at the moment they are posted. It achieves similar performance as related work [3], which focused on different types of question metrics, but did not look into metrics of the code itself. Combining their metrics with our metrics could increase the classification performance further. The mentioned paper also leveraged state-of-the-art learning genetic algorithm, which may also increase the performance.

We found that most important code metrics related to question quality are those most important for general code readability, such as length of the lines, whitespace occurrence or number of formatting errors. This suggests that presenting "clean" code is important when asking questions on SO. The Pearson correlations also show that certain constructs should be avoided. Print line statements, for instance, have a relatively high negative correlation with score, indicating that these subtract value from the code fragment.

Our findings can be used in a number of ways to improve question quality on SO. Firstly, the set of guidelines on SO could be updated to avoid complexity and certain constructs as much as possible. The second way our results can be used is by using the constructed classifier to automatically rate code fragments. When a question contains code that is not readable enough, or that is too complex, SO could warn the poster and the administrators. Posters can then use the feedback from the algorithm to improve their questions.

There are some threats that endanger the validity of our results. Due to reputation gaming on SO, score may not represent exact quality of a question. Moreover, many questions have a score equal to zero, therefore their quality is unknown. Since we focused only on code snippets, we are completely dependent on the askers and their proper marking of those snippets with the <code> tag. There are questions, with non-code text, like stack traces, marked with this tag that were not excluded; this could influence the effect of some metrics.

An important thing to note about classification with regard to the edit quality criteria is that we were not able to uncover the original code fragment. The metrics we extracted are based on the latest, thus edited, version of the question.

We also performed our analysis purely on Java questions. An analysis of questions concerning other programming languages might need a different set of metrics, which might have different importance for the classification algorithms.

VI. Conclusion

Q&A websites, such as SO, are becoming more and more popular. An unwelcome side effect that endangers this growth is low quality questions. By asking a high quality question, a user can expect a faster and better response, and the website gains valuable information. Therefore websites like SO could profit from having a tool that automatically detects low quality posts at the time they are posted.

Several papers researched quality of the questions on SO, however they have never tried to relate the quality of code snippets included with a question to the quality of the question itself. In this paper, we investigate code snippets in addition to code-to-text ratio on SO and study metrics to evaluate their influence on quality. We also examine which of these metrics contribute the most to a question's quality.

We defined the quality of a question in two ways: marking questions with positive score as good and with negative as bad and also those never edited as bad and edited as good. Using three different machine learning algorithms (Decision Trees, Logistic Regression and Random Forests) we classified the questions into these categories, using the metrics previously inferred and achieved a classification accuracy of 81.2%.

We also investigated the importance of observed metrics using the Random forest algorithm and Pearson's correlation analysis. We found that code-to-text ratio as used in most research is indeed the most important factor. However several metrics, most of them related to code readability, also significantly contribute to the quality of a question, underlining the importance of studying code fragments as a source of information.

References

[1] How do i ask a good question? [Online]. Available: http://stackoverflow.com/help/how-to-ask

[2] V. Bhat, A. Gokhale, R. Jadhav, J. Pudipeddi, and L. Akoglu, "Min(e)d your tags: Analysis of question response time in stackoverflow," in *Proc. of ASONAM 2014*, 2014, pp. 328–335.

[3] L. Ponzanelli, A. Mocci, A. Bacchelli, and M. Lanza, "Understanding and classifying the quality of technical forum questions," in *Proc. of QSIC 2014*, pp. 343–352.

[4] D. Correa and A. Sureka, "Fit or unfit: Analysis and prediction of 'closed questions' on stack overflow," in *Proc. of COSN 2013*, pp. 201–212.

[5] M. Squire and C. Funkhouser, "'a bit of code': How the stack overflow community creates quality postings," in *Proc. of HICSS 2014*, pp. 1425–1434.

[6] R. Buse and W. Weimer, "Learning a metric for code readability," *IEEE TSE*, vol. 36, no. 4, pp. 546–558, 2010.

[7] J. Yang, C. Hauff, A. Bozzon, and G.-J. Houben, "Asking the right question in collaborative Q&A systems," *Proc. of Hypertext 2014*, pp. 179–189.

[8] S. Subramanian and R. Holmes, "Making sense of online code snippets," in *Proc. of MSR 2013*, pp. 85–88.

[9] M. Allamanis and C. Sutton, "Why, when, and what: Analyzing stack overflow questions by topic, type, and code," in *Proc. of MSR 2013*, pp. 53–56.

[10] A. T. T. Ying, "Mining challenge 2015: Comparing and combining different information sources on the stack overflow data set," in *Proc. of MSR 2015*, 2015, p. to appear.