Maximizing Post Efficiency in Stack Overflow

Luís F. Correia Cleto
Delft University of Technology
l.f.correiacleto@student.tudelft.nl

Tiago Almeida Fernandes
Delft University of Technology
t.almeidafernandes@student.tudelft.nl

Alberto Bacchelli
Delft University of Technology
a.bacchelli@tudelft.nl

***Abstract*—The abstract will go here.**

I. Introduction

Software development can be described as a knowledge-intensive activity [1] where knowledge is often distributed among several individuals with different areas of expertise. This implies that practicing efficient knowledge management can be greatly important for software developers.

The widespread usage of social media and the Internet has originated in a rise of Q&A platforms such as Yahoo! Answers[1] or Answers[2] as a mean for exchanging knowledge. In these platforms users are encouraged to participate either by asking questions or answering them. The rewards for participating are usually based on some kind of point or karma system that rewards good answers and questions. While this enables quick and easy access to knowledge in areas unknown to the user, the quality of posts on these platforms can also vary widely from high quality posts made by experts to low quality or even misinforming posts, making quality detection for posts in Q&A websites an increasingly relevant area of research. For the domain of Software Development, Stack Overflow[3] has emerged as one of the dominant Q&A platforms where programmers can exchange knowledge between themselves. Since its foundation in August of 2008, over 10 million discussions have been started on Stack Overflow [2] along with around 17 million solutions.

The increase in usage of Q&A websites has brought a lot of new members as well as questions being asked. Similarly, the number of unanswered questions is also considerable. Stack Overflow alone had over 1.2 million unanswered questions on the 10th of December of 2015. This could be due to several factors, such as the questions being uninteresting, poorly formulated, already answered, too difficult or for other motives. The main purpose of this paper is to focus on identifying what influences the number of answers that a certain Stack Overflow question will get and provide an analysis on how to create high quality posts on this platform. Therefore, it was chosen to investigate the following research questions:

RQ1 *What influences the number of answers to a question on Stack Overflow?* With the study described in this paper we intend to determine what characteristics of a question might determine the number of replies that it receives. These characteristics can be quantitative (code/natural text ratio, length of title, number of tags, presence of certain popular tags, ...) or qualitative (easiness of a question, type of question [3], topic, ...). We expect that at least for certain topics the amount of replies will be greater than for other less popular topics.

RQ2 *Which questions are more likely to receive high quality answers?* As the number of answers alone might simply indicate a controversial topic, it is also important to measure the quality of the answers that have been given since the intent of a user when they post a question on Stack Overflow is to receive a high quality answer rather than several useless answers. For this case, we expect the easiness of a question to have a large impact.

RQ3 *What makes an answer more likely to be accepted on Stack Overflow?* Out of all the answers that are provided, only one gets accepted as the solution. Knowing what makes an answer more likely to be accepted can contribute to increasing the effectiveness of answer posts on Stack Overflow. One possible factor for determining this is if the answer was the first one to be posted.

Our findings will enable us to make recommendations on how individuals and companies in the domain of software development can leverage the knowledge and use Q&A websites effectively in order to provide high quality topics and similarly receive appropriate answers.

The remainder of this paper will be structured as follows: Section II focuses on the background of our research, including related work. Section III describes our approach to the problem at hand as well as the dataset and API that were used during the study. In section IV we present and analyze our findings and section V outlines our conclusions and contains recommendations for future work.

II. Background and Related work

Several studies have been undertaken to attempt to correlate features of a post in a Q&A platform with its quality. Many of these focus specifically on Stack Overflow and how to detect or improve the quality of a post on that platform.

Treude *et al.* undertook a qualitative approach to identifying and categorizing Stack Overflow questions. By manually analyzing a small sample of a few hundred discussions, they were able to identify 10 types of questions and determine that questions of type **review**, **conceptual**, **how-to** and **novice** were the types that got replied to the most often [3]. This might be due to the easiness of replying to **novice** questions, since

[1] http://answers.yahoo.com
[2] http://www.answers.com/
[3] http://stackoverflow.com/

they tend to be quite simple, as well as to *review* questions which mostly contain code fragments and are very concrete, often not requiring any external knowledge for the code to be understood. However, they also concluded that there are several other factors that influence the amount of replies a question can get, such as the length of the question, the presence of code snippets, the technology in question and even the day and time when it was posted.

On the topic of determining the quality of an answer, Hart and Sarma conducted a study to understand how novice users perceive the quality of answers on Stack Overflow [4]. By conducting a survey with 34 novices they found out that: *1*) social factors (like answerer reputation) have little impact on perceptions of answer quality; *2*) answer length is important insofar as longer answers tend to be more thorough; *3*) presentation is important - both code and prose are essential elements of a *high quality* answer. This contrasts with findings in other studies that give a higher relevance to popularity based metrics [5].

Ponzanelli *et al.* [6], by trying to decrease the size of Stack Overflow's review queue, added to the current metrics employed by Stack Overflow readability metrics and popularity metrics. They constructed a linear quality function for each metric and learned each one using genetic algorithms. The authors concluded that popularity metrics and readability metrics are the most useful metric sets to refine the low quality review queue. This suggests that these metrics might be the most adequate for distinguishing *low quality questions* from the rest.

In the domain of quality classification and prediction, Ponzanelli *et al.* [5] also devised and evaluated several metrics grouped into three different categories: Stack Overflow specific, readability, and popularity metrics. They then classified questions as 'very good', 'good', 'bad' or 'very bad'. The reported a precision of their methodology is approximately 80%.

III. Methodology

In order to conduct our research, we will resort to the StORMeD data set and API [7]. It has approximately 900.000 questions relating to the Java language and their respective answers and comments. Each question and answer, besides the usual metadata, has its body represented in a *heterogeneous abstract syntax tree* (H-AST) that encapsulates the content in structured fragments such as Java constructs, stack traces, XML/HTML fragments, JSON fragments or text. In addition, it contains metainformation for common analysis like term frequency data, variable names, and mentioned reference types. It is also worth noting that posts on Stack Overflow (both questions and answers) have a user-determined score which serves as a direct measurement of quality as it is perceived by the community, mitigating the problem of subjectiveness when a single individual is to determine the quality of this type of content.

For conducting this study a Scala framework was developed. It interacts with the StORMeD Devkit allowing the extraction of the specific variables present in the data set needed for our analysis. As there is a considerable amount of data present in the provided dataset, it was chosen to retrieve and process sample data. This data collection can be divided in the following steps:

1) **Extract a sample** - randomly extract discussions from the original data set.
2) **Calculate features** - extract or compute the desired features from the data.
3) **Dump data to a CSV** - save the resulting data to a CSV file, ready to be analyzed.

For *RQ1* and *RQ2* the same CSV will work. To answer *RQ3* a different CSV file must be computed with different features as this research question focus on the answers' content rather than in the questions' content.

The features for *RQ1* and *RQ2* are divided in two: questions' features (*QF*), which are features extracted from the question itself, and questions' classification (*QC*), which are features that allow to classify the question quality. Analogously, for *RQ3* there are: answers' features (*AF*) and answers' classification (*AC*). The metrics are explained in Table I.

TABLE I
METRICS' LIST

	Metric	Details
QF	Title Length	number of characters in question's title
	Tags count	number of tags associated with question
	Tags popularity	computed based on tags popularity
AF	first posted	0 or 1 - 1 if was first answer
	same day	0 or 1 - 1 if posted in question's day
QF & AF	Java %	Java's lines percentage
	JSON %	JSON's lines percentage
	XML %	XML's lines percentage
	Stack traces %	Stack traces' lines percentage
	Length	number of characters
	Words Count	number of words (text only)
	Day of Week	the day of the week (Monday-Sunday)
	Reputation	Author's reputation
	Intercaltions	number of text and code intercalations
QC	Score	question's score
	Answers count	number of answers in discussion
	Max(score)	max score among answers
	Avg(score)	average score among answers
	Min(score)	min score among answers
	Comments count	number of comments to question
	Max(length)	answers' max length
	Avg(length)	answers' average length
	Min(length)	answers' min length
AC	Score	answer's score
	Comments count	number of comments to answer
	Accepted	0 or 1 - 1 if the accepted answer
	Max(length)	comments' max length
	Avg(length)	comments' average length
	Min(length)	comments' min length

IV. Results

Results will go here. At this point, we've conducted an initial analysis to test our framework. We extracted only metrics relating score and number of tags with the number

of replies to a question. For the preliminary results that we found, it does not seem like the amount of tags has much of an influence but the results will need to be analyzed in more detail for a larger sample in order to present valid results.

V. Conclusion

The conclusion will go here.

References

[1] P. N. Robillard, "The role of knowledge in software development," *Commun. ACM*, vol. 42, no. 1, pp. 87–92, Jan. 1999.

[2] StackOverflow, "Stackoverflow," http://stackoverflow.com/10m, online December 2015.

[3] C. Treude, O. Barzilay, and M.-A. Storey, "How do programmers ask and answer questions on the web? (nier track)," in *Proceedings of the 33rd International Conference on Software Engineering*, ser. ICSE '11. New York, NY, USA: ACM, 2011, pp. 804–807.

[4] K. Hart and A. Sarma, "Perceptions of answer quality in an online technical question and answer forum," in *Proceedings of the 7th International Workshop on Cooperative and Human Aspects of Software Engineering*, ser. CHASE 2014. New York, NY, USA: ACM, 2014, pp. 103–106.

[5] L. Ponzanelli, A. Mocci, A. Bacchelli, and M. Lanza, "Understanding and classifying the quality of technical forum questions," in *Proceedings of the 2014 14th International Conference on Quality Software*, ser. QSIC '14. Washington, DC, USA: IEEE Computer Society, 2014, pp. 343–352.

[6] L. Ponzanelli, A. Mocci, A. Bacchelli, M. Lanza, and D. Fullerton, "Improving low quality stack overflow post detection," in *Software Maintenance and Evolution (ICSME), 2014 IEEE International Conference on*, Sept 2014, pp. 541–544.

[7] L. Ponzanelli, A. Mocci, and M. Lanza, "Stormed: Stack overflow ready made data," in *Proceedings of the 12th Working Conference on Mining Software Repositories*, ser. MSR '15, 2015, pp. 474–477.