Improving Low Quality Stack Overflow Post Detection

Luca Ponzanelli,[1] Andrea Mocci,[1] Alberto Bacchelli,[2] Michele Lanza,[1] David Fullerton[3]
1: REVEAL @ Faculty of Informatics – University of Lugano, Switzerland
2: Delft University of Technology, The Netherlands *3: Stack Exchange, New York, USA*

***Abstract*—Stack Overflow is a popular questions and answers (Q&A) website among software developers. It counts more than two millions of users who actively contribute by asking and answering thousands of questions daily. Identifying and reviewing low quality posts preserves the quality of site's contents and it is crucial to maintain a good user experience. In Stack Overflow the identification of poor quality posts is performed by selected users manually. The system also uses an automated identification system based on textual features. Low quality posts automatically enter a review queue maintained by experienced users.**

We present an approach to improve the automated system in use at Stack Overflow. It analyzes both the content of a post (*e.g.,* simple textual features and complex readability metrics) and community-related aspects (*e.g.,* popularity of a user in the community). Our approach reduces the size of the review queue effectively and removes misclassified good quality posts.

I. Introduction

Q&A websites, like Yahoo! Answers and Ask,[1] allow people to ask questions and receive knowledge through answers from the community. The popularity of Q&A websites has also emerged in software engineering, in particular as support for developers. Stack Overflow[2] is the prominent example of a technical Q&A website where developers exchange knowledge about programming problems. Treude *et al.* [1][2] investigated the interaction of developers with Stack Overflow, and reported how this interaction is providing a valuable knowledge base that can be leveraged during software development.

Stack Overflow is steadily growing both in the size of its community and in the amount of the contents it provides. Its data dump of March 2014 contains around 8.01M questions, 13.98M answers and a community of 2.87M users. The people involved in the Stack Overflow community are very active: In the last two years, 6,350 questions and 9,330 answers are created daily in average. Given such a high rate of creation of new posts, assuring the quality of the content in Stack Overflow is a challenge. According to Agichtein *et al.* [3], the quality of the content provided by Q&A websites varies, and ranges "*from high-quality questions and answers to low-quality, sometimes abusive content* [, thus making] *the tasks of filtering and ranking more complex than in other domains.*" In Stack Overflow, the task of keeping up the quality of questions is left to the crowd: Poor quality posts are identified by a selected subset of users in the community (*i.e.,* moderators) who have the rights to closing or delete questions.

Correa *et al.* [4] report that around 80% of the questions take at least 1 month or more to receive a delete vote, and approximately 14% receive 3 delete votes before being actually deleted. This slow deletion process is a symptom of the amount of effort required by moderators to guarantee a satisfiable level of quality in Stack Overflow.

Providing support to the moderators by automatically predicting the quality of a post is a solution to this problem. When the predicted quality of a post is low, it is reviewed manually by the selected users.

Many approaches focus on predicting the quality of Q&A posts. Jeon *et al.* [5] devised a framework based on stochastic processes to predict the quality of answers in Naver.[3] Arai *et al.* [6] presented a general model to predict quality of information in Q&A websites by using three classification algorithms (*e.g.,* Decision trees, Boosting, and Naïve Bayes). Correa and Sureka [4] developed a predictive model to identify deleted questions based on decision trees, by using features extracted from the question's text and the author's information. Adamic *et al.* [7] combine both user attributes (*i.e.,* number of best answer and number of replies) and answer characteristics (*i.e.,* reply length and thread length) to predict if an answer will be chosen as accepted answer by the asker. Stack Overflow implements a system to automatically identify and put low quality posts in an ad-hoc *review queue*. Their approach only relies on textual features (*e.g.,* smileys count, body length).

These approaches try to accelerate the low-quality post identification process by automating it. However, an automated approach with low precision would only move the bottleneck of the process from the identification phase to the review phase: Many misclassified posts lead to wasted time, because moderators have to spend time reviewing high quality posts.

We propose a complementary approach to identifying low quality posts. It uses both features concerning the content of a post (*e.g.,* from simple textual features to more complex readability metrics) and community-related aspects (*e.g.,* popularity of a user in the community). We apply our approach to refine the Stack Overflow review queue, by removing misclassified (*i.e.,* good quality) posts. Our results show that the presented approach is able to reduce the size of the review queue, with a minimal number of false positives.

II. The Stack Overflow Review Queue Process

Low quality posts in Stack Overflow are identified through a review queue system managed by moderators (a restricted set of users with enough reputation to unlock specific privileges[4]).

[1] See http://answers.yahoo.com and http://www.ask.com
[2] See http://stackoverflow.com
[3] http://www.naver.com
[4] http://stackoverflow.com/help/privileges/

Stack Overflow has 7 review queues:[5] Late Answers, First Posts, Low Quality Posts, Close/Reopen Votes, Suggested Edits, Community Eval. We focus on the queue that is of direct interest to us: the *Low Quality Posts* Queue. It contains posts that have been automatically determined as low quality, by using several system criteria that generates a post quality score, or that have been manually flagged by users.

We focus on improving the efficiency of the Low Quality Posts review queue. In particular, we propose an approach to refine the queue to remove misclassified (*i.e.,* good quality) post while retaining the bad quality posts in the review queue.

III. Quality Classification Approach

A. Dataset Construction

We built different datasets, based on the public data dump of September 2013, containing 5,648,975 questions. To avoid subjective classification of the question quality, we relied on the judgment of the crowd and we devised these two preliminary quality levels:

High Quality: Questions, neither closed nor deleted, with a score greater than zero and with an accepted answer; 1,110,260 questions fall into this category.

Low Quality: Questions with a score below zero, closed or deleted in their final state; 152,691 questions fall into this category.

In both cases we discard all questions whose score is 0. We assume that 0-scored questions have not attracted enough interests from the community to evaluate and classify their quality. Moreover, we exclude from the dataset modified questions since we want to avoid influence on the quality evaluation given by the crowd.

The variance of quality among posts considering only two levels of quality is remarkable [3], and thus a two classes classification is too coarse grained. In fact, the two sets also contain a lot of borderline quality questions.

To reduce this effect we further refine each quality class by identifying 'very good' and 'very bad' questions. Table I reports the distribution of the quality classes in our dataset with the related features.

TABLE I. Quality classes of the questions in our dataset.

Class	Description	Size
A	*Very good questions* (with accepted answer, not closed, not deleted, score > 7)	76,592
B	*Good questions* (with accepted answer, not closed, not deleted, score 1-6)	1,033,676
C	*Bad questions* (not closed, not deleted, score < 0)	70,837
D	*Very bad questions* (closed or deleted)	81,854
	Total	1,262,959

For the purpose of our study, we created two different datasets that we need for training and testing.

As we see in Table I, the four classes are unbalanced. In particular the class *Good* considerably differs from the other three classes. To reduce the bias in the classification phase, during the training of the quality function, we balanced the size of the classes in T_1 dataset by randomly downsampling the largest class [8]. Table II presents the two datasets with their related sizes. The dataset T_2 represents the remaining posts belonging to the classes described in Table I.

TABLE II. Datasets created for our study.

Dataset	T_1 (Training)	T_2 (Testing)
Very Good	5,000	71,592
Good	5,000	1,028,676
Bad	5,000	65,837
Very Bad	5,000	76,854
Total	20,000	1,242,959

B. Metrics Definition

We identified metrics that cover textual and non-textual features of Stack Overflow posts [9]. We report a short summary of the metrics listed in Table III.

Stack Overflow (M_{SO}) Metrics: Stack Overflow gave us a set of descriptions of simple textual metrics already in use.

Readability (M_R) Metrics: They capture other textual features and readability metrics. We included structural metrics, like words and sentences count, and the percentage of code. A lower readability could be a symptom of poor quality, therefore we include standardized readability metrics (*e.g., Automated Reading Index* [10]).

Popularity (M_P) Metrics: They concern the reputation[6] of the author of a question. We consider votes (*e.g.,* up and down votes), the total number of badges received, and a subset of specific questions badges (*e.g.,* Good and Stellar) and answer badges[7] (*i.e.,* Nice, Great, Good). These metrics are calculated at question *creation time*.

C. Classification Approach

The classification approach we proposed in our previous work is based on a linear quality function (QF) [9]. We combine all the metrics (m_i) described in Section III-B by assigning a weight (w_i) in the $[-1,1]$ interval. Metrics are normalized according to the minimum and maximum value calculated from the dump of September 2013, and range in the $[0,1]$ interval. The QF is learned to assign negative values for bad quality posts and positive values for good quality posts.

$$QF = \sum_{i=1}^{n} w_i \cdot m_i \quad w_i \in [-1,1] \quad m_i \in [0,1] \tag{1}$$

We constructed a different QF for each metric set devised in Section III-B, and we learned each one using genetic algorithms [11] implemented with the open source framework JGAP.[8] We trained the QF on the T_1 dataset, considering two levels of quality where *A* and *B* lie in the *Good* set, and *C* and *D* lie in the *Bad* set. We trained our genetic algorithm by using a population size of 64 individuals for 20 generations.

Figure 1 shows how we classify posts according to the distribution of a quality function. We consider the left and right tails based on specific quantiles of the quality function

[5] http://meta.stackexchange.com/questions/161390/

[6] http://meta.stackoverflow.com/questions/7237/how-does-reputation-work

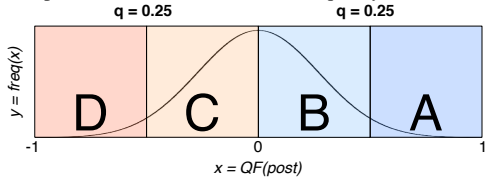
[7] http://stackoverflow.com/help/badges

[8] http://jgap.sf.net

TABLE III. STACK OVERFLOW (M_{SO}) METRICS, READABILITY (M_R) METRICS, AND POPULARITY (M_P) METRICS.

Metric	Description
Body Length	The length in characters of the question, including source code and HTML tagging.
Emails Count	The number of e-mail addresses found in the question.
Lowercase Percentage	The percentage of lowercase characters all over the question.
Spaces Count	The total number of spaces in the question.
Tags Count	The number of tags assigned to the question by the author.
Text Speak Count	The number of text speak (*e.g.,*'doesnt', 'wat', 'afaik', 'rotfl') found in the question.
Title Body Similarity	Textual similarity between title and body.
Title Length	The length in characters of the title of the question.
Capital Title	1 if the title begins with a capital letter, 0 otherwise.
Uppercase Percentage	The percentage of uppercase characters all over the question.
URLs Count	The number of URLs found in the question.
Average Term Entropy	Average entropy of terms in a question, according to the SO entropy index we devised. Each term's entropy is calculated on the SO dataset.
Automated Reading Index	$4.71 \cdot (\frac{characters}{words}) + 0.5 \cdot (\frac{words}{sentences}) - 21.43$
Coleman Liau Index	$0.588 \cdot L - 0.296 \cdot S - 15.8$ where L = average number of letters per 100 words, S = the average number of sentences per 100 words.
Flesch Kincaid Grade Level	$0.39 \cdot (\frac{total\ words}{total\ sentences}) + 11.8 \cdot (\frac{total\ syllables}{total\ words}) - 15.9$
Flesch Reading Ease Score	$206.835 - 1.015 \cdot (\frac{total\ words}{total\ sentences}) - 84.6 \cdot (\frac{total\ syllables}{total\ words})$
Gunning Fox Index	$0.4 \cdot [(\frac{words}{sentences}) + 100 \cdot (\frac{complex\ words}{words})]$
LOC Percentage	The percentage of lines of code declared between tags ¡code¿ all over the text of the question.
Metric Entropy	$(\frac{shannon\ entropy}{body\ lenght})$. It represents the randomness of the information in the question.
Sentences Count	Numer of sentences contained in the question, excluding ¡code¿ tags.
SMOG Grade	$1.0430 \cdot \sqrt{polysyllables \cdot (\frac{30}{sentences})} + 3.1291$
Words Count	The number of words in the questions, excluding <code>tags.
Accepted by Originator Votes	The number of accepted answer obtained by the user.
Approved Edit Suggestion	The number of accepted edit suggestions the user obtained.
Answer Badges Count	The number of badges obtained for answers (*e.g.,* Great Answer, Good Answer, Nice Answer).
Badges-Tags Coverage	The percentage of tags covered by the badges the user already possess.
Bounty Start Votes	The number of votes the user received for having started a bounty (*e.g.,* gift points for the answer she wants).
Bounty End Votes	The number of votes the user received for having ended a bounty.
Close Votes	The number of close votes the user received by the user for questions asked.
Deletion Votes	The number of deletion votes the user received for the questions asked.
Down Votes	The overall number of down votes the user received by the community.
Favorite Votes	The overall number of favorite votes the user received by the community.
Moderator Review Votes	The number of review votes the user received for her questions.
Offensive Votes	The overall number of votes the user received for contents considered offensive.
Reopen Votes	The number of close votes the user received for her already closed questions.
Question Badges Count	The number of badges obtained for questions (*e.g.,* Favorite Question, Stellar Question, Good Question).
Spam Votes	The overall number of votes the user received for contents considered spam.
Total Badges	The total number of badges the user obtained. It also includes badges for questions and answers.
Undeletion Votes	The number of undeletion votes the user received for her already deleted questions.
Up Votes	The overall number of up votes the user received by the community.

Fig. 1. Example of tails identification in the quality function distribution.



distribution. The left tail identifies posts with very bad quality (class D). The right tail identifies very high quality posts (class A). A given model may consider different (and more restrictive) quantiles to identify very bad or very good posts, and thus may be more or less precise to identify high/low quality posts.

IV. EVALUATING THE STACK OVERFLOW APPROACH

In the public data dump of Stack Overflow many tables and data are missing, *e.g.,* the data concerning review queues and the information concerning the quality score assigned to a post by the system are not disclosed. We collaborated with Stack Overflow to work on this private information. To evaluate the base performance of their approach we introduce two definitions of precision:

Hard Precision (HP): the percentage of posts in the review queue belonging to the class *D*.

Soft Precision (SP): the percentage of posts in the review queue belonging to the class *D* and *C*.

The rationale behind the hard precision is that the review queue should ideally contain low quality posts that need to be closed; the soft precision captures less problematic posts, which are low quality but do not need to be closed or deleted.

TABLE IV. REVIEW QUEUE (RQ) DISTRIBUTION FOR T_2

Class	RQ Size	Percentage
A	228	6.68%
B	991	29.01%
C	764	22.37%
D	1,433	41.94%
Total	3,416	100.00%

Table IV shows the number of posts classified as low quality by the Stack Overflow approach according to the dataset T_2 we constructed. The Stack Overflow approach has a hard precision of *41.9%*, indicating that over *58%* of posts in the review queue are of C-quality or better. If we consider the soft precision, the situation improves up to *64.31%*, where *35.69%* of the contents of the review queue are good posts. As we infer from Table IV, the Stack Overflow approach performs better when it comes to identifying high quality posts belonging to class *A*, while it fails when dealing with middle-high quality posts belonging to class *B*. With our approach we try to refine the review queue by removing high quality posts.

TABLE V. REVIEW QUEUE REDUCTION WITH OUR APPROACH.

Model	RQ Size	A	B	C	D	HP	SP	RQ Reduction	A Red.	B Red.	C Red.	D Red.
$RQ \setminus A(M_P, 0.25)$	3,108	166	799	735	1,408	45.30%	68.95%	9.02%	27.19%	19.37%	3.80%	1.74%
$RQ \setminus (A(M_P, 0.25) \cup A(M_{SO}, 0.05))$	2,650	107	559	664	1,320	49.81%	74.87%	22.42%	53.07%	43.59%	13.09%	7.89%
$RQ \setminus (A(M_R, 0.25) \cup A(M_P, 0.10))$	2,529	106	551	567	1,305	51.60%	74.02%	25.97%	53.51%	44.40%	25.79%	8.93%
$RQ \setminus A(M_P, 0.33)$	2,552	89	507	657	1,299	50.90%	76.65%	25.29%	60.96%	48.84%	14.01%	9.35%
$RQ \setminus A(M_R, 0.33)$	2,505	112	544	556	1,293	51.62%	73.81%	26.67%	50.88%	45.11%	27.23%	9.77%
$RQ \setminus A(M_R, 0.40)$	2,300	78	430	529	1,263	54.91%	77.91%	32.67%	65.79%	56.61%	30.76%	11.86%
$RQ \setminus A(M_P, 0.40)$	2,421	74	449	641	1,257	51.92%	78.40%	29.13%	67.54%	54.69%	16.10%	12.28%
$RQ \cap D(M_P, 0.40)$	2,244	64	393	600	1,187	52.90%	79.63%	34.31%	71.93%	60.34%	21.47%	17.17%
$RQ \cap (D(M_R, 0.40)$	1,912	33	251	468	1,160	60.67%	85.15%	44.03%	85.53%	74.67%	38.74%	19.05%

V. REFINING THE REVIEW QUEUE

We refine the review queue using the tails of the distribution. Table V reports a summary of the different approaches we followed to improve and reduce the low quality review queue.

We see two possibilities to refine the review queue using a quality function: We can remove very good posts (A class) or intersect the review queue with very bad quality posts (D class). Since we have three different set of metrics, we can also combine intersections and subtractions of tails for different QFs originated with different metric sets. Let $\{A, D\}(M_x, q)$ be a tail for class A or D, for a QF learned with the set of metrics M_x, and q be a quantile to identify the tails. We tried different models to refine the review queue and we compared the hard and soft precision we obtained against the one shown in Table IV. Table V shows the result for the best performing models. We can see that there is a tradeoff concerning the reduction of the review queue (RQ Reduction), the hard precision (HP), the soft precision (SP) and the percentage of D posts possibly removed.

For example, by removing A-class posts originated with the popularity metrics (M_P) QF on the 0.25 percentile, we are able to remove 27.19% of posts belonging to the class A, and *19.3%* of posts belonging to class B, while losing only 3.8% of class C posts and 1.74% of class D posts. We obtain a hard precision of 45.3% and a soft precision of 68.95%, with and effective review queue reduction of 9%. If we increase the percentile with the same approach (*e.g.,* 0.4 percentile), we can remove 65.79% class A post with a loss of 12.28% class D posts in the worst case, increasing the review queue reduction to 29% and the hard precision to 52%. Similar results can be obtained with readability metrics (M_R) where we are able to obtain the highest review queue reduction (44%) while considering the 0.4 quantile and intersecting D-class posts, with a loss of 19% of class D posts and a reduction of 85.5% of class A posts.

Lessons Learned. We can see that popularity metrics and readability metrics are the most useful metric sets to refine the low quality review queue. Readability metrics are effective alone and in combination with popularity metrics (Table V), thus connecting readability to good quality posts, and both metric sets complement the simple textual features currently in use at Stack Overflow. Thus, we recommend to take into account popularity and readability metrics in the quality assessment of posts. For example, while an automatic refinement system could be long-term goal, a simple recommendation system could be more effective in the short term to help reviewers to discard posts that score high in popularity and readability metrics, or review first posts that score particularly low.

VI. CONCLUSIONS

We discussed the review process of Stack Overflow, and how the low quality review queue plays an important role in identifying and filtering low quality contents. We have shown how the hybrid approach in use at Stack Overflow populates a review queue with a considerable amount of misclassified posts. We proposed an approach to identify misclassified posts in the review queue by analyzing the tail of a quality function created by means of genetic algorithms. We have shown that our approach is capable of reducing the low quality review queue by removing good posts (class A), with a minimal loss on real low quality posts (class D), thus saving time spent by the Stack Overflow moderators in reviewing posts not supposed to be reviewed.

Acknowledgments Ponzanelli and Lanza thank the Swiss National Science foundation for the financial support through SNF Project "ESSENTIALS", No. 153129.

REFERENCES

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

[2] C. Treude, "Programming in a socially networked world: the evolution of the social programmer," in *Proceedings of CSCW 2012 (15th ACM Conference on Computer Supported Cooperative Work and Social Computing)*. ACM, 2012, pp. 1–3.

[3] E. Agichtein, C. Castillo, D. Donato, A. Gionis, and G. Mishne, "Finding high-quality content in social media," in *Proceedings of WSDM 2008 (1st International Conference on Web Search and Data Mining)*. ACM, 2008, pp. 183–194.

[4] D. Correa and A. Sureka, "Chaff from the Wheat : Characterization and Modeling of Deleted Questions on Stack Overflow," in *Proceedings of WWW 2014 (23rd international conference on World Wide Web*. ACM, 2014.

[5] J. Jeon, W. B. Croft, J. H. Lee, and S. Park, "A framework to predict the quality of answers with non-textual features," in *Proceedings of SIGIR 2006 (29th Annual International ACM SIGIR Conference on Research & Development on Information Retrieval)*. ACM, 2006, pp. 228–235.

[6] K. Arai and A. N. Handayani, "Prediting Quality of Answer in Collaborative Q\A Community," *International Journal of Advanced Research in Artificial Intelligence*, vol. 2, no. 3, pp. 21–25, 2013.

[7] L. Adamic, J. Zhang, E. Bakshy, and M. Ackerman, "Knowledge sharing and yahoo answers: Everyone knows something," in *Proceedings of WWW 2008 (the 17th International Conference on World Wide Web)*. ACM, 2008, pp. 665–674.

[8] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 9, pp. 1263–1284, Sep. 2009.

[9] L. Ponzanelli, A. Mocci, A. Bacchelli, and M. Lanza, "Understanding and Classifying the Quality of Technical Forum Questions," in *In Proceedings of QSIC 2014 (14th International Conference on Quality Software)*. IEEE CS Press, 2014.

[10] E. Smith, R. Senter, and A. F. A. M. R. L. (U.S.), *Automated Readability Index*, ser. AMRL-TR-66-220. Aerospace Medical Research Laboratories, 1967.

[11] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, 1st ed. Addison-Wesley Longman Publishing Co., Inc., 1989.