Understanding and Classifying the Quality of Technical Forum Questions

Luca Ponzanelli[1], Andrea Mocci[1], Alberto Bacchelli[2], Michele Lanza[1]
1: REVEAL @ Faculty of Informatics – University of Lugano, Switzerland
2: Delft University of Technology, Netherlands

***Abstract*—Technical questions and answers (Q&A) services have become a valuable resource for developers. A prominent example of technical Q&A website is Stack Overflow (SO), which relies on a growing community of more than two millions of users who actively contribute by asking questions and providing answers. To maintain the value of this resource, poor quality questions—among the more than 6,000 asked daily—have to be filtered out. Currently, poor quality questions are manually identified and reviewed by selected users in SO; this costs considerable time and effort. Automating the process would save time and unload the review queue, improving the efficiency of SO as a resource for developers.**

We present an approach to automate the classification of questions according to their quality. We present an empirical study that investigates how to model and predict the quality of a question by considering as features both the contents 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). Our findings show that there is indeed the possibility of at least a partial automation of the costly SO review process.

I. Introduction

The advent of Q&A websites has changed the way people seek for information on the web. Q&A websites like Yahoo! Answers and Ask[1] have become prominent information venues.

People ask questions to the crowd of such communities to gather the information they need. The popularity of Q&A websites has also grown in software engineering. Stack Overflow[2] is a prominent example of a technical Q&A website, which allows developers to exchange knowledge about programming problems. Treude *et al.* [23] investigated the interaction of developers with Stack Overflow, and reported how this exchange of Q&A among developers is providing a valuable knowledge base that can be leveraged during software development. Stack Overflow discussions tackle topics, ranging from common programming and algorithmic problems to library-related API discussions, that can help in everyday developers' tasks.

Stack Overflow relies on a community that is steadily growing both in size and in the amount of the contents it provides. According to the SO data dump of September 2013[3] provided by the Stack Exchange network, Stack Overflow stores more that 5.5M questions, 10.2M answers and a community counting more than 2.3M users. The number of questions added each month has been steadily growing since the inception of Stack Overflow, as we see in Figure 1, and has reached peaks of more than 200,000 new questions per month.

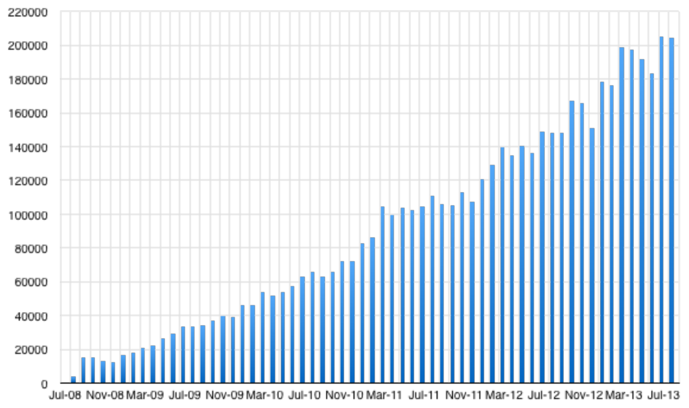

Fig. 1. Questions added monthly to Stack Overflow

The quality of the contents 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*" [1]. 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 of closing and deleting questions.

As reported by Correa *et al.* [6], 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 latency in the deletion process is a symptom of the amount of effort required by moderators to guarantee a satisfiable level of quality in Stack Overflow.

To solve this problem we propose an approach to automatize the filtering process. We have devised a quality predictor that helps moderators in identifying poor-quality questions at their creation time, thus reducing the review time. To do so, we have investigated the concept of quality for Stack Overflow questions and developed the classification approach.

Contributions. We make the following contributions:

- An empirical study where we mine the Stack Overflow data dump to understand the concept of bad and good questions and to build the related datasets.
- A set of metrics to quantify the concept of quality by considering both textual (*e.g.,* readability metrics) and

[1]See http://answers.yahoo.com and http://www.ask.com
[2]See http://stackoverflow.com
[3]http://www.clearbits.net/torrents/2155-sept-2013

non-textual features (*e.g.,* popularity of a user in the community), also showing which of those correlate with the quality.

- A classification approach to evaluate the quality of questions and to identify high quality and low quality questions at creation time.

Structure of the Paper. In Section II we survey related work. In Section III we discuss how we construct the datasets we use for our analysis. In Section IV we present the metrics that we use to construct our classifier. In Section V we then present our classifier and the results we obtain. In Section VI we discuss our findings and the threats to validity. We draw our conclusions in Section VII.

II. Related Work

Many studies analyzed how users in Q&A websites behave and interact among each others, with particular attention to the Stack Overflow community [23], [22]. Other studies focused on the contents of the discussions, and classified the topics and the trends on Stack Overflow. Barua *et al.* [4] presented an in-depth study of the topics discussed among developers by using Latent Semantic Indexing (LSI) [9] to cluster discussions. Allamanis *et al.* [2] constructed topic-models by means of Latent Dirichlet Allocation (LDA) [8] to associate concepts to questions.

A considerable amount of work has been done on prediction models. Stanley and Byrne [21] mined the Stack Overflow dataset and devised a Bayesian probabilistic model to predict hashtags, given a post. Kuo [15] compared different classification and topic-based model to perform word prediction, applying it on the Stack Overflow dataset to predict the tags of a post given its contents.

Pal *et al.* [18] analyzed users' behavior to early detect potential experts in Q&A websites by characterizing and estimating users' motivation and ability to provide help in the community. By means of machine learning approaches, they predicted the potential of a user given her first few weeks of activity in the community.

Many studies focused on predicting the quality of a post given both textual and non-textual features. Jeon *et al.* [14] devised a framework based on stochastic processes to predict the quality of answers in Naver.[4] They collected Q&A pairs from Naver, manually classified them in three categories (*i.e.,* bad, medium, good), and extracted 13 non-textual features to evaluate the quality.

Arai *et al.* [3] 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) provided by the Weka framework [24]. They took a snapshot of Yahoo! Answer for Indonesian people,[5] randomly selected and resolved 1,500 Q&A pairs, and extracted n-textual features, reaching a quality prediction with a precision close to 90% on the testing set. In our work we do not make use of annotators to create quality dataset of Q&A. Our approach is completely based on mining Stack Overflow data. We rely on the fact that the crowd defines the quality of question; this allows us to make use of a much larger dataset. Moreover, we analyze the information at disposal to infer different levels of quality.

Correa and Sureka [6] presented a large scale study on deleted questions on Stack Overflow where they show that the quality of questions decades in pyramidal fashion, and present an analysis of timing for closed and deleted questions. They developed a predictive model for deleted questions based on decision trees that uses 47 features, both non-textual and textual, extracted from the question's text and the author's information. Their model reaches a precision of 66% with 10-fold cross validation on their dataset.

We focus on Stack Overflow as Q&A website reference. We present a study aimed at detecting questions that can be excluded from the review queue of Stack Overflow reviewers. We do not only focus on deleted questions, but we focus on different levels of quality, and in particular on high quality questions. Indeed, the former would immediately be marked as bad, and removed from Stack Overflow, while the latter would be immediately excluded from the review queue. We also focus on both textual and non-textual features for a question, and we devised three distinct set of metrics concerning (i) readability metrics, (ii) author's popularity, and (iii) simple textual features in use at Stack Overflow. We analyze the pros and cons in using machine learning approaches like decision trees, and present an alternative approach based on a linear combination of features.

III. Dataset Construction

The September 2013 data dump contains 5,648,975 questions. Understanding the concept of quality for a question is very subjective if left to the judgment of a single person. On the other hand, in Stack Overflow low-quality question or a high-quality questions are defined by the crowd itself. We decided to rely on this information to identify quality. There are some actions the crowd can take to discriminate between bad and good posts: Every user can 'up vote' or 'down vote' a question or an answer, and moderators can vote for closing or deleting a question. Moreover, we can also consider information concerning the interaction with the question, *i.e.,* the answers. Indeed, a question with an accepted answer represents a discussion, or a posed problem, that has obtained the information needed. When an originator user (*i.e.,* the user who posed the question) accepts a specific answer, she is closing the discussion by pointing out the solution. Another aspect to take into account is the evolution of the question. Authors can modify their questions to clarify some parts, augment the information provided, and improve the overall quality. Modifying a question could have indirect side effects on the quality evaluation provided by the crowd. Therefore, we exclude from the dataset every question whose original body has been edited.

We also discard all questions whose score is 0. We assume that 0-scored questions have not attracted enough interests from the community, making it difficult to evaluate and classify their quality with the information at disposal.

After applying all the above mentioned filtering techniques we end up with a dataset of 1,262,959 questions, which we

[4]http://www.naver.com

[5]http://id.answer.yahoo.com

subdivide into high quality and low quality categories according to the following definitions:

- **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.

With only two categories a clear quality distinction between questions is not available: The variance of the quality among posts in Q&A websites is considerable [1], which in turn leads to very noisy data. For this reason, we want to further refine each quality class by identifying 'very good' and 'very bad' questions.

'Very bad' questions are low quality questions that have been closed or deleted in their final state, without considering the score they obtained. We do not consider as 'very bad' the ones that have been closed because they were duplicates of existing questions, since the closing was not due to quality-related issues (indeed, a duplicate can be a clone of a very good question). We obtain a set of 81,854 'very bad' questions.

To define a set of 'very good' questions one is naturally drawn to selecting those with very high scores. This raises the question about which score threshold one should pick. We picked as score threshold of 7, which generates a set of 'very good' questions of roughly the same size as the set of 'very bad' ones. This leads to a set of 76,592 'very good' questions.

Table I reports the distribution of the quality classes 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

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

A. Creating Datasets for Training and Testing

For the purpose of our study, we created four 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, we balanced the size of the classes in each dataset by randomly downsampling the largest class [13]. Table II presents the four datasets with their related sizes.

Dataset	T_1 (Training)	T_2 (Testing)	T_3 (Training)	T_4 (Testing)
Very Good	10,000	66,592	5,000	65,837
Good	0	0	5,000	65,837
Bad	0	0	5,000	65,837
Very Bad	10,000	66,592	5,000	65,837
Total	20,000	133,184	20,000	263,348

TABLE II. Datasets created for our study.

We created pairs of datasets with training and testing purposes respectively (see Section V). Each dataset in a pair is not interleaved with the other. The first pair, T_1 and T_2, excludes intermediate quality class, thus referring to our first rough definition of quality. The second pair , T_3 and T_4, refers to the extended definition of quality for Stack Overflow questions, thus including all four classes.

IV. Metrics Definition

We identified three sets of metrics that cover textual and non-textual features of Stack Overflow posts. All the reported metrics are calculated by considering the data available (*e.g.,* author's information) at post creation time. All metrics range between 0 and 1, being normalized according to their minimum and maximum value over all the dataset (Table I).

Stack Overflow (M_{SO}) Metrics (Table III): The staff of Stack Overflow provided us with a set of simple textual metrics currently in use. With such metrics Stack Overflow identifies the poor quality questions to be manually reviewed.

Most of the metrics are mainly character-based (*e.g., Title Length*, *Title With Capital Letter*, *Body Length*, and *Lowercase Percentage*); exceptions are *Emails Count*, *URLs Count* and *Tags Count* as they identify emails, urls, and the amount of tags respectively. Stack Overflow also checks for text speak (*e.g.,* 'wats', 'doesnt', 'afaik') and emoticons as additional symptoms of poor quality posts.

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.

TABLE III. Stack Overflow (M_{SO}) Metrics

Readability (M_R) Metrics (Table IV): We complement the Stack Overflow metrics with metrics that capture other textual features regarding readability. Focusing on the structure itself, we include in our analysis features like *Words Count* and *Sentences Count*. Another aspect characterizing a Stack Overflow question is the presence of code. Using the HTML structure of the posts, we identify the text within tags *¡code¿* to calculate the percentage of lines of code (*LOC Percentage*) in the full question's body.

We introduce *Metric Entropy* and *Average Terms Entropy* as features to evaluate the terms used in the textual part of a question. *Metric Entropy* is the Shannon entropy [7] divided by the length of the text, and represents the randomness of the information contained in the message. *Average Terms Entropy* measures the entropy of each term used in the question's text, against all the posts in Stack Overflow. We calculate the entropy

Metric	Description
Average Term Entropy	The average entropy of terms in a question, according to the Stack Overflow entropy index we devised. Each term's entropy is calculated on the Stack Overflow 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.

TABLE IV. Readability (M_R) Metrics

Metric	Description
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.

TABLE V. Popularity (M_P) Metrics

for each term in the Stack Overflow data dump of September 2013 and we calculate the average of the entropy of each term used in the question's text. As we did in our previous work [19], the entropy value describes the discriminating power of a word, therefore the lower the average of terms' entropy, the higher the use of uncommon terms.

To assess the question readability, we also compute six standardized readability indexes: *Automated Reading Index* [20], *Flesch Kincaid Grade Level* [10], *Coleman Liau Index* [5], *SMOG Grade* [17], *Gunning Fox Index* [12], and *Flesch Reading Ease Score* [10]. These represent the comprehension difficulty when reading a passage in English and are different approximations and representations of the U.S. grade level[6] needed to comprehend the text. We argue that a lower readability could be a symptom of a poor quality question. To calculate these indexes we first remove code snippets from the question's body. We use the Stanford NLP Parser[7] to extract sentences and words, and TeX hyphenation [16] to obtain syllables.

Popularity (M_P) Metrics (Table V): We also devise non-textual features to model the author writing the question. Analogously to M_{SO} and M_R metric sets, we require a snapshot of the status of authors when they created the question. The official data dump only reports the latest users' reputation levels (*i.e.,* computed in September 2013), thus we estimate Stack Overflow users reputation[8] by considering votes and badges received. The data dump provides all the votes a user received and the date when they were given. Representing the snapshot of the author's reputation at question-creation time requires to filter out votes and badges received after the question creation date. We consider three metrics: *Badged Answer Count*, *Badges Question Count*, and *Badges-Tags Coverage*. The first two represent the badges received concerning question (*e.g.,* 'Favorite Question' and 'Stellar Question') and answers (*e.g.,* 'Great Answer' and 'Good Answer'), while the latter refers to the coverage of author's badges with respect to the tags assigned by the author to the new question.

V. Data Analysis

In the previous sections, we discussed how to define the quality of a question in SO and we identified the features of a question and its author likely correlate with quality. In this section, we investigate such correlations through two empirical studies:

1) In Section V-A, we use machine learning, and in particular decision trees, to classify a question's quality using different combinations of metrics.
2) In Section V-B, we adopt a simpler approach that uses genetic algorithms to train a linear function expressing a measure of a question's quality, and then we investigate how such a measure can be used to perform more precise predictions on a question's quality.

A. Classification with Decision Trees

The first experiment we conducted involves the use of a simple machine learning algorithm to classify the quality of a question as bad or good, as defined in Section III.

[6]http://en.wikipedia.org/wiki/Grade_levels
[7]http://nlp.stanford.edu/software/index.shtml
[8]http://meta.stackoverflow.com/questions/7237/how-does-reputation-work

Considering the objective of our research, we want not only to predict the quality of a new question, but also to understand which classes of metrics influence the quality of a question. For this reason, we chose *Decision Trees* [24], a machine learning algorithm whose output can be easily interpreted. We conducted the experiments by considering all the different combinations of metric sets, to understand which ones give better precision in terms of identifying the quality of a submitted question.

A decision tree is a tree in which each internal node (non-leaf) is labeled with a feature, and arcs from any internal node are labeled with exclusive predicates that summarize possible distinct values for the feature. Finally, each leaf of the tree is labeled with a class. In Figure 2 we show a portion of a decision tree trained on T_4 using popularity metrics (M_P) as an example.

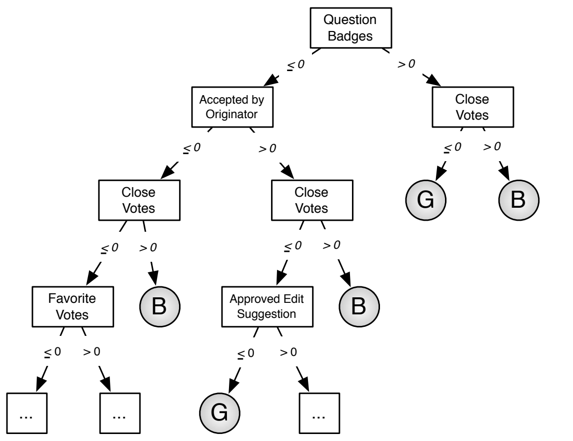

Fig. 2. Portion of a Decision Tree trained on T_4 and M_P

As we can see, in our case, the features are question metrics, and the class represents the quality to be assigned to the posts exhibiting the conjunction of predicates represented by arcs connecting the root to the leaf.

We trained decision trees on the largest datasets T_2 and T_4, considering a minimal amount of 50 posts per leaf and 0.25 confidence value. We performed 10-fold cross validation.

Results. Table VI shows the results of the first experiment on the two data sets (T_2 and T_4) that correspond to the different definitions of quality we identified (see Section III).

Dataset	Metrics	P_g	P_b	Average ROC
T_2	M_{SO}	62.9%	62.1%	0.667
	M_R	66.8%	62.1%	0.676
	M_P	74.3%	73.2%	0.795
	$M_{SO} \cup M_R$	66.3%	64.1%	0.702
	$M_{SO} \cup M_P$	74.0%	74.5%	0.808
	$M_R \cup M_P$	76.0%	75.1%	0.824
	$M_{SO} \cup M_R \cup M_P$	76.2%	75.2%	0.829
T_4	M_{SO}	61.2%	61.7%	0.655
	M_R	62.9%	61.1%	0.653
	M_P	73.0%	69.7%	0.734
	$M_{SO} \cup M_R$	63.3%	62.8%	0.676
	$M_{SO} \cup M_P$	72.7%	71.6%	0.780
	$M_R \cup M_P$	73.3%	71.9%	0.788
	$M_{SO} \cup M_R \cup M_P$	73.2%	72.0%	0.789

TABLE VI. Classification Results using Decision Trees

In both cases, considering a single set of metrics, popularity metrics (M_P) give the best results in terms of precision to identify both good and bad posts. The set of metrics considered by SO (M_{SO}) perform worse than readability metrics, and may also introduce noise in the classification: When combined with popularity metrics or readability metrics, it may actually decrease prediction precision. The combination of all the three sets of metrics does not significantly increase prediction performance compared to popularity metrics alone. This reveals that *the popularity of the author is more important than textual features to determine the quality of a new question*.

We interpret this fact as follows. First, a question's quality, in terms of how the crowd will react to it, is inherently related more to the semantics and intention of a question than the textual way it is formulated. Second, the history of a users' interaction with the community, which determined their actual reputation, will determine the quality of questions that they will ask in the future or, at least, will provide some bias towards this author by the community (in particular if we consider that moderators are elected by the crowd). This insight makes the interaction between authors and the community a notable component for predictions about a question's quality.

Java Subset. The overall level of precision reached by decision trees is relatively unsatisfactory for an automated process aimed at discarding bad questions, and in particular in the ideal dataset T_4. Given these results, we considered that questions about programming languages could be less noisy and better classifiable in terms of quality. We constructed a subset of T_4 containing only questions about the most popular programming language, Java, and we ran again the decision tree learning algorithm. Table VII shows the classification results for this subset. The results are only moderately better, leading us to the conclusion that it is not so much about what is being discussed, but by whom things are being discussed.

Dataset	Metrics	P_G	P_B	Average ROC
T_{Java}	M_{SO}	62.1%	61.7%	0.662
	M_R	63.8%	62.9%	0.672
	M_P	76.3%	77.0%	0.805
	$M_{SO} \cup M_R$	64.6%	64.3%	0.697
	$M_{SO} \cup M_P$	75.3%	75.5%	0.817
	$M_R \cup M_P$	76.9%	75.9%	0.818
	$M_{SO} \cup M_R \cup M_P$	76.3%	76.4%	0.823

TABLE VII. Classification Results using Decision Trees only on Java questions

Leaf Inspection. The inspection of the learned decision trees gives other important and detailed insights on which metrics influence the most the quality of questions, and suggests a different way to approach the problem of quality prediction. Each leaf on the learned decision tree is linked to a particular decision on the classification of a question's quality, *i.e.,* either good or bad. When a decision tree is trained and tested against a given data set, the learning algorithm also outputs the amount of data associated with a specific leaf, and the number of misclassified elements. Even if the overall precision of the decision tree is low, some leaves may exhibit a precision value that is particularly high, thus disclosing metrics more related to a particular quality class.

Table VIII shows examples of leaves on the decision trees that can correctly predict good or bad quality posts on subsets of data larger than 1% of the original dataset, having a precision greater than 75%. Nevertheless, the leaves that classify posts in such way are quite uncommon. Popularity metrics provide

Dataset	Decision Tree Leaf	Metric Set	Size	Perc.	Class	Precision
T_2	$QuestionBadges = 0 \ \wedge \ CloseVotes > 0$	M_P	13,033	9.9%	D	85.9%
	$QuestionBadges > 0 \ \wedge \ CloseVotes = 0$	M_P	17,480	13.2%	A	85.2%
	$QuestionBadges = 0 \ \wedge itCloseVotes = 0 \ \wedge \ AcceptedByOriginatorVotes > 0 \ \wedge \ ApprovedEditSuggestion = 0$	M_P	28,091	21.1%	A	75.4%
	$QuestionBadges = 0 \ \wedge CloseVotes = 0 \ \wedge \ AcceptedByOriginatorVotes > 0 \ \wedge \ ApprovedEditSuggestion = 0 \ \wedge LOCPercentage > 0.017705$	$M_P \cup M_R$	12,538	9.4%	A	88.7%
T_4	$QuestionBadges = 0 \ \wedge \ CloseVotes > 0$	M_P	14,258	5.4%	C+D	81.2%
	$QuestionBadges > 0 \ \wedge \ CloseVotes = 0$	M_P	26,426	10.0%	A+B	83.9%
	$QuestionBadges = 0 \ \wedge CloseVotes = 0 \ \wedge \ AcceptedByOriginatorVotes > 0 \ \wedge \ ApprovedEditSuggestion = 0$	M_P	46,655	17.7%	A+B	76.7%
	$QuestionBadges = 0 \ \wedge \ AcceptedByOriginatorVotes = 0 \ \wedge \ ClosedVotes = 0 \ \wedge \ FavoriteVotes = 0 \ \wedge wordsCount \leq 64 \ \wedge \ totalBadges \leq 1$	$M_{SO} \cup M_R \cup M_P$	33,879	12.9%	C+D	78.2%

TABLE VIII. Relevant Leaves on Learned Decision Trees

very good predictive performance even in this case, and we can mine the historical characteristics of users that influence future questions quality.

Consider, for example, the decision tree trained on dataset T_4 using popularity metrics. The precision P_B to predict bad posts is as low as 73%. If we inspect the decision tree, we can see that on a subset of the data made of $14,258$ questions (corresponding to 5.4% of T_4), the decision tree can correctly predict that 81.2% of them is bad with only 770 misclassified questions. This subset corresponds to a specific leaf of the decision tree, predicting bad quality questions candidates if a user has received no badges for questions, and has received a certain number of question closure votes. We also learn that if a user has instead received question badges but no closure votes, then her question will likely be of good quality, with a precision of 83.9%.

From the same decision tree, we also discover a very interesting leaf concerning bad quality posts. When a user obtained no question badges, neither close votes nor favorite votes, has no accepted answer in her history, possesses one badge at most, and asks a question of less than 64 words, it is likely to be a bad quality question with a precision of 78% on the 12.9% of the overall data in T_4. This finding remarks how the interaction of the user with the community matters and influences questions' quality estimation by the crowd. Indeed, some examples of users matching this leaf would be newcomers who have never interacted with the community, or people who provided neither notable questions (*i.e.,* no favorite votes, no question badges) nor accepted answers (*i.e.,* no accepted by originator), thus interacting not successfully with the community.

B. Linear Quality Function Classification

Given the limitations of the predictive performance of decision trees, and the fact that the analysis of leaves led to limited insights about what distinguishes good from bad questions, we decided to adopt a different approach for the classification of question quality, based on linear quality functions.

Intuitively, a quality function assigns a value to a post based on a given set of metrics. A quality function should assign a negative value to bad posts and a positive value to good posts. One of the benefits of such an approach to classification is that the predicted quality is not binary, but has a range and can therefore express intermediate levels of quality. To learn a quality function for a given metric set, we used genetic algorithms. A genetic algorithm [11] is a search algorithm inspired by the process of natural selection; we exploit such a search approach to find a set of coefficients of a quality linear function given a metric set and a training dataset.

In a genetic algorithm, possible candidate solutions (*individuals*) are evolved towards better solutions that tend to maximize a given *fitness function*. A candidate solution (*i.e.,* a *gene*) is composed of a set of properties (*i.e.,* its *chromosomes*) that are mutated and altered during the process of *evolution*. Evolution starts from a set of randomly generated individuals, and proceeds by modifying a *generation* of individuals through subsequent iterations. At the beginning of each iterative process, the fitness of individuals of a generation is evaluated. Usually, the fittest individuals are selected from the population, and randomly mutated or recombined to form a new generation, *i.e.,* to produce a new set of individuals for the next iteration. The algorithm stops when a pre-defined value of fitness for the best individual is found, or when a maximum number of generations has been produced. Overall, a genetic algorithm requires a definition of individuals through their chromosomes and a fitness function. Since we want to search for a linear quality function, we implemented the evolutionary search as follows:

- The chromosome of an individual is a set of coefficients, one for each metric in the considered metric set, ranging in the $[-1, +1]$ interval.
- The fitness function is determined from the number of posts in the training set that are correctly classified, *i.e.,* the posts for which the quality function outputs a negative value for a bad post and vice-versa. In other words, the fitness function is the classification precision on a given training set.

We implemented the evolutionary search by using an open source framework called JGAP.[9]

The fitness function evaluation is relatively costly, and depends on the size of the training set. Since each individual must be checked against the whole training set at each generation, it is impossible to search for quality functions using T_2 and T_4 as datasets, since they are too big. For these reasons, we used the smaller datasets T_1 and T_3 to train quality functions. We trained the genetic algorithm by using a population size of 64 individuals for 20 generations, and we constructed a quality function for each distinct set of metrics.

Results. Table IX summarizes the classification results for

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

Metrics	Top-3 $Features_G$	w_G	Top-3 $Features_B$	w_B	Neutral	w_N
Trained on T_1, tested on T_2						
M_{SO}	Tags Count	0.75	Text Speak Count	-0.99	Uppercase Percentage	-0.17
	Title Length	0.72	Body Length	-0.93	Title-Body Similarity	-0.15
	Spaces Count	0.39	Lowercase Percentage	-0.82	URLs Count	0.11
M_R	LOC Percentage	0.92	Gunning Fox Index	-0.87	Automated Reading Index	-0.19
	Coleman-Liau Index	0.88	Flesch Reading Ease Score	-0.54	Flesch Kincaid Grade Level	0.44
	Words Count	0.83	Sentences Count	-0.49	SMOG Index	0.45
M_P	Favorite Votes	0.83	Approved Edit Suggestion	-0.91	Total Badges	-0.02
	Down Votes	0.74	Moderator Review Votes	-0.90	Bounty Start Votes	0.02
	Accepted By Originator Votes	0.50	Spam Votes	-0.84	Badges-Tags Coverage	0.03
Trained on T_3, tested on T_4						
M_{SO}	URLs Count	0.93	Text Speak Count	-0.98	Title With Capital Letter	0.17
	Body Length	0.90	Uppercase Percentage	-0.67	Emails Count	0.32
	Title Length	0.84	Title-Body Similarity	-0.46	Tags Count	0.36
M_R	LOC Percentage	0.98	Coleman Liau Index	-0.74	Flesch Kincaid Grade Level	-0.09
	Average Term Entropy	0.33	Sentences Count	-0.69	Flesch Reading Ease Score	-0.07
	Automated Reading Index	0.33	Gunning Fox Index	-0.61	SMOG Index	0.14
M_P	Accepted By Originator Votes	0.98	Favorite Votes	-0.22	Total Badges	-0.02
	Offensive Votes	0.97	Approved Edit Suggestion	-0.15	Up Votes	-0.02
	Down Votes	0.93	Moderator Review Votes	-0.06	Close Votes	0.00

TABLE X. QUALITY FUNCTIONS METRIC WEIGHTS

Metrics	Quantile	Left Tail	P_B	Right Tail	P_G
Trained on T_1, Tested on T_2					
M_{SO}	0.25	34,718	62.0%	34,106	58.3%
	0.10	14615	67.2%	14,466	58.2%
	0.05	7,341	69.5%	7,288	60.0%
	0.01	1,364	77.0%	1,740	57.4%
M_R	0.25	30,912	64.2%	39,528	61.9%
	0.10	11,906	64.2%	16,270	49.0%
	0.05	5,896	64.7%	8,091	39.1%
	0.01	1,237	66.9%	1,625	30.2%
M_P	0.25	46,016	68.4%	25,841	85%
	0.10	17,542	74.1%	11,474	88.3%
	0.05	8,495	78.2%	6,931	89.5%
	0.01	1,718	81.0%	2,251	90.1%
Trained on T_3, Tested on T_4					
M_{SO}	0.25	63,944	61.9%	66,888	59.2%
	0.10	25,114	66.6%	26,081	60.8%
	0.05	11,984	69.0%	12,781	60.4%
	0.01	2,291	73.8%	2,487	58.6%
M_R	0.25	61,630	62.8%	69,679	50.3%
	0.10	23229	63.5%	28,381	40.0%
	0.05	11,696	63.1%	14,421	34.9%
	0.01	2,338	61.7%	2,772	30.52%
M_P	0.25	63,152	64.7%	69,542	68.9%
	0.10	21,987	70.4%	24,350	70.9%
	0.05	10,480	71.3%	11,054	73.3%
	0.01	1,787	71.3%	1,661	90.8%

TABLE IX. CLASSIFICATION RESULTS USING QUALITY FUNCTIONS

quality functions. After training the quality functions on T_1 and T_3, we tested their predictive performance on T_2 and T_4, respectively. With quality functions, we can easily identify questions with very high or very low predicted quality. We consider the distribution of qualities as evaluated on the training set as reference, and we calculate 4 different quantile values, of decreasing size, corresponding to the left and right tail of the distribution. Then we project the quantile values on the testing set and we consider the projected left and right tails, on which we calculate corresponding precisions, respectively P_B and P_G on Table IX. Even in the case of quality functions, popularity metrics exhibit the highest precision on the testing sets. However, on the noisy dataset, and considering the smallest quantile size, the metrics on use at SO could predict bad posts with a slightly higher precision compared to popularity metrics (73.8% vs 71.3%).

To get a rough overview of how the classification function behaves over the quantiles we have depicted in Figure 3 the percentage of bad (orange) vs. good classified questions (blue).

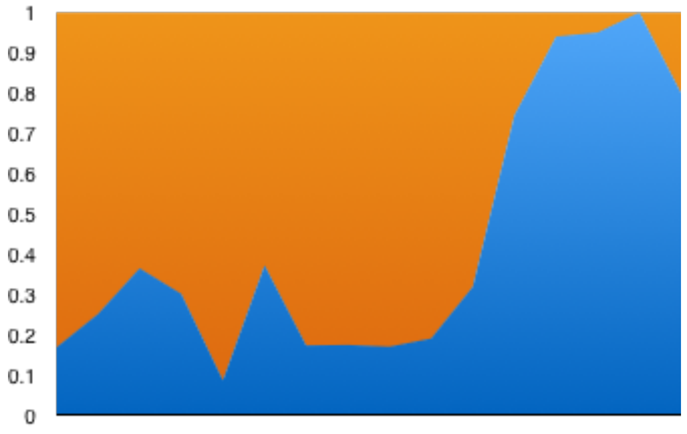

Fig. 3. Overall classification trend over quantiles, trained on T_1 and tested on T_2 using M_P.

On the left side the questions classified as bad are predominant, while on the right side there is a steep increase of questions classified as good. This trend reflects the data reported in Table X, where for M_P we obtain a precision of 68.4% on the left 0.25 quantile (bad tail), while we have a precision of 85% on the right 0.25 quantile (good tail).

Learned Quality Function Inspection. The structure of learned quality function reveals important insights about the metrics to determine good or bad quality of posts. Table X shows, for each learned quality function on a given training set, the role of each metric. In particular:

- Each coefficient with a strong positive value, close to 1, contributes to increase a question's quality.
- Each coefficient with a strong negative value, close to -1, negatively contributes to a question's quality.
- Each coefficient with a value close to 0 essentially does not contribute to determine quality.

Although the generalizability of the results can be questioned, the following findings emerge:

- For both datasets, *the number of down votes received by a user is a strong component of quality*. Essentially, if users have received a significant amount of down votes, they will be more likely to formulate good quality questions to improve their reputation.

- Another strong component of high quality is having answers that have been accepted by the originator user. In other words, having produced very good answers in the past has impact on producing good questions in the future.
- On the contrary, *up votes received on the past do not influence quality*. This is counterintuitive; intuitively, it means that the fact that users performed well in terms of questions that the crowd appreciated does not have an impact on the quality of their future questions;
- A good *tagging of code elements* in a question determines high quality. We expected this, for a Q&A website like SO.
- *Text speak determines bad quality*. Moreover, a low number of sentences in the question negatively influences quality.

The following characteristics influence one of the quality functions in the two data sets and become irrelevant on the other one:

- The number of urls in a post seem to be related to high quality ($w_G = 0.93$) post in the noisy data set, but the contribution is only minimal ($w_N = 0.11$) on the ideal dataset T_2.

A few metrics are related in completely opposite way to a post's quality if we consider the more noisy dataset T_4 instead of T_2 where the quality classes are clearly separated. In particular:

- Favorite votes received by a user seem to be a strong component to determine high quality posts ($w_G = 0.83$), while instead it is a relatively strong component for bad quality ($w_B = -0.22$) in the noisy dataset. This means that users that received favorite votes are somehow prone to produce high quality questions but also a great number of questions that are not to be deleted, but receive, on average, negative scores.
- Word count seems to characterize very good posts ($w_G = 0.83$), but when intermediate quality questions are added in the data set, a high number of words determines bad quality, even if not strongly ($w_B = -0.44$, value not shown in Table X).
- On the opposite side, body length relates to very bad quality ($w_B = -0.93$) on T_2, but with strong quality ($w_G = 0.90$) on the noisy dataset T_4.

Interaction between Metric Sets. Each quality function, associated with a given metric set, shows a relatively good predictive performance, which varies considering the dimension of the quantile to identify individuals with very low or very high quality. We manually inspected the smaller quantiles for each metrics set and we noted that each set contained questions with different features that would classify them as good or bad, as expected. In other words, each metric set captures different characteristics of a question quality and of the user who posted it, and it is reasonable to expect that we can achieve better precision by identifying questions who have very good or very bad values for quality functions of more than one set of metrics. A possible approach to investigate would be to train genetic algorithms with bigger chromosomes (corresponding to larger sets of metrics); however, this would be relatively expensive, and might introduce classification noise. While such a method might be worth investigating, in the scope of this paper, we prefer to try a simpler, and hopefully more effective approach, to combine predictions of quality functions.

We considered a larger set of quantile sizes with respect to Table IX, and we studied the prediction precision of intersections of such quantiles, which correspond to posts which show very good or very bad quality as predicted by more than one quality function associated to a metric set. We obtained a large set of possible predictive models based on such intersections, which are summarized in Table XI.

Each combination of quantile sizes identifies a different set of questions, and with decreasing size of such set one can achieve better precision. It is an expected trade-off between an identified set of questions and the precision to be obtained. In Table XI we present models with top precision in three different range sizes, from around 1% to 20% of original testing set. We can achieve very high precisions on the intersections of tails. On separated dataset T_2, we can reach precisions as high as 97.4% to identify good questions, and as high as 89.2% to identify bad questions. On the noisy dataset T_4, precision reaches values around 80% for both good and bad questions on smaller portions of the testing set.

Class	Size Range	Quantile Intersection M_{SO}	M_R	M_P	Size	P
Trained on T_1, Tested on T_2						
D	10-20%	0.5	0.5	0.5	22063	80.4%
		–	0.5	0.2	16808	81.0%
	5-10%	–	0.25	0.2	8532	81.5%
		0.25	0.5	0.5	12303	83.5%
	1-5%	0.1	0.5	0.5	5447	85.7%
		0.05	0.2	0.5	1341	89.2%
A	10-20%	–	0.5	0.25	15759	91.3%
		–	0.25	0.5	16729	88.9%
	5-10%	–	0.25	0.25	8293	95.8%
		–	0.25	0.2	6752	96.1%
	1-5%	–	0.25	0.1	3954	96.7%
		0.5	0.25	0.05	1544	97.4%
Trained on T_3, Tested on T_4						
C+D	10-20%	0.5	0.5	0.5	26987	76.4%
		0.5	0.25	0.5	43006	74.3%
	5-10%	0.2	0.25	0.5	14695	78.4%
		0.25	0.5	0.5	24256	76.9%
	1-5%	0.05	0.2	0.5	3965	81.4%
		0.1	0.2	0.5	7255	80.0%
A+B	10-20%	–	0.5	0.2	31694	74.4%
		–	0.5	0.25	40060	74.3%
	5-10%	0.25	0.5	0.25	14286	79.2%
		0.5	0.5	0.25	25712	77.6%
	1-5%	0.25	0.5	0.05	2916	83.2%
		0.5	0.25	0.05	4948	81.6%

TABLE XI. QUANTILE INTERSECTION MODELS

VI. DISCUSSION

We defined more than 40 metrics that capture different aspects of a question at its initial stage. We classified these metrics in three distinct sets concerning (i) readability metrics, (ii) author's popularity, and (iii) simple textual features in use at Stack Overflow.

Decision Trees. Initially, we adopted decision trees, a machine learning approach whose output can be easily interpreted, and we considered two possible datasets: an *ideal* one, where we selected only posts with very low or very high quality, and a *noisy* one. Results of classification with decision trees exhibited poor predictive power: slightly better than a coin-flip. Inspection of decision tree leaves gave us preliminary insights on which metrics influence the quality of significant sets of questions. Overall, the author's popularity metrics better discriminated bad and good posts than the other two sets of metrics, reaching a precision above 70%. According to the public data dump of September 2013, about 6,000 questions are asked every day. A precision of 70% would lead to many misclassified questions to be reviewed that do not need to be closed or deleted from Stack Overflow.

Quality Functions. We considered a linear quality function-based model for each set of metrics and we trained the weights by means of a genetic algorithm. The linear model was built to classify poor quality questions with negative values, and high quality questions with positive values. The models obtained could not correctly classify bad and good questions in general, thus we measured the precision of the classification for questions residing in the tails, that is, the more positive and the more negative value ranges. To this end, we trained our linear model on a subset of 20,000 questions (with balanced bad and good posts) and we tested it against a balanced testing set. To verify the precision of the classification of questions lying in the tails, we adopted two approaches:

1) We took every metric set and we verified the precision in classifying the elements in the tails, by choosing different sizes for each of them (*i.e.,* 1%, 5%, 10%, and 25%). In the ideal data set, the model trained on popularity metrics was able to correctly classify from 85% questions out of 25,841 good posts lying on a bigger tail and up to 90.1% questions out of a tail containing 2,251 posts. On the other hand, the same model correctly classified from 68.4% bad questions out of a tail of 46,016 questions and up to 81.0% bad questions out of a tail of 1,718 questions. Results on the noisy dataset were worse but proportionally similar. While metrics in use at Stack Overflow performed badly, readability metrics seemed to slightly better classify bad questions with a precision of 73.8% on a set of 2,291 posts.
2) We inspected the coefficients of quality functions and obtained insights about which metrics influence good and bad qualities of posts. For example, we found that users who received down votes in the past are more prone to post high quality questions in the future, probably to raise their reputation.
3) By looking at the intersection of the posts classified by each metrics set, we noticed that each set seems to identify different types of bad and good posts. For this reason, we verified the precision of the intersection of these models by varying and mixing the tail sizes (*e.g.,* 50% popularity metrics, 10% readability metrics, 10% Stack Overflow metrics). With this approach, the best model was able to correctly classify 96.2% out of 2,464 good posts and 74% out of 2,230 bad posts. We also identified a model that improves the classification of bad posts up to 85% out of 1,194 questions.

An important insight we derived from the classification models above is that author's popularity metrics are the most effective feature in deciding if a post is of a good quality or not. If we consider that reviewers are users selected inside the community, then having prominence on the popularity of an author matches the organizational behavior of the community itself. Last but not least, the prediction power given by author's popularity can be complemented by taking into account structural properties of the posts given by the other two sets of metrics.

A. Threats to Validity

Construct Validity Threats to construct validity are concerned with whether what one measures is what one intends to measure. In our case, there could be several reasons why the considered quality of the questions is incorrect. We rely on the judgement of the SO crowd to differentiate the quality of questions, which is a potentially error-prone process. In fact, the perceived quality might be different from person to person, and might be based on different definitions. This issue is alleviated by the fact that we also manually examined more than 100 questions not only to get insights on their features, but also to verify whether the choices made by the users were reasonable. Moreover, Stack Overflow relies on the same criteria.

Another issue regarding quality is the definition of the class A in our dataset (Table I): 'very good' questions. We defined 'very good' questions as those with a Stack Overflow score higher than 7. We chose this threshold to obtain balanced datasets, and as a reasonable trade-off between choosing a too high value (which would have only included questions regarding trending topics) and a very low one (which would have included questions only inspected by few users).

Finally, the choice of balanced dataset could have impacted the results of the machine learning models. Nevertheless, this should not be a problem when enough training data is available.

Statistical Conclusion threats concern the fact that the data is enough to support claims. We considered statistically significant samples in our experiments; this was possible because we relied on the crowd assessment and not on manual inspections of questions.

External Validity threats are concerned with the generalizability of results. The approaches we tried may show different results when applied to a Q&A website other than Stack Overflow. To alleviate this issue, we chose to include questions related to any valid topic in the technical forum, thus including a very large population. An evaluation of our approach that involves other Q&A websites could measure the effect of this threat.

VII. Conclusion

Understanding and classifying question quality is essential to maintain a good user experience of Q&A services. In particular, it is fundamental to filter out poor quality questions that may hinder the value of an important resource like a Q&A service. In fact, an automatic classifier of question quality could improve, and in some cases even replace, manual review processes like the ones nowadays implemented in Stack Overflow.

We devised, implemented and illustrated an approach to classify question quality, and in the same way understand what fundamentally influences and characterizes it. We devised three sets of metrics that capture both textual features of a question and the reputation of the user who asked it. Together with these metrics, we constructed two datasets from a Stack Overflow data dump which captured what the community identifies as a question's quality. We began by devising two types of quality for a question in Stack Overflow (*i.e.,* 'Good Quality', 'Bad Quality') and we then extended our definition to four level of quality (*i.e.,* 'Very Good', 'Good', 'Bad', 'Very Bad'), by imposing some empirical thresholds, based on data balancing.

We also conducted a twofold empirical study aimed at classifying Stack Overflow questions' quality and understand how the metrics we devised influence it. In the first part, we used a machine learning algorithm to infer decision trees. While the predictive power of such artifacts was relatively poor, we discovered that metrics expressing author's popularity are best predictors of a question's quality. In the second part, we used genetic algorithms to learn linear quality functions that describe a question's quality. We encoded quality as a function classifying bad quality questions with negative values, and high quality questions with positive values, thus representing different shades of quality where the extremes represent very bad and very good levels. By analyzing the tails of such quality functions, and in particular intersections of them, we were able to

1) reach prediction results that can be beneficial for an automatic quality classification approach, and
2) confirm that popularity metrics are the best predictors, and identify which specific metrics strongly influence good or bad quality.

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

References

[1] 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)*, pages 183–194. ACM, 2008.

[2] M. Allamanis and C. Sutton. Why, when, and what: Analyzing stack overflow questions by topic, type, and code. In *Proceedings of MSR2013 (10th Working Conference on Mining Software Repositories)*, pages 53–56. IEEE Press, 2013.

[3] K. Arai and A. N. Handayani. Prediting Quality of Answer in Collaborative Q\A Community. *International Journal of Advanced Research in Artificial Intelligence*, 2(3):21–25, 2013.

[4] A. Barua, S. W. Thomas, and A. E. Hassan. What are developers talking about? an analysis of topics and trends in stack overflow. *Empirical Software Engineering*, 19(3):619–654, June 2014.

[5] M. Coleman and T. L. Liau. A computer readability formula designed for machine scoring. *Journal of Applied Psychology*, 60(2):283–284, April 1975.

[6] 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.

[7] T. Cover and J. Thomas. *Elements of Information Theory*. Wiley-Interscience, 1991.

[8] M. J. David Blei, Andrew Ng. Latent Dirichlet Allocation. *Journal of machine Learning research*, 3:993–1022, 2003.

[9] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. Indexing by latent semantic analysis. *JOURNAL OF THE AMERICAN SOCIETY FOR INFORMATION SCIENCE*, 41(6):391–407, 1990.

[10] R. Flesch. A new readability yardstick. *Journal of Applied Psychology*, 32(3):p221 – 233, June 1948.

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

[12] R. Gunning. *The Technique of Clear Writing*. McGraw-Hill, 1952.

[13] H. He and E. A. Garcia. Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, 21(9):1263–1284, Sept. 2009.

[14] 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)*, pages 228–235. ACM, 2006.

[15] D. Kuo. On word prediction methods. Technical report, EECS Department, University of California, Berkeley, Dec 2011.

[16] F. Liang. *Word Hy-phen-a-tion by Com-put-er*. PhD thesis, Stanford University, 1983.

[17] H. G. McLaughlin. SMOG grading - a new readability formula. *Journal of Reading*, pages 639–646, May 1969.

[18] A. Pal, R. Farzan, J. A. Konstan, and R. E. Kraut. Early detection of potential experts in question answering communities. In *Proceedings of UMAP 2011 (19th International Conference on User Modeling, Adaptation and Personalization)*, pages 231–242. Springer, 2011.

[19] L. Ponzanelli. Mining StackOverflow to Turn the IDE into a Self-confident Programming Prompter. In *In Proceedings of MSR 2014 (11th Working Conference on Mining Software Repositories)*, pages 102–111. ACM, 2014.

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

[21] C. Stanley and M. D. Byrne. Predicting Tags for StackOverflow Posts. In *Proceedings of ICCM 2013 (12th International Conference on Cognitive Modeling)*, 2013.

[22] 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)*, pages 1–3. ACM, 2012.

[23] 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)*, pages 804–807. ACM, 2011.

[24] I. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques, Second Edition*. Morgan Kaufmann Publishers Inc., 2005.