

Moving to Stack Overflow: Best-Answer Prediction in Legacy Developer Forums

Fabio Calefato
fabio.calefato@uniba.it

Filippo Lanubile
filippo.lanubile@uniba.it

Nicole Novielli
nicole.novielli@uniba.it

University of Bari
Bari, Italy

ABSTRACT

Context: Recently, more and more developer communities are abandoning their legacy support forums, moving onto Stack Overflow. The motivations are diverse, yet they typically include achieving faster response time and larger visibility through the access to a modern and very successful infrastructure. One downside of migration, however, is that the history and the crowdsourced knowledge hosted at previous sites remain separated or even get lost if a community decides to abandon completely the legacy developer forum.

Goal: Adding to the body of evidence of existing research on best-answer prediction, here we show that, from a technical perspective, the content from existing developer forums might be automatically migrated to the Stack Overflow, although most of forums do not allow to mark a question as resolved, a distinctive feature of modern Q&A sites.

Method: We trained a binary classifier with data from Stack Overflow and then tested it with data scraped from Docusign, a developer forum that has recently completed the move.

Results: Our findings show that best answers can be predicted with a good accuracy, only relying on shallow linguistic (text) features, such as answer length and the number of sentences, combined with other features like answer upvotes and age, which can be easily computed in near real-time.

Conclusions: Results provide an initial yet positive evidence towards the automatic migration of crowdsourced knowledge from legacy forums to modern Q&A sites.

Categories and Subject Descriptors

[Software creation and management]: Collaboration in software development; [Information systems]: World Wide Web—*Web applications*

Keywords

Developer forums, Stack Overflow, Q&A sites, Best-answer prediction

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org.

ESEM '16, September 08-09, 2016, Ciudad Real, Spain
© 2016 ACM. ISBN 978-1-4503-4427-2/16/09...\$15.00
DOI: <http://dx.doi.org/10.1145/2961111.2962585>

1. INTRODUCTION

The growing popularity of modern, community-based question answering sites (Q&A) like Stack Overflow descends from mailing lists and web-based discussion forums. As software grew in complexity, developers more and more needed to seek support from experts outside their inner circles [27]. This trend, in fact, has been steadily increasing over the last two decades. At first, mailing lists were useful enough because they allowed developers to archive and search the generated knowledge. Still, searching the archived content required a separated web interface. Then, web-based discussion forums with integrated search represented a step forward in both ease of use and efficiency, preventing same questions to be asked over and over. Nonetheless, the inability to mark help requests as resolved made all that user-generated knowledge difficult to access because, in popular threads, a complete and useful answer is often located several pages away from the start. The ability to highlight in the first position the accepted answer in a thread and the combined psychological effects of gamification are the main factors that boosted the success of Q&A platforms like Quora and Yahoo! Answers [14, 19].

As such, recently, more and more developer communities are abandoning their legacy support forums, moving onto Stack Overflow [27]. Being the first and largest Q&A site of the Stack Exchange network, Stack Overflow is a community where millions of programmers ask questions and provide answers about software development on a daily basis. The motivations are diverse, yet they typically include: achieving faster response time, reaching out to a larger audience of experts, increasing visibility and having a free access to a modern and very successful infrastructure. To further encourage and facilitate the move, Stack Overflow even allows to define custom tags that identify threads from specific developer communities. Despite these evident benefits, one consequent downside of moving onto another platform is that the history and the crowdsourced knowledge generated through a legacy support forum is going to remain separated from the new one or, if a community decides to completely dismantle it, even get lost. For instance, based on the analysis of 21 developer support communities that moved to Stack Overflow between 2011 and 2014 [27], we found that about 20% of them did not archive their content, thus causing a loss of knowledge that still appears in Google searches but turns out to be inaccessible. One might question the value of 'old' knowledge hosted at legacy forums. Indeed, information obsolescence in knowledge platforms is a challenging research problem – i.e., even questions in Stack

Table 1: Breakdown of Docusign dataset, arranged by category.

Forum category	Questions threads	Questions resolved (%)	Answers	% Answers accepted
Java	103	43 (41.75%)	383	11.23%
.Net	490	183 (37.35%)	1660	11.02%
Ruby	156	39 (25%)	553	7.05%
PHP	144	53 (36.81%)	616	8.60%
Misc	679	155 (22.83%)	1538	0.97%
<i>Tot</i>	<i>1572</i>	<i>473 (30.08%)</i>	<i>4750</i>	<i>9.96%</i>

Exchange may become old. However, recent research by Anderson *et al.* [3] has observed the emergence of many long-lasting value conversations in Stack Overflow as an effect of its shift from supporting the initial information-seeking scenario towards the one of creating crowdsourced knowledge that remain valuable over time.

To date, none of the available modern Q&A platforms allows to import existing content from other sources. The migration of content poses several challenges, such as coping with different interaction styles, ensuring quality of imported content, dealing with user reputation and the lack of information about accepted answers and resolved questions. Still, the migration of the whole history of question threads (plus the identification of the accepted answers) from a legacy forum towards a modern Q&A site such as Stack Overflow would be beneficial to all developers who browse the web seeking to solve their technical problems. Some initial work has been performed by Vasilescu *et al.* [30], who investigated how to match the identities of the R-help community members after the migration onto the Stack Exchange platform from the original mailing list. Here, instead, we demonstrate the technical feasibility of content migration. Building on our previous research on answer quality in Stack Overflow [8], we design and run an experiment where a binary classifier is trained to identify the accepted answers in question threads from a legacy developer forum. Specifically, we use a training dataset obtained from Stack Overflow consisting of answers from question threads both closed (i.e., resolved, with one accepted answer in the thread) and still open (i.e., unresolved, with no answer in the thread marked as accepted). The classifier is then evaluated using a test dataset obtained from the legacy support forum of Docusign¹, an electronic signature API for securing digital transactions, which has been recently abandoned as its community moved to Stack Overflow.

The contribution of the paper is twofold. First, the results show that we are able to identify best answers with a good performance (accuracy $\sim 90\%$, $F=.86$, $AUC=.71$), only relying on easy to compute and shallow text features, such as the number of words and sentences, the values of which are then ranked [13, 12]. Our classification model does not rely on any user-related features (e.g., user reputation, badges, number of accepted answers) because they are generally not available in old support forums. Besides, even when available, user-related features are very dynamic and need to be constantly recomputed. Second, we use two corpora from different data sources, thus strengthening the generalizability and robustness of our best answer classifier.

The remainder of the paper is structured as follows. In

Section 2, we present the two datasets used to train and evaluate our classifier. In Section 3 and Section 4, respectively, we report the features included in our model and report the results from the experiment on best answer prediction. The findings and their limitations are first discussed in Section 5 and then compared to previous work in Section 6. Finally, we conclude in Section 7.

2. DATASETS

Two datasets are used to run the experiment reported in this paper. The first dataset comes from the dump of the legacy forum used by the developer community of Docusign.² Since June 2013, the original forum has become read-only, inviting community members to use instead Stack Overflow for development-focused requests, following the custom tag `docusignapi`.

To obtain a dump of the content from the now-abandoned Docusign developer support forum, we built a custom scraper using the Python library Scrapy³. The scraper downloaded all the question threads from the forum and stored them in a local database for convenience. The API is available for several programming languages, namely Java, .NET, Ruby and PHP. Consequently, the forum content was organized according to the programming language, other than having a *miscellaneous* category that contains questions related to API (e.g., SOAP errors) but not connected specifically to one specific language. As shown in Table 1, overall the dataset contains 4750 answers to 1,572 questions threads. Besides, we notice that the dataset is skewed towards unaccepted answers since only $\sim 10\%$ is marked as accepted solution.

The Docusign dataset is intended to be used as test set for a binary classifier discussed next (see Section 3), trained using another dataset retrieved from Stack Overflow. We opportunistically selected Docusign as test set because it allowed the question asker to select one answer in the thread as the accepted solution. Although this is typical of Stack Overflow and other modern Q&A sites, such feature is hard to find in legacy web forums. As such, because the dataset is already annotated with accepted answers, it avoids the need for manually creating a gold standard, i.e., using experts to identify among many answers the accepted solutions (if any) for each thread in the dump. Such procedure is obviously more prone to errors than relying on a dataset like Docusign in which an accepted answer is always marked by the original asker, i.e., the one user who had a problem and sought out a solution from others. Therefore, albeit not common in other legacy forums, here Docusign represents an optimal choice

¹<https://www.docusign.com>

²The dataset is available at <http://tiny.cc/QAmigration>

³<http://scrapy.org>

Table 2: Information elements extracted from the two data sources.

	Info Elements	Stack Overflow	DocuSign
Thread content	Type (quest./answer)	Yes	Yes
	Body	Yes	Yes
	Title	Yes	Yes
	Author name	Yes	Yes
	Answer tags	No	Yes
	Comments	Yes	No
Thread metadata	URL	Yes	Yes
	Question ID	Yes	Yes
	Question closed	Yes	Yes
	Answer count	Yes	Yes
	Accepted answer	Yes	Yes
	Date/time	Yes	Yes
	Answer views	No	Yes

for validating our approach.

Regarding the training set, it consists of over 232,000 answers and about 91,000 question threads. It was built using the following procedure. We first downloaded the official dump from Stack Overflow released on March 2015⁴, consisting of almost 8 million questions and over 13 million answers. Then, as the DocuSign forum is categorized per programming language, from the dump we randomly selected question threads with at least a tag in the following list: `java`, `.net`, `php` and `ruby` (i.e., tagged by the asker as related to the programming language). Eventually, we removed questions closed with self-answers, that is, those questions where the accepted answer is provided by the same asker. Like the DocuSign dataset, the Stack Overflow training set is also skewed towards unaccepted answers. We use the accepted answers (24.83%) as positive examples for training the classifier for the task of best-answer prediction whereas the remaining unaccepted answers represent the negative examples.

Finally, the two dumps came in different formats and structures. To facilitate their representation and analysis, we first converted their content into a common format and then stored it into a local database. To do so, the same information elements had to be extracted from the question threads in each dump. Table 2 provides an overview of which elements are available from each data source and eventual differences. The information elements extracted relate to either the *thread content* (e.g., the question body, the tags associated) or the *thread metadata* (e.g., when an answer was entered, the number of views received by a thread, question upvotes). Crossed out elements are discarded, whereas the others are retained.

In particular, we observe that *comments* to either questions or answers are not available in DocuSign and, therefore, they are not included in the datasets. Besides, we discarded the *views* information element because it was inconsistently used across the two data sources. In Stack Overflow, the number of views refers to the whole thread, whereas in Do-

cusign to a single answer. Yet, in the DocuSign forum we found no direct links to answers, only to threads, and, therefore, it is unclear how answers in the same thread would generate a different number of views. As regards the *author* information element, in the DocuSign forum only the name is available. As such, we discarded from the Stack Overflow dataset any information element related to user profile, such as reputation and badges. Finally, as per *tags*, the DocuSign forum allows to define tags also for the answers, unlike Stack Overflow, which attaches labels to the whole question thread. As such, answers tags in the DocuSign dataset are discarded.

3. FEATURE DESCRIPTION

In this section, we describe the features employed to train and test our classifier. From the information elements presented in the previous section, we define four main categories of features: *linguistic*, *vocabulary*, *meta* and *thread*. As further discussed in Section 6, we do not exploit any user-related features (e.g., reputation score) because in legacy web forums such information is generally not available. Furthermore, user-related content changes frequently over time and thus, the proper use of user-related feature would require a real-time data analysis.

The overall set of features (22) is reported in Table 3, arranged by categories. Both *linguistic* and *vocabulary* feature categories are concerned with *readability*. Specifically, *linguistic* features represent the attributes of questions and answers, and are intended to estimate their quality. Such *linguistic* features are called ‘shallow’ because they measure readability through the ‘surface’ properties of a text, such as the number of words and the average word length [22]. As such, they are also computationally cheap. For example, the *linguistic* features include *length* (in characters), *words count* and *average number of words per sentence* in an answer. We also add *contains hyperlinks* to this category of features because in our previous works on factors affecting the probability of answer to be accepted we found that the presence of links to external resources is positively related to the perception of its completeness [8].

Besides the *linguistic* features, in order to estimate the readability of an answer we also employ two *vocabulary* features, *normalized log likelihood* and the *Flesch-Kincaid Grade*. The normalized log likelihood (LL_n , hereinafter), already employed in previous work [22, 13, 12], uses a probabilistic approach to measure to what extent the lexicon in an answer is distant from the vocabulary used in the whole forum community. Specifically, (LL_n , hereinafter) is defined as follows:

$$LL_n = \frac{LL = \sum_{w_s} C(w_s) \log(P(w_s|Voc))}{UC(w_s)} \quad (1)$$

Given s , a sentence in an answer, $P(w_s|Voc)$ is the probability of the word w_s to occur, according to the background corpus Voc , and $C(w_s)$ is the number of times the word w_s occurs in s . LL is normalized by dividing it over the number of unique words occurring in s . The normalization is necessary to take into account answers of different lengths.

The *Flesch-Kincaid Grade* ($F-K$, hereinafter), defined in [18] and already used by Burel *et al.* in [7], is a readability metric for English calculated as follows:

⁴<https://archive.org/details/stackexchange>

Table 3: Summary of the features in our model, arranged by category.

Category	Feature	Also ranked
Linguistic	Length (in characters)	Yes
	Word count	Yes
	No. of sentences	Yes
	Longest sentence (in characters)	Yes
	Avg words per sentence	Yes
	Avg chars per word	Yes
Vocabulary	Contains hyperlinks	Yes
	LL_n	Yes
Meta	$F-K$	Yes
	Age (hh:mm:ss)	Yes
	Rating score (upvotes - downvotes)	Yes
Thread	Answer count	

$$F-K_{p_i}(awsp_{p_i}, asps_{p_i}) = 0.39 awps_{p_i} + 11.8 asps_{p_i} - 15.59 \quad (2)$$

In the definition (2) above, for any given post p_i , $awps_{p_i}$ is the average number of words per sentence and $asps_{p_i}$ is the number of syllables per word.

Two are the features belonging to the *meta* category. The first one, *age* applies only to answers and it computes the time difference since the question has been posted. The second one, *rating score* is the score (i.e., the number of upvotes minus the number of downvotes) a post received by users and reflects its perceived usefulness. The *thread* features include only the *answer count*, i.e., the number of answers to a question, a measure that reflects the popularity of a thread.

Furthermore, as shown in the rightmost column of Table 3, for all the feature categories except *thread*, we also assign ranks after computing their numeric values. In other words, for each question thread, we group all answers, compute a feature, and then rank the values in ascending or descending order (in the following, we refer to this procedure as *ranking*). For instance, for the *word count* linguistic feature, the answer in the thread with the largest value ranks 1, the second largest ranks 2 and so on (i.e., descending order) because we assume that long answers are more accurate and elaborate and, hence, have a larger chance of being accepted. For the *age* feature, instead, we assign the rank 1 to the quickest answer (i.e., ascending order), because previous research has demonstrated that responsiveness plays a major role in getting answers accepted in Q&A websites [19]. This ranking approach has been inspired by the *discretization* of features reported by Gkotsis *et al.* in [13, 12] who found that feature discretization makes binary classifiers for best-answer prediction robust across different Q&A sites, albeit all belonging to the same Stack Exchange platform. This is appealing to us because we use different datasets for training and testing our model.

Finally, to conclude the description of how our classifier is built, we clarify our definition of ‘best answer’. In our study, as well as in Stack Overflow and Docusign, the best answer is the one marked as accepted by the original asker.

However, sometimes, the best answer in a thread is not actually the one accepted – i.e., found to be useful by the original asker, but rather the one that receives more upvotes – i.e., found to be the most useful by the whole community. Therefore, we clarify that in our study we are not considering the ‘*absolute best answer*’, but rather the ‘*fastest and good-enough answer*’ that provides a prompt and effective solution to the problem reported by the asker. This is because, on one hand, we believe that the time dimension is a predictor that cannot be overlooked. Hence, we included the *meta* feature *age* in our model. On the other hand, this is also the same approach taken by Stack Overflow. Therefore, being it the target platform of our migration experiment, we remain consistent with Stack Overflow conceptualization of best answer.

4. EVALUATION

The experiment reported here aims at assessing the fit of our feature set for best-answer prediction (Section 4.1) as well as evaluating how each of the 22 features in our model affects best-answer prediction (Section 4.2).

As a preliminary step, we compare the performance of several tree classifiers, which have been found to be the most successful in similar contexts (e.g., see [7, 2, 24]). In particular, we experiment with Alternating Decision Trees (ADT), J48, Random Forest and Random Tree. We evaluate the performance of each classifier in a 10-fold cross-validation setting, using the Stack Overflow dataset and measuring *precision* (P), *recall* (R) and *F-measure* (F). In fact, these metrics are regarded as a standard for assessing a classifier effectiveness in text categorization tasks [25]. According to the definition provided by Sebastiani [25], the precision of a classifier is the probability that if a random text is classified as belonging to a given class (e.g., ‘answer accepted’ or ‘answer not accepted’), this prediction is correct. Analogously, the recall of a classifier is the probability that if a random text belongs to a given class, then this class is predicted. These probabilities are estimated by observing the behavior of the classifier on the validation set. In our scenario, the precision is computed as the proportion of retrieved best answers that are actual best answers in the dataset (3) and, for recall, in terms of the proportion of best answer successfully retrieved over the total best answers (4).

$$P = \frac{TruePositives}{TruePositives + FalsePositives} \quad (3)$$

$$R = \frac{TruePositives}{TruePositives + FalseNegatives} \quad (4)$$

F-measure (also known as F-score or F1) synthetically describes a classifier’s performance since it is computed as the harmonic mean of precision and recall. Precision, Recall and F-measure all assume values in $[0, 1]$, reaching best score at 1 and worst at 0.

For the sake of comparison with previous research (see related work in Section 6), we also calculate the Area Under Curve (AUC) as a further measure of the each classifier performance [10]. AUC is defined as a plot of the true positive rate (as the y coordinate) versus the false positive rate (as the x coordinate). AUC expresses a measure of the overall performance of a prediction test: the bigger its AUC is, the better the overall performance. The highest observable

Table 4: Model comparison on Stack Overflow training set with 10-fold cross-validation.

Classifier	PA	P	R	F	AUC
ADT	78.9%	0.77	0.79	0.77	0.83
J48	78.2%	0.77	0.78	0.77	0.74
Random Forest	72.0%	0.75	0.77	0.76	0.78
Random Tree	72.0%	0.72	0.72	0.72	0.63

variable for AUC is 1 whereas AUC=.50 corresponds to the curve associated with random prediction.

We ran our experiments using Weka⁵ [15], adopting the default configuration setting of parameters as provided by the tool. Table 4 reports the performance achieved by each classifier, exploiting the complete set of features listed in Table 3. For the sake of completeness, we also report the overall prediction accuracy (PA), that is the percentage of correctly classified cases.

Consistently with previous research [7, 13] the ADT classifier [11] achieves the best performance in terms of both F1 and AUC. We also performed the same comparison using *non-tree* classifiers, namely, Logistic Regression, SVM and Naïve Bayes. The results, omitted due to space constraints, show that all these models are outperformed by the tree classifiers. Therefore, we adopt ADT to further investigate the suitability of our approach in terms of robustness and generalizability, as described in the following sections. Although all the tree-learning algorithms produce comparable outcomes, we operate this for a better comparison with the results obtained in previous work [7, 13, 12], the research settings of which are similar to ours (see related work in Section 6).

4.1 Results: Best-Answer Prediction

The selected ADT algorithm is used to train the best answer classifier using the dataset of answers from Stack Overflow. Then, the classifier is tested using the dataset from the Docusign developer forum. Table 5 presents, for each row, the performance of our best answer classifier when different features are enabled. The last rows, instead, shows the performance with all the features enabled.

From the results reported in the first row of the table, we observe that the performance (AUC=.56) obtained only enabling the *linguistic* features (the largest type of features available in our experiment) is barely above random prediction (AUC=.50). The performance of the classifier improves as soon as we enable also the ranked version of *linguistic* features, as reported in the second row (F=.82, AUC=.69). Then, in the third and fourth row, respectively, we also enable the *thread* and the *meta* features, that is, the two other types of features that are computationally cheap. In both cases, we observe an increase to AUC=.70. In particular, the fourth combination with *meta* features manages to also achieve slightly higher values in terms of precision and recall (P=.87, R=.90, F=.86).

Furthermore, in the fifth combination of features, we enable together the *linguistic*, *meta* and *thread* features, both ranked and non-ranked. We observe that this feature set produces results identical to the last row where all the features are enabled, including the *vocabulary* category (PA=89.6%,

P=.85, R=.90, F=.86, AUC=.71). Here we opportunistically choose to enable the *vocabulary* features only at the end because, unlike all the other features, they are computationally intensive. For example, the LL_n requires the definition of a vocabulary containing all the stems of the unique words in a corpus and the computation of a distribution matrix to assess how frequently each word occurs in the corpus. Yet, the uniformity of results in the last two rows implies that, despite the complexity in calculation, the *vocabulary* features have a limited effect on best-answer prediction on the Docusign dataset.

Finally, to complete the performance assessment, we build two rule-based classifiers and use them as baselines to benchmark our ADT classifier. These naïve classifiers (see Table 6) require no training as they are based on simple heuristics that select as best answer the (i) most upvoted answer (i.e., with the highest *rating score*), (ii) the first answer received (i.e., with the smallest *age*) and (iii) either of the two. We selected the *age* and *rating score* features because they have been reported as two of the most important predictors in the related literature (e.g., see [7]), as we also observe next (see Section 4.2). Looking at the results reported in Table 6, we notice that fastest answers (PA=68%, row (ii)) tend to be accepted as solutions more often than the top scoring ones (PA=22.8%, row (i)). The combination of the two rules produce a performance (PA=21%, F=.19, AUC=.54, row (iii)) that is, respectively, worse than (ii) and comparable to (i). Besides, all the naïve classifiers are slightly above random prediction (AUC values between .54 and .59). Therefore, they are all largely outperformed by the ADT classifier (PA=89.6%, F=.86, AUC=.71), as observed in row (5) of Table 5.

4.2 Results: Feature Comparison

The results of the classification experiment from the previous subsection show that some features contribute more than others to boost the performance of the classifier when selected. Therefore, here we assess of the importance of each of the 22 available features to the task of best-answer prediction, while also measuring the benefits of applying ranking to them.

This assessment relies on computing the *information gain* (IG), a measure based on entropy that is largely used in machine learning research in combination with decision trees for classification tasks [20]. Information gain is defined using the entropy H measurement as follows:

$$IG(C, A) = H(C) - H(C|A) \quad (5)$$

In the definition (5) above, A represents an attribute (e.g., the *word count* feature) and C is the class (i.e., ‘answer accepted’, ‘answer not accepted’). In other words, here information gain provides us with a measure of how a single feature helps to classify each answer in the Docusign test set *correctly* as either accepted or not accepted. The higher the IG score of a feature, the better its contribution to the classifier.

Table 7 shows the value of information gain computed for each feature in our set, grouped by category (first three columns from the left). Note that in the fourth column we report also the value of the information gain for ranked features. The last column of the table reports the relative change in information gain when ranking a feature. The table also displays in gray the top 10 features, sorted by

⁵<http://www.cs.waikato.ac.nz/ml/weka>

Table 5: Results for best-answer predictions on Docusign dataset using Alternating Decision Trees (ADT) with different sets of features enabled.

Features enabled	PA	P	R	F	AUC
(1) Linguistic	90.0%	0.81	0.90	0.85	0.56
(2) Linguistic + Ranked Linguistic	81.0%	0.84	0.81	0.82	0.69
(3) Linguistic + Ranked Linguistic + Thread	85.0%	0.85	0.85	0.85	0.70
(4) Linguistic + Ranked Linguistic + Meta + Ranked Meta	90.0%	0.87	0.90	0.86	0.70
(5) Linguistic + Ranked Linguistic + Meta + Ranked Meta + Thread	89.6%	0.85	0.90	0.86	0.71
<i>All (including Vocabulary)</i>	<i>89.6%</i>	<i>0.85</i>	<i>0.90</i>	<i>0.86</i>	<i>0.71</i>

Table 6: Performance baselines from rule-based classifiers tested on Docusign.

Rule	PA	P	R	F	AUC
(i) Top-voted answer (<i>rating score</i>)	22.8%	0.11	0.96	0.20	0.55
(ii) First answer (<i>age</i>)	68.0%	0.15	0.47	0.23	0.59
(iii) (i) or (ii)	21.0%	0.11	0.97	0.19	0.54

information gain in descending order.

First, from Table 7 we can observe that the worst performing feature for the classification of the Docusign dataset is the linguistic feature *contains hyperlinks*, as it provides a null information gain in both versions (ranked and not), followed by the linguistic features *avg. words per sentence* (IG=0.0037) and the vocabulary feature *F-K* (IG=0.0102), respectively. The low gain of both versions of the *F-K* feature explains the limited effect of vocabulary features on best-answer prediction observed in Section 4.1.

Second, the most important feature, which belongs to the set of *metadata* features, is the *rating score*, both in the ranked (IG=0.1255) and non-ranked (IG=0.0789) version. The relative change in IG between the two version of *rating score* is +0.59, which represents a +37% gain in favor of the ranked version as compared to the non-ranked one.

Finally, we notice that 7 out of 10 features in the top 10 are ranked. The benefit of ranking is even more evident when considering that the process improves the information gain for all the features except *F-K*, for which we observe a small negative changes (IG=-0.22). For all the other features, instead, the relative change is positive and ranging between +0.59 and +13.5.

5. DISCUSSION

In this section, we first discuss the findings from the experiment presented in the previous section and then, the benefits of employing ranked features in our classifier. Finally, we outline the limitations of our work.

5.1 Best-Answer Prediction Performance

We provide evidence that predicting best answers from legacy developer forums like Docusign can be performed by only relying on features that are easy to compute in real time. These include features like the number of answers in a thread and the number of upvotes given to an answer, as well as shallow text features, such as the length of an answer and the average number of words per sentence. In fact, we do not include any user-related feature (e.g., user reputation, number of answers accepted) because, other than being often unavailable in legacy forums, they are highly dynamic and problematic to compute in real time.

Table 5 presents the results achieved by the classifier, which is able to correctly identify almost 90% of the accepted answers in the Docusign dataset (AUC=.71). Although AUC is commonly used to present results for binary decision problems like best-answer prediction, for the sake of completeness, the table also reports the values of prediction accuracy, precision, recall and F-measure. This is because, when dealing with highly skewed datasets as in our case (i.e., uneven numbers of accepted and unaccepted answers), AUC can give a more informative picture of a binary classifier performance [9]. Furthermore, we found that *vocabulary* features are non-essential to our classifier (see the last two rows of Table 5), as there is no performance improvement, and therefore we can drop this category from the model without any performance loss in best-answer prediction. This is another important finding from our experiment because, like those user-related, *vocabulary* features are computationally intensive.

Moreover, the good performance coming from the inclusion of *linguistic*, *meta* and *thread* features is strengthened by the experimental setup. To the best of our knowledge, this is the first work to present an experiment on best-answer prediction where a classifier is trained and tested with corpora from different sources. In fact, relying on distinct training and test datasets suggests that the final set of features in our model is able to capture intrinsic constructs related to the quality of an answer that are not specific of either the Docusign forum or its ‘community vocabulary’. In future work, we intend to further this evidence by replicating the experiment on datasets coming from other legacy developer support forums.

To further assess the classifier performance, we focus on the results achieved when the *meta* features are not enabled (see row (3) in Table 5). This classification experiment is particularly relevant because the set *meta* includes *rating score* (i.e., answer upvotes), a feature that has been so popularized by Stack Overflow that it has been later ‘backported’ to old legacy forums, such as Docusign. Still, most of these forums do not support answer upvoting and, therefore, the experiment in row (3) provides an idea of how our classifier might perform in future replications when such piece of information is not available. Specifically, here our classi-

Table 7: Summary of Information Gain (IG) per feature and the relative IG change between the non-ranked and ranked versions in Docusign.

Feature type	Feature name	IG	IG Ranked Feature	IG relative change due to ranking
Linguistic	Length	0.0112	0.0598	+4.34
	Word count	0.0097	0.0585	+5.03
	No. of sentences	0.0061	0.0541	+7.87
	Longest sentence	0.0063	0.0499	+6.92
	Avg words per sent.	0.0020	0.0290	+13.5
	Avg chars per word	0.0037	0.0340	+8.19
	Contains hyperlinks	0.0000	-	-
Meta	Age	0.0126	0.0471	+2.40
	Rating score	0.0789	0.1255	+0.59
Vocabulary	LL_n	0.0040	0.0366	+8.15
	$F-K$	0.0102	0.0079	-0.22
Thread	Answer count	0.0657	-	-

Cells in gray are the features in the top 10, sorted by IG in descending order

fier achieves good results (AUC=.70), equal or comparable to those achieved when *rating score* is enabled, as reported in rows (4) and (5) of Table 5. Albeit further replications are certainly needed, this result suggests that our classifier might perform well enough even when answer upvotes are unavailable in legacy Q&A sites.

Finally, one might argue that, given the limited size (at least, compared to Stack Overflow) of legacy developer forums, simpler approaches that do not require many features and machine learning might be employed to retrieve best answers. Thus, we have benchmarked our classifier against a few rule-based, naïve classifiers (see Table 6), which build on simple heuristics that select the best answer from a question thread, respectively, as the (i) top voted answer, (ii) the fastest answer and (iii) either of the two. Tested on Docusign, we found these simple ‘strawman’ models to achieve a poor performance, with AUC values close to random prediction (i.e., between .54 and .59). As such, they are all largely outperformed by our ADT classifier under most of the conditions (see Table 5, rows (2)-(5)). In particular, regarding the first heuristic, such a poor result is explained by the low number (18%) of question threads in the Docusign dataset containing at least one answer that has received upvotes. For example, on the Stack Overflow dataset the same ‘strawman’ model successfully identifies the accepted answer in 93% of the cases, showing that *rating score* is a highly reliable predictor of best answers for modern Q&A sites where the practice of upvoting answers is widely used and part of the community culture. Overall, these results suggest that, while complex classifiers based on machine learning might be superfluous for Stack Overflow, the best-answer prediction in legacy forums is a challenging task that requires the development of more complex predictive models with multiple features.

5.2 Benefits of Ranking

The good performance of our classifier is also due to the benefits of employing ranked features. Except for the *thread* features, all the others in our feature set come in two versions, not-ranked and ranked. The ranking process, i.e., sorting the values for a feature in a given thread and then assign ranks according to the position, has been previously

proposed by Gkotsis *et al.* [13, 12] with the name of *discretization*. They investigated best-answer prediction in 21 sites from Stack Exchange, including Stack Overflow. Despite their analysis showed that the characteristics of accepted answers vary significantly across the sites, the adoption of discretization made their classifier robust enough to keep good performances on each one. Therefore, to conduct our experiment, we have borrowed the idea of feature discretization because of our experimental set up, with two distinct training and test sets retrieved from entirely different platforms.

Our classifier achieves performances that are comparable to those reported by Gkotsis *et al.* (see Section 6 for more details). Besides, the information gain measures reported in Table 7 show that the ranked features contribute to the correct classification of the answers in the Docusign forum more than those non-ranked. In fact, in the top 10 features sorted in descending order by information gain, 7 are ranked features. Furthermore, comparing the information gain of each ranked feature to its non-ranked counterpart, a positive increase in information gain can be observed in 9 out of 10 cases. Therefore, our findings show that the ranking (or discretization) process applied to features makes the best answer classifier robust even outside of Stack Exchange.

5.3 Limitations

As regards limitations, we first acknowledge the limited size of the training and test sets retrieved from Stack Overflow and Docusign, respectively. The datasets employed in similar research work contain tens of thousand, if not million, questions and answers. We intend to mitigate this limitation through future replications of the current experiment, which will consequently strengthen and augment the generalizability of our current findings. Still, none of the existing legacy developer forums has ever been as successful as Stack Overflow is. As such, no matter how large they might be, training and test set sizes will differ in size in future replications as well.

Furthermore, we have to improve our automatic approach of best-answer prediction to cope with cases where no best answer is available in a thread at all. Especially in old support forums, it is likely to find threads where none of the

received answer is correct or even good. Instead, our current approach ‘naively’ forces the selection of one answer in any case. Another related limitation is that our approach currently does not distinguish between answers and comments. Albeit, this limitation does not affect current results (i.e., Docusign allowed to enter only answers, not comments), we will have to address it in future replications involving other data sources.

Our approach, albeit not platform-dependent, is here targeted to legacy, discontinued forums. As such, we assume that no further answers are added to question threads. This is not the case when dealing with still active Q&A sites, where the answer identified as the best in a thread may become obsolete if a newer, better answer is entered later. Such scenario is not unlikely, as shown by Oktay *et al.* [21], who proved that answers in Stack Overflow are added to question threads for a considerable amount of time and even after an answer has been marked as accepted.

Finally, we acknowledge that the migration from legacy developer forums to Stack Overflow is currently not practically implementable because the current API (ver. 2.2) does not allow to add content programmatically. Still, the API development roadmap⁶ shows that write access will be available from ver. 3.

6. RELATED WORK

The largest share of previous work in mining community Q&A sites has focused on: (i) assessing the quality of questions e.g., studying specifically how low quality post detection can be improved [23] and how software developers interact, as well as the main characteristics of successful questions [29]; (ii) how to write good answers, e.g., helping developers to quickly earn good reputation scores [6]; the impact of sentiment on the chance of getting an answer accepted [8] and the role played by social cues on the perceived quality of an answer [16]; (iii) the main topics being discussed by developers [4, 5].

Instead, not much attention has been devoted to the problem of best-answer prediction, which is critical to support a migration from a legacy developer forum to a community Q&A site. Table 8 shows a breakdown of strictly related papers in the field that we were able to find, sorted from the least to the most recent. In the following, we review each of them, comparing their results to ours.

In [1], Adamic *et al.* report the first experiment on best-answer prediction using data only from the *Programming & Design* category on the Yahoo! Answers. The prediction accuracy achieved (73%) is lower than ours, even though their logistic regression-based classifier relies also on user-related features that are not included in our experiment because too computationally-intensive. A similar work is that of Shah and Pomerantz [26] who run another experiment with logistic regression, using a dataset from Yahoo! Answers, which is comparable in size to ours. Albeit the performance of their classifier is close to ours, their feature set contains user-related features as in the work of Adamic *et al.* In addition, their dataset from Yahoo! Answers also contains non-technical content that is not allowed in Stack Overflow. The findings from the experiment by Tian *et al.* [28] are directly comparable with ours because they train a classifier

on a dataset obtained from Stack Overflow and do not rely on user-related features. Besides, their feature set is similar to our set of non-ranked features. Their classifier, running a 2-fold cross-validation, achieved a lower prediction accuracy (72%) compared to that of our classifier when relying on ranked features. We note that these three works report the prediction performance using only accuracy, a metric that has been observed to be unreliable when used alone with datasets, as in our case, which are intrinsically skewed (i.e., where the class of non-accepted answers is naturally larger than that of accepted answers) [17]. Accordingly, we also report the F-measure and AUC measures, which are instead reliable in case of data skewness [17].

Burel *et al.* [7] compare the performance of their classifier across two different Q&A platforms, namely a SAP-specific Q&A site (SAP Network Community), and Stack Exchange, from which they pick one technical site (Server Fault) and one non-technical site (Cooking). Their results are good, with AUC ranging between .89 and .92, and also interesting because, as in our case, they found ADT to be the best performing tree learning algorithm. Besides, one interesting similarity is the poor performance of the vocabulary F - K feature that was used to assess the readability of answers. However, the comparison of results is difficult because they do not employ separate training and test sets (only cross-validation) and employ user-related features that we exclude.

Finally, the closest comparison is with the work of Gkotsis *et al.* [13, 12], because they do not rely on user-related feature and make use of ranked features. The authors employed a very large dataset consisting of 21 Stack Exchanges sites, including Stack Overflow. Overall, the main findings from their work are consistent with ours, and so are the sets of features employed, all computationally cheap. Their classifier also achieves the best performance using the ADT learning algorithm. Compared to our results, their findings report higher AUC scores for Stack Overflow (.85) and, on average, for all the sites (.87). However, although the prediction score of our classifier is lower (AUC=.71), the difference can be explained by the usage of cross-validation instead of different training and test sets like us. In fact, our classifier achieves similar results when running a 10-fold cross-validation on Stack Overflow (AUC=.83). Besides, they found that, thanks to ranking (called discretization by Gkotsis *et al.*), their classifier maintains good performance across the different Q&A sites in their dataset, albeit all belonging to the Stack Exchange platform. Likewise, we found that our classifier trained on Stack Overflow maintains good performance on the legacy Docusign forum. Therefore, while their work shows that the process of ranking features within a question thread is robust and generalizable to any Stack Exchange site, our results go further, suggesting that the approach is also generalizable across different Q&A platforms. Combined together, these results indicate that the identified set of features, augmented through ranking, define a generalizable classifier that is able to accurately model cross-domain and cross-platform aspects that are intrinsic to the problem of best-answer prediction, while discarding the superfluous domain- and platform-specific ones. Yet, further evidence is needed, using training sets from other modern Q&A platforms, such as Quora and Yahoo! answers, and test sets from other legacy developer forums.

⁶<http://stackapps.com/questions/1999/announcing-api-version-1-1-and-roadmap>

Table 8: Breakdown of related work on best-answer prediction.

Reference	Dataset (# questions / answers)	Feature categories (# total)	Feature ranking	Experimental setting	Results
Adamic <i>et al.</i> (2008) [1]	Yahoo! Answer - Programming & Design (N/A)	user, thread, linguistic (4)	No	10-fold cross- validation with logistic regression	Pred. accuracy= \sim 73%
Shah & Pomerantz (2010) [26]	Yahoo! Answer (\sim 1.3K/5K)	user, thread, meta, linguistic (21)	No	10-fold cross- validation with logistic regression	Pred. accuracy= \sim 84%
Tian <i>et al.</i> (2013) [28]	Stack Overflow (\sim 103K/196K)	thread, meta, linguistic (16)	No	2-fold cross- validation with random forest	Pred. accuracy= \sim 72%
Burel <i>et al.</i> (2012) [7]	SAP Commu- nity Net (SCN)* (\sim 95K/427K) Server Fault (SF) [†] (\sim 36K/95K) Cooking (C) [†] (\sim 2K/7K)	user, thread, meta, lin- guistic, vo- cabulary (19*/23 [†])	No	10-fold cross- validation with ADT	P=.83 R=.84 F=.83 AUC=.88 (SCN) P=.85 R=.85 F=.84 AUC=.91 (SF) P=.87 R=.87 F=.87 AUC=.92 (C)
Gkotsis <i>et al.</i> (2014-15) [13, 12]	21 Stack Ex- change sites including Stack Overflow (SO) (\sim 4M/8M)	thread, meta, linguistic, vo- cabulary (14)	Yes	10-fold [⊙] and leave-one-out [⊗] cross-validations with ADT	P=.82 R=.66 F=.73 AUC=.85 (SO) [⊙] P=.84 R=.70 F=.76 AUC=.87 (avg) [⊗]

7. CONCLUSIONS AND FUTURE WORK

As more and more developer support forums abandon their legacy websites to move onto Stack Overflow, lots of crowdsourced knowledge is at risk of being left behind. For instance, today a Docusign developer seeking for a solution to a problem on the web might follow a link to a related discussion thread in the legacy Docusign forum only to get a page with a “*Discussion no longer exists*” error and an invite to post new questions at Stack Overflow using the community tag `docusignapi`.

In order to show the technical feasibility of an automatic content migration from legacy developer forums to modern Q&A sites, we set up an experiment where we trained a binary classifier using data from Stack Overflow and then tested it on a dataset retrieved from the legacy Docusign support forum. The classifier was built by only including easy to compute and shallow text features (i.e., no computationally intensive features). The results showed that, when all features are enabled, our model is able to identify best answers with a good performance (accuracy \sim 90%, F=.86, AUC=.71). Albeit the Docusign dump comes with the answer rating information that is not generally available in legacy forums, our experiment proved that our model is capable of coping with the lack of such piece of metadata, achieving a performance (accuracy \sim 85%, F=.85, AUC=.70) comparable to the best one obtained with the full set of features. Finally, to the best of our knowledge, this is the first experiment on best answer prediction to use different training and test sets, an indicator of the strength and generalizability of the model achieved also including the ranking of the features computed for each thread.

Although the migration of content from legacy developer forums to Stack Overflow is not practically implementable,

nonetheless our work may provide some practical insights for improving the design of modern Q&A platforms. In particular, this research contains insights to cope with the problem of lack of resolved question. In fact, the Stack Overflow dump used in the experiment consists of almost 8 million questions of which less than 5 millions (58%) are provided with an accepted answer. Sometimes, questions are not closed because no perfect answer was provided; other times, it is just because the question askers did not take any action. Yet, in every thread there are some answers that are better than others. Especially when threads are popular, this situation is detrimental to developers because it forces them to go through all the answers, looking for a solution. Hence, our research presents insights upon which to build tools that rely on features other than the only number of upvotes to help developers tell good answers from bad ones.

As future work, to gain further evidence of the robustness and generality of our approach, we will run replications of this experiment using larger training and test sets. Currently, we are building training sets from the content of Q&A platforms other than Stack Overflow, such as Quora and Yahoo! Answers, where threads are categorized by topic and we can filter out non-technical questions.

Acknowledgements

This work is partially funded by the project ‘Investigating the Role of Emotions in Online Question & Answer Sites’, funded by MIUR (Ministero dell’Università e della Ricerca) under the program “Scientific Independence of young Researchers” (SIR).

8. REFERENCES

- [1] L. A. Adamic, J. Zhang, E. Bakshy, and M. S. Ackerman. Knowledge sharing and yahoo answers. In *Proceeding of the 17th international Conf. on World Wide Web - WWW '08*. ACM, 2008.
- [2] E. Agichtein, C. Castillo, D. Donato, A. Gionis, and G. Mishne. Finding high-quality content in social media. In *Proc. of the Int'l Conf. on Web search and web data mining - WSDM '08*. ACM, 2008.
- [3] A. Anderson, D. Huttenlocher, J. Kleinberg, and J. Leskovec. Discovering value from community activity on focused question answering sites. In *Proc. of the 18th ACM SIGKDD Int'l Conf. on Knowledge discovery and data mining - KDD '12*. ACM, 2012.
- [4] K. Bajaj, K. Pattabiraman, and A. Mesbah. Mining questions asked by web developers. In *Proc. of the 11th Working Conf. on Mining Software Repositories - MSR 2014*. ACM, 2014.
- [5] 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, nov 2012.
- [6] A. Bosu, C. S. Corley, D. Heaton, D. Chatterji, J. C. Carver, and N. A. Kraft. Building reputation in StackOverflow: An empirical investigation. In *2013 10th Working Conf. on Mining Software Repositories (MSR)*. IEEE, may 2013.
- [7] G. Burel, Y. He, and H. Alani. Automatic Identification of Best Answers in Online Enquiry Communities. In *Lecture Notes in Computer Science*, pages 514–529. Springer, 2012.
- [8] F. Calefato, F. Lanubile, M. C. Marasciulo, and N. Novielli. Mining Successful Answers in Stack Overflow. In *2015 IEEE/ACM 12th Working Conf. on Mining Software Repositories*. IEEE, may 2015.
- [9] J. Davis and M. Goadrich. The relationship between Precision-Recall and ROC curves. In *Proc. of 23rd Int'l Conf. on Machine learning - ICML '06*. ACM, 2006.
- [10] T. Fawcett. An introduction to ROC analysis. *Pattern Recognition Letters*, 27(8):861–874, jun 2006.
- [11] Y. Freund and L. Mason. The Alternating Decision Tree Learning Algorithm. In *Proc. of the 16th Int'l Conf. on Machine Learning*, ICML '99, pages 124–133, San Francisco, CA, USA, 1999. Morgan Kaufmann.
- [12] G. Gkotsis, M. Liakata, C. Pedrinaci, K. Stepanyan, and J. Domingue. ACQUA: Automated Community-based Question Answering through the Discretisation of Shallow Linguistic Features. *Journal of Web Science*, 1(1):1–15, jun 2015.
- [13] G. Gkotsis, K. Stepanyan, C. Pedrinaci, J. Domingue, and M. Liakata. It's all in the content. In *Proc. of the 2014 ACM Conf. on Web science - WebSci '14*. ACM, 2014.
- [14] S. Grant and B. Betts. Encouraging User Behaviour with Achievements: An Empirical Study. In *Proc. of the 10th Working Conf. on Mining Software Repositories*, MSR '13, pages 65–68, Piscataway, NJ, USA, 2013. IEEE Press.
- [15] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The WEKA data mining software. *SIGKDD Explor. Newsl.*, 11(1):10, nov 2009.
- [16] K. Hart and A. Sarma. Perceptions of answer quality in an online technical question and answer forum. In *Proc. of 7th Int'l Workshop on Coop. and Human Aspects of Softw. Eng. - CHASE'14*. ACM, 2014.
- [17] H. He and E. A. Garcia. Learning from imbalanced data. *Knowledge and Data Engineering, IEEE Transactions on*, 21(9):1263–1284, 2009.
- [18] J. Kincaid, R. J. Fishburne, R. Rogers, and B. Chissom. Derivation of new readability formulas (Automated Readability Index, Fog Count and Flesch Reading Ease Formula) for Navy enlisted personnel. *Research Branch Report*, 8(75), 1975.
- [19] L. Mamykina, B. Manoim, M. Mittal, G. Hripcsak, and B. Hartmann. Design Lessons from the Fastest Q&A Site in the West. In *Proc. of the SIGCHI Conf. on Human Factors in Computing Systems*, CHI '11, pages 2857–2866, New York, NY, USA, 2011. ACM.
- [20] T. M. Mitchell. *Machine Learning*. McGraw-Hill, Inc., New York, NY, USA, 1 edition, 1997.
- [21] H. Oktay, B. J. Taylor, and D. D. Jensen. Causal discovery in social media using quasi-experimental designs. In *Proc. of the 1st Workshop on Social Media Analytics - SOMA '10*. ACM, 2010.
- [22] E. Pitler and A. Nenkova. Revisiting readability. In *Proc. of the Conf. on Empirical Methods in Natural Language Processing - EMNLP '08*. ACL, 2008.
- [23] L. Ponzanelli, A. Mocci, A. Bacchelli, M. Lanza, and D. Fullerton. Improving Low Quality Stack Overflow Post Detection. In *2014 IEEE Int'l Conf. on Software Maintenance and Evolution*. IEEE, sep 2014.
- [24] M. Rowe, S. Angeletou, and H. Alani. Predicting Discussions on the Social Semantic Web. In *The Semantic Web: Research and Applications*, pages 405–420. Springer Science + Business Media, 2011.
- [25] F. Sebastiani. Machine learning in automated text categorization. *CSUR*, 34(1):1–47, mar 2002.
- [26] C. Shah and J. Pomerantz. Evaluating and predicting answer quality in community QA. In *Proceeding of the 33rd Int'l ACM SIGIR Conf. on Research and development in information retrieval - SIGIR '10*. ACM, 2010.
- [27] M. Squire. Should We Move to Stack Overflow? Measuring the Utility of Social Media for Developer Support. In *2015 IEEE/ACM 37th IEEE Int'l Conf. on Software Engineering*. IEEE, may 2015.
- [28] Q. Tian, P. Zhang, and B. Li. Towards Predicting the Best Answers in Community-based Question-Answering Services. In E. Kiciman, N. B. Ellison, B. Hogan, P. Resnick, and I. Soboroff, editors, *Proc. of the 7th Int'l Conf. on Weblogs and Social Media - ICWSM '13*. The AAAI Press, 2013.
- [29] C. Treude, O. Barzilay, and M.-A. Storey. How do programmers ask and answer questions on the web? In *Proceeding of the 33rd Int'l Conf. on Software engineering - ICSE '11*. ACM, 2011.
- [30] B. Vasilescu, A. Serebrenik, P. Devanbu, and V. Filkov. How social q&a sites are changing knowledge sharing in open source software communities. In *Proc. of the 17th ACM Conf. on Computer Supported Cooperative Work, CSCW '14*, pages 342–354, New York, NY, USA, 2014. ACM.