Predicting Answer Quality Using Post and Domain-Based User Information

Lorraine Fan, Alexander Hsu, and Christina Kao

Abstract

Stack Overflow has a well-established reputation system that gives users incentives to ask and answer questions, as well as to evaluate content generated by other users. In this study, we predicted whether an answer in a specific domain, Java, would receive at least two net upvotes from other users, and whether an answer would be accepted by the user who had posted the question. Our feature set consisted of answer post-specific measures, answerer's history in the predicted domain, and the answerer's history in the domain most related to the predicted domain. We obtained test accuracies of 70 to 80%, and identified that a set of just three features—response order, number of answers in the question, and the relative rank of the number of accepted answers of the answerer among those who responded to the same question—performed nearly as well as a full set of features. We also discussed the importance of accounting for competition among answers to the same question.

Motivation

We predicted whether an answer in a specific domain on Stack Overflow would receive at least two net upvotes from other users, and whether an answer would be accepted by the user who had posted the question. The general interpretation of this prediction was, given a question in the specific domain, how likely the answerer would give a quality response.

Past research on Stack Overflow has suggested that post-specific predictors, such as the arrival time of the answers, are associated with reactions to answers.[1] Other research on another developer forum has proposed that user-specific characteristics are positively correlated with human assessment of domain expertise.[2] We explored whether proxies for the domain expertise of answerers are also associated with the reactions to answers.

With respect to user expertise, we hoped to improve the Stack Overflow's current reputation model, which relies on various user actions that may not necessarily relate to actual skill levels. In particular, we believed user expertise can be better proxied by certain user history measures in the domain of the question, and user history in the most related domains.

One possible application of our prediction problem is system recommendation or highlighting of high-quality answers to enhance the knowledge creation process of the website. Another application is to gauge whether identifying the level of domain expertise of each user is viable in a peer-assessment environment, after the characterization of domain expertise is refined. Such an application may be useful for assessing online participation in classes.

Data

We obtained a complete trace of all the actions on Stack Overflow from its inception on July 31, 2008 to September 6, 2013, which is publicly available at the website. We processed the data using SQLite3 in Python.

We considered each hashtag to be a domain and focused on predicting reactions to answers in the "Java" domain, which is among the top five most active domains on Stack Overflow. Descriptive statistics on this domain are presented in Figure 1.

Figure 1.Statistics on "Java" Since Website Inception[3]

Total Users	Total Questions	Total Answers	Total Accepted Answers	Average questions per users	Average answers per user	Average accepted answers per ser	Average score per user (up/downvotes)
230,859	458,254	458,254	39,873	1.98	4.09	0.17	8.02

Because we wanted to explore using answerers' history in related domains as features, we looked into cross-domain activity. As shown in Figure 2, the number of users decreases exponentially when the number of associated domains increases. Among the 1,011,197 users in the 20 most popular domains, 484,714 users have activities in only one domain, and 213,223 users have activities in two domains.[3] Consequently, we decided to explore the one domain most similar to Java, which is Android (Figure 3).

Figure 2. Distribution of number of associated domains[3]

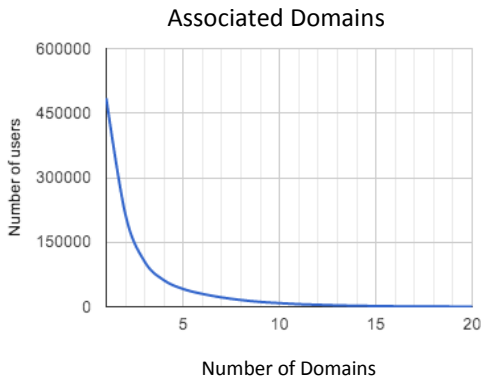
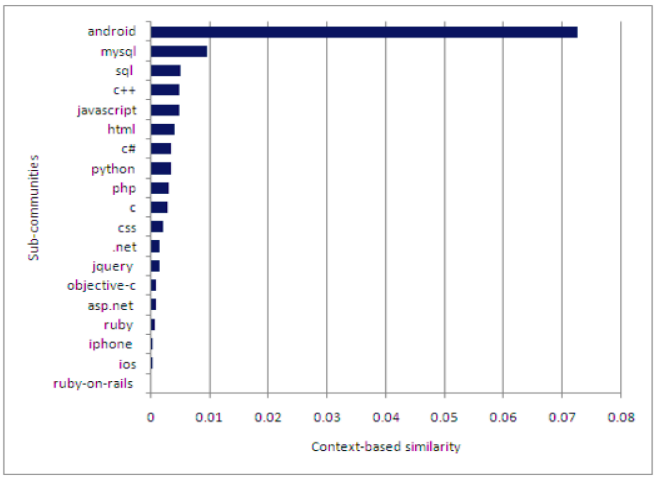


Figure 3. Domain Similarity based on Intersecting Questions[3]



Features

The three sets of features included were answer post-specific measures, answerers' history in the predicted domain (Java) between July 31, 2008 and March 5, 2013, and answerers' history in the most related domain (Android) (Figure 4). For answerer history, we tabulated the history between July 31, 2008 and March 5, 2013 of 789,032 unique answer providers, who formed 2,025,647 user-domain pairs.

Figure 4. Three sets of features

Answer Post-specific measures	Answerer's history Java between 7/31/08 and 3/5/13	Answerer's history in Android
○ Response time ○ answer edited or not ○ number of comments on the answer ○ number of comments on the question ○ question page's view count ○ question's favorite count ○ question's answer count	○ Number of answers ○ number of questions ○ number of net upvotes from answers ○ average number of net upvotes per answer ○ percentage of answers receiving at least 2 votes ○ number of answers accepted, percentage of answers accepted ○ number of not accepted answers, number of net upvotes from questions ○ number of answers with negative net upvotes, number of questions with negative net upvotes	Same as second column

Algorithms

We ran logistic regression, support vector machine, naïve Bayes logistic regression, and linear discriminant analysis using scikit-learn in Python, as well as in R's base, e1071, e1071 and MASS packages, respectively.

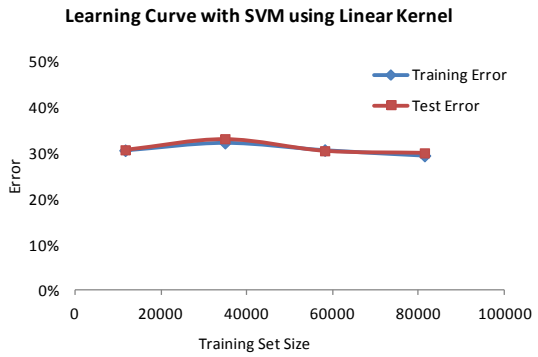
Results and Error Analysis

Approach 1

For our first approach, we predicted whether each answer posted between March 6, 2013 and September 5, 2013 would receive at least two upvotes using answerers' history in Java only. The actual proportion of answers that met the upvote threshold was 42.3%.

As shown in Figure 5, Training a SVM with a linear kernel from 11,632 to 81,420 examples, the training and test error rates were approximately 30% across the board (the radial basis / Gaussian kernel reduced the error rates by about 4 percentage points). The uniformly high rates of error suggest that our model suffered from high bias, which in turn suggested that more features were needed.

Figure 5. Learning Curve with user-in-Java history features only, SVM using Linear Kernel



Approach 2

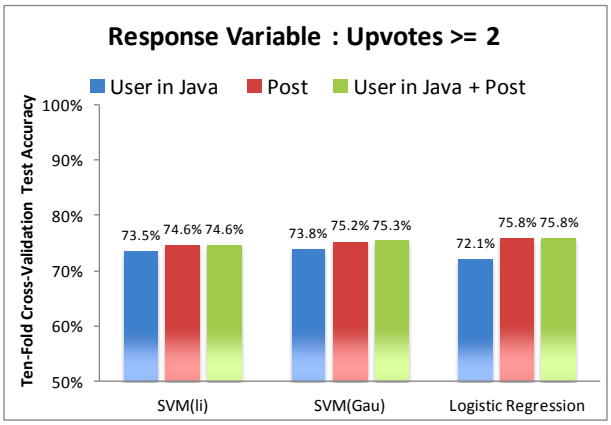
For our second approach, we predicted whether each answer posted between March 6, 2013 and April 5, 2013 would receive at least two upvotes or be accepted using answers' history in Java only. Concentrating on valuable questions, which we defined as questions with at least 200 views, the data set diminished to 4,216 examples. The actual proportions of answers that met the upvote threshold and were accepted were 29.1% and 26.8%, respectively.

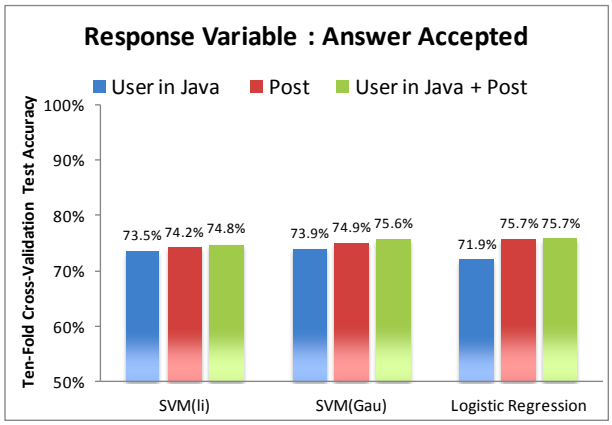
We analyzed feature sets with 1) answerers' history in Java only, 2) answer post-specific measures only, and 3) combining both. Within user history measures, we noticed that three measures about the answerers' past history in asking questions had 90% zero entries and were worsening our predictive performance. Therefore, the three features were removed. It has been found in past research that StackOverflow experts rarely ask questions but are active in answering questions.[1] With the remaining features, we checked the stepwise AICs of logistic regression models, which suggested that further removing features might not be useful.

For both response variables, ten-fold cross-validation test accuracies using SVM and logistic regression remained around 75% (Figures 6 and 7). Post features delivered higher test accuracy than did user features. Moreover, the addition of user features to post features did not have much influence on the outcome. We believed post features captured much of what user features had to offer. There was a clear issue that insufficient positive test examples were classified as such, and the issue was more serious from SVM than from logistic regression.

On a separate note, our model worked better for answers of questions with higher view counts, reaching approximately 85% test accuracy.

Figures 6. Predicting answer receiving at least two upvotes: Test accuracy
Figure 7. Predicting answer accepted: Test accuracy





Approach 3

In our last approach, we focused on predicting answers being accepted. As we were predicting which answer got selected as the best answer by the user who asked the question, we eliminated all answers that did not have a question with an accepted answer. Furthermore, we removed all answers that were posted after an answer had already been accepted because each question could only have one accepted answer. To mitigate the shrinkage in the sample size, we expanded the prediction period to include all answers posted in March 6 to May 5, 2013 for questions posted during the same period. Concentrating on valuable questions, which we defined as questions with at least 200 views and had accepted answers within the same time frame, we were left with 1,041 samples. Within this constrained sample, 70.8% of answers were accepted.

We introduced new features after realizing that, in order for us to better predict whether an answer would be accepted or not, we needed to take into account the presence of other answers under the same question. For example, user A might have really good background history in the domain and really fast response time, but that did not mean his answer would necessarily be accepted because he had to be compared with other answerers. User B might have even better background history and even faster response time. Therefore, where applicable, for each existing feature which pertained to user history in Java or the post, we added a within-question rank feature. Importantly, within each question, we ranked each answerer's number of accepted questions in the past to represent the answerer's relative standing among its "competitors" for the question. We also ranked the response time of each answer within each question because response time itself was extremely right-skewed, even after taking log.

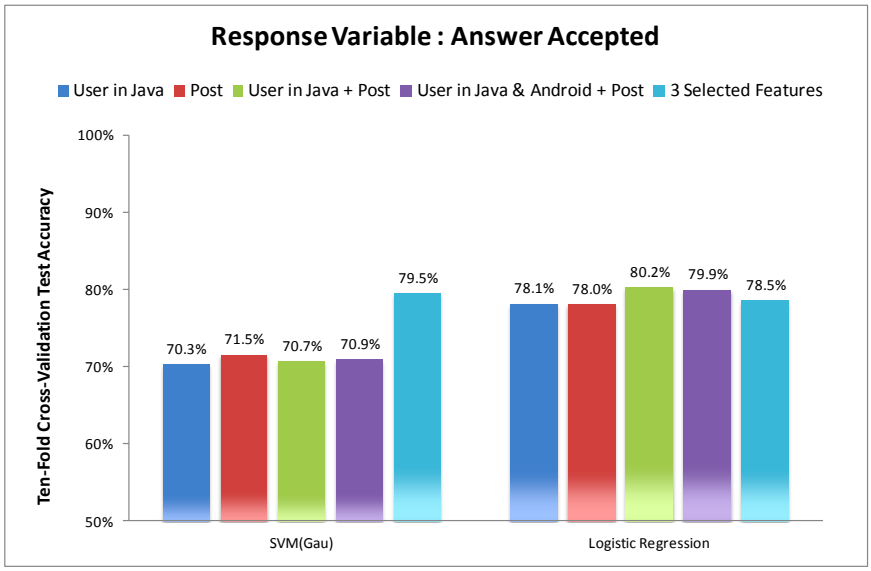
We analyzed feature sets with 1) answerers' history in Java only, 2) answer post-specific measures only, 3) combining user-in-Java and post-specific measures, 4) combining user-in-Java, user-in-Android, and post-specific measures, and 5) our stepwise search result of combining response order, number of answers in the question, and relative rank of the number of accepted answers of the answerer among those who responded to the same question.

Figure 8 shows that, with logistic regression, combining user-in-Java and post-specific measures yielded the highest test accuracy of 80.2%. However, it was only marginally better than the 78.5% obtained with our selection of only three features. Given the proximity of training and test errors, we believed the problem of high bias had persisted. Naïve Bayes and linear discriminant analysis results were very similar to those from logistic regression.

SVM with a Gaussian kernel and a cost parameter of 3 painted a somewhat different picture. The model was overfitting, obtaining 100% training accuracy but 71% test accuracy with large sets

of features (the second to fourth sets aforementioned). Test accuracy benefitted from reducing the number of features; it improved substantially to 79.5% with our selection of only three features.

Figure 8. Predicting answer accepted with original and rank features: Test accuracy



For both logistic regression and SVM, we observed that false positives were the biggest source of our test error, especially for SVM. We believed a flaw of our approach was that we allowed the model to predict acceptance for more than one answer for the same question, as it performed the classification of each example independently from that of other examples.

Conclusion and Future Work

For this project, we came up with several approaches in tackling the two predicting tasks and obtained reasonable results. During our efforts in improving performance in the last approach, we discovered several more complex problems in predicting whether an answer would be accepted or not. If we not only looked at features of an answer post and the answerer's history but also kept in mind its "competitors," we could improve our results significantly. A similar approach could be applied to the upvotes predicting problem given the reasoning that users would probably only vote once under the same question because casting a vote is not free in Stack Overflow. Although we incorporated rankings within each question, our approaches remained to suffer from not taking account into competition constraints such as "only one answer can get accepted in a question." Finally, for future work, we propose to examine features related to the contexts of the questions and answers, such as body lengths and word associations, to improve performance.

References

[1] A. Anderson, D. Huttenlocher, J. Kleinberg, and J. Leskovec. Discovering value from community activity on focused question answering sites: A case study of Stack Overflow. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2012.

[2] J. Zhang, M.S. Ackerman, and L. Adamic. Expertise networks in online communities: Structure and algorithms. *Proceedings of the 16th international conference on World Wide Web*, 2007.

[3] C. Chan, A. Hsu, and C. Yea. Status differentiation between sub-communities in stack overflow. 2013.