Eliciting Answers on StackOverflow

CS294-1: Behavioral Data Mining

Derrick Cheng, Michael Schiff, Wei Wu
derrick@eecs.berkeley.edu, mschiff@berkeley.edu, weiwu@berkeley.edu

May 11, 2013

Abstract

Using data provided from StackOverflow, we attempted to learn which features of a question make it most likely to be answered. Initially features were limited to only those found in the question itself, and which were known at the time the question was asked. Additional experiments attempted to improve classification results by expanding the feature set to include features of the question author, and by modifying the definition of what it means to be an answered question. Classifications were made with four different algorithms, Logistic Regression, Random Forests, Support Vector Machines, and K-Nearest Neighbors.

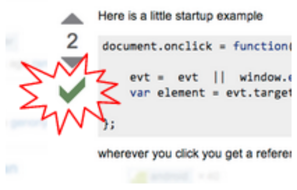
1 Introduction

StackOverflow, and the entire StackExchange network, is among the most popular question and answer-based forums. StackOverflow specifically focuses on a wide range of topics in computer programming. It is a rich community with over 1.8 million users who have together contributed over 4.5 million questions and 8.75 million answers. Valuable questions and answers receive votes from users. The respective authors of these posts gain reputation on the site, and can earn badges as experts of certain topics. Fast response times and high quality answers attribute to StackOverflows success and popularity as a Q&A site [9].

As with any type of forum, one needs to craft their question to increase the probability that they receive an acceptable answer. We aim to help StackOverflow users maximize this probability by learning which features of a question are most indicative of a question with an answer. This means that we are primarily interested in the features of a question that are available at the time the question is asked. This restriction can be loosened, without compromising the use-cases of our classifiers, by including features of the author at the time the question was asked. Other features, which may be predictive, were ignored, as those which are not available at the time the question is asked, can not be used to help the author better their question.

2 Related Work

All user-generated data is licensed under a Creative-Commons license, making StackOverflow the center of past work related to collective action and reputation systems. Posnet et. al mined longitudinal data on StackOverflow to understand how reputations vary over time and found that users do not gradually become more skilled at answering questions, despite computer-mediated communication theories that suggest the contrary [11]. Hanrahan and Convertino create a model for question difficulty and user expertise and found little correlation between harder questions and the involvement of more expert users [8]. StackExchange themselves have used a considerable amount of effort making predictions about questions on their site. This work has largely gone into detecting questions that are off-topic, or are likely to be marked as closed. A StackExchange-sponsored Kaggle competition sought to crowd-source an algorithm capable of making these predictions [5]. Perhaps the most related work thus far to our project is Nasehi et. al's analysis of what makes for good answers on StackOverflow [10]. No research has, however, tried to predict which questions will be answered, something of far more use to the typical user asking a question. For this reason, we believe that there is a need for this type of predictive algorithm.



Here is a little startup example

```
document.onclick = function(

    evt =  evt  ||  window.e
    var element = evt.target

};
```

wherever you click you get a referen

Figure 1: An accepted answer

3 Data Extraction

The StackExchange network provides a window into their database in the form of their interactive Data Explorer [7]. This application provides the ability to run SQL queries on their data. We chose to use the data explorer, as opposed to existing dumps due to the ease of joining tables and selecting for desired attributes. This allowed us to extract only the aspects of the data related to our question. To label our data into ”answered” or ”unanswered,” we first defined an answered question as one that had an accepted answer. Using the Data Explorer we collected a set of posts which had an accepted answer, yielding a table of post ids and the label 1, and a set of posts which did not have an accepted answer, yielding a second table of post ids and the label 0. The union of these two tables is the set of labeled data. We join this labeled data back with the original posts on the post id field, and then select for all of the features we need, yielding a final table of labeled examples [2].

4 Labels and Features

4.1 Labels

We define a question to be answered if the original poster of the question returns and marks it as an Accepted Answer as can be seen in Figure 1.

4.2 Featurization

Our inital feature set comprises of different characteristics of the question itself. This totalled to about 14000 features. We used Java [3] in order to build this Matrix.

Words Counts

For each of the words in the question that is not in the code section, we counted up the number of times that word occurred in that question. We selected to only use the top 10,000 most frequent words in our whole corpus of questions.

Tag Counts

Every question can be tagged to be under specific categories such as Python, MapReduce, etc.. For each tag we mark its presence in the question with a zero or one. This totalled to about 7500 features.

Part of Speech Tag Counts

In order to identify part of speech tags of words in the question body text, we utilized CMU's LingPipe, a suite of Java libraries for linguistic analysis [4]. We specifically used the Brown Parser, which provides a parser for the NLTK distibution of the Brown Corpus [1]. This allows us to tag up to 98 different base tags such as apostrophes, sentence closers, adjectives, and prepositions. For each of the 98 features, we count up the number of occurrences per question.

Miscellaneous

Number of edits, time from creation to first edit, number of code examples, number of words in code examples, number of words in question body, number of sentences in question body

5 Classification

In order to determine if we could distinguish between questions with accepted answers and those without, we tried both linear and non-linear classifiers. We used four classification techniques: SVMs with RBF Kernel, Logistic Regression, K-Nearest Neighbors (specifically a 10-Nearest Neighbors), and Random Forests.

After going through a few machine learning tool kits we ultimately chose to use SciKit-Learn, which is a Python library for general-purpose machine learning [6].

One downside of using SciKit-Learn was that it could only reasonably handle a smaller amount of data than more scalable machine learning tools. It ended up being able to handle 10,000 training and testing examples quite well. However, we believe this amount was sizeable and sufficient enough to not overfit, because we had at max 14000 features.

5.1 Results and Analysis

Our entire dataset was 10,000 Questions, where 80% was used for training and 20% was using for testing. Training set and test set were randomly partitioned. In order to determine the impact of specific features in our feature set, we decided to test on three distinct subsets of our full feature set (as described in Section 4.2):

1. Part of Speech Tag Counts, Misc Features
2. Part of Speech Tag Counts, Misc Features, Tag Counts
3. Part of Speech Tag Counts, Misc Features, Tag Counts, Word Counts

In order to evaluate each model, we looked into the AUC, F1, Precision, Recall, and Accuracy of our models.

Classifier	Subset	Acc	Precision	Recall	F1	AUC
LR	#1	0.81	[0 0.81]	[0 1]	[0 0.9]	0.50
	#2	0.81	[0 0.81]	[0 1]	[0 0.9]	0.50
	#3	0.81	[0 0.81]	[0 1]	[0 0.9]	0.51
RF	#1	0.79	[0.21 0.81]	[0.04 0.96]	[0.07 0.88]	0.52
	#2	0.80	[0.23 0.81]	[0.04 0.97]	[0.07 0.89]	0.52
	#3	0.79	[0.22 0.81]	[0.04 0.97]	[0.07 0.88]	0.53
SVM	#1	0.81	[0 0.81]	[0 1]	[0 0.90]	0.49
	#2	0.81	[0.14 0.81]	[0 1]	[0.0013 0.90]	0.50
	#3	0.81	[0.25 0.81]	[0 1]	[0.0013 0.90]	0.50
KNN	#1	0.80	[0.19 0.81]	[0.02 0.98]	[0.04 0.89]	0.51
	#2	0.80	[0.17 0.81]	[0.02 0.97]	[0.04 0.89]	0.51
	#3	0.80	[0.22 0.81]	[0.02 0.98]	[0.04 0.89]	0.50

Table 1: Inital Results.

As can be seen from Table 1, the different subsets did not effect the values of the evaluation metrics.

In addition the AUC values of these classifications were close to 50%. The higher accuracies that we see of around 80% mainly may be attributed to the skew between positive and negative labels (+:8133/ -:1867).

The poor performance of the linear classifiers (i.e. Logistic Regression) implies that our data may not be linearly separable. This means that we may have to add more features in order to capture more information. The poor performance of the non-linear classifiers imply that our labels may not be correlated with features. Thus, in order to attain better results we may have to redefine what it means for a question to be answered.

We explore these two ideas in the following two experiments.

6 Experiment #1

We hypothesize that whether a question gets an accepted answer is dependent on the original poster returning to the post. In order to test this, we incorporate the last login time and the reputation of the author. We then ran the our 4 classifiers over the same 3 feature subsets as before, where we now include these user features into the miscellaneous feature category. We ran with 10,000 examples and a 80-20 Training-Test split.

Classifier	Subset	Acc	Precision	Recall	F1	AUC
LR	#1	0.86	[0 0.86]	[0 1]	[0 0.93]	0.58
	#2	0.86	[0 0.86]	[0 1]	[0 0.93]	0.57
	#3	0.86	[0 0.86]	[0 1]	[0 0.93]	0.5
RF	#1	0.84	[0.24 0.86]	[0.06 0.97]	[0.093 0.91]	0.57
	#2	0.85	[0.32 0.87]	[0.067 0.98]	[0.11 0.92]	0.58
	#3	0.85	[0.28 0.86]	[0.054 0.97]	[0.09 0.92]	0.57
SVM	#1	0.86	[1 0.86]	[0.018 1]	[0.0036 0.93]	0.53
	#2	0.86	[0.53 0.87]	[0.026 1.0]	[0.049 0.93]	0.54
	#3	0.86	[0.29 0.86]	[0.019 1.0]	[0.035 0.92]	0.53
KNN	#1	0.85	[0.36 0.87]	[0.09 0.97]	[0.14 0.92]	0.59
	#2	0.85	[0.32 0.87]	[0.064 0.98]	[0.11 0.92]	0.57
	#3	0.50	[0.35 0.87]	[0.073 0.98]	[0.12 0.92]	0.57

Table 2: Results for Experiment #1 with User features.

As can be seen from Table 2. The AUC scores did improve, but the change was minimal. This slight change is not enough to suggest a strong correlation between user information and receiving an accepted answer.

7 Experiment #2

We hypothesize that the accepted answer flag is not a good definition for an answered question. This is implied by the lack of dependence between the label and the features of the question as seen in Table 1.

Thus we redefine whether a question is answered to whether it receives an answer with more than a threshold of X votes. In our experiments, we set X to be 10. We again run with all 3 original feature sets for each of our 4 classifiers again with 10,000 samples and a 80-20 Training-Test split. For the training set, 5185 were positive and 2815 were negative.

Classifier	Subset	Acc	Precision	Recall	F1	AUC
LR	#1	0.65	[0.65 0]	[1 0]	[0.79 0]	0.40
	#2	0.48	[0.74 0.38]	[0.32 0.79]	[0.45 0.51]	0.60
	#3	0.65	[0.65 0]	[1 0]	[0.79 0]	0.40
RF	#1	0.66	[0.68 0.53]	[0.89 0.23]	[0.77 0.32]	0.61
	#2	0.68	[0.70 0.57]	[0.89 0.28]	[0.78 0.37]	0.64
	#3	0.67	[0.68 0.59]	[0.93 0.18]	[0.78 0.28]	0.63
SVM	#1	0.65	[0.65 0.36]	[0.99 0.0071]	[0.78 0.01]	0.55
	#2	0.65	[0.65 0.29]	[1.0 0.0014]	[0.79 0.0029]	0.56
	#3	0.65	[0.65 0.063]	[1.0 0]	[0.79 0.00071]	0.56
KNN	#1	0.62	[0.68 0.43]	[0.80 0.28]	[0.73 0.34]	0.57
	#2	0.61	[0.68 0.42]	[0.76 0.32]	[0.72 0.36]	0.56
	#3	0.61	[0.68 0.43]	[0.77 0.32]	[0.72 0.36]	0.57

Table 3: Results for Experiment #2 using new labels.

The results show an overall significant improvement as compared to our other tests. Random Forests received the best results with an AUC of 0.64 (Figure 2 and Figure 3), as compared to 0.52 (Table 1) that we received using our original accepted answer definition.

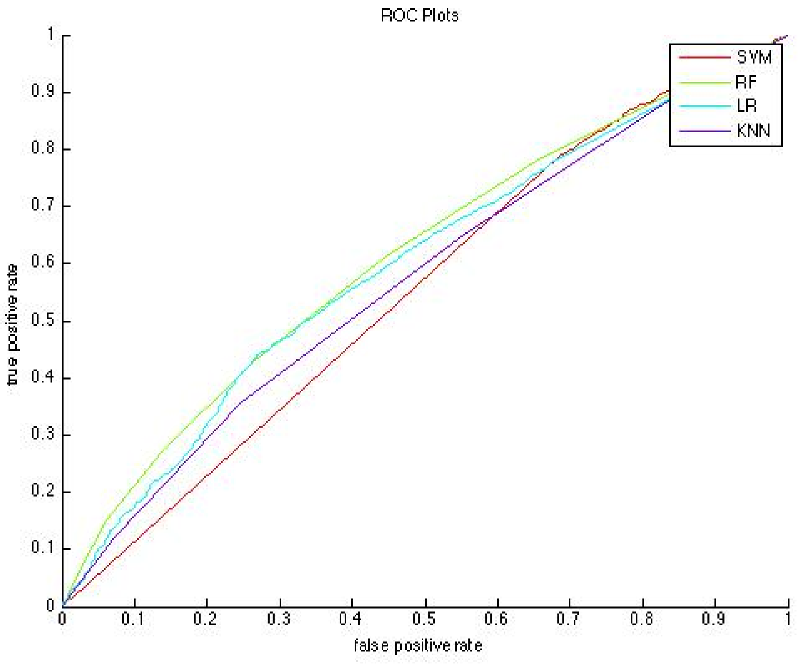


Figure 2: Comparison of Classifier ROC Plots for Experiment #2

This provides promising indication that the content of question may serve as a signal for question

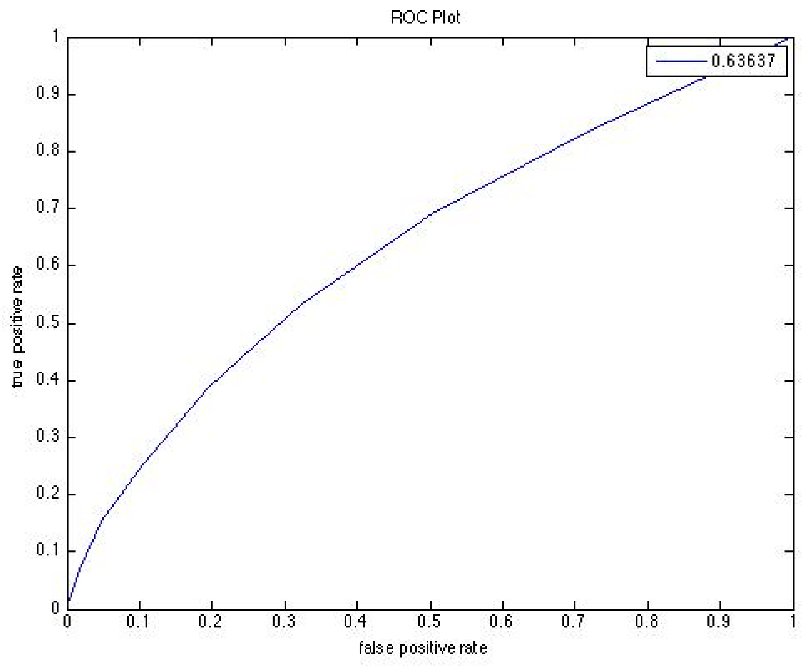


Figure 3: Random Forest ROC Plot for Experiment #2

answerability.

8 Future Work

The lack of success with our current iteration of classifiers has lead us to two explanations for our failure, both of which point to possible improvements we can make moving forward. Two aspects of the question that were not captured in this project but may be indicative of the likelihood of a question being answered are (1) question visibility and (2) question difficulty. Currently, our classifiers do not make use of either of these pieces of information. Question visibility was discounted by choice, as it is not a feature that is accessible at the time the question is asked. Question difficulty was ignored because the data was not labeled according to difficulty, and partitioning it as such would have been the work of another project. Going forward we could use these features to condition our data, such that we are learning with data that is consistent in terms of question visibility and difficulty. This would allow us control for these two confounding factors, so that we can isolate features of a question itself that most indicative of getting an answer. This would also allow for a more fine grained result, as we could see which were the indicative features at various levels of question difficulty and visibility.

9 Lessons Learned

From this project, we were able to understand the difficulty of feature selection to make a problem classifiable as well as the importance of selecting a good label. Once we changed our labels, our results were much improved. In addition, we realized the importance of data extraction. We believe that if we had limited our data to only questions that were of the same difficulty or of the same visibility, then our results could potentially be improved. We were also able to firsthand experience how increasing feature size dramatically slowed down our classifiers.

References

[1] Brown corpus. http://icame.uib.no/brown/bcm.html.

[2] Data extraction query. http://data.stackexchange.com/stackoverflow/query/105553.

[3] Java. http://www.java.com/en/.

[4] Lingpipe linguistic analysis. http://alias-i.com/lingpipe/.

[5] Predict closed questions on stackoverflow. http://www.kaggle.com/c/predict-closed-questions-on-stack-overflow.

[6] Scikit learn. http://scikit-learn.org/stable/.

[7] Stack exchange data explorer. http://data.stackexchange.com.

[8] Hanrahan, B. Convertino, G. N. L. Modeling problem difficulty and expertise in stackoverflow. *CSCW (Companion)* (2012), 91–94.

[9] Mamykina, L. Manoim, B. M. M. H. G. H. B. Design lessons from the fastest q&a site in the west. *Proceedings of CHI 2011* (2011).

[10] Nasehi, S. Sillito, J. M. F. B. C. What makes good a good code example? *Proceedings of 28th IEEE International Conference on Software Maintenance* (2012).

[11] Posnet, D. Warburg, E. D. P. F. V. Mining stack exchange: Expertise is evident from initial contributions. *ASE/IEEE International Conference on Social Informatics* (2012).