CS 446 : Project Report
Predicting Tags for Stack Exchange Questions

Aditya Rastogi
University of Illinois
at Urbana-Champaign
arastog2@illinois.edu

Chaitanya Datye
University of Illinois
at Urbana-Champaign
cdatye2@illinois.edu

Chinmay Kulkarni
University of Illinois
at Urbana-Champaign
ckulkarn@illinois.edu

1 Introduction

Identifying tags or keywords from a collection of text has been a very important application of text analysis and data mining that has come up in recent years. In case of Question Answer websites, such as Stack Exchange, Quora, etc, identifying the tags belonging to the questions which facilitates flexible grouping and fast item retrieval allows the users to explore more related content, choose specific topics that interest the user and easily contribute towards those questions that are related to the users expertise. Also, it has been widely observed that tagging posts provides a better user experience and easier management of the data.

StackExchange is a popular network of question and answer websites where users can post questions and label them with different tags that are relevant to the topic or content of the question. Within the network there are websites devoted to answering questions on different topics ranging from Computer Science , Mathematics to Films and Science Fiction. These tags basically allow a particular user to choose which questions are typically of interest to him and need to be followed, or which questions the user can contribute to by providing answers based on the area of expertise.

In this project, we have chosen a large dataset consisting of question and answers aggregated from several stack exchange websites. This dataset comes from the Facebook Recruiting Challenge III that was hosted on Kaggle.com in 2013. As part of this project, we build an automated tagging system, which given a question, extracts the title and body of the questions and outputs a set of tags that the question can be categorized into. Our project focuses on different ways in which we can extract the features and also on the various classification algorithms that can be used in order to gain an insight into the effectiveness of these algorithms in such keyword extraction scenarios.

In this paper, we have discussed the various learning approaches taken in order to predict the class labels given a question. Initially we discuss various ways of data cleansing that were done in order to get reasonably clean data which was then used in our algorithms for training. We have compared three learning approaches, namely,Stochastic Gradient Descent, Logistic Regression and Support Vector Machines. All these algorithms were run in a multi-label scenario using their One-Versus-All variants. We have also experimented on an approach to filter out infrequent tags using the FP-Growth Association Rule Mining Algorithm. The intuition behind this is that we will get a more confident and relevant prediction if we were to limit the label space to frequently occurring tags. In this paper we aim at providing a comprehensive explanation of the various learning approaches that we have adopted, along with a comparison between all these approaches in both scenarios: when using frequent pattern mining methods, and when not using this method. We also provide certain interesting facts that were learnt from the data and our results that were obtained based on the classifiers we used in order to reach the desired conclusion. After analyzing our various approaches, we arrive at the conclusion that Stochastic Gradient Descent and

Support Vector Machines perform considerably better than Logistic Regression, both when using FP-Growth and without using FP-Growth.

2 Problem Definition and Algorithms

2.1 Task Definition

From the Kaggle competition dataset, we obtained a training set with 6,034,195 StackExchange posts that consisted of the title, body and the tags of the questions. The testing set contained 2,013,337 posts which had the title and body. Our job is to predict correctly the tags for these test questions. After our initial collection of data, we had around 7GB of text data. With the constraints of memory and time we used around 250,000 posts out of all these in order to study the data and to arrive at the first analysis of the data distribution.

Each input is of the form
$< Id, Title, Body, Tag >$ for the training set. The test set examples are of the form
$< Id, Title, Body >$ and we have to predict the set of tags. This is a pure Multi-label classification problem where we can have multiple tags for a particular question.

We used the F1 score as the evaluation method for our system. The reason is that in our problem, the accuracy of the binary classifier may not be a good indication of the effectiveness, since for any tag, the vast majority of examples should not be assigned that tag. Therefore, a trivial predictor that always predicts the top 5 frequent tags would in fact reach very high accuracy, even though it is practically useless. In cases like these, we must incorporate a notion of precision and recall rates for the predictor in order to find the overall utility of that predictor. Thus, an F1 score is a good measure to measure the test's accuracy since it uses both precision and recall values. The F1 Score is given by:

$$F1 = \frac{2 * \text{precision} * \text{recall}}{\text{precision} + \text{recall}}$$

We use the Mean F1 Score in order to measure the performance of our classifiers.

2.2 Algorithm Definition

Since the classification task involves multi-label classification with a large number of possible output classes, we used the One vs All multi-label classification strategy with linear classifiers.

In addition to different classifiers, we felt that the problem of predicting tags could also be modeled as a problem of finding association rules using frequent patterns mining of the tags and the data. The classifiers we used and the frequent pattern mining process is briefly described below:

2.2.1 SVM

SVMs are often used for document classification tasks and have been seen to perform reasonably well. Using the One Vs All strategy we trained the SVM classifier for every tag in the set of frequent tags for the dataset and predicted the tags for an example based on the distance of the example from the learned SVM classifiers for the respective tags.

2.2.2 Logistic Regression

Logistic Regression is another popular classifier that like Naïve Bayes', allows interpreting the output of the classifier as a probability however, unlike Naïve Bayes' can be used used as a linear classifier in the One vs All scheme.

2.2.3 Stochastic Gradient Descent

The Stochastic Gradient Descent classifier is a linear classifier that offers inexpensive computation and at the same time can be used as an online classifier.

2.2.4 Frequent Pattern Mining and Association Rule Mining

Frequent Pattern Mining involves finding values that occur together frequently in the data. FP growth and Aprioiri are two of the most popular frequent pattern mining algorithms that produce sets of frequent items for a given support threshold. For every frequent item-set (an item-set with support greater than the given threshold) all its subsets also figure in the list of frequent item-sets. As a result, frequent pattern mining can be used to discover tags that occur frequently together and therefore may be representative of a topic distribution that is commonly observed in the training data. We make use of this observation in preprocessing the output labels for the training examples by using the mined frequent patterns as class labels.

After discovering frequent patterns, the frequent patterns could be mined to discover association rules of the form

$$A \rightarrow B$$

where we say that B occurs frequently together with A, whenever A occurs with a significant support. To predict the tags for a new question then, association rules of the form

$$\text{word in document} \rightarrow \text{tag}$$

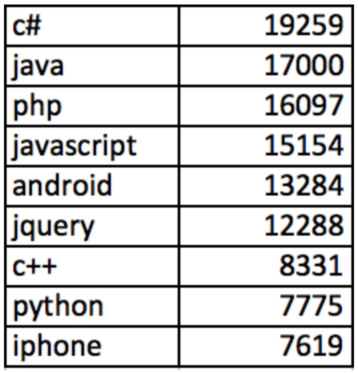
can be used to predict the tags for the question.

3 Experimental Evaluation

3.1 Methodology

3.1.1 Data Analysis and Preprocessing

Our initial analysis over the training data concluded in giving us 25,334 total tags. Top 10 tags sorted by their frequencies are depicted in Figure 1. These top 10 tags account for about 17.26% of all the tags, the top 100 tags constitute roughly 40% of all the tags. We also plotted the distribution of top



c#	19259
java	17000
php	16097
javascript	15154
android	13284
jquery	12288
c++	8331
python	7775
iphone	7619

Figure 1: Top 10 tags and their frequencies

500 tags and we can clearly see from Figure 2 that this follows a power law distribution. We also performed the tag frequency computation on the title and body separately which illustrates the distribution of top tags that were obtained from the title and the body. These results have been shown in Figure 3 and Figure 4 respectively. We can see that these 2 graphs clearly are in line with our top tag distribution and are seen to follow the power law distribution as well. Each question has at most 5 tags.

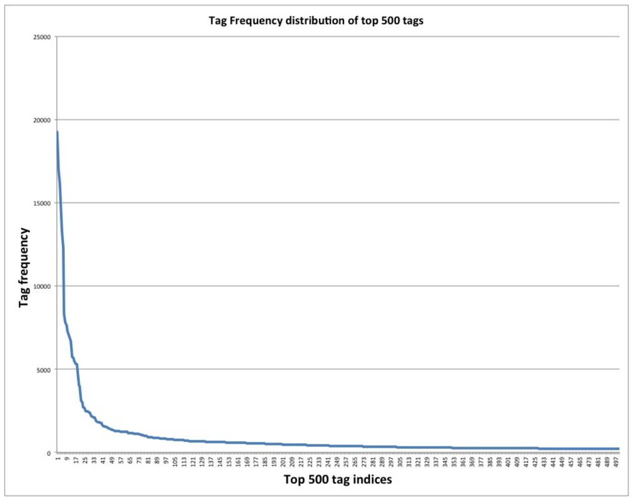


Figure 2: Distribution of top 500 tags

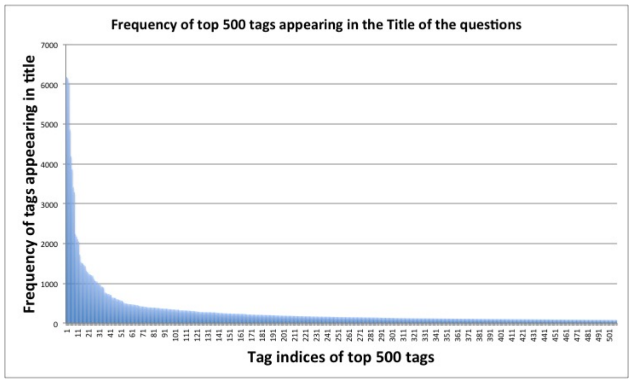


Figure 3: Distribution of top 500 tags obtained from the title of the posts

We also analyzed the distribution of number of tags across the data. We observed that the number of tags has been roughly modeled by a normal distribution with mean 2.9 and standard deviation of 0.73. Observations are summarized below in Figure 5.

3.1.2 Experimental Methodology

Running experiments on the full data set was computationally very expensive , so we had to conduct our experiments on four samples from the dataset of sizes 5000, 10000, 20000 and 40000. We ran the three classifiers on each dataset first without the class labels pruned and then with the class labels pruned with frequent pattern mining.

We preprocessed the question title and body text to produce TF-IDF weighted vectors for the questions in the training and test dataset.

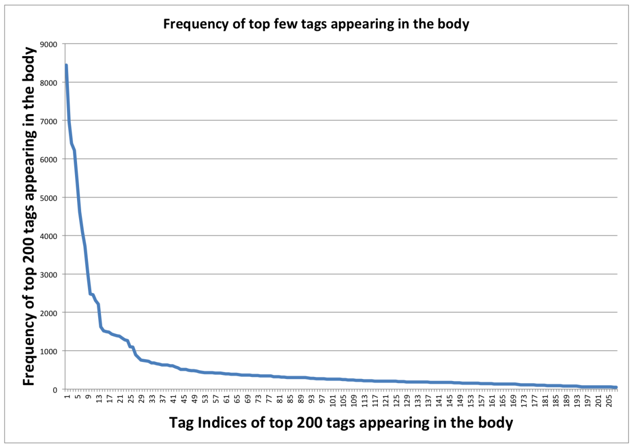


Figure 4: Distribution of top 208 tags that occurred frequently in the body of the posts

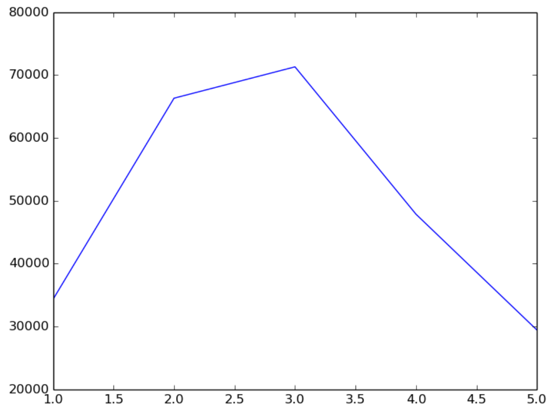


Figure 5: Distribution of number of tags across posts

To arrive at the class labels to keep, we observed from the distribution of questions across tags that 97 percent of the questions in the training data were labelled with at least one of the top 1812 tags (with support count > 50) out of a total of nearly 25,334 tags that were present in the training data. Therefore we decided to keep only these 1812 tags and removed from the dataset all those examples where none of these tags appear in the label.

For the second set of experiments, we used the One-vs-All scheme to perform multi-class classification, where we treated each frequent pattern of combination of the tags as a different class label since we expected the class labels thus created to be closer to the true labels that would be assigned to the example. With this scheme, for each training example we assigned the training example the class label of the largest frequent pattern of tags that matched with the true tags of the examples.

The following configurations were used for training the classifiers

Classifer	*Parameters*
SVM	C=0.8, L2 loss
Logistic Regression	L2 loss, C=0.8
SGD classifier	L2 loss

In addition to the above linear classifiers, we also used an Association Rule based predictor where we predicted the tags by looking for association rules that matched the words in the title and body of the text.

3.2 Results and Discussion

In our first approach, we used the three classifiers SVM, Logistic Regression and SGD where we trained these as a Multi-class One vs All classifier over the tags, and picked the top 3 tags from the predictions obtained. We choose top 3 because, as we have seen earlier, the tag length or the number of tags per question follows a normal distribution with a mean roughly 3. For each classifier, we calculate the F1-score of the classifier over dataset of 5000, 10000, 20000 and 40000 examples. We use 80% of the samples as training data and 20% as the evaluation set. The following performance graph (Figure 6) was obtained as a comparison of these classifiers. From the figure, we can see that the SVM and SGD perform significantly better over Logistic regression in all datasets. Following are some of the predictions that the SVM makes on the test data:

Actual: ['python', 'arrays', 'numpy'] $\rightarrow$
Predicted: ['numpy', 'arrays', 'python']
Actual: ['android', 'activity'] $\rightarrow$
Predicted: ['java', 'android', 'eclipse']
Actual: ['css', 'optimization', 'web', 'smartphone'] $\rightarrow$
Predicted: ['c#', 'css', 'html']

As we can see, the above tags predicted are reasonably accurate for the first and second case. But, since we are only predicting the top 3 tags, if a particular question has less than 3 or more than 3 tags,

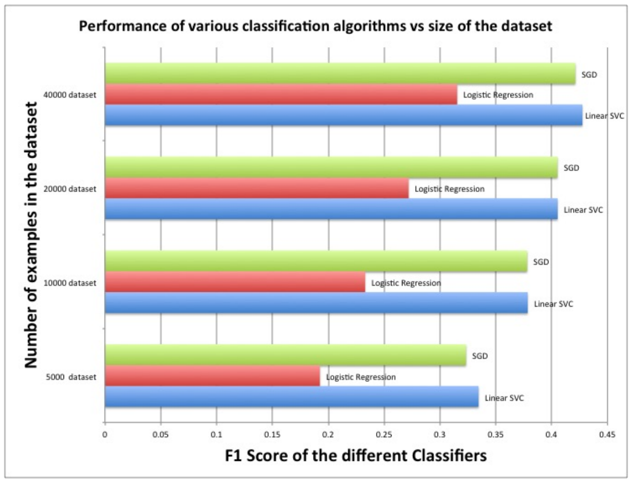


Figure 6: Performance comparison of SVM, SGD and Logistic Regression

our prediction accuracy goes down. For example, in the third case where there are 5 tags in actual, the classifier ends up predicting some irrelevant tags. Similar results have been obtained with the other two classifiers.

In order to provide more relevant predictions over the questions and eliminate the constraint of choosing the top 3 classes, in our second approach, we used FP Growth algorithm to find out the frequent patterns amongst the tags. Using this we predicted only the maximal length frequent item-set consisting of the tags predicted by our classifiers. Figure 7 depicts the results of the performance comparison when the three classifiers were used coupled with he FPGrowth algorithm.

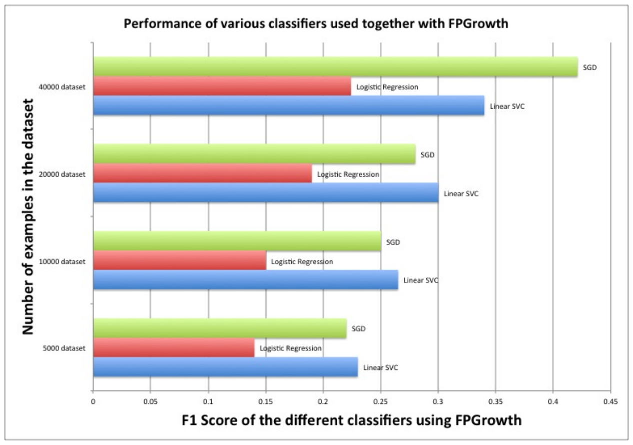


Figure 7: Performance comparison of SVM, SGD and Logistic Regression coupled with FPGrowth

Following are some of the predictions that SVM makes on the test data when used coupled with FP-Growth:

Actual: ['python'] $\rightarrow$ Predicted: ('python',)

Actual: ['android'] $\rightarrow$ Predicted: ('android',)

Actual: ['python', 'version'] $\rightarrow$ Predicted: ('python',)

We can see that the predictions in case of this approach are more relevant even though they are less in number. This is because we cannot generate full frequent item sets. So, the false positives in this case are less.

Apart from these approaches, we used association rule mining, and used the Apriori algorithm to predict the tags. We got an F-1 measure of around 0.41 which was similar to SVM and SGD as seen above. Following are some of the predictions made by the Apriori algorithm

Actual: ['nhibernate', 'fluent-nhibernate'] $\rightarrow$

Predicted: ['fluent-nhibernate', 'nhibernate-mapping']

Actual: ['xaml', 'richtextbox', 'paragraphs'] $\rightarrow$

Predicted: ['c#', 'wpf', 'xaml', 'richtextbox', 'paragraph']

4 Future Work

The problem of predicting tags for newly posted questions on stack-overflow or any question answering site in general is an interesting and relevant problem and an automated system to do this can significantly improve the quality of answers that questions receive, while at the same time allowing experts, community moderators to easily identify questions of their interest and categorize them. In this paper we suggested several different approaches to solve this problem and through experimental evaluation demonstrated that a combination of frequent pattern based approach and a Naïve Bayes' classifier achieves the best results.

The inability to model a question as arising from a mixture of topics where the topics are not derived from the text of the documents but from an external set of labels. We tried to explore several alternative using topic modelling approaches such as pLSA and LDA but could not come up with a convincing EM model that builds on this approach. Future work could concentrate on attacking the problem from this direction.

We found out that tags which occur less than 50 times in our entire data set of 250,000 posts that we considered for our experiments, led to only 8663 questions. So, in order to gain a better classification and reduce the rate of false positives, we straightaway ignored these questions which corresponded to the very less frequent tags.

We also found out the word distribution across the data which gave us the frequency of each word occurring in the dataset. Figure 6 shows us the word distribution graph. Based on which we decided to eliminate certain useless words and thus gain a more clean dataset which we used for our experiments. The questions consisted of code enclosed in the $< code >< /code >$ tags which was removed. All the stop words were eliminated, all the html tags were eliminated. We thus got a cleansed data which consisted of relevant words that would be required to predict the tags for a given particular question.

References

https://www.kaggle.com/c/facebook-recruiting-iii-keyword-extraction

Y Song and Roth. 2014. *On Dataless Hierarchical Text Classification*. AAAI, 2014.

Gabrilovich E and Markovitch S. 2007. *Computing Semantic Relatedness using Wikipedia-based Explicit Semantic Analysis*, IJCAI 2007.

Ramage, Daniel, David Hall, Ramesh Nallapati, and Christopher D. Manning. 2009. Labeled LDA: A supervised topic model for credit attribution in multi-labeled corpora. *In Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, Volume I - pp. 248-256, Association for Computational Linguistics, 2009.

Rubin, TimothyN., Chambers, America, Smyth, Padhraic, Steyvers, Mark. 2012. *Statistical topic models for multi-label document classification.*. Machine Learning, 88, 157-208

Dell Zhang and Wee Sun Lee. 2003. *Question classification using support vector machines.* In Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval (SIGIR '03), 88, 157-208.

Landauer, Thomas K., Peter W. Foltz, and Darrell Laham. 1998 *An introduction to latent semantic analysis.* Discourse processes 25.2-3 (1998): 259-284.

Thabtah, F.A., Cowling, P. Peng, Y, 2004 MMAC: A New Multi-class, Multi-label Associative Classification Approach, paper presented to Proceedings of the 4th IEEE International Conference on Data Mining, ICDM '04.

1999 McCallum, A. (1999), 'Multi-label text classification with a mixture model trained by EM', paper presented to Proceedings of the AAAI' 99 Workshop on Text Learning.