Keywords Extraction from StackExchange Posts

Bhargavi Chalasani
School of Informatics and Computing
Indiana University Bloomington
bchalasa@indiana.edu

Sachin Kariyattin
School of Informatics and Computing
Indiana University Bloomington
skariyat@indiana.edu

Abstract

Tags or labels are important for a post or document to understand it's relevance to a topic and for easy retrieval of information. In this paper, we try to predict the tags for a StackExchange post so that users in the community can gain better access to the information in the post. Since each post can have multiple tags, we modeled this as a multi-label classification problem and tried to predict the best possible tags for a question.

1 Introduction

Tagging is important to have the huge set of Q&A data in an organized format and it provides a mechanism for the user to easily access the relevant topics. The success of tagging a body of text to identify related content started a trend where more and more sites began to implement it. Today tags can be found on a variety of websites like Twitter, Youtube, internet forums, etc. Stack Exchange is an online question-and-answer community that covers topics in specific disciplines; for instance, Stack Overflow, that is part of this community, is a very popular computer programming question-and-answer site.

In this paper, our objective is to build a model for tagging Stack Exchange questions spanning across math, programming and non-programming topics with keywords which help classify them into meaningful categories. We try to solve the multi-label classification problem where each item (in this case, question) can belong to multiple classes (tags). Prediction of the tags (or keywords) for a Stack Exchange post will be done by analyzing only the title and the body of the question.

The dataset that we have used for this project contains content from disparate stack exchange sites like StackOverflow, Mathematics Q&A, Linguists Q&A, Science Fiction and Fantasy, Cooking, Movies & TV containing a mix of both technical and non-technical questions. We have tried to predict tags for the posts using the following models: Multinomial Naïve Bayes, Bernoulli Naïve Bayes, Linear Support Vector Classification, and Stochastic Gradient Descent Classifier.

2 Related Work

A tag is a word or phrase that describes the topic of the question. Tags are a means of connecting experts with questions they will be able to answer by sorting questions into specific, well-defined categories. Tags can also be used to help you identify questions that are interesting or relevant to you. Figure 1. Shows an example post from StackOverflow website which is tagged with specific tags.

Fig 1. Showing the tags for a StackOverflow post

Tagging is also important to have the huge set of Q&A data in an organized format and provides a mechanism for the user to easily access the relevant topics.

Tagging also plays an important role in Search Engine Optimization (SEO) of the website. Tags effect the page rank of the website and also helps the visitors and web crawlers understand the content of each web page.

Prior research has been done on the task of predicting tags for Stack Overflow questions. [Kuo 2011] used a cooccurrence model that predicts tags based on tokens extracted from the question title and body, and the cooccurrence to their tags. Online tag recommendations was one of the tasks in the ECML PKDD Discovery Challenge 2009 and the winning solution [Lipczak et al. 2009] used a 3 stage tag recommender with the second and third stages modeling title ↔ tag and tag ↔ tag relationships using graphs. Tags were predicted by combining scores from the three stages. [Stanley and Byrne 2013] used an ACT-R based Bayesian probabilistic model to predict hashtags based on the author of the post. The idea was that the tags used by an author are based on their history of prior use and the strength of association between the tag and the content of words in the title and body of the post. An offset measurement (mean of the top five tags for the post) was used to equalize the top activated tags across posts. This was used to predict one tag per post on average.

3 Dataset

The dataset consists of posts from StackExchange website which can be obatained in .csv format through Google Big Query or the StackExchange data explorer (http://data.stackexchange.com). Currently, the StackExchange website contains over 13,000,000 questions from StackOverflow and more than 4,000,000 questions from other StackExchange websites combined. We decided to consider a subset of 100,000 questions from the available data. The split of the dataset for different domains is given below

Domain	**Questions**
StackOverflow	50,000
Mathematics	10,000
Linguists Q&A	10,000
Cooking	10,000
Movies	10,000

Table 1. Split of posts from different domains

We composed the following query to fetch the Id, Title, Body, Tags of a Post where none of the fields in the posts are empty

SELECT Id, Title, Body, Tags FROM Posts WHERE Title != ' ' AND Body != ' ' AND Tags != ' '

We used the same query to get the data for all the domains and finally we combined the data to form a single .csv file of 108 MB.

3.1 Exploratory Analysis

The tag frequency for the top 20 tags is shown in fig 2.

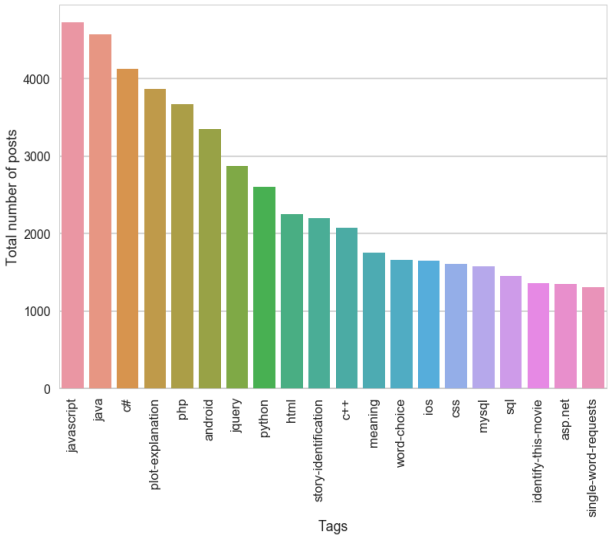

Fig 2. Frequency of top 20 tags

It can be observed that most of the top tags are from technical domain which is expected as StackOverflow data has the most number of questions in the considered dataset. We can also see tags like plot-explanation, meaning and single word request which are from different domains that we had considered.

We also plotted the correlation between the top 20 tags (fig 3)

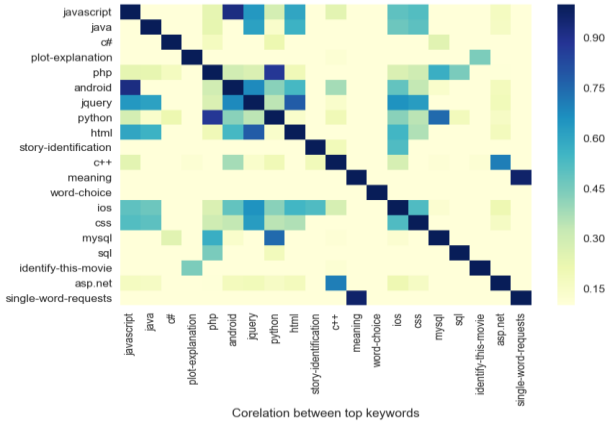

Fig 3. Correlation between top 20 tags

As it can be seen from fig 3, most of the top 20 tags are correlated i.e., they tend to appear together in many of the questions.

The maximum number of tags that a question can have in a StackExchange website is 5. Fig 4 shows the tag length of the posts from 1 to 5 tags

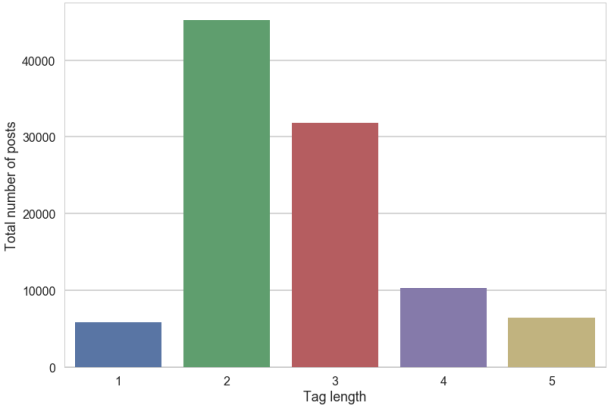

Fig 4. Number of posts with different tag length

From fig 4, we can see that most of the posts have 2 or 3 tags.

The posts in the actual StackExchange site are labelled either automatically (when user types a question) or manually (user has an option to manually add the tags to a question).

4 Methods

We choose 4 models viz. Multinomial Naïve Bayes, Bernoulli Naïve Bayes, Linear Support Vector Classification, and Stochastic Gradient Descent Classifier. Predicting tags given the body and title of the question involved preprocessing of text as well.

4.1 Preprocessing

The title and body of the text were combined into a single string and the tags for each question were converted to a list of tags. The <code> part of the body was removed using the Beautiful Soup package. Even though this information would have been useful in some cases, we believed we can ignore this part since we are considering non-technical data as well.

4.2 Features and Models

We used Tf-Idf vectorizer to get the important features. Tf-Idf is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus. The Tf-Idf value increases proportionally to the number of times a word appears in the document, but is often offset by the frequency of the word in the corpus, which helps to adjust for the fact that some words appear more frequently in general (Wikipedia). We initially worked with one-gram as a parameter to the Tf-Idf vectorizer but we noticed that considering one-grams, two-grams and three-grams together gave better results.

We used supervised methods for the prediction tasks. The dataset was split into training and testing set in the ratio 75:25 respectively. Since, predicting the number of tags a post contains is in itself a major task, we have set the number of tags that we are predicting to be equal to 5. To implement multilabel classification we used sklearn.OneVsRestClassifier. This classifier compares each class to all other classes by considering them as a single unit. The results obtained from this classifier are very interpretable. The description of the machine learning models that we used is given below. All the models were used from scikit-learn package.

4.2.1 Multinomial Naïve Bayes

The multinomial naive Bayes model is typically used for discrete counts. E.g., if we have a text

classification problem, we can take the idea of bernoulli trials one step further and instead of 'word occurs in the document' we have 'count how often word occurs in the document', you can think of it as "number of times outcome number x_i is observed over the n trials"

We used the sklearn.MultinomialNB for the prediction task with parameters like 'alpha' and 'fit prior'.

4.2.2 Multi-variate Bernoulli Naive Bayes

This binomial model is useful if the feature vectors are binary (i.e., 0s and 1s). One application would be text classification with a bag of words model where the 0s 1s are 'word occurs in the document' and 'word does not occur in the document'. So basically this model is used if we are just interested in the presence or absence of a particular word in the document.

sklearn.BernoulliNB was used for the prediction task with parameters like 'alpha' and 'fit prior'.

4.2.3 Linear Support Vector Classifier

Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier (Wikipedia)

Sklearn.LinearSVC was used for the prediction with parameters like 'penalty', 'loss' and 'dual'.

4.2.4 Stochastic Gradient Descent Classifier

It is a stochastic approximation of the gradient descent optimization method for minimizing an objective function that is written as a sum of differentiable functions (Wikipedia).

Prediction was done using Sklearn.SGDClassifier with parameters like 'penalty', 'alpha' and 'iterations'

4.2.5 Parameter Tuning

Optimal parameters for each classifier are obtained using the GridSearchCV which is available in scikit learn package. Grid-search is a way to select the best of a family of models, parametrized by a grid of parameters. The best parameters that we obtain are stored in a python pickle file and then they are used to classify the posts in the testing set.

5 Evaluation Metric

We used f1 score as a success metric for our evaluation. The F1-score, commonly used in information retrieval, measures accuracy using the statistics precision p and recall r. Precision is the ratio of true positives (TP) to all predicted positives (TP + FP). Recall is the ratio of true positives to all actual positives (TP + FN). The F1 score is given by:

$$F1 = 2(p.\ r) / (p+r)$$

$$\textit{Where Precision, } p = TP/(TP+FP)$$

$$\textit{Recall, } r = TP/(TP+FN)$$

The F1 metric weights recall and precision equally, and a good retrieval algorithm will maximize both precision and recall simultaneously. Thus, moderately good performance on both will be favored over extremely good performance on one and poor performance on the other.

We have used the weighted F1-score from sklearn metrics. Weighted F1-score makes sense for a data which has huge class imbalance which is the case for distribution of tags in our posts dataset.

6 Results, Discussion, and Conclusion

Initially we ran all the 4 models on the subset of 50,000 data points and the results are shown in fig 5

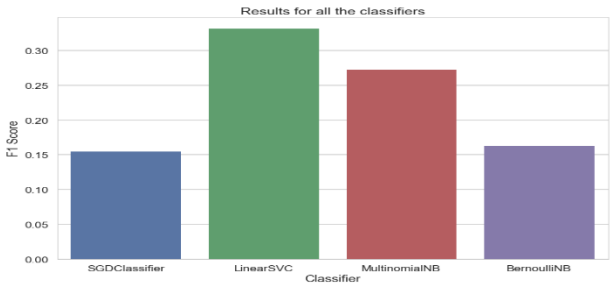

Fig 5. F1 scores for 50,000 posts

Classifier	F1-Score	Train-Time	Test-Time
SGDClassifier	0.154392	12113.401	365.323
LinearSVC	0.3313774	2805.665	359.251
MultinomialNB	0.2722954	333.671	605.435
BernoulliNB	0.1626471	602.734	584.115

Table 2. Results for 50,000 data points

It can be seen from the above table that the F1-score for Linear SVC is high for the initial analysis.

Further we ran the whole dataset of 100,000 data points on SGD and Linear SVC. We did not include Naïve Bayes classifiers as we faced memory issues while running the whole dataset. Fig 5 shows the results for the whole dataset

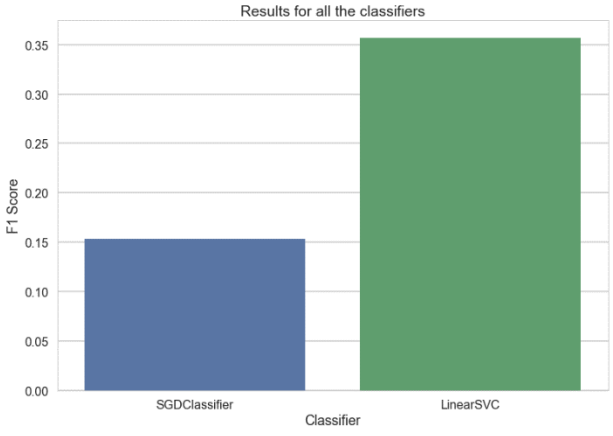

Fig 6. F1 scores for 100,000 posts

Classifier	F1-Score	Train-Time	Test-Time
SGDClassifier	0.1534597	28396.294	563.585
LinearSVC	0.3568806	8562.367	558.488

Table 3. Results for 100,000 data points

As one can see from Table 3 LinearSVC still performs better than SGD and the training time is very less compared to that of SGD.

6.1 Restricting the Number of Tags

As mentioned earlier, we were assuming the number of tags to be 5. We tried running the models by restricting the tags to 2 and 3, i.e. we ran our models on the subset of posts containing 2 and 3 tags. The results can be seen in fig 7 and fig 8 respectively

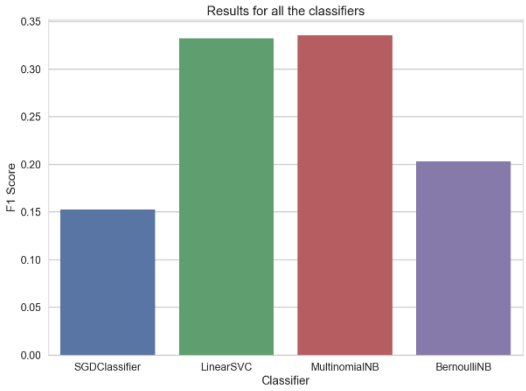

Fig 7. F1 scores after restricting tags to 2

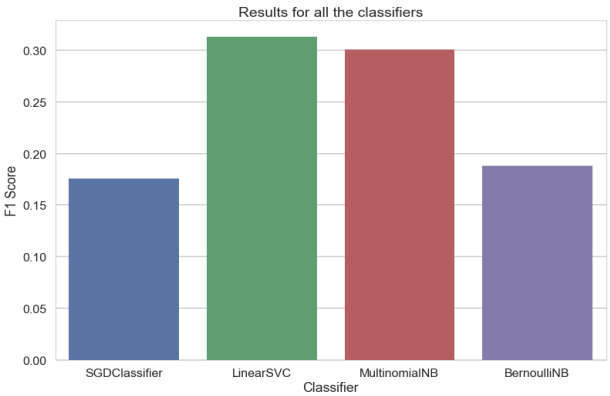

Fig 8. F1 scores after restricting tags to 3

The F1 score for MultinomialNB increased considerably well after restricting the number of tags.

6.2 Predicting only StackOverflow tags

We further wanted to see whether only considering data from one domain will make any differences to the F1 scores. So, we considered only those posts which belong to StackOverflow domain and predicted the number of tags. Again, here we had restricted the number of tags to 2 and 3. The results for StackOverflow data can be seen in fig 9 and fig 10

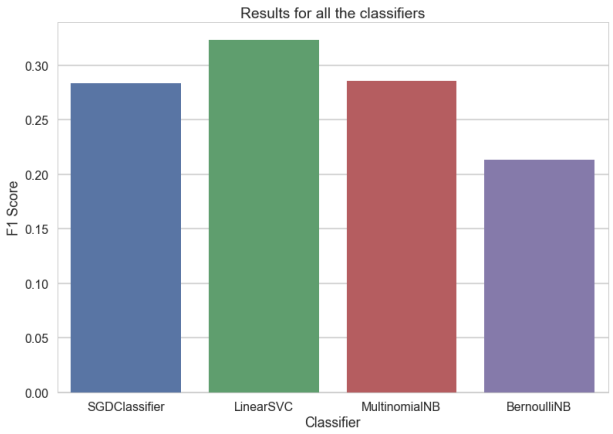

Fig 9. F1 scores for StackOverflow posts after restricting tags to 2

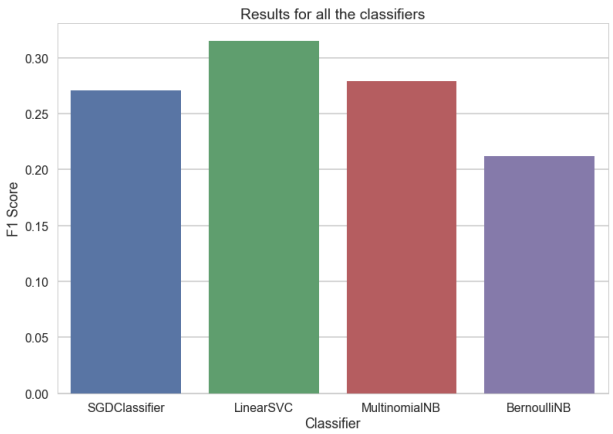

Fig 10. F1 scores for StackOverflow posts after restricting tags to 3

It can be seen in the above analysis that while predicting domain specific tags, the F1 score of SGD increased considerably. However, still LinearSVC gave the best results.

So, to summarize, Linear Support Vector Classifier gave the best results for all the datasets. Although SGD classifier showed an improvement when predicting tags when domain specific data is considered, Linear SVC still performed better.

6.3 Limitations

We wanted to implement more complex models like Perceptron for this prediction problem but could not finish it due to lack of better computing resources and time-constraints. We initially looked at few unsupervised learning algorithms like Latent Semantic Analysis to model the topics in the questions on a small subset of the dataset. The idea was to check if dimensionality reduction using LSA and applying classifiers on this sparse data will give better results than using Tf-Idf to get the keywords. Knowing the number of tags beforehand, rather than assuming the number of tags equal to 5, would have yielded better results. But we could not think of any methods to predict the exact number of tags that a post might have, given the body and title.

6.4 Future directions

In our experiments, we understood that correctly predicting the number of tags for a post is a difficult task on unseen data. Though we feel this is an understandable restriction we would like to see if this can be tackled by using association rules to get the family of keywords which are similar to a tag by using domain knowledge. In our paper, we noticed that specifying the number of tags beforehand to the classifiers give better results. We also observed that training the model specific to one category of questions (say programming vs cooking) yielded better results. We believe that training the model with more data points would increase the accuracy further. Also ,this model can be made more generic by adding a layer of domain knowledge.

7 References

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

[2] Creative Commons Data Dump. (2011). Retrieved from http://data.stackexchange.com/about

[3] Darren Kuo. 2011. On Word Prediction Methods. EECS Department, University of California,Berkeley. http://www.eecs.berkeley.edu/Pubs/TechRpts/2011/EECS-2011-147.html

[4] Marek Lipczak, Yeming Hu, Yael Kollet, and Evangelos Milios. 2009. Tag sources for recommendation in collaborative tagging systems. ECML PKDD discovery challenge (2009)

[5] Clayton Stanley and Michael D Byrne. 2013. Predicting tags for stackoverflow posts. In Proceedings of ICCM, Vol. 2013.