Stack Overflow Tag Prediction

Akanksha Nichrelay
University of Virginia
an9sx@virginia.edu

Arjun Malhotra
University of Virginia
am2cj@virginia.edu

ABSTRACT

Predicting the tags for user questions is an extremely important problem for the business model of the various Stack Overflow websites. Assigning the proper tags to these questions results in a more satisfying experience for both answerers and question-seekers. In this project we aim to use different machine learning models on the Stack Overflow questions to predict appropriate tags.

CCS CONCEPTS

• **Big data** → *Data Mining*; • **Unsupervised algorithm**; • **Machine Learning** → *Clustering*;

KEYWORDS

Natural Language Processing, Machine Learning, Tag Prediction, Multi-label classification

ACM Reference Format:
Akanksha Nichrelay and Arjun Malhotra. 2018. Stack Overflow Tag Prediction. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, Article 4, 3 pages. https://doi.org/10.475/123_4

1 INTRODUCTION

Tag prediction is an important problem for the business model of Stack Overflow. If the tagging is done correctly, Stack Overflow can determine who should the question be sent to for a correct and a fast response on the question. If Stack Overflow knows that Alice has answered a lot of question of C# in the past, i.e. it knows that Alice is an expert in C# , then using the tags it can send the question whose tag is predicted to be C# to Alice for an answer. This creates a great ecosystem, for getting a fast answer by sending the right question to the right person.

Our objective is, given the title and the description of a question on Stack Overflow, to build a model that will predict the tags associated with that question. We picked our data set from Kaggle contest [2] which comprises of a train set which is 6.75 GB approx. in size and has about 6 million rows. It has 4 columns - ID, Title, Body/Description and Tags. There is a separate test set provided but we plan to split the train set itself to get a test set with labels to verify our performance.

In the tag prediction scenario, we can have multiple tags associated with a question, so this is a multi-label classification problem. A question related to Java could also be related to Interfaces as well as Inheritance at the same time. So, we will have to choose our machine learning model and performance metrics accordingly.

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 123-4567-24-567/08/06.
https://doi.org/10.475/123_4

2 RELATED WORK

Stack Overflow allows users, who post questions, to enter up to five tags for a question. Entering all five tags makes a question more reliable and specific. While entering five tags is advantageous, most users find it easy to enter one or two tags but may not accurately enter other tags. In other related work, for the user-centered approaches - mining usage patterns from current users, collaborative ltering (CF) can be applied to suggest tags from users who share similar tagging behaviors [1]. Filtering step selects the best tags from those similar users for the recommendations. Drawback of this approach is obvious- a new user that does not have recorded history are unable to benet from this approach at all since the similarities with existing users cannot be calculated. More recently, there has been work on hybrid auto-tagging system for Stack Overflow [3]. The auto-tagging system includes a) programming language detection system b) SVM based question classification system. This system will suggest tags once a user enters a question. In our project, we plan to extend the idea of programming language detection system using LTSM in our future work.

3 PROPOSED METHOD/ EXPECTED OUTCOME

Since this is a multi-label classification problem, we plan to simplify it into a series of multi-class classification problems. We will train a one-versus-rest logistic regression classifier where we will train on each tag individually. In our train set, we have about 42K unique tags, which would result in a large number of classifiers. So we only use a subset of tags as our train labels which we determine based on our data analysis as explained in the next section. We will convert our title and description data into vectors using TF-IDF n-grams and Word2Vec when training different machine learning models. Currently we are have only used one-versus-rest logisitic regression classifer with TF-IDF n-grams vectors since linear classification model works well with sparse data sets. We will in future use Word2Vec with other machine learning models and also apply LSTM for our final presentation.

Ideally we would want our model to have high precision as well as high recall for all the tags. We want high precision in order to be certain that the tags actually belong to the question and we want high recall because if a tag is supposed to be present for a question, we want it to be present most of the times while predicting. If our model predicts an incorrect tag or if it misses a tag, that will result in a negative user experience. If the model predicts an incorrect tag then that damages our precision while missing on a correct tag is harmful for our recall.

Since this is a multi-label setting, to evaluate the performance of our models, we use mean F-1 score, which is also called micro average F-1 score. This is the weighted average of the F1 score of each class.

$$MeanFScore = F1_{micro} = \frac{2Precision_{micro}Recall_{micro}}{Precision_{micro} + Recall_{micro}}$$

where,

$$Precision_{micro} = \frac{\sum_{k \in C} TP_k}{\sum_{k \in C} TP_k + FP_k}$$

and

$$Recall_{micro} = \frac{\sum_{k \in C} TP_k}{\sum_{k \in C} TP_k + FN_k}$$

We will focus on F-1 micro as our key performance indicator for our evaluation.

4 EXPERIMENT

To start things off we performed data analysis. The dataset has 6,034,196 questions. Next, we removed the duplicate quesitons. There are 42,048 unique tags in our dataset. Some of the tags were '.a', '.app', '.asp.net-mvc', '.aspxauth', '.bash-profile', '.class-file', '.cs-file', '.doc', '.drv', '.ds-store'. In the figure below, we see the distribution of the top 100 tags by their frequency. The most frequent tag is ranked as 1, it appears 331,505 times over all the questions. This graph shows that not all the 42k tags appear in equal distribution. On further analysis we found that the top 500 tags cover appx 90% of the questions. Going forward we shall only use the top 500 tags to train our model.

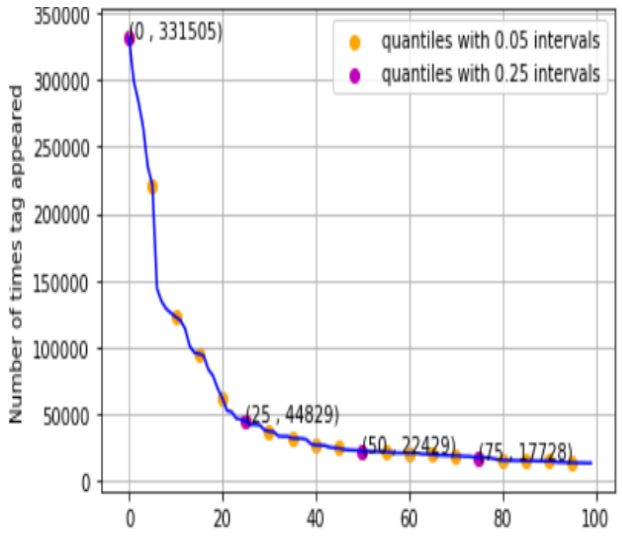

Figure 1: first 100 tags with respect to the number of times they appear in all the questions of the dataset.

4.1 Pre-Processing

For the step of pre-processing of the data set, we used 500 thousand questions as our train set because of the computation restrictions. Each row in our data set has a body column, which contains the description of the question and the code of some programming language relevant to the question. Since using LSTM on our code snippets - to predict the coding language tag - is a step for future work, while pre-processing the data set we separated out the code snippets from the body/description. We also removed special characters from the title and the body/description of the question. Further, we also removed the stop words. We made sure that we don't remove the character 'C' as one of our stop words. Next, we removed HTML tags, and converted all the characters to lower case. We used Snowball stemmer for stemming the tokens.

4.2 Featurization

For each question, we converted the tags into a binary vector. For this we restricted the number of tags to 500 because these 500 tags cover 90.956 % of the data set questions. To vectorize the questions we used TF-IDF vectors. In future we plan to incorporate Word2Vec to featurize our questions to keep semantic meaning intact. A couple of points worth mentioning here would be that while featurizing a question, we merged the title and the body/description (all the non-code text) of the question. As usually the title is more relevant and informative to the tag prediction, we gave three times more weightage to the title. Also, while converting the text into TF-IDF vectors we restricted the features to top 200,000 and we considered our n-grams in the range of 1 to 3.

4.3 Model

We used Logistic Regression (One-vs-Rest) with L1 norm. In future we plan to tune the hyper parameter as well using GridSearch.

5 RESULTS / EVALUATION

Since we used TF-IDF to vectorize our data set, we have a high dimensional vector to represent each each row. Keeping that in mind the model we used Logistic Regression for now. This will act as our base model. On applying Logistic Regression, we found the following results:

Table 1: Micro-average performance indicators

Precision	Recall	F1-Micro
0.7172	0.3672	0.4858

Table 2: Macro-average performance indicators

Precision	Recall	F1-macro
0.5570	0.2950	0.3710

As we can see that the recall of our model is quite low. We would want to increase that in the future work. Our precision is not that bad. We would also want to see a higher F1-micro. This result is

related to our proposal, as we already mentioned that our Key Performance Indicator (KPI) is F-1 micro.

6 CONCLUSION

The Logistic Regression will act as our base model and going forward we will try to improve our model. Few things we would like to implement in the future are: featurize our questions using the Word2Vec, apply LSTM for code recognition, apply more sophisticated models when we use Word2Vec. We also intend to tune the hyperparameter for our current Logistic Regression model, using Grid Search.

REFERENCES

[1] Grigory Begelman, Philipp Keller, and Frank Smadja. 2006. Automated Tag Clustering: Improving search and exploration in the tag space. In *Proceedings of the Collaborative Web Tagging Workshop at the WWW 2006*. Edinburgh, Scotland. http://.pui.ch/phred/automated_tag_clustering/

[2] Facebook. 2013. Facebook Recruiting III - Keyword Extraction. https://www.kaggle.com/c/facebook-recruiting-iii-keyword-extraction/data

[3] V. Smrithi Rekha, N. Divya, and P. Sivakumar Bagavathi. 2014. A Hybrid Auto-tagging System for StackOverflow Forum Questions. In *Proceedings of the 2014 International Conference on Interdisciplinary Advances in Applied Computing (ICONIAAC '14)*. ACM, New York, NY, USA, Article 56, 5 pages. https://doi.org/10.1145/2660859.2660970