StackOverflow Tag Predictor
Machine Learning Project Report

Navya Aggarwal
IIIT Delhi
navya18349@iiitd.ac.in

Nitin Gupta
IIIT Delhi
nitin18251@iiitd.ac.in

Sandeep Kumar Singh
IIIT Delhi
sandeep18363@iiitd.ac.in

I. Abstract

As the times moves, the amount of data present on the internet is increasing rapidly, and so is the data stored by the online education forums. Some of the common education forums are GeeksforGeeks, Stack Overflow, Chegg. Large-scale datasets for these websites are available online. We are trying to develop a system for Stack Overflow in which we are processing a publicly available dataset, to predict the correct tag by merely looking at the provided title and the body of the question. For this, we are using 6000 most frequently used tags in the available dataset.

Keywords *- Stack Overflow, tags, prediction, SVM, Logistic Regression*

Github Repository: https://github.com/itissandeep98/StackOverFlow-Tag-Predictor

II. Introduction

Stack Overflow is the largest, most trusted online community for developers, students, and other educationists throughout the world. It is an educational resource available to everybody to clarify doubts, ask questions, answer questions, vote questions, and answers up or down and similarly, post questions and answers like Wiki or Quora.

.

Stack overflow's popularity is due to its high response time from fellow users; this happens because when a question is posted then for every tag other users who have shown interest in those topics get notified. Also, users earn points which further help them gain some benefits on the website like the ability to comment or upvote or answer other questions. All of these play a crucial role in creating such a responsive and informative website.

For each question on StackOverflow, tags are used to relate different questions to categories. Tags are assigned to questions by users only. Further, users can subscribe to tags on a particular topic to receive digests of new questions for which they might be curious to learn and share knowledge. StackOverflow allows users to assign between one to five tags to a posting manually.

Even when an old or experienced user posts a question on Stack Overflow, he faces difficulty in tag identification which describes the question correctly. Most users use the terms present in the questions as the tags, but in most of the cases, it does not describe the question adequately. Due to this, some questions are given incorrect tags which do not represent their question.

The tag predictor can be used for the following scenarios :

1) All the questions belonging from the same topics can be grouped for the user to navigate easily.
2) We can show the post to the user related to the questions he or she is asking on the Stack Overflow

.

III. Literature Review

Many articles and similar projects are available on the internet, which gives us an insight into the tag assignment to a question or an answer. Students, educators, and computer science enthusiasts from all across the globe have contributed.

A paper titled 'Predicting Tags for Stack Overflow Questions Using Different Classifiers' was presented by Taniya Saini and Sachin Tripathi of ISM Dhanbad. According to the paper, relevant data extraction from the data was a crucial task, and text classification is a multi-class and multi-label classification problem.

The approach that was followed in this paper was to use a one-vs-rest classification method. In the one-vs-rest strategy, we fit one classifier per class. Each class is fitted against the rest of the classes. The different classifiers used in this paper were Logistic Regression, Naive Bayes, SVM.

Another paper titled 'Predicting Tags for StackOverflow Questions' by PhD students at Stanford demonstrates an approach which helps in building a predictor. According to the paper, tags for a question can be divided into two parts.

- Firstly, all questions can be broadly classified using a tag indicating the technology or the programming language they are related. This type of methodology to classify files according to language is also employed by GitHub.
- Lastly, the questions can be related to a subtopic or paradigm, and a separate classifier can be used to predict them.

This dividing approach can be used to predict tags more concisely. Thus, building classifiers for different types of classification can help predict labels with greater accuracy.

IV. Dataset

The dataset used is available at Kaggle as part of a past competition. The dataset has 60,34,195 data points, which are

Stack Overflow questions with their associate tags. For the project, we will be restricting ourselves to a subset of the dataset. Each data point in the dataset consists of 3 columns specifying question, title, body and the related tags for the question. The body contains the extracted HTML text from the web site.

Dataset Description	
ID	Integer
Title	String
Body	String
Tags	Space Separated List

A. Data Preprocessing

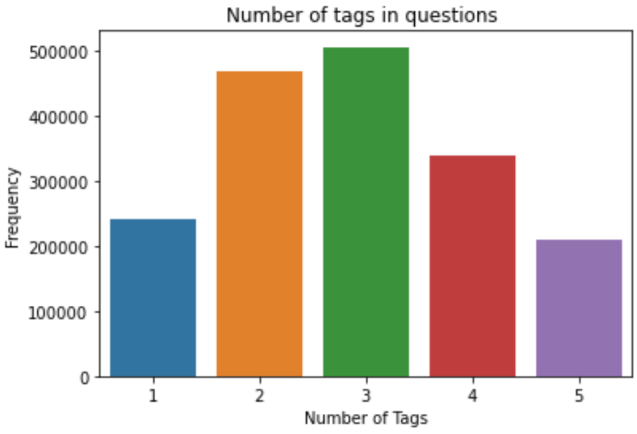
- **Remove HTML tags:** Since the data was directly scraped from the website, it contains HTML tags which are present in every data point and create noise and hence, needs to be removed.
- **Remove extra whitespaces and other special characters:** There were some user generated extra spaces and special characters which were contributing to the noise, and were removed.
- **Lowercase all texts:** To standardise the dataset, we converted all the text to Lowercase.
- **Convert numeric form to number words:** Since some questions contained numeric data which can't be processed in text processing. Hence, it was converted to number words.
- **Remove stopwords:** For text preprocessing words such as "as", "is", "a", "the" etc. creates noise and hence, were removed.
- **Remove duplicate entries:** There were duplicate data poits in the dataset, which would have affected our prediction later, and hence, were deleted.
- **Vectorization:** convert a list of tags into a set of columns denoting the presence of the tags.
- **Lemmatization:** Removing the inflectional endings and stored the base or dictionary form of a word.
- **Tokenization:** Breaks the raw text into words, sentences called tokens. These tokens help in understanding the context and developing the model.
- **Class normalization:** Converting all those words that resemble the same technology, e.g. converting "ReactJS", "React-JS", "React-Js", "react-JS" etc. to "reactjs".
- **Convert accented characters to ASCII characters:** Convert accented characters to its equivalent. For example, covert ė, ę to its ASCII equivalent "e".
- **Expand contractions:**Expand words like "don't", "can't" to "do not" and "can not" respectively.

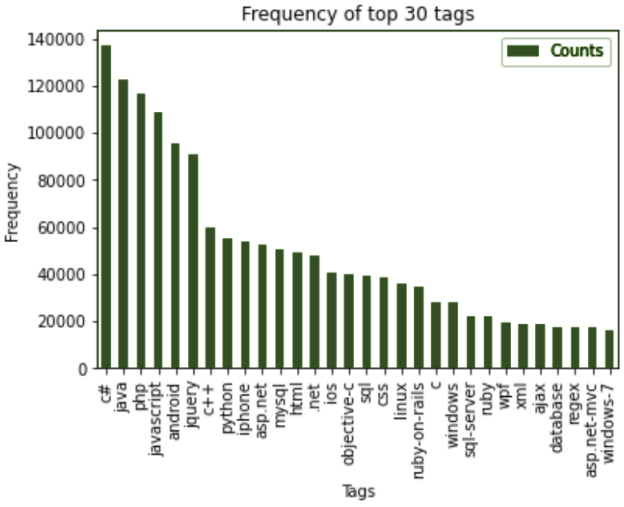
B. Data Visualization

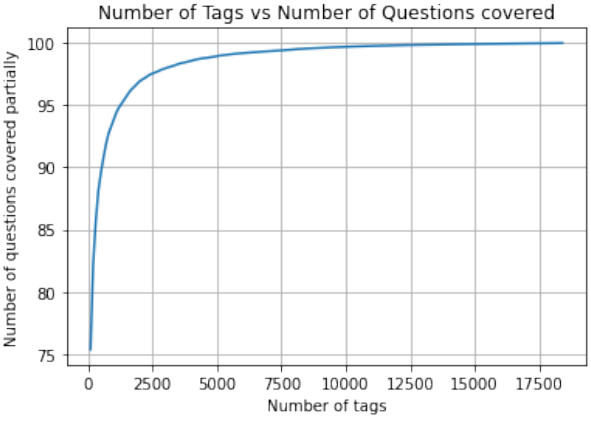
Initially, data size that we used for sampling was 2 million. Total of 232713 duplicate entiries was found. After removing duplicate entries we had 1767287 entries. We then plotted graph for frequency vs number of tags and frequency of tag of top 30 in the dataset.

1) Maximum Number of tags in any question - 5
2) Minimum Number of tags in any question - 1
3) Total Number of unique tags - 38427
4) Most Occuring tag - C#

We then analysed how many number of questions were covered with respect to the number of tags. We then plotted the graph for the same.







V. Methodology

A. Data Collection

Firstly we needed to collect data to start working. We visited the different website over the internet to find the data of our need. Finally, we were able to find the data on Kaggle.

B. Data Preprocessing

We then moved on to processing our data to remove unwanted symbols, spaces, duplicate entries etc. We applied multiple preprocessing techniques, as mentioned above in the data preprocessing section.

C. Working with Different Machine Learning Models

Next, we started training our model on the data set. We have vectorized both training data and class labels in the preprocessing of the data, and we are training our models on these data.

D. Classifiers

We ran all the models mentioned below for ten different data sizes, and then we reported Accuracy, Macro F1 score, Micro F1 score and hamming loss for each size. We then obtained following results.

1) Logistic Regression:
We used the sklearn library for implementing One-vs-Rest and logistic regression and grid search. We have applied grid search to find the optimum value of learning rate for each data size.We then obtained following results.

Size	Accuracy	Macro F1	Micro F1	H loss
10000	0.3732	0.1246	0.2352	0.0179
20000	0.4014	0.1615	0.2792	0.0167
30000	0.4018	0.1858	0.2983	0.0166
40000	0.3965	0.1915	0.2836	0.0168
50000	0.4187	0.2029	0.3123	0.0164
60000	0.4150	0.2106	0.3236	0.0163
70000	0.4089	0.2066	0.3166	0.0166
80000	0.4110	0.2154	0.3130	0.01668
90000	0.4110	0.2260	0.3201	0.0166
100000	0.4142	0.2269	0.3224	0.0165

Table 1: Analysis for Logistic Regression

2) SVM:
We used the sklearn library for implementing One-vs-Rest and SVM. We then obtained following results.

Size	Accuracy	Macro F1	Micro F1	H loss
10000	0.3807	0.3986	0.4431	0.0183
20000	0.4483	0.4058	0.4836	0.0156
30000	0.4591	0.3986	0.4893	0.0152
40000	0.4680	0.3897	0.4808	0.0147
50000	0.4826	0.3981	0.4955	0.0143
60000	0.4802	0.3981	0.4982	0.0142
70000	0.4693	0.3700	0.4785	0.0146
80000	0.4727	0.3828	0.4854	0.0146
90000	0.4779	0.3824	0.4900	0.0144
100000	0.4786	0.3787	0.4911	0.0143

Table 2: Analysis for SVM

3) Decision Tree:
We used the sklearn library for implementing One-vs-Rest and decision tree and grid search. We have applied grid search to find the optimum value of depth for each data size. We then obtained following results.

Size	Accuracy	Macro F1	Micro F1	H loss
10000	0.3496	0.3875	0.4282	0.0218
20000	0.3750	0.3970	0.4497	0.0202
30000	0.3788	0.3864	0.4489	0.0204
40000	0.3685	0.3726	0.4268	0.0210
50000	0.3834	0.3924	0.4476	0.0204
60000	0.3741	0.3992	0.4489	0.0205
70000	0.3720	0.3869	0.4470	0.0208
80000	0.3743	0.4002	0.4477	0.0209
90000	0.3778	0.4074	0.4563	0.0205
100000	0.3810	0.4095	0.4602	0.0203

Table 3: Analysis for Decision Tree

4) Bernoulli Naive Bayes: We used the sklearn library for implementing One-vs-Rest and BernoulliNB. We have applied grid search to find the optimum value of depth for each data size. We then obtained following results.

Size	Accuracy	Macro F1	Micro F1	H loss
10000	0.3061	0.1234	0.2435	0.0327
20000	0.3054	0.1967	0.3100	0.0374
30000	0.2842	0.2331	0.3304	0.0406
40000	0.2665	0.2513	0.3282	0.0435
50000	0.2601	0.2711	0.3395	0.0453
60000	0.2411	0.2881	0.3498	0.0464
70000	0.2294	0.2955	0.3579	0.0460
80000	0.2320	0.3026	0.3540	0.0481
90000	0.2085	0.3142	0.3635	0.0476
100000	0.2209	0.3126	0.3609	0.0487

Table 4: Analysis for Bernoulli Naive Bayes

VI. Result and Analysis

We evaluated our problem using 4 different score metrics namely Accuracy, Macro F1 score, Micro F1 score, hamming loss for diffferent datasize by setting tag size to 50. For more analysis, we set the dataset size to 50,000 and change the tag size from 50 to 200.

The plot of different accuracy of the models with respect to datasizes is given below.

A. Metric Analysis

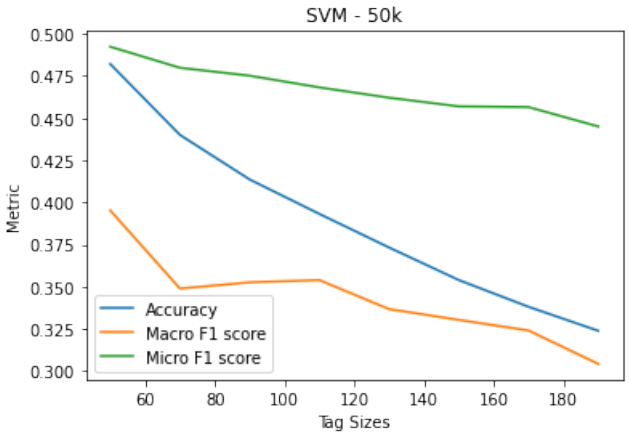


Fig. 1. Score metrics with respect to tag sizes

1) SVM: We plotted SVM model's accuracy, micro F1 score, macro F1 score with respect to different tag sizes(classes to be predicted) by taking a 50000 datapoints. A huge drop in Macro F1 score and accuracy was observed when we changed the tag size from 50 to 200 tags. But the Micro F1 score decrease is quite slow when compared with other two parameters.

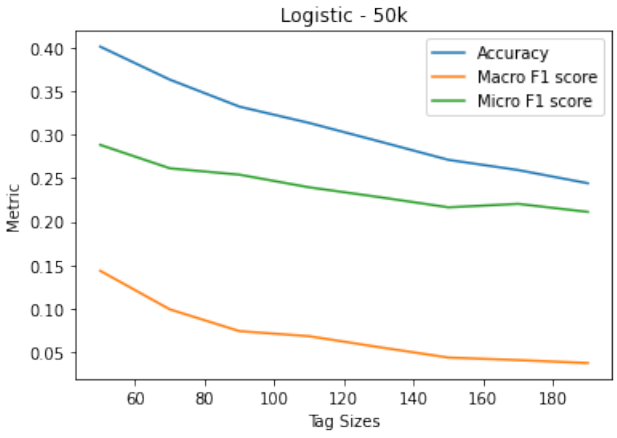


Fig. 2. Score metrics with respect to tag sizes

2) Logistic Regression: We then plotted Logistic Regression model's accuracy, micro F1 score, macro F1 score with respect to different tag sizes(classes to be predicted) by taking a 50000 datapoints. The value of Macro F1 score of the logistic regression model is quite low even for the 40 tags and its value keeps on decreasing as we are increasing the tag sizes. The Micro F1 score is quite stable as we are increasing the tag sizes.

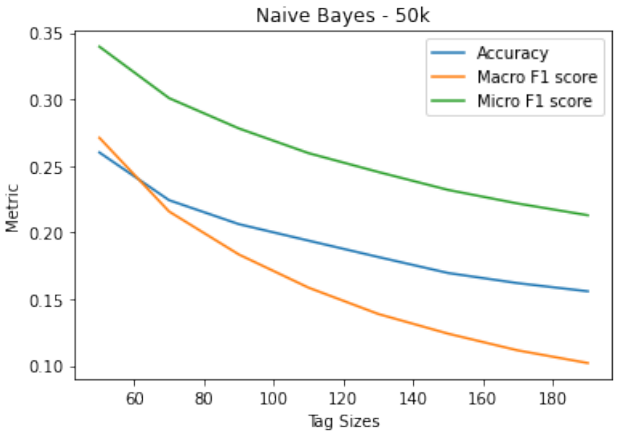


Fig. 3. Score metrics with respect to tag sizes

3) Bernoulli Naive Bayes: We then plotted Naive Bayes model's accuracy, micro F1 score, macro F1 score with respect to different tag sizes(classes to be predicted) by taking a 50000 datapoints. All the 3 score metric parameters of Naive Bayes are comparatively lower when compared to other models and these model even starts to go even lower as we are increasing the tag size. The Macro F1 score almost reached 0.10 as we moved the value of the tag size to 200.

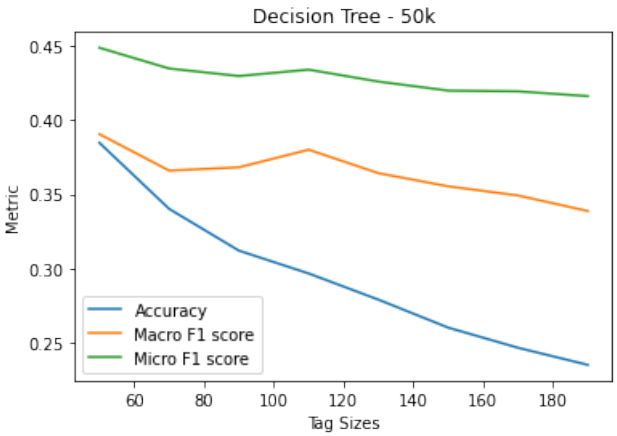


Fig. 4. Score metrics with respect to tag sizes

4) Decision Trees: We finally plotted Decision tree's accuracy, Micro F1 score, Macro F1 score with respect to different tag sizes(classes to be predicted) by taking a 50000 datapoints. A major drop in accuracy was observed which started from 0.38 and went down to 0.1832 as we keep on increasing the tag size. While the Macro F1 and Micro F1 score decline was much more slower as we changed the tag size.

Comparing all the above models, we can observe that SVM works in the best capacity, whereas, Bernoulli Naive Bayes

works as the worst when compared in the terms of accuarcy scoring metrics.

B. Accuracy over Different sizes of Dataset

We then plotted bar graph the accuracy of different models as we keep on increasing the size of the dataset.

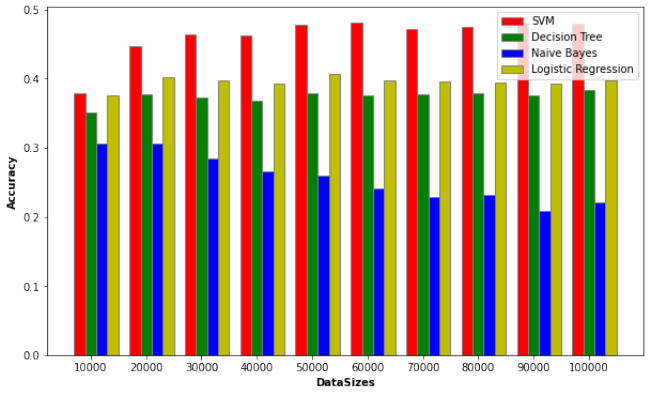


Fig. 5. Histogram of accuracy of different models is as follows

Observe the performance of different models over datasets of different sizes and different tag sizes we get following results

- We can see that SVM tends to perform better in all the scoring metrics that we have calculated as we are increasing the size of the dataset.
- The performance of SVM is increasing as we are increasing the size of the dataset.
- Performance of Decision Trees and Logistic Regression is independent of size of the dataset.
- For Bernoulli Naive Bayes, the performane degrades when the size of the datasize increases.
- As we are increasing the number of tags, all the three scoring metrics are decreasing at a rapid rate for all the models on which we have worked.

VII. Conclusion

A. Learnings from the project

- How to handle multi-label and multi-class dataset.
- Various text pre-processing techniques like lemmatization, tokenization etc. work.
- New ways to visualize and understand large datasets using graphs and plots.
- New methods to analyze errors in prediction. For example: hamming loss.

B. Contribution

- **Nitin**: Literature review, data visualization, feature extraction, working with Support Vector Machine(SVM), report documentation.
- **Sandeep**: Data visualization, data preprocessing, feature extraction, working with Logistic Regression, report documentation
- **Navya**: Literature review, data preprocessing, working with Bernoulli Naive Bayes, working with Decision Tree, report documentation.

C. Future Work

- We are planning to try this on other Q&A based sites like GeeksforGeeks, Chegg, Quora etc.
- We will further increase the accuracy using other different classifiers with varying hyperparameters.
- We will create an interface for the users so that users can enter the text and the app will provide the tags to users.

References

[1] **Predicting tags for stack overflow questions using different classifiers** - https://ieeexplore.ieee.org/document/8389059

[2] **Predicting Tags for StackOverflow Questions** - http://cs229.stanford.edu/proj2013/SchusterZhuCheng-PredictingTagsforStackOverflowQuestions.pdf

[3] **Predicting tags for the question in StackOverflow** - https://medium.com/datadriveninvestor/predicting-tags-for-the-questions-in-stack-overflow-29438367261e

[4] **TagStack: Automated System for Predicting Tags in StackOverflow** - https://www.researchgate.net/publication/338370635_TagStack_Automated_System_for_Predicting_Tags_in_StackOverflow

[5] **Sklearn One Vs Rest Classifier** - https://scikit-learn.org/stable/modules/generated/sklearn.multiclass.OneVsRestClassifier.html

[6] **Sklearn Support Vector Machine and Logistic Regression** - https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDClassifier.html

[7] **Sklearn Bernoulli Naive Bayes** - https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.BernoulliNB.html