Disclaimer This project is work of students of IIT Kanpur who are mentioned in the document and any part of it has not been copied from somewhere else or is being re-used from somewhere else. The information and figures taken from other sources has been referenced properly to the best of our knowledge.

Question Tagging of StackOverflow

ADITIYA JAIN
18111003

HARIOM
18111019

SANJAY KUMAR
18111058

SAURABH KUMAR
18111062

SHIKHAR BARVE
18111065

VIKASH KUMAR
18111082

1 Introduction

Nowadays community-driven question answering forums are blooming. These portals are overflowing with the questions and intellectual people. Stackoverflow, piazza are typical examples of such forums. These forums are abundant with questions seeking particular information becomes very tedious and difficult. Typically stackoverflow allows user to manually describe the domain of the question by associating it with at most 5 tags. These tags can help experts who are proficient in certain domain to attend and answer the questions quickly. Therefore these tags might help the questioner as well as experts.

In addition to this users can also subscribe to the tags in which they are interested. The problem with tagging is that the users needs to be experienced about the topic otherwise he might incorrectly tag the question. Also, Since there is no standard templates defined for tags, multiple tags differing by a small margin can represent the same tag. For e.g. ”androidstudio” and ”android-studio” means the same here.

To resolve this problem instead of manual tagging of the question, each question can be tagged automatically to their respective field or can be suggested to the user.

In this term project we have developed a predictor(classifier) that predicts the most relevant multiple tags for every question asked depending the body of the question.

2 Prior Work

1. One approach developed by Brooks and Mantanez used three terms with top TF/IDF scores from each post and suggest them as tags.
2. A binary array indexed by the tag is created. where value ”1” represents that the question belongs to the that tag.A classifier treats every tag prediction as One vs Rest problem. And sets ”1” if the question belongs to that tag and 0 otherwise.
3. A model has been proposed by Stanley et al. (Stanley and Bryne, 2013) which used calculation of activation score using the words in the title of the question and the body which depends on the co-occurrence statistics.

3 COMPONENTS USED

3.1 Hardware

1. Intel Core i7
2. RAM 16GB
3. HDD Space :- 20GB

3.2 Software

We have preferred python 3.7 coding environment with Jupyter notebook and following libraries

3.2.1 PANDAS

It is an open-source Library supported by many intelligent individuals worldwide. It provides high performance data structures and data analysis tools. For ex, for reading files, group by etc.

We have used it for manipulating the data.

3.2.2 Beautiful Soup

It is a python library for parsing and pulling data out of HTML and XML files.

For parsing HTML data and extracting text present in the question.

3.2.3 NLTK

It is a platform for building python programs for working with human language data.

For filtering data (prepositions, conjunctions, determiners etc.)

3.2.4 TF/IDF Vectorizer

The vectorizer is used for converting Textual Data into numerical form. We have used this component to convert the questions into vectors.

Input to the TF/IDF vectorizer with the following parameters

1. stopwords :-nltk.stopwords
2. max_df :- 0.8
3. min_df :- 50
4. max_features :- 3000
5. smooth_idf :- True
6. sublinear_tf :- True
7. token_pattern :- Space separated

Output :- 1million x 3000 sparse feature matrix.

4 Our Approach

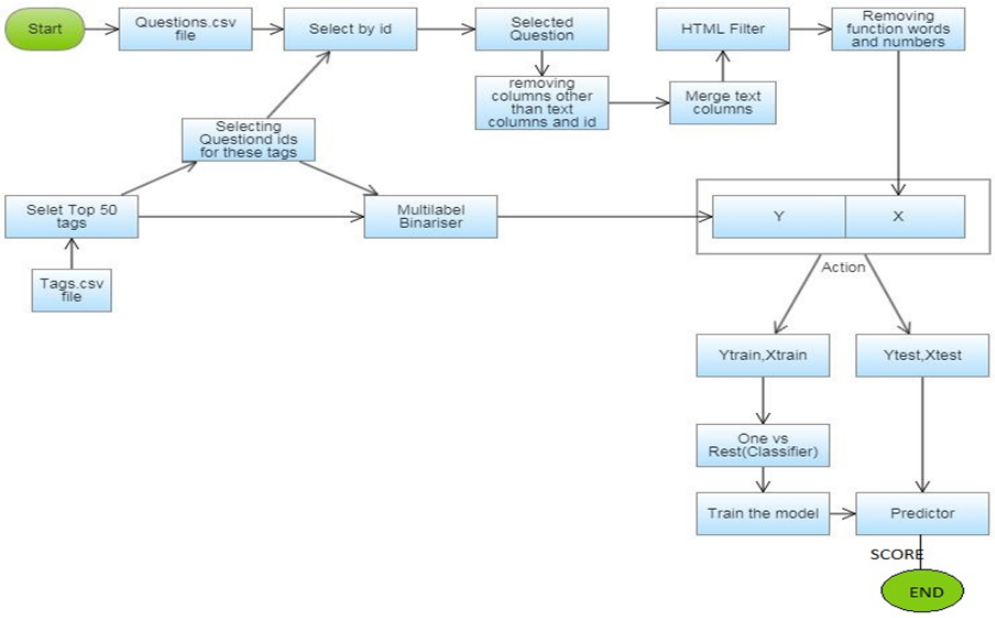

Figure 1: FLOW CHART DIAGRAM

In general the prediction is divided into 3 major parts.

4.1 Converting Questions into Vectors

Dataset have more than 37000 tags and more than 4 million questions. But due of unavailability of computation power we have reduced our dataset to top 50 tags and 7.5 lacs for training and cross validation and the remaining 2.5 lacs for testing.
Questions corresponding to the selected 50 tags are converted into TF/IDF vectors using TF/IDF vectorizer and the 50 tags are converted into multi label binary matrix.

4.2 Training classifier

We have trained the dataset on three models namely MLP, random forest and SGD

1. **Random Forest**
   It is a type of ensemble method in which subset of features are used to train various tree models and gives output as mode of the class predictions in classification problem.we used this bagging algorithm which combines 200 tree models. Gini index is used for sorting features according to importance.Since dataset is imbalanced we are providing balanced class weights. It took a day to train the model.
2. **Multi Layer Peceptron(MLP)**
   MLP classifier is feed forward artificial neural network divided into input, hidden layer and output. We have used hundred neurons in a single hidden layer. It took about a day to train the model.we used Early Stopping on 10% validation dataset with ReLU as nonlinearity function
3. **Stochastic gradient descent(SGD)**
   As the time taken by our classifiers was way too much so we tried to move towards a simple classifier. For this we used 4 versions SGD classifier. The different versions of SGD are as follows:
   1. l1 regularised SGD with hinge loss
   2. l2 regularised SGD with hinge loss

3. l1 regularised SGD with logistic loss
4. l2 regularised SGD with logistic loss
All the SGD approaches took only minutes to complete.

4.3 Tags Prediction

During cross validation we found that the test accuracy that we were getting using the different classifiers varies in range 28.40% to 42.93%.The low accuracy(28%)that we were getting was due to SGD based approaches.This is due to the fact that variance observed in SGD implementation was very high. To reduce this we could have used instead of SGD we could have used Minibatch SGD.

5 Conclusion and Results

From our observation we see that most of models give accuracy between the range 28.40% to 42.93%. The best accuracy is obtained by MLP model with hundred neurons in a single hidden layer that may be due to the fact that each hidden unit of the first layer extract different important features from the input. It is used to do a multi class classification which isn't the case for SGD or random forest.

Random forest is also giving the accuracy in similar range that may improve if we increase the number of estimators.

The four versions of SGD are giving less accuracy. Even though we have large dataset but the data seems to be not enough for running SGD because only uses a randomly choosen example for updating.

CLASSIFIERS	EVALUATION MATRIX			
	SUBSET ACCURACY	HAMMING LOSS	ZERO ONE LOSS	JACKARD SIMILARITY
MLP	0.4293	0.0161	0.5707	0.5781
RANDOM FOREST	0.387	0.0178	0.6129	0.5213
SGD(LOG LOSS, L2 PENALTY)	0.284	0.0205	0.715	0.3854
SGD(LOG LOSS,L1 PENALTY)	0.3643	0.0181	0.6357	0.4972
SGD(HINGE LOSS,L2 PENALTY)	0.3937	0.0171	0.6062	0.5254
SGD(HINGE LOSS, L1 PENALTY)	0.3747	0.0178	0.6252	0.5144

Figure 2: TEST RESULTS OF EACH CLASSIFIER

6 Future Work

- **Using Meta data**:- Question Forum such as Stack Overflow, Quora, also saves Meta data about users. We can use Meta data for better performance like owner id to improve our result. For ex. If a user mainly asked Java related questions in past then our model would be biased for predicting Java tags for that user.
- **Tagging using Answers**:- If we are not able to predict tags with high probability just only with questions we can use top few answers for tagging questions. It would provide better performance.
- **Complete Dataset** :- Processing and training model using complete dataset needs lots of computational power. Due to resource constraint, we trained model on partial dataset. Using complete dataset would provide better results.
- **Feature Extraction**:- We used tf-idf vectorizer to generate feature vector for each question which is an implementation of Bag-of-words approach. We can use NLP techniques like stemming or lemmatization to get better features.
- **Using Complex Models**:- Using multiple hidden layers in MLP classifier would learn a complex and better classifier which can learn better decision boundaries.

7 References

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

[2]S. Schuster, W. Zhu, and Y. Cheng. (2013). Predicting tags for StackOverflow questions. CS 229 Machine Learning, Stanford Univ. [Online]. Available: http://cs229.stanford.edu/projects2013.html

[3]Saha, A.K. ; Univ. of Saskatchewan, Saskatoon, SK, Canada; Saha, R.K. ; Schneider, K.A., " A discriminative model approach for suggesting tags automatically for Stack Overflow questions", Mining Software Repositories (MSR), 2013 10th IEEE Working Conference , May 2013

[4]Susan Li.Multi Label Text Classification with Scikit-Learn Available:Multi Label Text Classification with Scikit-Learn

[5]Dataset.https://www.kaggle.com/stackoverflow/stacksample