Stack Overflow Tag Prediction

Team: Cluster

Nisarg Sheth (2020201049)
Pujan Ghelani (2020201083)
Karan Bhut (2020202015)

Problem Statement

Implement a system that is able to automatically assign tags to questions from the question-answering site StackOverflow which consists of a programming language detection system using content-based features.

More formally, given a question q containing a title consisting of n words a_1 , ..., a_n and a body consisting of m words b_1 , ..., b_m , we want to assign $1 \leq k \leq 5$ tags t_1 , ..., t_k from a limited list of tags T.

Theory

StackOverflow allows users to assign tags to questions in order to make them easier for other people to find.

This is multilabel classification problem. In which, the training set is composed of instances each associated with a set of labels, and the task is to predict the label sets of unseen instances through analyzing training instances with known label sets.

Dataset

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

As suggested in the research paper. Dataset contains 2 files: Train.csv and Test.csv having size reqpectively 6.75 GB and 2 GB.

Train.csv contains 4 columns: The question's Id, Title, Body and Tags associated with it.

Test.csv contains the same columns but without the Tags.

Number of rows in Train.csv = 6034195

Experiments

Downloading dataset

We have downloaded dataset from Kaggle. We have used only Train.csv file. As Test.csv does not contain Tag column which is model's output, we can't compare accuracies without that.

Experiments

Preprocessing:

Since the dataset is huge with over 60 lakh entries, we sampled it down to 5,35,909 entries as suggested in the research paper.

```
#Randomly sampling 535,909 rows from 6034195 rows as mentioned in research-paper
df = df.sample(n=535909)
```

```
[ ]  df.to_csv('/content/gdrive/MyDrive/Colab Notebooks/Train_5l.csv')
```

Removing duplicates

```
# group rows having same value of title body and tags
no_dups = df.groupby(['Title', 'Body', 'Tags'],as_index=False).size()
```

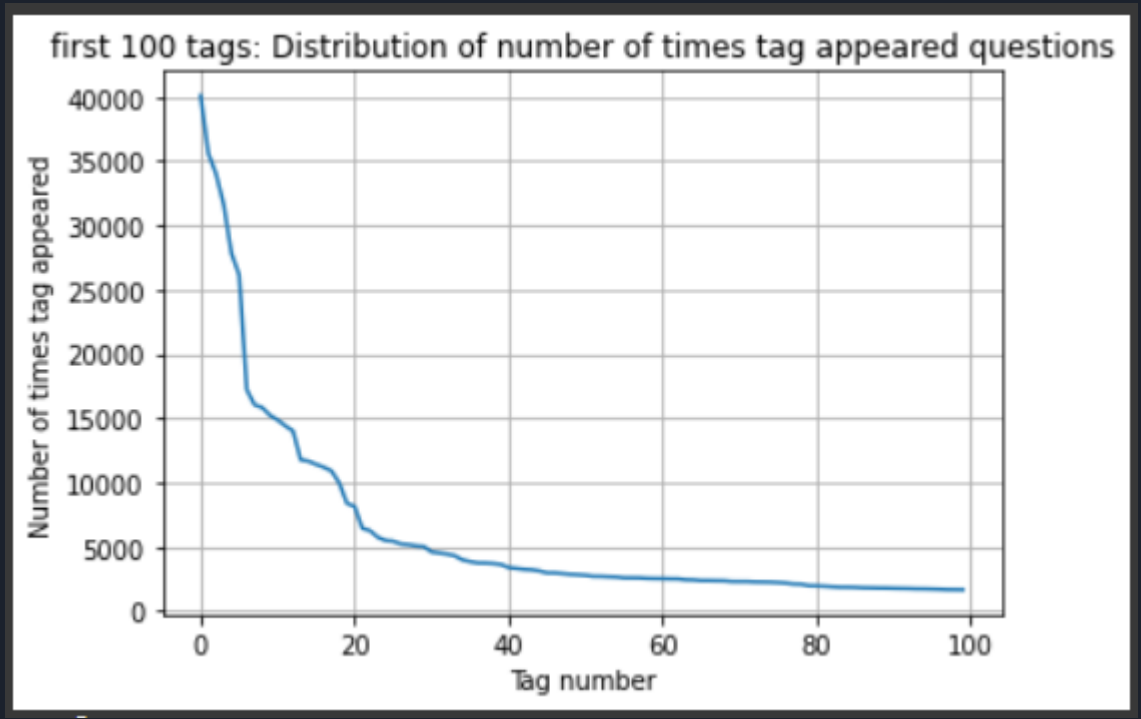
```
[ ] print(len(no_dups))
```

```
519406
```

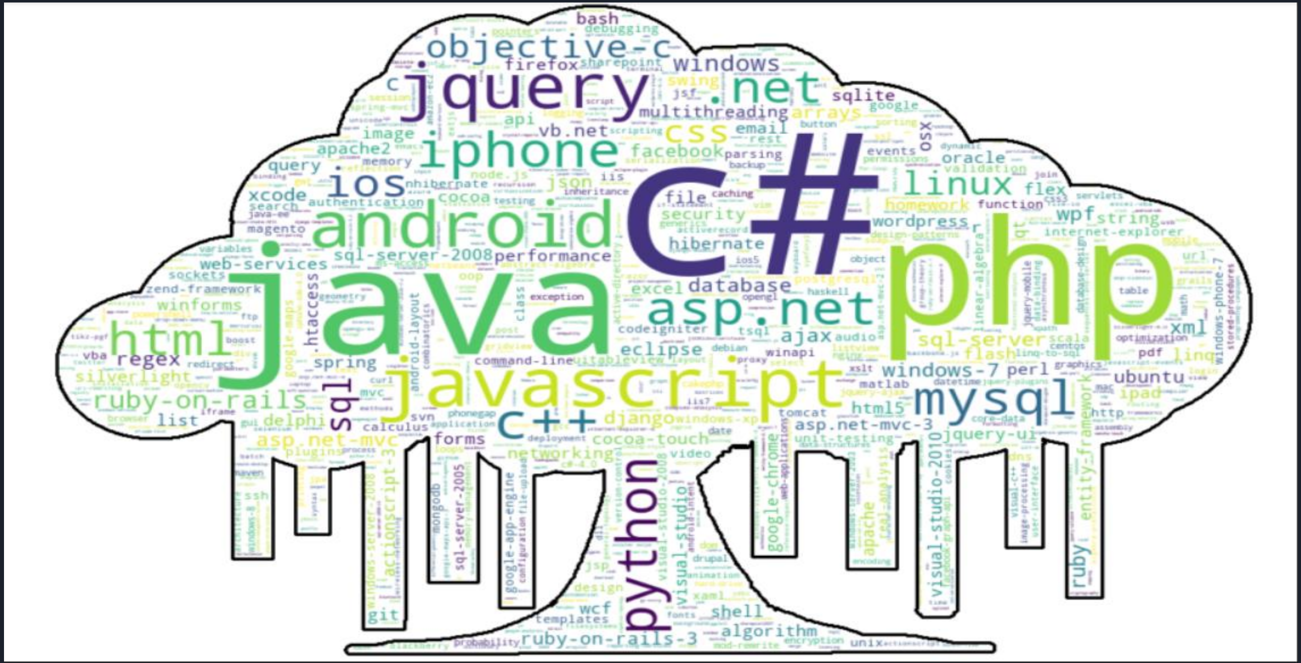
After removing duplicates present in dataset, we got 5,19,406 uniques questions.

Then we represented data in different forms:

Graph representation of 100 most frequent tags in different questions.



Wordcloud containing different tags



Then we performed following operations:

- Separate out code-snippets from Body.
- Remove HTML Tags
- Remove Special characters from Question title and description.
- Remove stop words.
- Convert all the characters into small letters
- Use SnowballStemmer to stem the words.
- We have added title 3 times into question to improve accuracy.

Then we stored processed questions in processed_df.csv file

Preprocessing time (without thread)

```
493000 rows processed
510000 rows processed
Time taken to process data: 11901.351175069809 Seconds
```

Preprocessing time (using multithreading)

```
40000 rows finished for thread-7
Length of processed dataframe: 519406
Time taken: 3782.379077911377 seconds
```

Feature Extraction

```
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(tokenizer = lambda x: x.split(), binary='true')
multilabel_y = vectorizer.fit_transform(preprocessed_data['tags'])
```

```
[ ] print(multilabel_y.shape)
```

```
(519406, 30868)
```

Questions are then converted into matrix form where each row represent unique question and each column represent unique tag.

After that, we chose most frequent 100 tags from 30868 tags. It covered approximately 80% questions.

Then we splitted data into training and testing sets.

Then we performed feature extraction on questions to get matrix form.

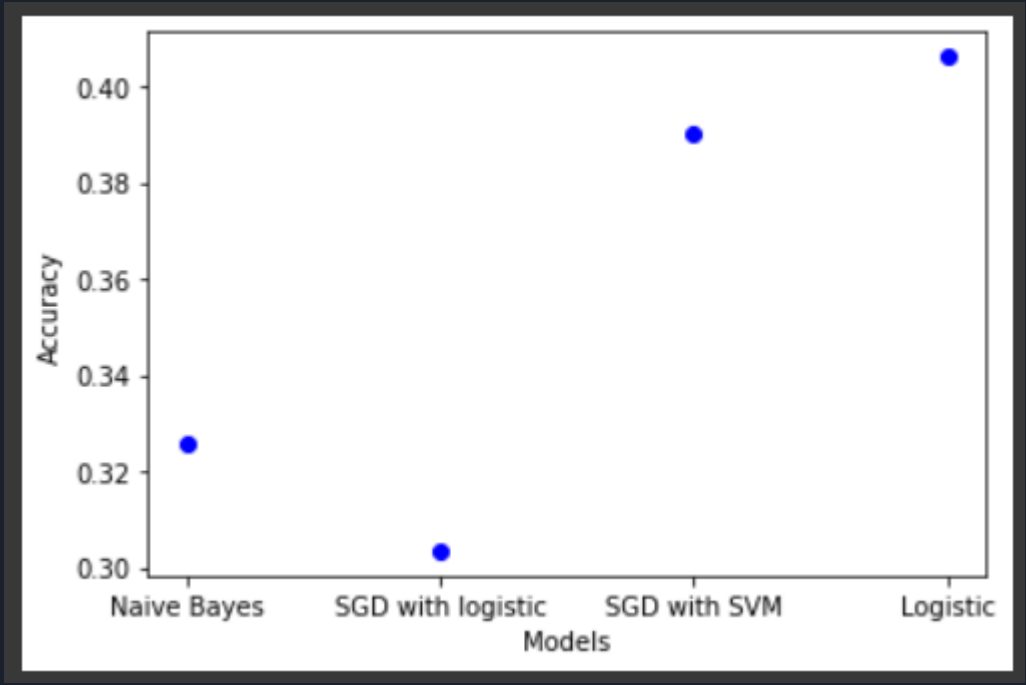
After that, we saved x_train, y_train, x_test and y_test into google drive for efficient operations later.

Model defination

We have Implemented different models on different data size as per model efficiency.

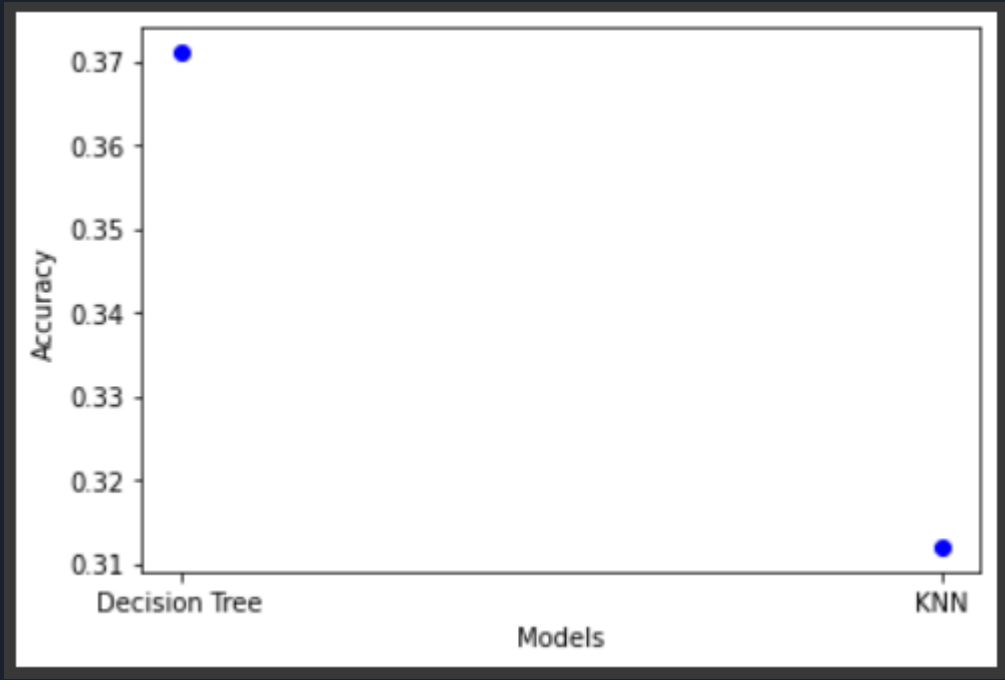
On 5,19,406 data:

1. Naive-bayes classifier
2. SGD training with Logistic Classifier
3. SGD training with SVM Classifier
4. Logistic Classifier


0.40
0.38
0.36
0.34
0.32
0.30
Accuracy
Naive Bayes
SGD with logistic
SGD with SVM
Logistic
Models

On 80,000 data:

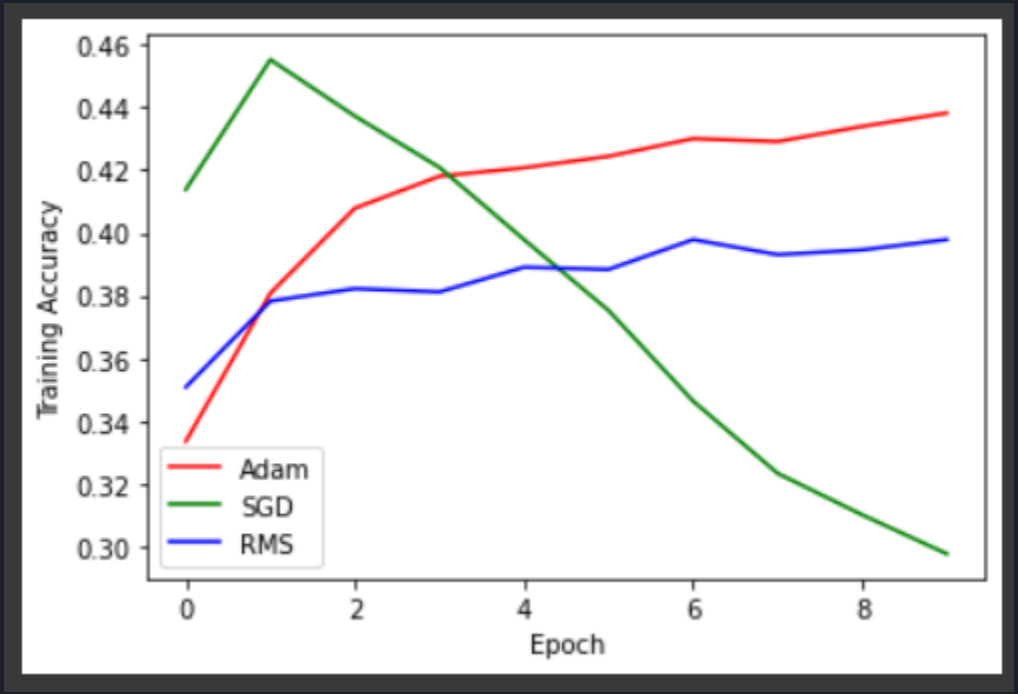
1. KNN classifier
2. Decision tree classifier



On 35,000 we performed:

1. Random Forest Classifier
2. Classification using Neural-Network

In neural network, we have experimented model with different optimizers:



Result

After different experiments, we realized that Logistic Classifier works best and gives highest accuracy among all models but takes more time than SGD and naive bayes classifiers.

Individual Contribution

Nisarg : Data Preprocessing, Naive Bayes, Logistic Classifier and SVM Classifier with SGD Training

Pujan : Data Preprocessing, Logistic Classifier, KNN, Decision Tree Classifier

Karan : Data Preprocessing, Random Forest classifier, Classification using Neural Network

Thank You