Tag Prediction from Stack Overflow Questions

Aditya Gupta
IIIT-Delhi
aditya19138@iiitd.ac.in

Deepak Gupta
IIIT-Delhi
deepak19158@iiitd.ac.in

Ishan Mehta
IIIT-Delhi
ishan19309@iiitd.ac.in

1. Introduction

In information retrieval and natural language processing, assigning keywords is a critical activity. Automated tagging is a subtype of keyword assignment and a sort of text classification in and of itself, with applications in online forums and automated document screening.

Stack Overflow is the largest, most trusted online community for developers to learn, share their programming knowledge, and build their careers. Stack Overflow is something which every programmer use one way or another. In this study, we predicted the tags based on the content of the questions posted on the forum. We used Logistic Regression, SGDClassifier, Multinomial Bayes, and RandomForestClassifier in order to predict tags.

2. Related Work

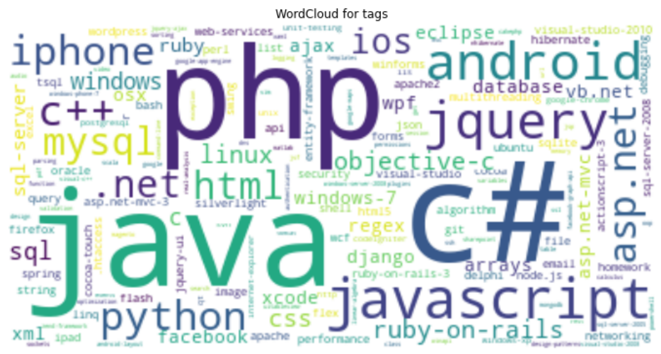
In other research papers[2] that we go through the most commonly used models are logistic regression, Multinomial Bayes, Linear SVM, and Simple Neural Network. The initial steps in all those research papers were text pre-processing which involved techniques like feature extraction, stemming, lemmatization, excluding punctuation, and applying regex.

After the data processing, models were applied to the resulted dataset, out of which, a few models were highly competitive with similar results like SNN predicted result with the precision of 0.78, and Logistic Regression predicted with the precision of 0.79 but the best result was predicted by the Linear SVM i.e. 0.866.

3. Dataset and Evaluation

3.1. Description

Our data[1] consists of around 6 million questions, titles, and tags from the site Stack Overflow. Each question has between one and five tags(Fig.2); tags represent the themes and keywords that appear in a question(Fig.1). In order to avoid computation constraints with training and evaluation of the data, and in order to properly bound the problem space, we decided to limit our tag prediction to the top 1,00 most commonly occurring tags, and conducted training and testing on a subset of 200,000 rows. We removed all questions that did not have a tag from the 100 most common tags or which were duplicates. Then we splitted test and validation set with ratio 80:20.



3.2. Preprocessing

In order to create features from title and question text,we first cleaned the text by removing HTML markups, stopwords, special characters, URL links and replaced the words with lemmatization.

3.2. Feature Extraction & Selection

We use TFIDF vectorizer to get the features from our questions and title text and then selected top 1000 features using chi2 feature selection technique for computational purposes. Since words in the title are more important we gave more weightage to title text while extracting features.

3.3. Evaluation Metric

Since this is a multi-label classification task, and accuracy is not a good option for us since the data is imbalanced. So we needed to choose an appropriate set of evaluation metrics that properly weigh precision and recall. The most common is F1 score, precision and recall. These can further be split into micro-averaged and macro-averaged. Macro computes for each label and averages across all the labels. Micro does so using a weighted average, where common labels receive more weight. We chose micro average in this study for measuring model performance.

4. Methodology & Analysis

4.1. Data Processing

In data processing, our first step was to remove all the HTML tags, and to remove them we used Regex. So after the removal of HTML tags, we converted all the text data into lowercase and in order to do that we used the '.lower()' function.
After this, our Question also included the code for the problem, so our next step was to remove that code, and in order to do so we used 'findall' and 'sub function of the 're' library.
After removing the code we further proceed to remove URL links from the Question. As the web URL is of no use, they are simply redundant pieces of information.
Further, we also removed the Special characters from the Question tile and body.
And In the last, we removed all the stopwords except 'C' as it stands for language C and lemmatized all the left out words, and also expanded the contracted words.

For label output, we just simply vectorized them using Countvectorizer.

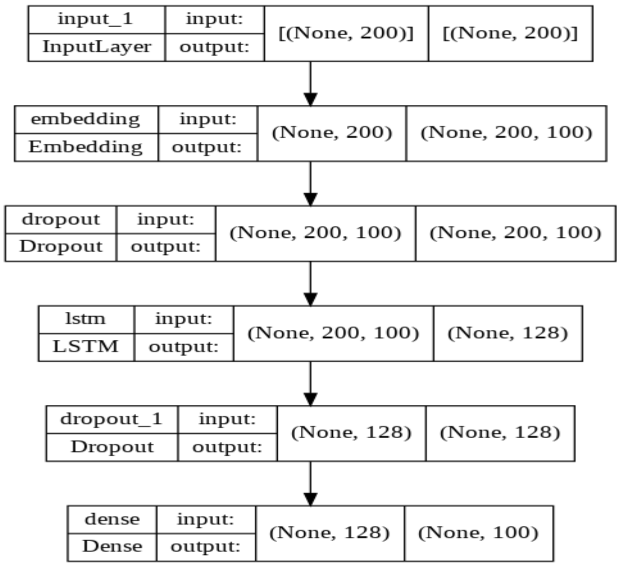
4.2. Approach & Model Building

Since this is a multi-label classification problem, we needed to predict multiple values (i.e. a probability indicating the likelihood of a given tag) per question which meant that we needed to either have multiple classifiers (one for each label). So for that we chose One-vs-rest paradigm in which a separate classifier is trained for each label.
So far we have implemented and tested 5 models i.e, SGDClassifier, LogisticRegression, kNN, Gaussian NB, RandomForestClassifier, Multinomial Naive Bayes.

We further used advanced neural networks model like LSTM and also applied BERT model,

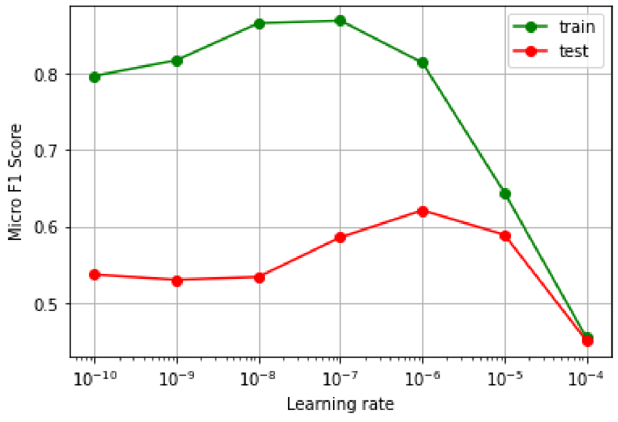
BERT:So for the bert model we just simply gave our text which was processed by bert preprocessing model and later the output were fitted into bert classifier model with batch size of 16 and learning rate of 10^-5..

LSTM : For the given problem we used LSTM model with pretrained Glove embeddings to reduce the dimension of vectorized features and convert text to numerical sequences. Dropout(0.2) layers are also added in between layers to avoid overfitting. Adam optimizer has been used with batch normalization and batch size 120


input_1 | InputLayer | input: | output: | [(None, 200)] | [(None, 200)]
embedding | Embedding | input: | output: | (None, 200) | (None, 200, 100)
dropout | Dropout | input: | output: | (None, 200, 100) | (None, 200, 100)
lstm | LSTM | input: | output: | (None, 200, 100) | (None, 128)
dropout_1 | Dropout | input: | output: | (None, 128) | (None, 128)
dense | Dense | input: | output: | (None, 128) | (None, 100)

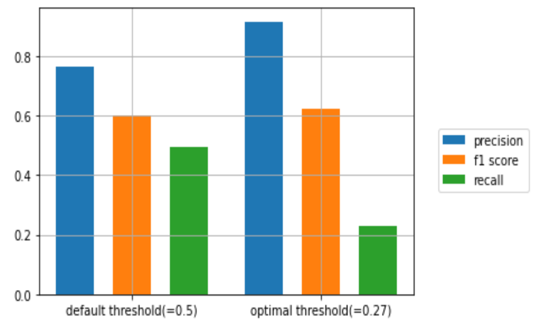
4.3. Hyper parameter Tuning

To ensure we get good results, we tuned the parameters for all our models. For SGDClassifier and Linear Regression we tuned the learning rate and applied L1 penalty to avoid overfitting. Best learning rate for our model was coming out to be around 10^{-6}
For BERT we used batch size, learning rate and different model to get better results.



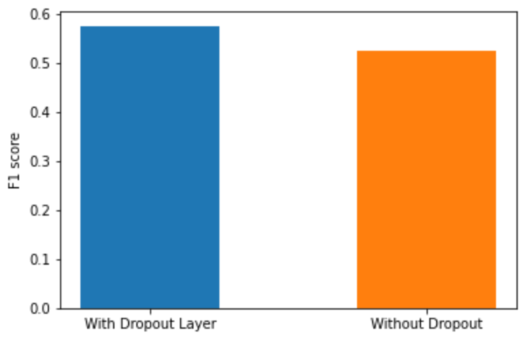
4.4. Challenges Faced & Error Analysis

- Initially, we used 20,000 features but due to computational constraints, we were regularly encountering system crash(due to high memory requirement), So in order to avoid that we selected only top 1000 features using chi2 feature selection.
- In models like SGDC, we were not able to directly calculate the probability of every label because it uses hyper plane to calculate the labels, so we have to wrap the SGDC into CalibratedClassifierCV to get probability of each label.
- On analysis of our linear model,we observed that our models were not able to predict any tags at all for some vagues questions.This is problematic in our case because if a tag prediction system fails to recommend at least one relevant tag to a user, the user experience will most likely be poor. This happens because our models outputs a probability of that tag belonging to the text,and since the default threshold is 0.5, it was best for our case, so we experimented with different thresholds and found the optimal threshold which improved our overall f1 and recall score significantly(Fig.3).



- In SGDClassifier, for alpha=0.01 our training and test performance was quite low,this was due to large learning rate and hence the model was diverging. After hyperparameter tuning we got our best alpha value =10**-7. (See Fig.5)

- While training LSTM model our model gave better performance with dropout layer which can be due to reduced overfitting.



5. Results

5.1. Logistic Regression

In this model,we got our best test performance(f1 score=0.63) with no regularization(means no overfitting) and liblinear solver.(See Fig,4)

5.2. SGD Classifier

Took SGD as baseline model and performed hyperparameter tuning for a better performance.
Best performance(f1=0.65) was seen at alpha=10**-7 with L1 regularization, after this our model starts diverging. (See Fig.5)

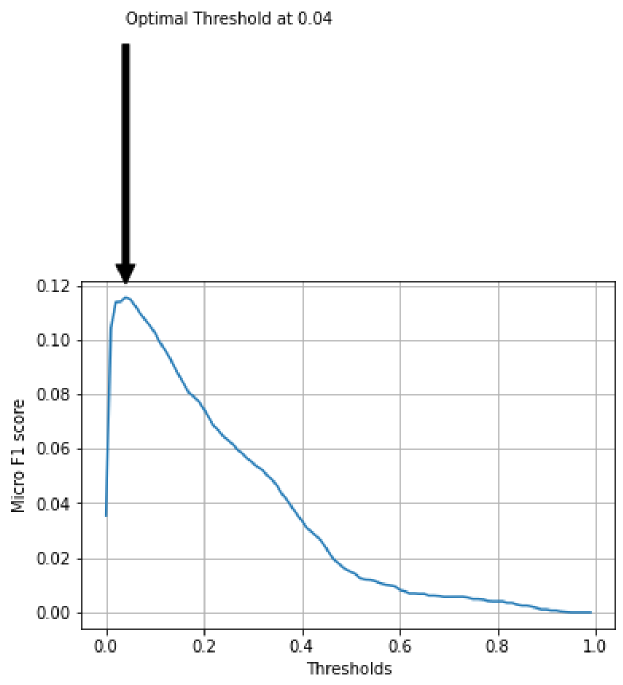
5.4. k-Nearest Neighbors

In this Model the best performance for training data was 0.93 and for test data was 0.44 after optimizing probability threshold(see Fig 8)

5.5 Random Forest Classifier

In this Model the best performance for training data was 0.998 and for test data was 0.5741 without optimizing probability threshold, and after optimizing the probability threshold test performance increased to 0.65(see Fig 9)

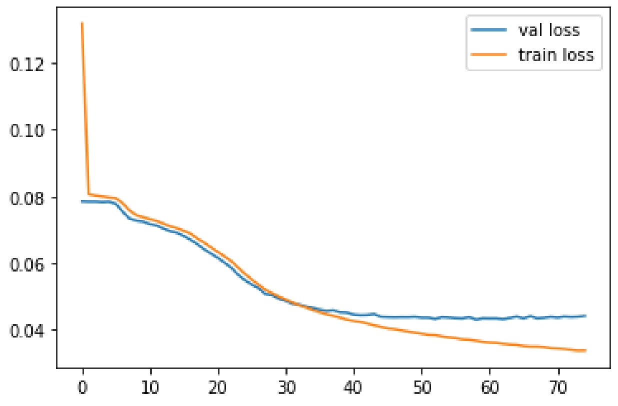
5.6 BERT



The performance of BERT model wasn't as good as expected, it gave us an accuracy of 0.36 and its best F-1 score was about 0.12 which is very less compared to LSTM and the rest other models.

5.7 LSTM

For LSTM our best f1 score was coming out to be 0.62 with 100 epochs and batch size 120 and the train loss was also converging.



Model	Micro F1 score	Macro F1 score
Logistic Regression	0.63	0.65
SGD Classifier	0.65	0.66
SVM	0.53	0.4
KNN	0.44	0.43
Random Forest Classifier	0.57	0.58
BERT	0.12	0.1
LSTM	0.62	0.58

6. Contribution

6.1 Deliverables

Initially we planned to perform the training on large dataset(>5lakhs) but due to computational constraint required to training neural network we were unable to do it.
We have implement advance DL models as proposed earlier.All the models have been saved for future reference and evaluation.
Programming language was also considered while feature extraction which was also proposed by us earlier.

6.2 Individual Contribution

Aditya:
- Data Preprocessing, linear models
- LSTM model, Error Analysis

Deepak:
- Data Preprocessing, SVM and kNN
- Bert model, Error Analysis

Ishan:
- EDA, Literature Review
- Random Forest

6.3 References and Citations

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

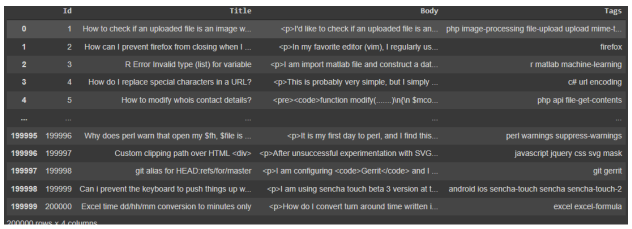
[2] Jalal, Kevin and Reid. Tag Prediction from Stack Overflow Questions 2019.

[3] Sebastian, Wanying, Yiying. Predicting Tags for StackOverflow Questions

6.4 Files

1. Dataset Folder
2. StackTag.ipynb
3. Models folder
4. glove.6B.100d.txt

Plots and figures



	Id	Title	Body	Tags
0	1	How to check if an uploaded file is an image w...	<p>I'd like to check if an uploaded file is an...	php image-processing file-upload upload mime-t...
1	2	How can I prevent firefox from closing when I ...	<p>In my favorite editor (vim), I regularly us...	firefox
2	3	R Error Invalid type (list) for variable	<p>I am import matlab file and construct a dat...	r matlab machine-learning
3	4	How do I replace special characters in a URL?	<p>This is probably very simple, but I simply ...	c# url encoding
4	5	How to modify whois contact details?	<pre><code>function modify(.......)\n{\n $mco...	php api file-get-contents
...	...	...	...	...
199995	199996	Why does perl warn that open my $fh, $file is ...	<p>It is my first day to perl, and I find this...	perl warnings suppress-warnings
199996	199997	Custom clipping path over HTML <div>	<p>After unsuccessful experimentation with SVG...	javascript jquery css svg mask
199997	199998	git alias for HEAD:refs/for/master	<p>I am configuring <code>Gerrit</code> and I ...	git gerrit
199998	199999	Can i prevent the keyboard to push things up w...	<p>I am using sencha touch beta 3 version at t...	android ios sencha-touch sencha sencha-touch-2
199999	200000	Excel time dd/hh/mm conversion to minutes only	<p>How do I convert turn around time written i...	excel excel-formula

200000 rows × 4 columns

Fig. 1

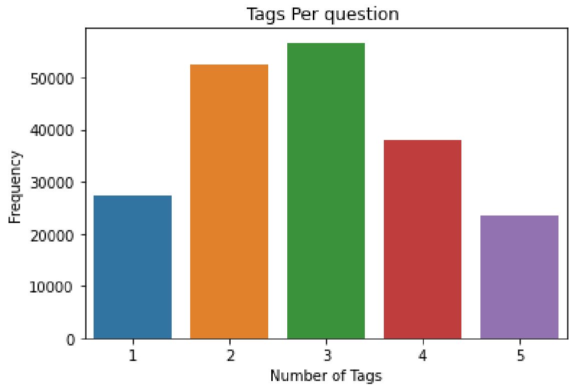
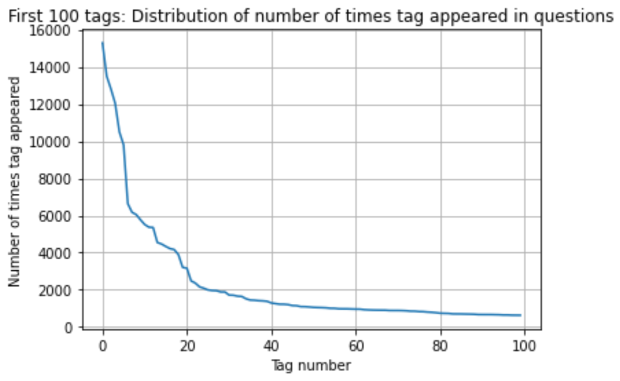
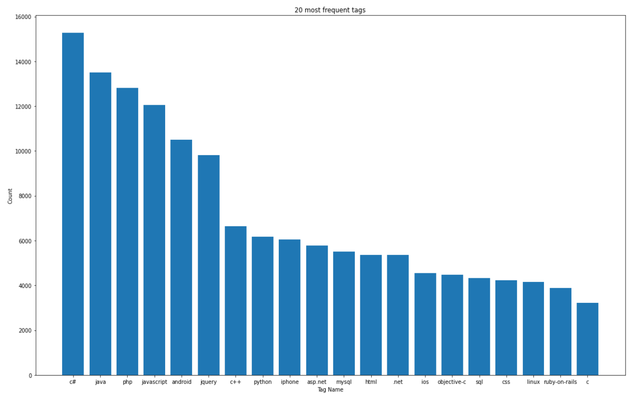
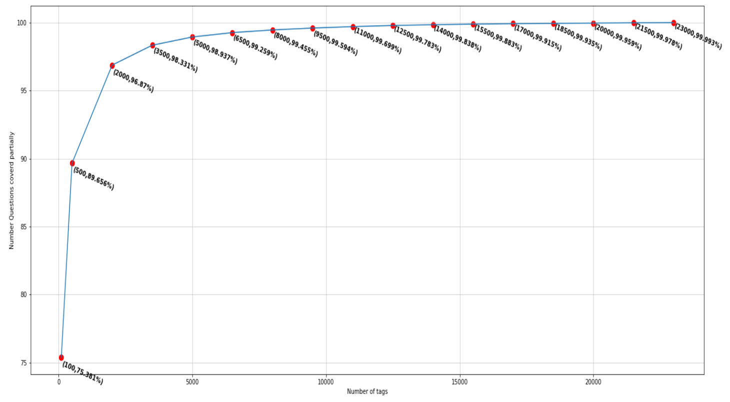


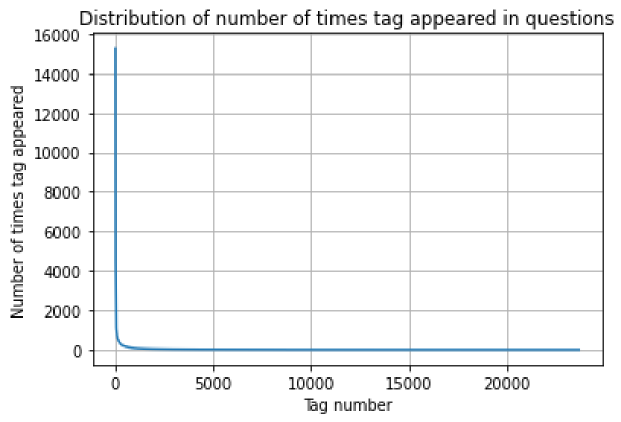
Fig.2







Tags vs % Questions covered partially



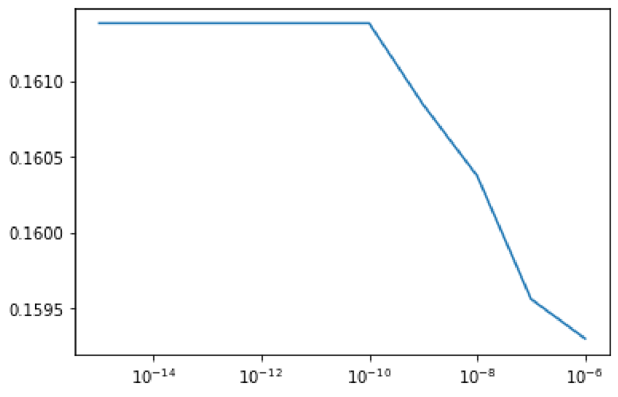
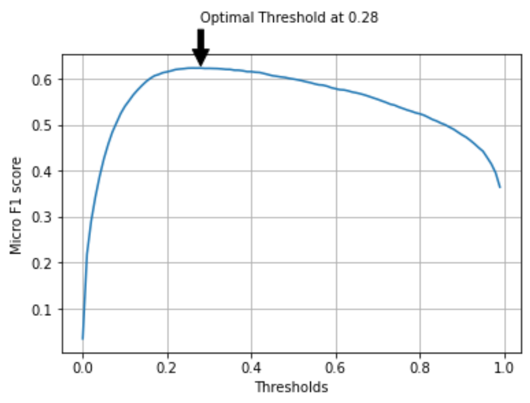


Fig.6. Multinomial bayes(F1 score vs alpha)



Optimal threshold for LR model

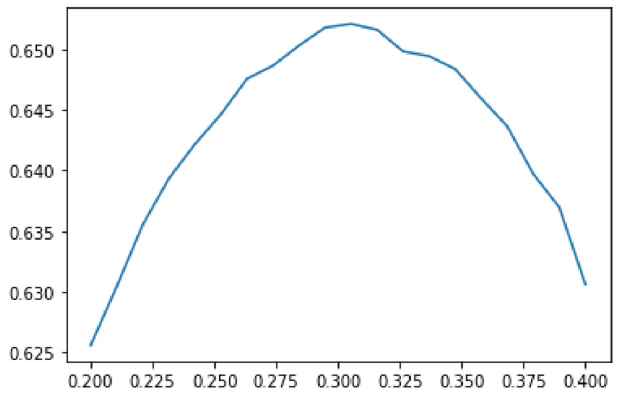
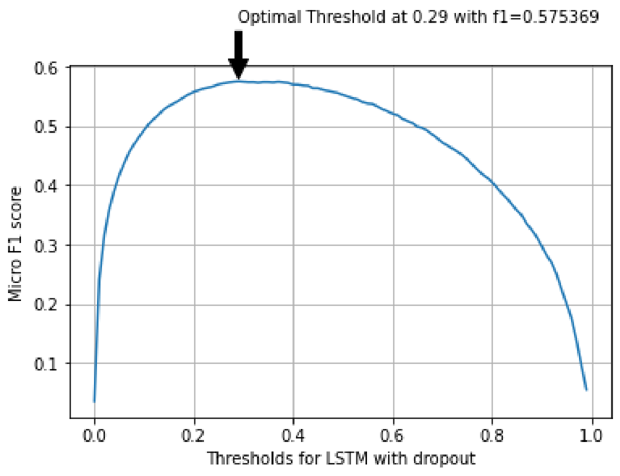


Fig 9: random forest classifier (F1 Score vs threshold)

Fig.3



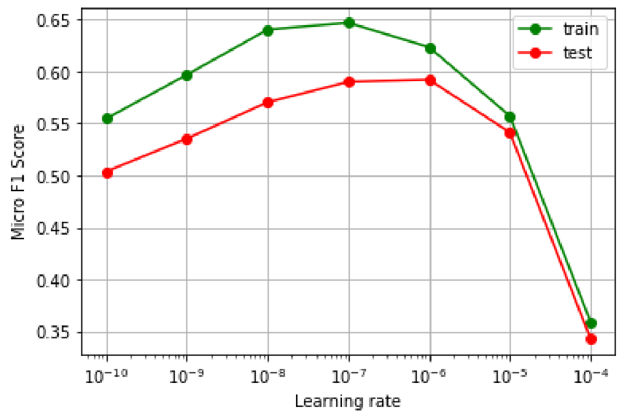
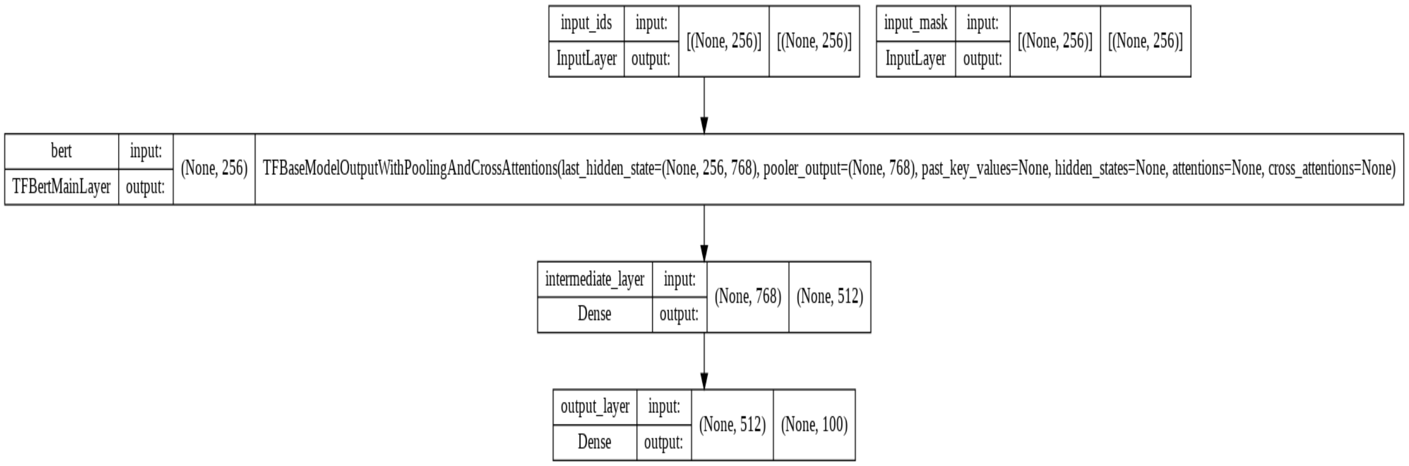


Fig.5. SGDClassifier(alpha vs f1)



[Bert Model Architecture]