ITEC-876-Applications of Data Science

Introduction

The main goal of the project is to apply text classification on the problems which we face in day to day activities like spam detection, credit defaulters etc. For the project, we have used a library called as "FastText" which is an efficient way of learning word representations and sentence clarifications. We have used different types of language models namely: Epoch, learning Rate, N-grams, Hierarchical SoftMax and Multilabel Classification to improve the precision (efficiency) of the outcome.

• **Epoch**: By default, FastText trains our data only 5 times so to improve the quality of the outcome we have changed the parameters of Epoch to get higher accuracy.

• **Learning Rate**: To improve the learning rate of the algorithm we tune with the parameters of the Learning Rate, ideally it is in between 0.1 To 1.0.

• **N-grams**: By using the word n-gram, we can improve the precision of preprocessing of the data. N-grams works on the concept that unigrams will check by taking a single word at a time, bigrams will scan 2 words at a time and similarly the process continues.

• **Hierarchical SoftMax**: It works as the tree hierarchy, where each word is represented by leaves or parent nodes; and internal nodes represents the probabilities of child node, we use this function for faster computation of larger datasets

• **Multilabel Classification**: When number of labels are used to predict the threshold of predicted probability; here we tune arguments to improve the efficiency of the data.

Replication of Original Work

We tried to replicate the output on the cooking dataset, which is given in fast text classification tutorial, here are the snippets of the replicated output on the original data

```
(base) taylorswift@taylorswift-virtual-machine:~/github/fastText$ head cooking.stackexchange.txt
__label__sauce __label__cheese How much does potato starch affect a cheese sauce recipe?
__label__food-safety __label__acidity Dangerous pathogens capable of growing in acidic environments
__label__cast-iron __label__stove How do I cover up the white spots on my cast iron stove?
__label__restaurant Michelin Three Star Restaurant; but if the chef is not there
__label__knife-skills __label__dicing Without knife skills, how can I quickly and accurately dice vegetables?
```

Figure 1

```
(base) taylorswift@taylorswift-virtual-machine:~/github/fastText$  ./fasttext test model_cooking.bin cooking.valid
N       3000
P@1     0.139
R@1     0.06
```

Figure 2: Precision Before preprocessing the text file

```
(base) taylorswift@taylorswift-virtual-machine:~/github/fastText$  ./fasttext test model_cooking.bin cooking.valid
N       3000
P@1     0.139
R@1     0.06
```

Figure 3: Precision After preprocessing

```
(base) taylorswift@taylorswift-virtual-machine:~/github/fastText$ ./fasttext test model_cooking.bin cooking.valid
N       3000
P@1     0.517
R@1     0.224
```

Figure 4: Precision at Epoch 25

Figure 5: Precision at learning rate 1.0

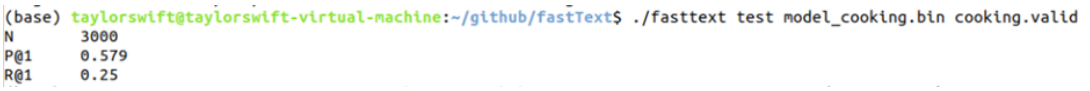


Figure 6: Precision at learning rate 1.0 & epoch 25

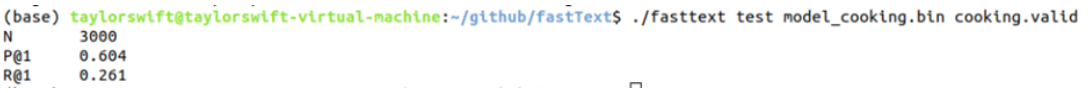


Figure 7: Precision at learning rate 1.0, epoch 25 & wordngram2

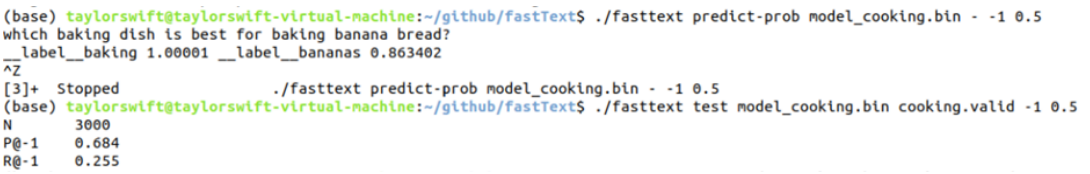


Figure 8: Multilabel

Upon Looking at the results from the Original dataset, the precision of the pre-processed data (which is model efficiency in identifying the labels) is increased when compared with the precision of the data set before pre-processing. we can see that Precision@1 is 0.139, and after we pre-processed the data i.e, converting all the upper to lower, removing special characters, we find the Precision to increase by 3.5%, which is now at 0.174.

When we ask the question:" Why not put the knives in the dishwasher", the 5 predicted labels by the model were given as follows: "baking, food-safety, equipment, bread, substitution"; only one label out of those predicted labels is correct so Precision is 0.2. Out of the 3 correct labels only one is predicted, so leaving Recall at 0.03.

Epoch: After comparing the precision with the pre-processed data, we can see that the Precision has improved by 34.3%, which is very drastic change.

Learning Rate: we changed the learning rate and compared the result with the Epoch, where we have seen further improvement in precision by 5.9%.

Combining Epoch & Learning Rate: With the change in Epoch and learning rate, we saw improvement in the Precision, hence we tried to bring both together and yet we saw Precision to improve a bit more.

Word gram: To improve the efficiency furthermore we have used bigram that has improvement in the Precision by 2.5%.

Now after running the multi label model on the original data, we had given the same question asked as above and it predicted better labels as compared to before.

⇒We learned that with increase in Epoch, learning rate, and word gram the precision will also increase.

⇒ Multilabel model will try to improve in predicting more no of original labels when compared with the initial model Prediction.

⇒ We use the trends that are observed from the original data set as the base line comparison for all our data sets.

⇒ We will be trying to explore these models in the coming sections with some parameter tuning and compare the Precision at each stage, to find which model is doing better.

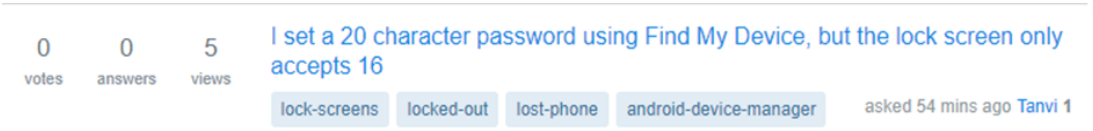
Operating system: Linux, Windows
Ram: 8GB
Processors: 4 cores
Software: Jupyter from Anaconda

Construction of new data:

We have used python programming language with beautiful soup to establish the connection with stack exchange and extracted the data by web scrapping using the html parser. We are in need of question and tags from the division section that are used for the text classification as label and question in the later part of the analysis.

The question and tags were extracted and directly converted them to a text file that fast text can understand, which is in the form of labels and questions.

Below is the snippet of the data which is before scrapping from the stack exchange.



The snippets of all the 4 datasets which has been scrapped from the stack exchange and converted into the fast text format is given below:

Dataset 1 : Music & movies

```
__label__dialogue __label__heist What does the line 'Your weight and your fate right here..' mean?
__label__star-wars __label__return-of-the-jedi Why did Leia not want to tell Han about Luke being her twin brother?
__label__plot-explanation __label__district-9 Why was District 9 alien ship stuck for so long?
__label__parasite What is the significance of the rock?
```

Dataset 2 : Python

```
__label__python __label__variables __label__airflow Create airflow environment variables
__label__python __label__django Django's POST method remove “+” symbol from html input form
__label__python __label__nginx __label__system-calls Nginx: how to return result of a python script?
__label__python __label__nlp __label__spacy Difference between context-sensitive tensors and word vectors
```

Dataset 3 : Android

```
__label__wi-fi Random rebooting issues after the Android 10 update on my oneplus 7 pro [Logcat inside]
__label__android-sdk Android 3.5.1 Gradle Sync Bug [on hold]
__label__termux ssh to termux server disconnects after X seconds
__label__9.0-pie __label__android-one How to disable Direct Share on an Android One device with Android 9?
__label__5.0-lollipop __label__recovery-mode __label__boot __label__asus-zenfone-5 Android Lollipop Boot Issue
```

Dataset 4 : Electronics

```
__label__1983 What year is 1983 set?
__label__suits How old is Sheila Sazs supposed to be?
__label__plot-explanation __label__character __label__3-ninjas-knuckle-up How can Jack Harding also control the Mayor?
__label__the-lord-of-the-rings __label__dubbing __label__korean-cinema Korean Dub of Lord of the Rings Film? [on hold]
__label__ending __label__dark Does Dark have an ending?
```

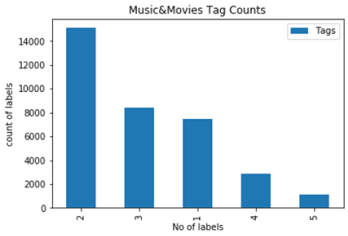
Different types of labels in each dataset

Dataset 1-Music & movies: The data have been scrapped from stack exchange where we have all the questions which are related to the music and movies and we have merged the music and movies data and got approximately 35000 of questions which are used for the fast Text analysis .
In the below histogram we can get the count of labels per question. The number of questions which are having 2 labels is around 14000 being highest.

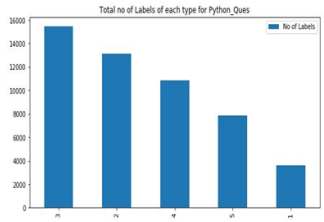
Dataset 2-Python We scrapped these data from stack exchange where we have all the questions related to the python and got approximately of 51000 of questions which are used for the fast Text analysis.In the below histogram we can get the count of labels per question. The number of questions which are having 3 labels are around 35000 being highest.

Dataset 3-Android The Android data is scrapped from stack exchange that has questions related to the android OS and got approximately of 50000 of questions which are used for the fast Text analysis.The histogram plotted below represents the count of labels per question. The number of questions which are having 2 labels are around 15000 being highest.

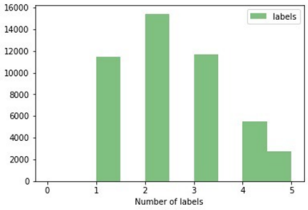
Dataset 4-Electronics We got this electronic dataset from the stack exchange as well with questions related to electronics and got approximately 50000 questions which are used for the fast Text analysis.In the below histogram we can get the count of labels per question. The number of questions which are having 2 labels are around 16000 being highest.



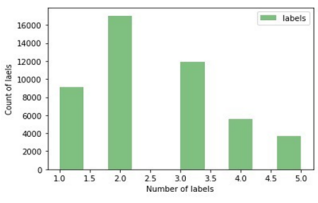
(a) Dataset 1-Music & movies



(b) Dataset 2-Python



(a) Dataset 3-Android



(b) Dataset 4-Electronics

Results on new data

In this section we are showing the results from three Data sets mainly Music-Movies, Python and Electronics. All the Data sets results can be found in the supporting section in their

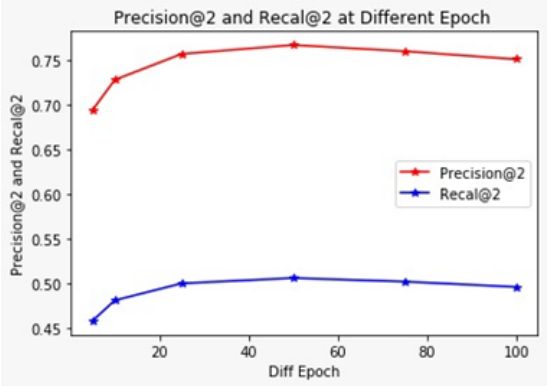
respective folders. On Dataset Python when we run fast text with just the default parameters the precision@1 is coming as 0.997, it's because there is a common label named label_python for all the questions so the model is predicting the correct label at almost 100% all the time, so we decided to look at precision@2, that has given the precision at 0.644, and precision@3 is 0.464.

we performed tests with all the models on this Python Data Set and compared the precision and Recall value at position 2 for the comparisons to be valid. First, we cleaned the dataset by converting all the upper cases letters to lower case letters (Since Both and upper case would represent the same meaning) as well as removing some special characters.

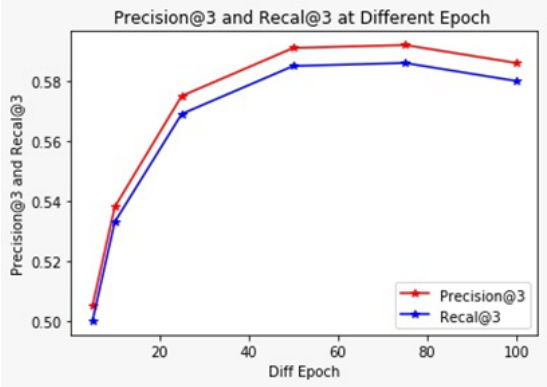
After Doing some pre-processing the precision at 2 is coming as 0.694 (precision is increased by 5% when compared this with precision at 2 before pre-processed) and precision at 3 is 0.505. There are several ways we can increase the efficiency of fast text some of them are Epoch, Learning rate, Word-N-Gram, Multilabel, Hierarchical SoftMax.

Epoch

Below are the graph's that represent Precision and Recall at 2, 3 for different number of Epoch's (Results are extracted from Python Dataset)



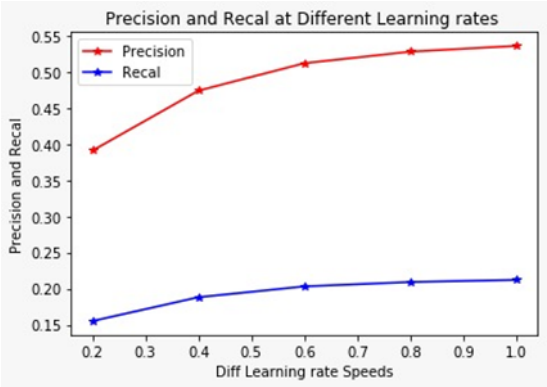
From the graph above, we can clearly see that the Precision@2 and recall@2 rate for python dataset is increasing with increase in number of Epochs (Which is no of times the program will look at the data sets) until Epoch 50, later on we can see decrease in precision by a bit.



The precision@3 and Recal@3 is increasing with increase in the number of Epochs as what we have seen from the original data set.

Learning Rate

Below are the graphs that represent Precision and Recall at for different number of Learning rates (Results are extracted from Music and Movies Dataset)

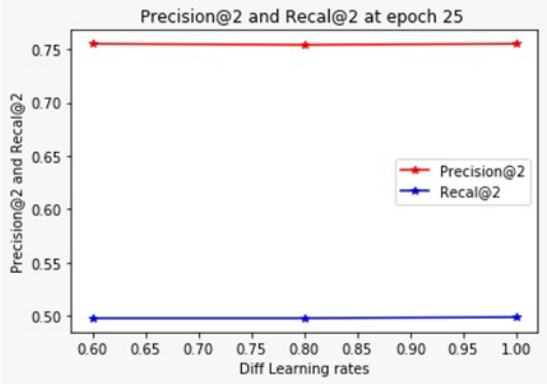


From The graph above we can see that with increase in Learning rate Number there is an increase in Precision and Recall. From the above two observations we can see that Precision is Directly Proportional to the Learning rate and Epoch.

Now we try to look at the results when we have Both Learning rate and Epoch.

Below are the graphs that represent Precision and Recall at 2,3 for different number of Learning rates and Epoch (Results are extracted from Python Dataset)

Learning rate & epoch:



The graph above is representing the Precision and Recall at position 2 for different learning rates when epoch is 25. Usually the precision should increase but here We can see that the precision is decreasing by a very small value its because of having one same label for all the Questions so, we try to look at Precision at position 3. The rest of the data sets are like the original data set results.

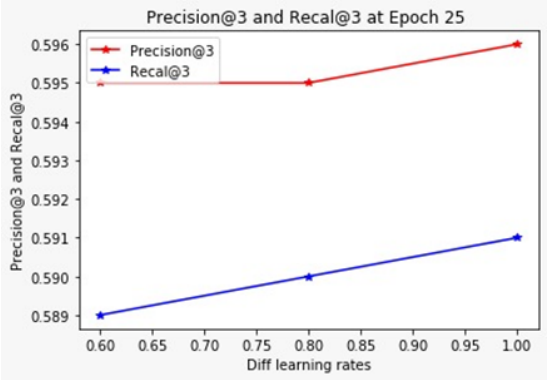


Figure 11

The above graph is representing the Precision and Recall at position 3 for different learning rates when epoch is 25. Now we can see that the Precision and Recall both are Increasing with increasing in the learning rate.

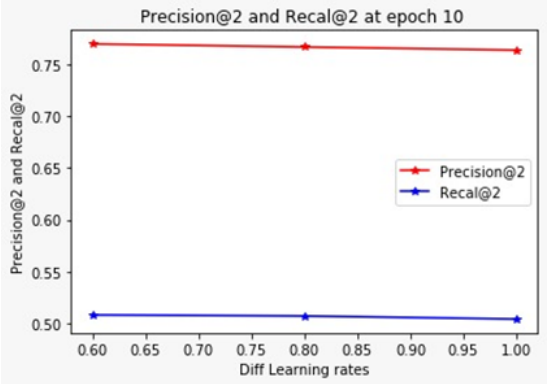
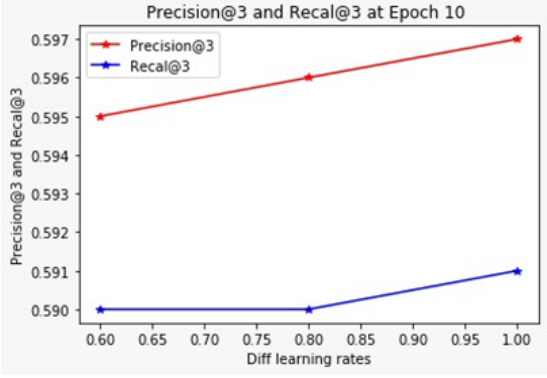


Figure 12

The Precision and Recall at position 2 for different learning rates when epoch is 10 is shown from the above graph. Usually the precision should increase but here We can see that the precision is decreasing by a very small value it's because of having one same label for all the Questions so, we try to look at Precision at position 3. The rest of the data sets are like the original data set results.



The above graph is representing the Precision and Recall at position 3 for different learning

rates when epoch is 10. Now we can see that the Precision and Recall both are Increasing with increasing in the learning rate.

Word n grams

The followig results are from the Electronics Data sets.

Usually with increase in Word N Grams the precision should increase but When running word n grams 2 with epoch 25 and learning rate 1.0 we can see the precession rate is 0.50 and recall 0.18 and with increase in the wordNgram to 3 we can observe the precession is reduced to 0.48 and recall to 0.17 and it is because of the label problem in the dataset Electronics.

```
praveen@DESKTOP-73GJ1HJ:~/fastText-0.9.1$ ./fasttext supervised -input Processed.train -output model_processed -lr 1.0 -
epoch 25 -wordNgrams 2
Read 0M words
Number of words:  30432
Number of labels: 5713
Progress: 100.0% words/sec/thread:    1707 lr:  0.000000 loss:  4.134550 ETA:   0h 0m
praveen@DESKTOP-73GJ1HJ:~/fastText-0.9.1$ ./fasttext test model_processed.bin Processed.valid
N       9996
P@1     0.502
R@1     0.181
```

```
praveen@DESKTOP-73GJ1HJ:~/fastText-0.9.1$ ./fasttext supervised -input Processed.train -output model_processed -lr 1.0 -
epoch 25 -wordNgrams 3
Read 0M words
Number of words:  30432
Number of labels: 5713
Progress: 100.0% words/sec/thread:    1593 lr:  0.000000 loss:  4.183177 ETA:   0h 0m
praveen@DESKTOP-73GJ1HJ:~/fastText-0.9.1$ ./fasttext test model_processed.bin Processed.valid
N       9996
P@1     0.489
R@1     0.176
praveen@DESKTOP-73GJ1HJ:~/fastText-0.9.1$
```

Multilabel

The Following outputs are from the Python Data Sets

```
(base) taylorswift@taylorswift-virtual-machine:~/github/fastText$ ./fasttext supervised -input FT1.train -output model_FT1 -lr 0.5 -epoch 25 -wordNgrams 2 -bucket 200000 -dim 50 -loss one-vs-all
Read 0M words
Number of words:  22108
Number of labels: 6245
Progress: 100.0% words/sec/thread:    8810 lr:  0.000000 avg.loss:  5.823550 ETA:   0h 0m 0ss
(base) taylorswift@taylorswift-virtual-machine:~/github/fastText$ ./fasttext test model_FT1.bin FT1.valid -1 0.5
N       10200
P@-1    0.847
R@-1    0.535
(base) taylorswift@taylorswift-virtual-machine:~/github/fastText$ ./fasttext predict-prob model_FT1.bin - -1 0.5
reportlab text in new line with flx column width
__label__python 1.00001 __label__reportlab 0.938134 __label__pdf 0.749097
create airflow environnet variables
__label__airflow 1.00001 __label__variables 1.00001 __label__python 1.00001
^Z
[1]+  Stopped                 ./fasttext predict-prob model_FT1.bin - -1 0.5
(base) taylorswift@taylorswift-virtual-machine:~/github/fastText$ ./fasttext supervised -input FT1.train -output model_FT1 -lr 0.5 -epoch 25 -wordNgrams 3 -bucket 200000 -dim 50 -loss one-vs-all
Read 0M words
Number of words:  22108
Number of labels: 6245
Progress: 100.0% words/sec/thread:    8877 lr:  0.000000 avg.loss:  6.429079 ETA:   0h 0m 0ss
(base) taylorswift@taylorswift-virtual-machine:~/github/fastText$ ./fasttext predict-prob model_FT1.bin - -1 0.5
^Z
[2]+  Stopped                 ./fasttext predict-prob model_FT1.bin - -1 0.5
(base) taylorswift@taylorswift-virtual-machine:~/github/fastText$ ./fasttext test model_FT1.bin FT1.valid -1 0.5
N       10200
P@-1    0.877
R@-1    0.518
(base) taylorswift@taylorswift-virtual-machine:~/github/fastText$ ./fasttext test model_FT1.bin FT1.valid -1 0.9
N       10200
P@-1    0.928
R@-1    0.492
```

Since we got multilabel data we tried to use multilabel functions loss one-vs-all to get a better Precision. From the above, we can clearly see that Precision rate increased from 0.847 to 0.877 with increase in word n gram from 2 to 3. We can even see that the model is now good at predicting all the labels with the respective probability assigned to each of the labels.