MACQUARIE University
SYDNEY·AUSTRALIA

Applications of Data Science (COMP8240)
Faculty of Science & Engineering

Major Project : fastText Implementation

Group S

Author's
Agam Kachhal (45762643)
Rohit Manral (45710864)
Shubham Rana (45812713)
Kripali Gandhi (45712158)

November 06, 2020

Abstract

Text classification is a fundamental obstacle to many applications, like spam detection, sentiment analysis or smart replies. The objective of text classification is to assign documents to one or multiple labels. To build such classifiers, we required datasets pre-loaded with labels or gold standard labelled data (Questions with their labels), which consists of documents and their corresponding categories (or tags, or labels).

Keywords: Text classification, Questions, labels

Figure 1: fastText ∼ Faster, better text classification

1. ## Why fastText:

   In our day to day life, we need to understand the meaning of words that roll off our tongue as we talk, or our fingertips as we tap out posts and this is one of the biggest technical challenges facing artificial intelligence researchers today. With the growing amount of data online data, there is a need for more flexible tools to better understand the content of very large datasets, in order to provide more accurate classification results. To exactly address this need the Facebook AI Research (FAIR) lab has open -sourced fastText which is a library to help achieve scalable and efficient solutions for learning of word representations and sentence classification.

2. ## Replication of original work:

   - ***Environment :*** AWS Virtual Machine and server - Ubuntu 18.04 LTS , Google Colaboratory
   - ***Code Management (Version Control, Code History):*** GitHub
   - ***For Reporting:*** Latex + Overleaf

   The Stack exchange cooking dataset has been obtained from Cooking section of Stackexchange. We are running fastText in the ***AWS Virtual Machine and server - Ubuntu 18.04 LTS – Hardware assisted virtualization, SSD volume type - ami-0dba2cb6798deb6d8 (64-bit x86) / ami-0ea142bd244023692 (64-bit Arm) with t2.micro instance along with a 30 GB's RAM***. In order to get this working, we needed labelled data to train our supervised classifier. We are interested in building a classifier to automatically recognize the topic of a Stackexchange question about cooking. We first downloaded the dataset as a .tar file and unzipped the file to get the file in .txt format which is required as an input to fastText. Each line of the text file contains a list of labels, followed by the corresponding document. All the labels start by the **__label__ prefix**, which is how fastText recognize what is a label or what is a word. The model is then trained to

predict the labels given the word in the document. Before training our first classifier, we needed to split the data into train and validation. We used the validation set to evaluate how good the learned classifier is on new data.

We trained a number of classifiers on the training data using multiple hyperparameters like **(epochs, learning rate, loss as hierarchical softmax, wordNgrams)** and we could see a increase in precision and recall based on the way we performed hyper parameterization. We got very close to perfect results as in the original cooking dataset mentioned here Original Results. Please refer to the GitHub Repository link for the .py script that we used to run in the Virtual Machine to replicate the original results which can be found in the GitHub Repository[1].As a note, we are not restricted to only providing the above-mentioned hyper parameters, so we can make changes in the script and play around with different parameters and check on the accuracy score thereafter.

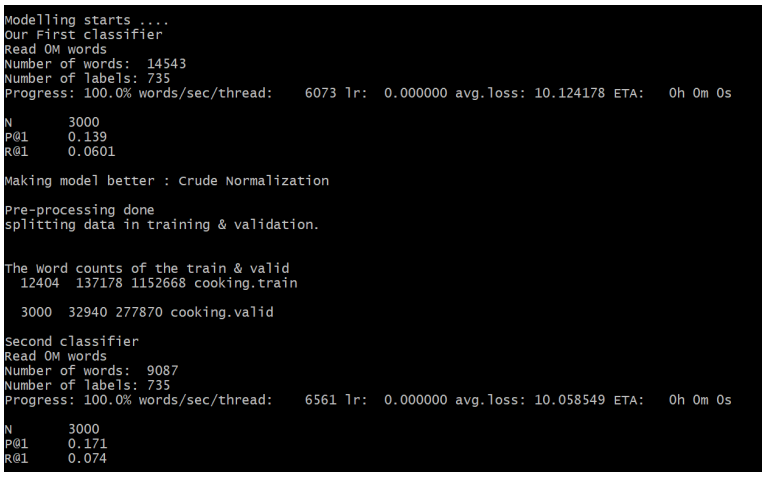


Figure 2: Original Work Results (Part_1)

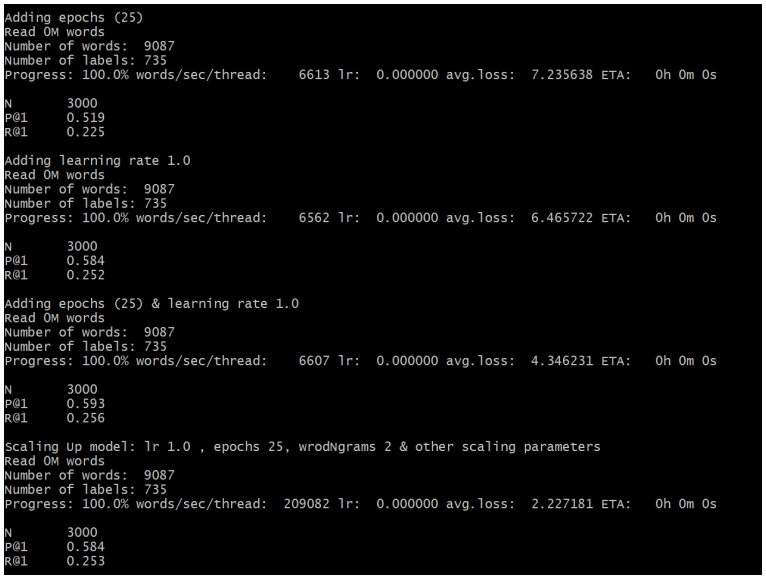


Figure 3: Original Work Results (Part_2)

[1]Python Script to Replicate Original work

3. Construction of New Datasets:

The original dataset on which fastText was run was the tutorial dataset i.e. Stack exchange Cooking Data. This contains examples of questions and the associated tags. Similar to the nature of this dataset, we constructed four different datasets, out of which three were from different stack exchange communities namely: Photography Dataset, Astronomy Dataset and Ask Ubuntu Dataset, and the fourth dataset consisted of StackOverflow Posts that was retrieved by querying the google cloud platform Big Query and it has multiple columns, out of which the main columns of interest were Questions(title column) and Tags. The data was fetched by querying the StackOverflow public database through Google Colab.

(a) Data Scraping using BeautifulSoup:

For the first three datasets that were from different stack exchange communities, we used Python's BeautifulSoup library to scrap Questions posted by people and their respective Tags. We utilized the Inspect element functionality for a respective webpage to identify the HTML tags that consisted of Questions and Tags which were of interest to us. For example, there was a tag with the class "Summary" that had a sub tag <h3> that consisted of the question description that we wanted to scrape. In addition, there was a tag with the class where all the respective tag names associated with the question were displayed. We created two empty lists in python for questions and tags and scraped data accordingly for multiple web-pages by iterating over all the required webpages using a for loop. Later, after data scraping, we converted the scraped Questions and Tags into a pandas Data frame and pre-processed the data by removing all the special characters, bad characters and then converting the data into a format which is accepted by fastText by prefixing label to each tag of the form _label_.

Challenge in Data Scraping:

After we finally scraped data for the three Stack exchange datasets mentioned, we witnessed that there were two rows in the Questions list which were getting scraped by default for each webpage. After analyzing in depth, we came to the conclusion that those were not questions posted by end users on the stack exchange platforms but was some default text that was being scraped and was appended to the top of each webpage. Due to this issue, the length of Questions and Tags was uneven and our fastText model was not running well due that inconsistency issue with Questions and tags and the accuracy scores were not good. We handled that using code to get rid of those two default questions, and we cross checked the lengths of Questions and tags post to which the lengths were same and then re-trained our fastText model and we could see a significant difference in the precision scores from our initial model. There was a major increase in the accuracies with around a 10-12% jump after this change was made.

i. **Photography Dataset:** [2]: It has 24,266 questions with labels. The data has been splitted into training and validation in ratio of 85:15 respectively.

	Questions	Tags
2	Is it possible for a camera to wear out if it ...	webcam|durability
3	Why does the iPhone SE (2016) have worse image...	image-quality|iphone
4	Are any of these photos any good? [closed]	canon|dslr|lighting|focus|macro
5	What should I be careful about while buying us...	equipmen|recommendation|fujifilm|used-equipment
6	How can I make part of an image transparent in...	lightroom|photo-editing

Figure 4: Photography Dataframe

ii. **Astronomy Dataset:** [3]: It has 9,792 questions with labels. The data has been splitted into training and validation in ratio of 85:15 respectively.

	Tags	Questions
2	__label__galaxy __label__galactic-dynamics	Calculate the absolute magnitude for a multi-s...
3	__label__gravity __label__rotation	Do the stars in irregular galaxies orbit anyth...
4	__label__star __label__temperature __label__sp...	By how much does Haumeas fast rotation affect ...
5	__label__star __label__galaxy __label__dark-ma...	Is there an O1 or O0 star?
6	__label__natural-satellites __label__gas-giant...	Evolution of galaxies with time

Figure 5: Astronomy Dataframe

iii. **Ask Ubuntu Dataset:** [4]: It has 21,000 questions with labels. The data has been splitted into training , validation & testing in ratio of 70:15:15 respectively.

	Tags	Questions
2	__label__apt __label__windows-subsystem-for-linux	I am currently trying to install CMake to Ubuntu-20.04 via WSL but I get dependency error
3	__label__shortcu __label__keys __label__keyboard-layout __label__xdotool	key map: combined keys to combined keys
4	__label__boot __label__nvidia	Ubuntu stuck at spinning logo at boot time
5	__label__drivers __label__nvidia __label__20004 __label__graphics	Fresh ubuntu 20.04 install but unclaimed nvidia graphic card?
6	__label__checksums	Unable to install ubuntu Checksum "No such file or directory"

Figure 6: Ask Ubuntu Dataframe

(b) Data Retrieval from Google Big Query:

BigQuery is a fully-managed, server less data warehouse that enables scalable analysis over petabytes of data. It is a Platform as a Service that supports querying using ANSI SQL. It also has built-in machine learning capabilities. We can access BigQuery by using the Cloud Console, by using the bq command-line tool, or by making calls to the BigQuery REST API using a

[2]Photography Data Analysis
[3]Astronomy Data Analysis
[4]Ask Ubuntu Data Analysis

variety of client libraries such as Java, .NET, or Python. We had used **pandas.io.gbq.read_gbq method to extract data** (the main method a user calls to execute a Query in Google BigQuery and read results into a pandas DataFrame).

We executed a SELECT query with Like functions on ? ("question mark") to retrieve 500K questions with tags along with other columns such as answer_count, created_at, user_id, many more.

StackOverflow Posts [5]: It has 100,000 questions with labels. The data has been splitted into training , validation & testing in ratio of 70:15:15 respectively.

	tags	title
0	__label__javascript __label__syntax-highlighting	How Code Color is Set in StackOverflow?
1	__label__xsd __label__guid __label__xsd.exe	What is the correct way of using the Guid type...
2	__label__perforce	How can I grab my local changelist and send it...
3	__label__c# __label__linq __label__distinct	Why is there no Linq method to return distinct...
4	__label__c# __label__.net __label__winforms __...	Is it Possible to Make a Generic Control in .N...

Figure 7: StackOverflow Posts

4. Results on New Data:

For all the four individual datasets from different Stackexchange communities and Stack overflow posts, we validated the accuracy scores achieved (the level of precision and recall) after implementing fastText. Looking at the data, we observed that some words contain uppercase letter or punctuation. One of the first step to improve the performance of our model was to apply some simple pre-processing. A crude normalization was obtained using command line tools such as sed and tr. After applying normalization, we could see that the model accuracy improved significantly on all the four datasets. After normalization, the next step was to train the fastText classifier using hyper parameterization which consists of epochs, learning rate, wordNgrams, bucket, dimensions and others.

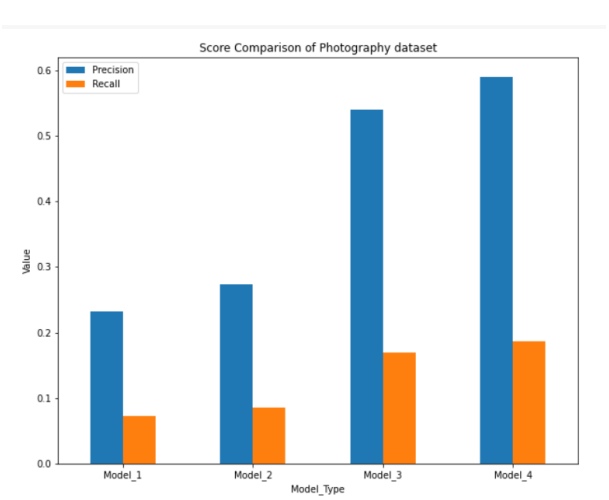
We increased the number of epochs for training different classifiers and we could clearly see that when the classifier saw each training example for 15 or 25 times during training, that resulted in an enhanced accuracy on the validation dataset. When we used wordNgrams, we saw that the precision went further up. With a few steps, we were able to go from a precision of around 15% to around 60% for all the four datasets. Including important steps:

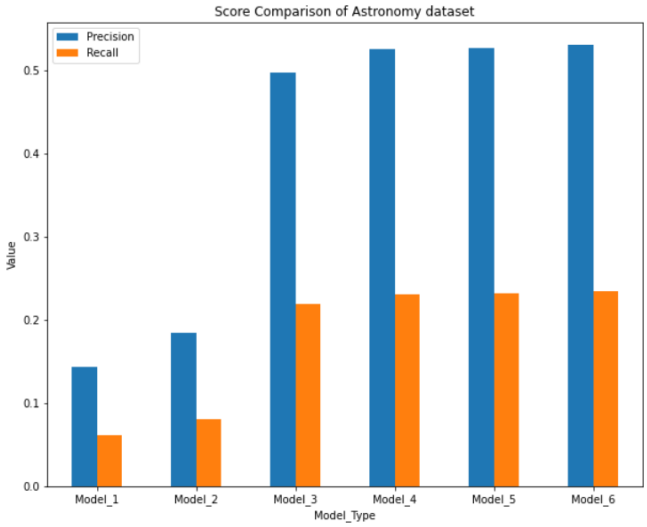
- preprocessing the data;
- changing the number of epochs (using the option -epoch, standard range 5 - 50);
- changing the learning rate (using the option -lr, standard range 0.1 - 1.0);
- using word n-grams (using the option -wordNgrams, standard range 1 - 5);
- using hierarchical softmax loss (hs);
- models scaled in the end using bucket and dimensions.

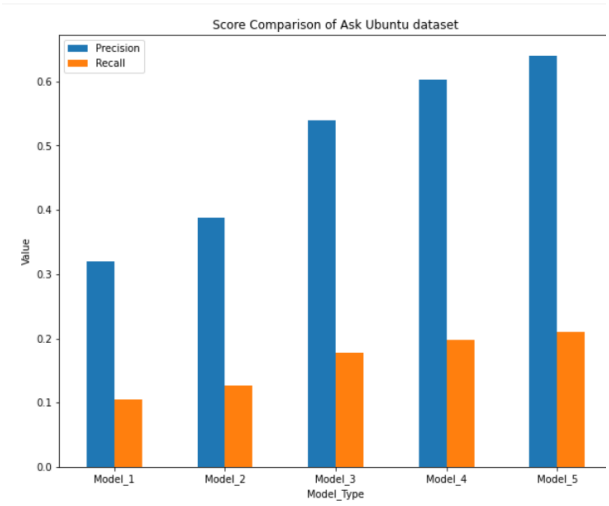
[5] StackOverflow Posts Data Analysis

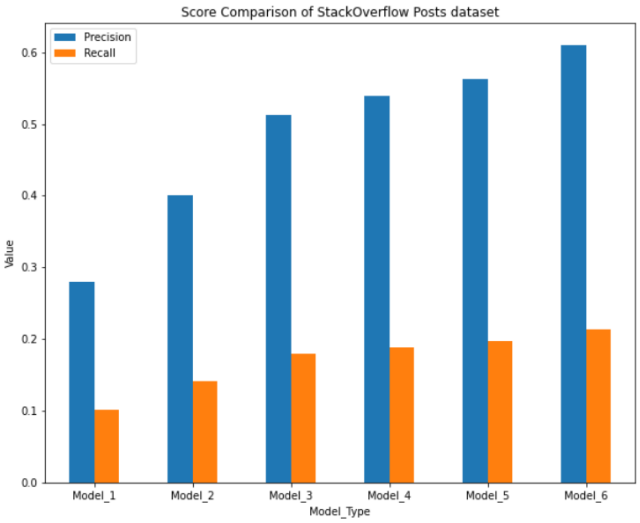
(a) **Results Visualisation**:

i. ***Below, we are displaying the scores (Precision and Recall) for all the four datasets post fastText implementation using a bar chart.***









ii. ***Below, we are displaying the scores (Precision,Recall) for all the four datasets post fastText implementation in comparison with Cooking dataset in a tabular format:***

Model	Cooking Data	Photography Dataset
Initial Model	(0.139,0.060)	(0.232,0.073)
Model (Pre-processed data)	(0.171,0.074)	(0.274,0.0861)
Model (25 epochs)	(0.515,0.223)	(0.54,0.17)
Model (with Scaling)	(0.588,0.254)	(0.59,0.186)

Table 1: Comparison of Scores - Cooking Data vs Photography Data

Model	Cooking Data	Astronomy Dataset
Initial Model	(0.139,0.060)	(0.143,0.0629)
Model (Pre-processed data)	(0.171,0.074)	(0.184,0.0812)
Model (25 epochs)	(0.515,0.223)	(0.498,0.219)
Model (25 epochs, lr)	(0.573,0.248)	(0.525,0.231)
Model (25 epochs, lr,softmax)	(0.573,0.248)	(0.527,0.232)
Model (with Scaling -softmax loss)	(0.575,0.249)	(0.531,0.234)

Table 2: Comparison of Scores - Cooking Data vs Astronomy Data

Model	Cooking Data	Ask Ubuntu Dataset
Initial Model	(0.139,0.060)	(0.32,0.105)
Model (Pre-processed data)	(0.171,0.074)	(0.388,0.127)
Model (15 epochs)	(0.515,0.223)	(0.54,0.178)
Model (25 epochs)	(0.515,0.223)	(0.602,0.198)
Model (with Scaling)	(0.588,0.254)	(0.64,0.21)

Table 3: Comparison of Scores - Cooking Data vs Ask Ubuntu Data

Model	Cooking Data	StackOverflow Posts
Initial Model	(0.139,0.060)	(0.28,0.102)
Model (Pre-processed data)	(0.171,0.074)	(0.40,0.141)
Model (15 epochs)	(0.412,0.178)	(0.511,0.179)
Model (25 epochs)	(0.515,0.223)	(0.548,0.192)
Model (25 epochs,lr,wordNgram)	(0.604,0.261)	(0.569,0.199)
Model (with Scaling)	(0.608,0.263)	(0.613,0.215)

Table 4: Comparison of Scores - Cooking Data vs StackOverflow Posts

Note:The above presented results in the table have been achieved after training fastText Classifier with same parameters on both the Cooking Data and the New Datasets.

Comments: As a general convention, trade-off means increasing one parameter leads to a decrease in the other and vice versa. Let us explain this in context to multiclass multilabel classification, i.e. classification where there are more than two labels, and where each instance can have multiple labels.

In case of our datasets, we are considering the multi-label classification and we can consider the following: η is the number of questions.

- Y_i is the actual label(s) assignment to the i^{th} question.
- x_i is the i^{th} question.
- $(\hat{Y_i})$ is the predicted label(s) for the i^{th} question.

Questions based Justification: The metrics are computed in a per question manner. For each predicted label its only score is computed, and then these scores

are aggregated over all the questions.

• Precision $=\frac{1}{n}\sum_{i=1}^{n}\frac{Y_i \wedge \hat{Y}_i}{\hat{Y}_i}$, The ratio of how much of the predicted is correct. The numerator finds how many labels in the predicted vector has common with the ground truth, and the ratio computes, how many of the predicted true labels are actually in the ground truth.

Precision is the fraction of correct positives among the total predicted positives. It is also called the accuracy of positive predictions $\sim \frac{TP}{TP+FP}$

• Recall $= \frac{1}{n}\sum_{i=1}^{n}\frac{Y_i \wedge \hat{Y}_i}{Y_i}$, The ratio of how many of the actual labels were predicted. The numerator finds how many labels in the predicted vector has common with the actual (as above), then finds the ratio to the number of actual labels, therefore getting what fraction of the actual labels were predicted.

Recall is the fraction of correct positives among the total positives in the dataset. It is indicating how many total positives of the actual dataset were covered(classified correctly) while doing prediction $\sim \frac{TP}{TP+FN}$

Definition of TP, FP, TN, FN:

1. TP: represents the true positives, which is the number of positive predictions which are actually positive.

2. FP: represents the false positives, which is the number of negative predictions incorrectly classified as a positive ,i.e. they were identified as positives though they were from a negative class.

3. TN: represents the true negatives, which is the number of negative predictions correctly classified as negative.

4. FN: represents the false negatives which is the number of positive instances incorrectly identified into the negative class, i.e. they were identified as negative but in reality they were from the positive class.

For our datasets, we have trained different models with different input parameters i.e. we haven't kept the same model for training every time and we have also changed threshold of the final model while training. Optimal Hyper parameters have been provided while training different classifiers, hence we saw that both Precision and Recall increased while applying better hyper parameters in our every new model.

To exemplify this,we trained StackOverflow Posts on different models with different hyper parameters. Thus, after applying some optimal hyperparameters in every successive model of Stack Overflow posts we noticed that both Precision and Recall increased simultaneously.

5. Reflection:

As per the noise in the dataset and the dataset is not perfectly separable, there might be some questions of one label closer to another label. In such cases, shifting the decision boundary can either increase the precision or recall but not both, increasing one parameter leads to decrease the other. In other words, multilabel classifier will miss classify some labels always. Miss classification means classifying a question from a particular label as another label this miss classification rate is

either compromising precision or recall score.Note that the increasing or decreasing threshold is similar to shifting the decision boundary.

To cite this, in terms of the Stack Overflow posts, when we increased the threshold or probability of the final model to 0.5 while training, the Precision skyrocketed to 93% from 61% and the Recall decreased moderately to 4.29% from 21.4%. (We had trained 70K(70%) questions and check scores on 15K(15%) validation records.)

6. References:

1) https://fasttext.cc/docs/en/supervised-tutorial.html

2) https://www.dataquest.io/blog/web-scraping-beautifulsoup/

3) https://stats.stackexchange.com/questions/21551/how-to-compute-precision-recall-for-multiclass-multilabel-classification

4) https://cloud.google.com/bigquery/docs/introduction

5) https://medium.com/opex-analytics/why-you-need-to-understand-the-trade-off-between-precision-and-recall-525a33919942