Natural Language Processing (CSE4022)

Digital Assignment - 1

Name: Anmol Pant

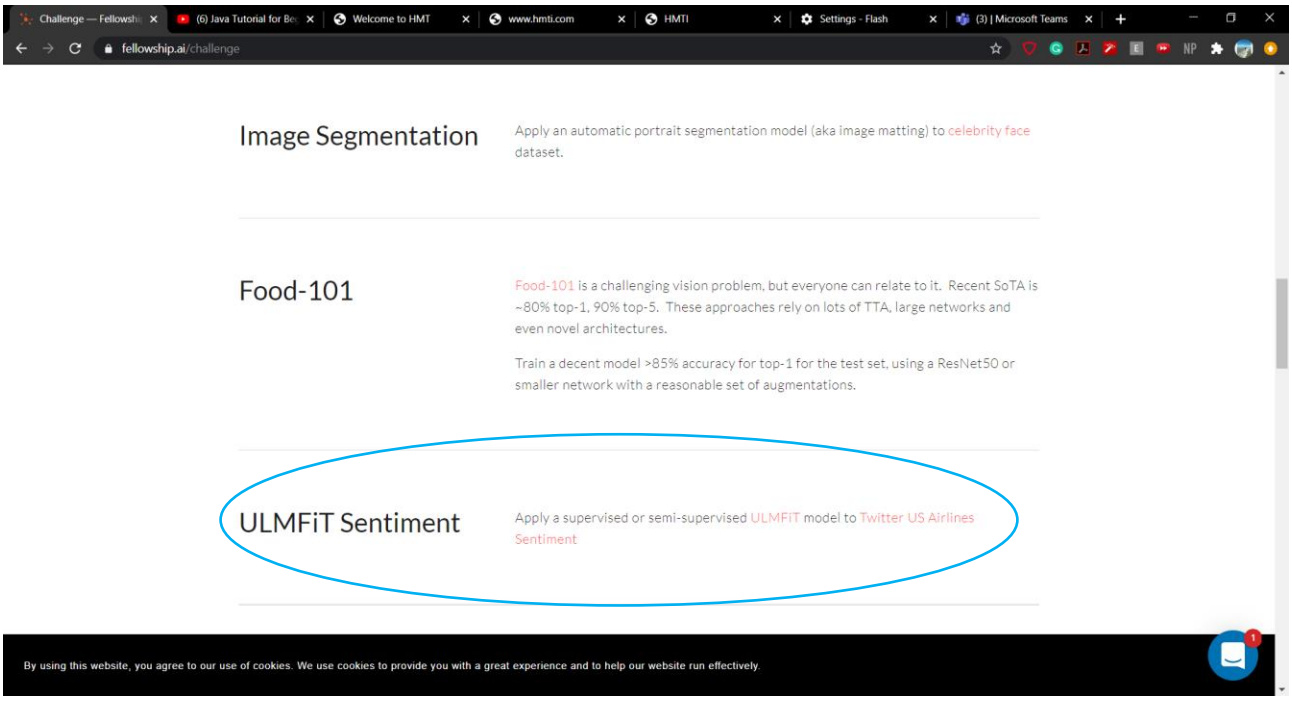
Reg. No: 18BCE0283

Slot: E1 + TE1

Faculty: Sharmila Banu K

Topic: ULMFiT Sentiment Challenge for Fellowship.ai

In the early days of the lockdown I was a candidate to be a part of the Asia AI fellowship offered by **fellowship.ai**. After clearing the resume round, I and the other candidates were given an individual challenge problem to work on, we had 10 days to select and work on a particular problem out of the list of challenge problems listed on their website. The details about the fellowship program can be checked out in the link of the website provided in the reference section [1].



Above are some of the problem challenge topics posted on their official website [2]. The topic I chose was sentiment analysis using the **ULMFiT** technique, unlike the conventional sentiment analysis approach this makes use of a semi supervised RNN deep learning model for sentiment analysis. The semi-supervised ULMFiT model can be implemented with the help of the fastai [5] library for text preprocessing and data manipulation, which I was completely unfamiliar with, hence to get a working implementation up and running within a span of 10 days, was definitely a challenge. My submission did not make the final cut in this fellowship competition as it is one of the most competitive ml fellowship programs out there, selecting only about 10-15 practitioners out of the hundreds that apply. But this competition was a learning venture like none other and hence I decided to select the same as the subject matter for this assignment.

1. Original Writing/Analysis

The method used in the above challenge problem is inspired from the official UMLFiT paper [3]. To understand what ULMFiT is, one must first familiarize oneself with the basics of Language Modeling [6]. A language model is a probability distribution over sequences of words. The language models can then be used as base models for various natural language processing tasks including text classification, summarization, analysis and more.

Universal Language Model FIne-Tuning (ULMFIT) is a transfer learning technique which can help in various NLP tasks. It has been state-of-the-art NLP technique for a long time, before it was dethroned by techniques like BERT and XLNet.

According to the official ULMFiT paper [3], the advantage it boasts of over its counterparts is the fact that deep learning techniques, that often require very large datasets, tend to overfit when the dataset [4] given to us is relatively smaller and domain specific, like in this case. ULMFiT helps address this very predicament by first breaking our data into batches known as a databunch, we then fit and train our deep learning model for a few cycles by running 1 epoch at a time, then unfreezing some layers and running subsequent epochs to fine tune the same, instead of the conventional approach of model training by running all epochs at once. The model is saved after every epoch, while adjusting the hyper-parameters after every epoch to help our model learn various edge cases in our test set better.

2. My Work

Under this challenge, I was expected to automate the detection of different sentiments from textual comments and feedback and perform sentiment analysis using the ULMFiT method on the crowdflower dataset of United States airline Sentiment [4].

I. **Solution Approach**: As both the dataset [4] and the technique to be used for sentiment analysis were already given to me in the problem statement, the first few steps consisted of the usual data loading, splitting and pre-processing by plotting the various correlations between the different parameters given to me and filtering out the most important ones, followed by tokenization, removal of stop words and formulating regular expressions to weed out usernames and links from the tweets that might give the model an inaccurate picture of the dataset.
Next I trained two language models making use of the ULMFiT method in the fastai library that makes use of LSTMs and RNNs to first predict the word that might follow the given word and then the sentiment associated with the whole block of text in general. By following this approach, I could achieve an overall accuracy of over 82.2% on unseen data i.e. the test set.

II. **Assumptions**: I have made the safe assumption that the only necessary columns from the dataset are the text and its associated sentiments and hence I dropped columns like 'airline_sentiment_confidence' which are not relevant in the long run. I have also assumed that the dataset, being an informal one, contains abbreviations and unconventional word combinations and hence split these abbreviations into their root words.

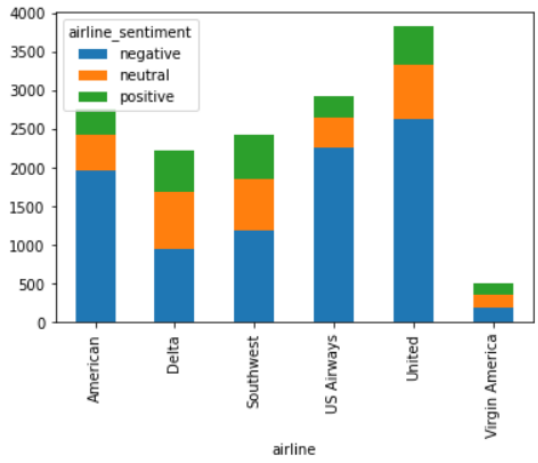
III. **Algorithms**: The approach and algorithm adopted is taken from the ULMFiT research paper which makes use of data bunches to train two language models, the first one tries to predict the word that might follow the word at hand and the second language model predicts the overall sentiment. To train these language models we train by gradually unfreezing layers of our neural network and training our

model one epoch at a time instead of performing all the epochs at once, in accordance to the suggestions in the ULMFiT paper [3]. This method of training not only ensures better accuracy, but also helps our model learn the edge cases to perfection.

IV. **Project Code Snippets and diagrams**: The prominent code snippets apart from model training were mainly the ones that plotted graphs that analyzed various trends and correlations and were also used for setting hyper parameters to further fine tune our model.

Attached herewith are some of the screenshots and a brief insight into the role these diagrams played in our project.

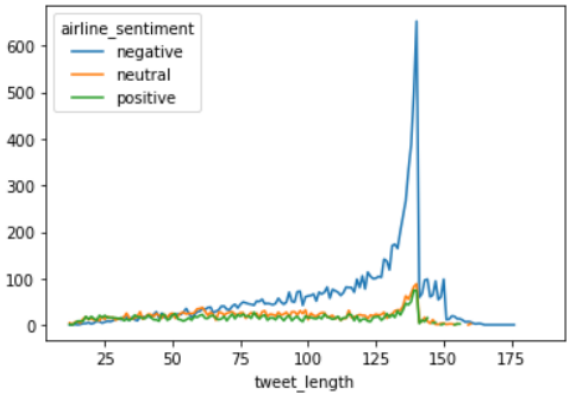
```
df_tweets.groupby(['airline','airline_sentiment']).size().unstack().plot(kind='bar', stacked=True)
```



Airline vs the kinds of sentiment associated with them.

```
df_tweets['tweet_length'] = df_tweets['text'].apply(len)
df_tweets.groupby(['tweet_length','airline_sentiment']).size().unstack().plot(kind='line', stacked=False)
```

Tweet length vs sentiment.



Language Model

In [30]:
```
#Learn = language_model_learner(data, AWD_LSTM, drop_mult=0.3)
```

Defining the language model and setting the leaning rates.

In [31]:
```
learn = language_model_learner(data, AWD_LSTM, drop_mult=0.5, model_dir='/tmp/models')
learn.freeze()
```

```
Downloading https://s3.amazonaws.com/fast-ai-modelzoo/wt103-fwd.tgz
```

In [32]:
```
learn.lr_find()
```

50.00% [1/2 00:08]

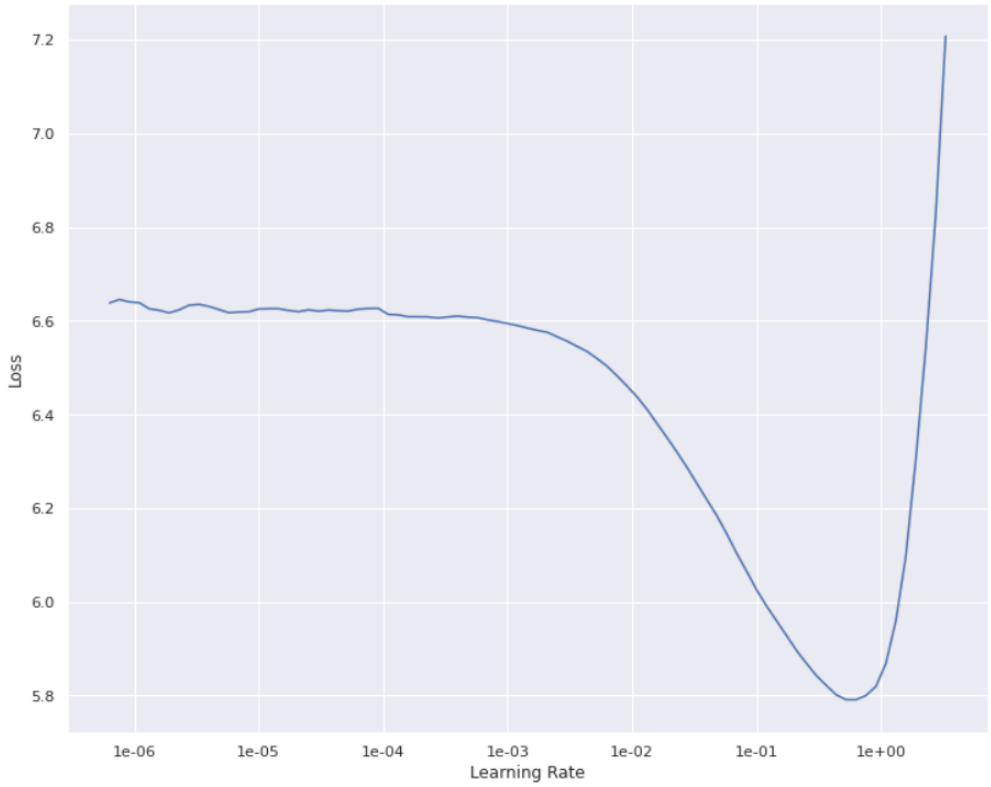
epoch	train_loss	valid_loss	accuracy	time
0	6.184167	#na#		00:08

37.50% [27/72 00:02]

```
LR Finder is complete, type {learner_name}.recorder.plot() to see the graph.
```

In [33]:
```
learn.recorder.plot()
```

Initializing the language model, setting rates and plotting the Loss function vs learning rate curve for first language model (to fine tune alpha).



Loss function vs learning rate for first language model (to fine tune alpha)

```
In [44]: learn = text_classifier_learner(data_classifier, AWD_LSTM, drop_mult = 0.5, model_dir = '/tmp/m
         learn.load_encoder('fine_tuned_enc')
         learn.freeze()

In [45]: learn.lr_find()
```

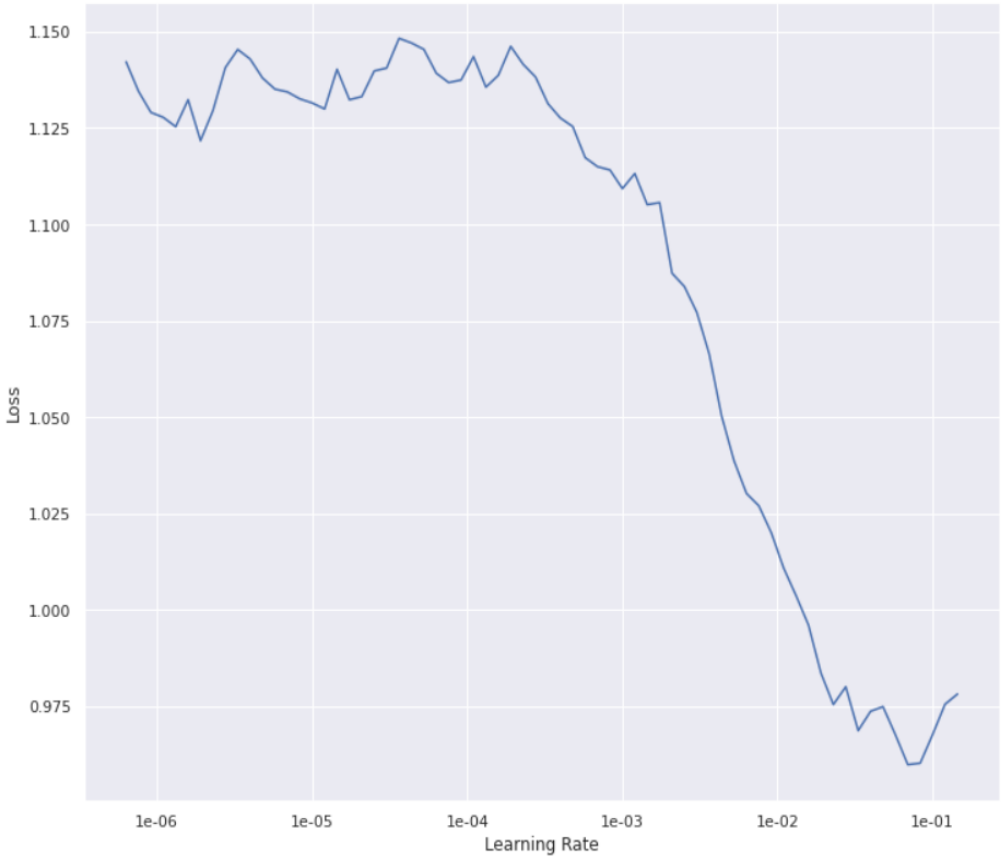
0.00% [0/1 00:00]

epoch	train_loss	valid_loss	accuracy	time

17.00% [84/494 00:01]

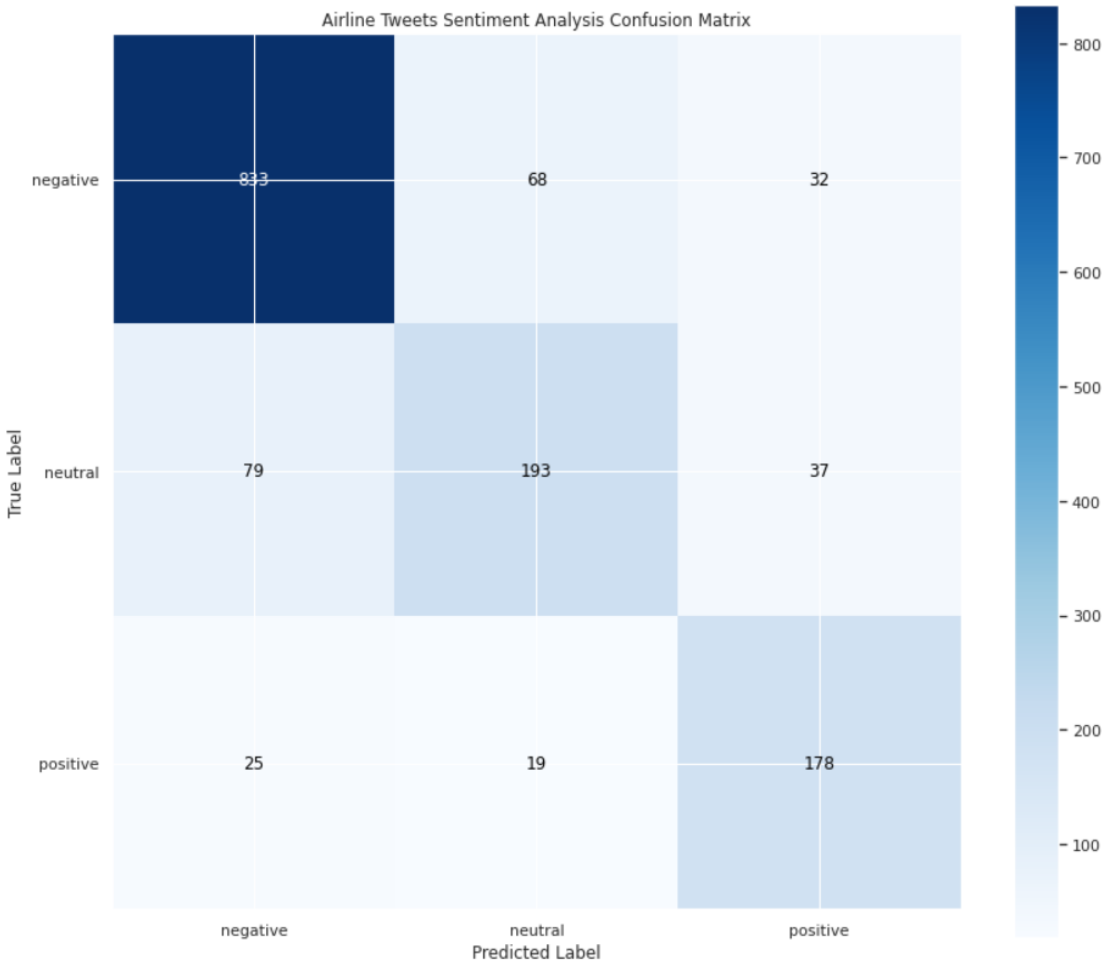
```
LR Finder is complete, type {learner_name}.recorder.plot() to see the graph.

In [46]: learn.recorder.plot()
```

Model fine tuning.



Loss function vs learning rate curve for our final fine-tuned predictive model that detects the sentiment of the text given.

The other code snippets on model training and evaluation on test set can be found in the Jupyter Notebook in the GitHub Repository [7].



Confusion matrix obtained after evaluating on the test set – shows us that our model performs reasonably well and can generalize to unseen data having a high accuracy for true positives and true negatives.

Since the challenge problem submission was a very lengthy one, it is not possible for me to attach all the code snippets in the limited scope of this digital assignment. The fellowship accepted submissions in the form of a GitHub Repository [7], the link to which I've enclosed in the reference section. The repository has been initialized with a readme explaining the approach I adopted and the associated Jupyter Notebook file is also extremely well documented, explaining almost every step, in accordance to the guidelines issued by the fellowship.ai team.

3. Choice of topic and challenges faced.

Being unfamiliar with both the fastai library and the ULMFiT approach, getting a working implementation of the project running in mere 10 days was definitely a challenge, despite that I managed to achieve an overall accuracy of about 83% on the test set. Although the submissions that got selected boasted of over 89-90% accuracy and my entry didn't make the cut, it was a great learning venture nonetheless.

Also some challenges that I faced working with informal social media data like tweets, were more because of the nature of the data and the anomalies of the dataset, for example, some of the tweets carried negative sentiment aimed at other twitter users instead of the airline. In addition, there are other things to consider with informal writings such as sarcasm, and improper grammatical structures and short forms that might end up confusing our model.

Moreover some of the negative tweets have been mislabeled in the dataset as neutral and vice versa, which can be another factor hampering the accuracy of our model.
The overall accuracy is good for true positives, but some of the positive tweets do get classified as negative.

4. Conclusion and scope of improvement

In the future it could be a good idea to spend more time on feature engineering to account for improper grammatical structures and misspellings while fitting the language model, so that the model hence formulated is more suited to the vocabulary of the language used in informal writings like tweets. Even as humans, it is sometimes hard to judge the sentiment of a given tokenized piece of text and hence our model does a good job in predicting the labels with about 82% accuracy. One possible way of handling the accuracy issue could be to make use of a much larger corpus with maximum possible training examples that are likely to be encountered in tweets when developing the initial language model.

5. References

[1] https://fellowship.ai/

[2] https://fellowship.ai/challenge

[3] https://www.aclweb.org/anthology/P18-1031.pdf

[4] https://data.world/crowdflower/airline-twitter-sentiment

[5] https://www.fast.ai/

[6] https://towardsdatascience.com/understanding-language-modelling-nlp-part-1-ulmfit-b557a63a672b

[7] https://github.com/anmolpant/ULMFiT-Sentiment

* * *