**A brief text documentation**

**Last week link description:**

* Initially while submitting last week’s project initial report, our topic that we submitted was ‘Sentiment Analysis Using Twitter Data’ and we had created a developer account and waiting for twitter to approve it. We waited for their email till today morning,but unfortunately, we did not receive any kind of response from them and so we choose to keep the methodology and techniques same but use Amazon review data for out Sentiment Analysis.

**This week’s progress**

**Project Description:**

* We are implementing Sentiment Analysis using Amazon reviews. We are using datasets to fetch data and implement text segmentation using NLP. The text is the reviews and it will be divided into positive, negative and neutral review also, some special characters like hashtags etc. Using various models, our goal is finding the best accuracy using a best fit model.
* To perform text reading, the wording is in a simplified format. The first word gives the mark, so we need to translate that to a number and then take the rest as the statement.
* For data cleaning, normalization is performed using a regular expression.
* A regular expression is a special sequence of characters that helps you match or find other strings or sets of strings, using a specialized syntax held in a pattern. Regular expressions are widely used in UNIX world. The module re provides full support for Perl-like regular expressions in Python.
* The first thing we must do to process the text is to lower it all, and then delete non-word characters. After replacing these with spaces as most would be punctuation. Only then, delete any other character.
* Initially we decided to use lexicon method for the sentiment analysis, where in bag of words are created from the data and performed text analysis using NLP.
* As by research we found that using tokenization, results are more accurate and faster and we decided to move ahead with it.
* Tokenization is the process of tokenizing or splitting a string, text into a list of tokens.
* Tokenizer in out project is running using the top 12000 words as features.
* To efficiently use sets, we would need to take the sequences and convert them into the same-length sequences. I would only make it the longest sentence in the training package here.
* We first perform Convolutional Neural Net model where in it has 64, 3-layer embedding with the first two matching normalization and max pooling. The results are the dense layers followed by the output.
* We perform the Recurrent Neural Net model on the train and test datasets and we get accuracy score of 95%.
* Given a character, or a sequence of characters, what is the most probable next character? This is the task we're training the model to perform. The input to the model will be a sequence of characters, and we train the model to predict the output—the following character at each time step.
* Since RNNs maintain an internal state that depends on the previously seen elements, given all the characters computed until this moment, the next character can be predicted.

**Reference:**

* https://www.kaggle.com/bittlingmayer/amazonreviews/kernels
* https://michael.hahsler.net/#Teaching

**Group member contributions:**

* We are group of three namely:

Shrutika Mokashi

Yashmi Sevak

Janaki Sajja

* Reading of text and Text processing was implemented by Me Yashmi Sevak.
* Followed by Shrutika Mokashi who performed the Convolutional neural net model
* And lastly the accuracy using Recurrent neural net model was performed by Janaki Sajja