Proposal :Tweet Sentiment Extraction

Pavan Poosarla ; Springboard Capstone 2

Introduction

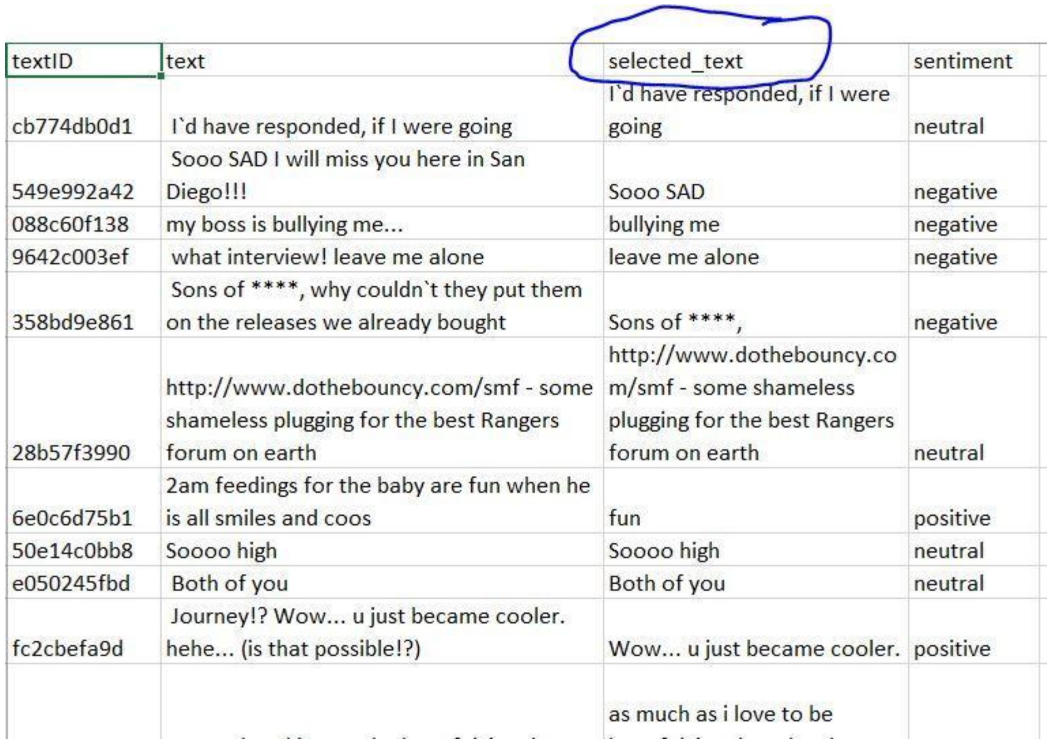
The problem we attempt to solve is an extension of the classical sentiment analysis problem for twitter data. In this problem, we would like to pick out the exact phrase of the tweet that informs the sentiment, given the sentiment. The inspiration and data for this problem is obtained from the Kaggle Competition in the link below

Link : https://www.kaggle.com/c/tweet-sentiment-extraction/data

Problem Statement

Given a tweet and the sentiment associated with it, determine the best words in the tweet that capture the sentiment

For example, in the snip of the dataset below, predict the column that is annotated with blue circle



textID	text	selected_text	sentiment
cb774db0d1	I`d have responded, if I were going	I`d have responded, if I were going	neutral
549e992a42	Sooo SAD I will miss you here in San Diego!!!	Sooo SAD	negative
088c60f138	my boss is bullying me...	bullying me	negative
9642c003ef	what interview! leave me alone	leave me alone	negative
358bd9e861	Sons of ****, why couldn`t they put them on the releases we already bought	Sons of ****,	negative
28b57f3990	http://www.dothebouncy.com/smf - some shameless plugging for the best Rangers forum on earth	http://www.dothebouncy.com/smf - some shameless plugging for the best Rangers forum on earth	neutral
6e0c6d75b1	2am feedings for the baby are fun when he is all smiles and coos	fun	positive
50e14c0bb8	Soooo high	Soooo high	neutral
e050245fbd	Both of you	Both of you	neutral
fc2cbefa9d	Journey!? Wow... u just became cooler. hehe... (is that possible!?)	Wow... u just became cooler.	positive
		as much as i love to be	

Motivation

This is a further value addition from sentiment analysis when the company/ government or person interested in sentiment analysis wants to have ‘human-readable’ descriptions associated with the sentiment in question.
In addition to the application to this specific Kaggle project, the methods explored in this competition will have broader applications in NLP. This can be an approach for text summarization, where large sections of text are reviewed to select information related to a particular theme. For example, this can be used to process customer feedback text to summarize the large body of feedback text while capturing the reason for given rating. In this example, we may also be able to extract specific excerpts of the reviews related to user experience, reliability, etc.

Data Sources

Data sources used will be in the link below. The training data consists of more than 27000 entries containing the tweets, sentiment associated with the tweets and the selected text associated with the sentiment. In addition to these, there is a unique *textID* associated with each entry.
LInk : https://www.kaggle.com/c/tweet-sentiment-extraction/data

Initial Approach

The initial approach for attacking the problem is as follows:
1. Data Wrangling: Use regex on tweets to capture URL’s , substitutions for expletives(***, F***, etc.). We would also need to correct typos (potentially separate intentional misspelling from unintentional), etc. However, we need to do this without losing the original text

2. Exploratory Data Analysis: We need to identify which words appear strongly in positive and negative connotations. We will need to treat the words that can be used in both contexts differently

3. Analysis : Plan to use a two part approach:

   a. classify all words into two categories - 1. Which have strong sentiment and which do not: One potential way to do this is to use Naive Bayes on bag-of-words vectorizer on the dictionary
   b. Next, we will use parse trees approach to get the section of text containing the selected word as the selected text. The larger text around the word can be obtained by extracting the noun-chunk containing the word or selecting a fixed number of tokens neighbouring the selected word.

Data Wrangling

The raw dataset consists of four columns, one is a unique textID for each data point. One of the column is the full text of the tweet. The remaining two columns are for sentiment associated with the tweet and the selected text from the full tweet that expresses the said sentiment.
The first step in data cleaning is to look for and deal with missing values in dataset. We see that one row has the tweet and selected text missing. This row is dropped, which leaves us with 27480 lines of training data
The next step is to create a cleaned text dataset with the following cleaning steps

1. Remove leading and lagging white space
2. Remove URL's in the text and extract them into a different column consisting of list of url's
3. Convert all text into lower case

In addition to the cleaned_text column consisting of the tweet preprocessed with above steps, we create the following additional columns in the training dataset

1. *word_count* : Contains the number of words contained in the text
2. *sen_count* : Contains the number of sentences in the tweet
3. *url_list* : List of urls in tweet. List is [ ] for tweets with no url's
4. *char_count* : Number of characters in the tweet
5. *selected_char_count* : Number of characters in *selected_text*
6. *jaccard* : Lists the jaccard score of the original text and the selected text

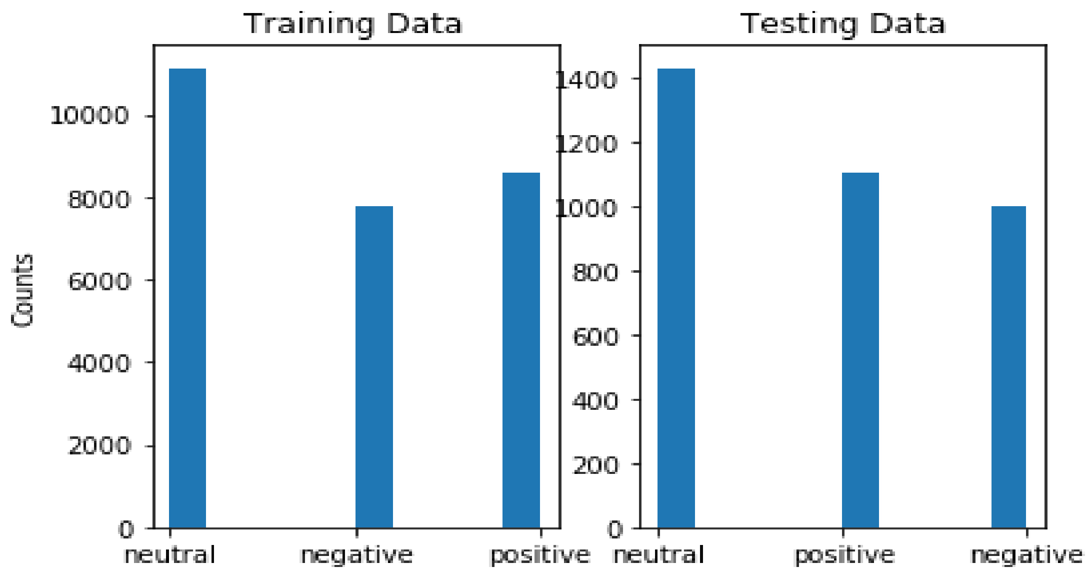


Figure Distribution of the number of tweets by sentiment in the training and test data

Exploratory Data Analysis

Once the dataset is cleaned, we look at the trends in the data. First, lets look at the distribution of the number of tweets with each sentiment in training and the text data. As can be seen from the figure, we do notice that the classes are imbalanced. The number of tweets with neutral sentiment seem to be higher for both test and train dataset

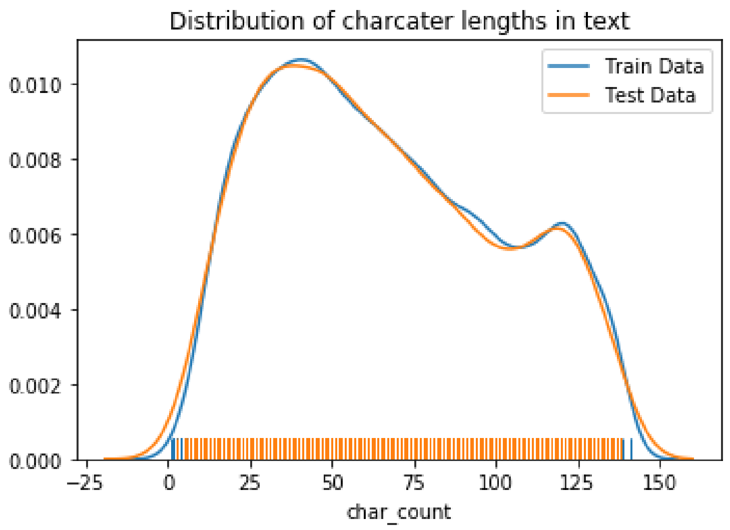


Figure Distribution of character counts in training data and the test data

Similarly, we see that the distribution of character lengths in tweets is also similar between the test and training set, as shown in the figure. Furthermore, the character count distribution does not seem to be impacted by the sentiment of the tweet. However, the distribution of character counts in selected text shows a dependence on the sentiment. It appears that for neutral tweets, most of the actual tweet is selected while for the positive and the negative tweets, only a small part of the tweet is selected for most cases.

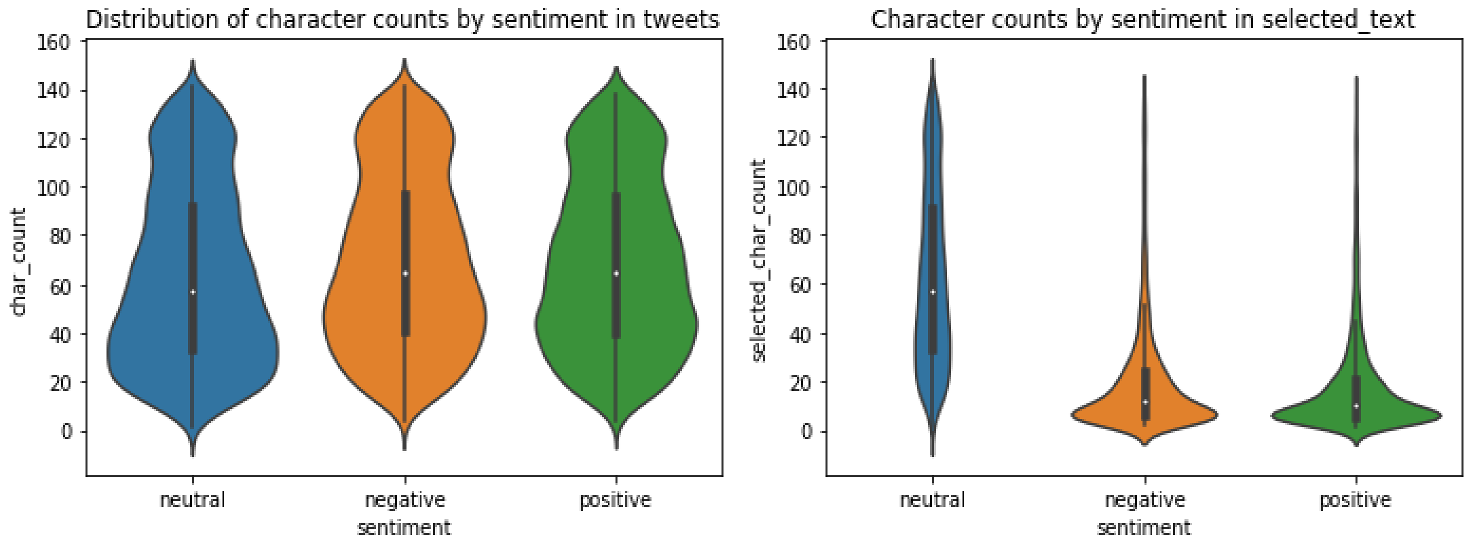


Figure Distribution of character lengths by sentiment in the full tweet and in the selected text

This is also shown in another way in the scatterplot of the character counts of selected text plotted against the main tweet. As we expect, we see that most of the data points for positive and negative tweets lie toward the right-bottom corner of the plot. This shows that in most cases, the selected point is smaller than the main tweet for positive and negative sentiments.

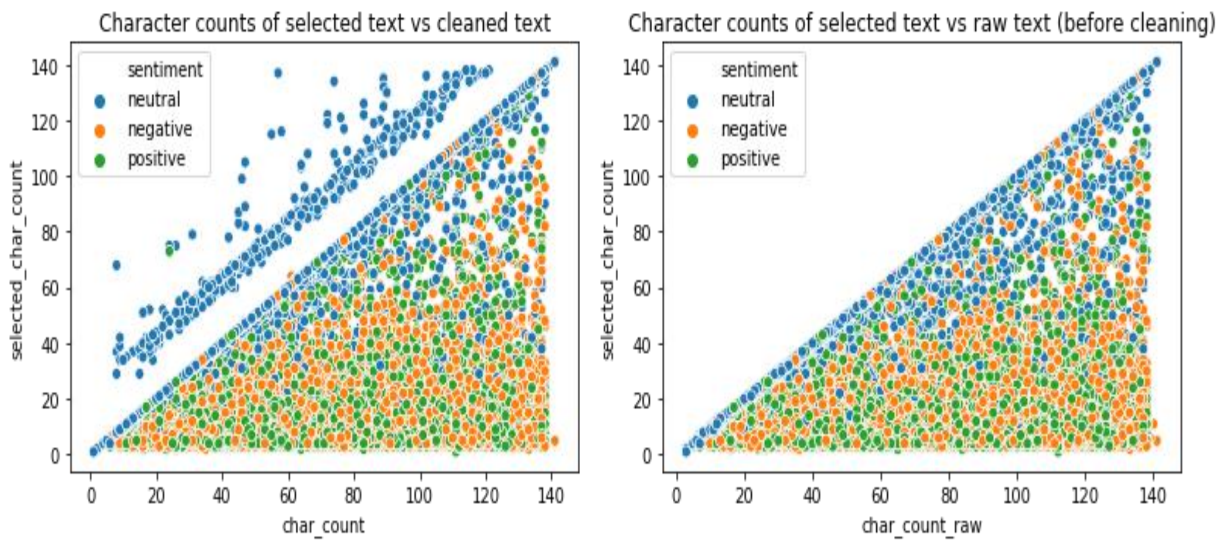
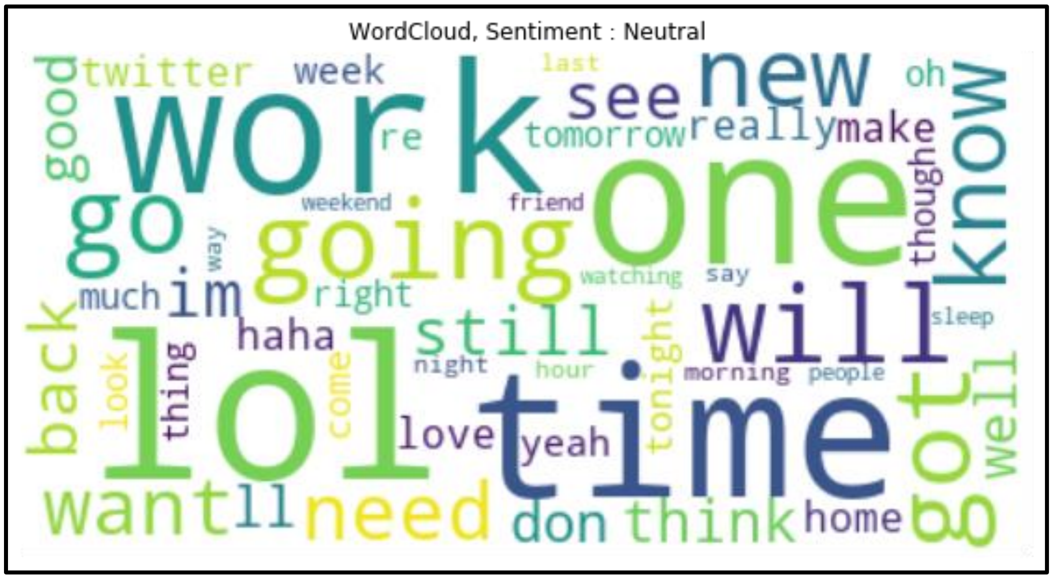


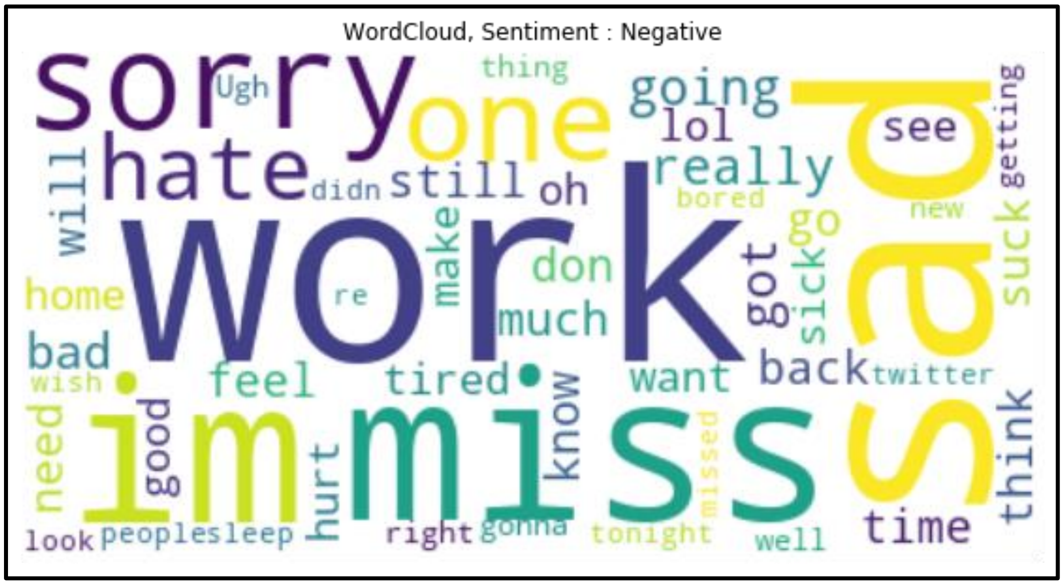
Figure Scatter plot of character counts in selected text vs main text, with character counts in tweet extarcted after removing URL's and before

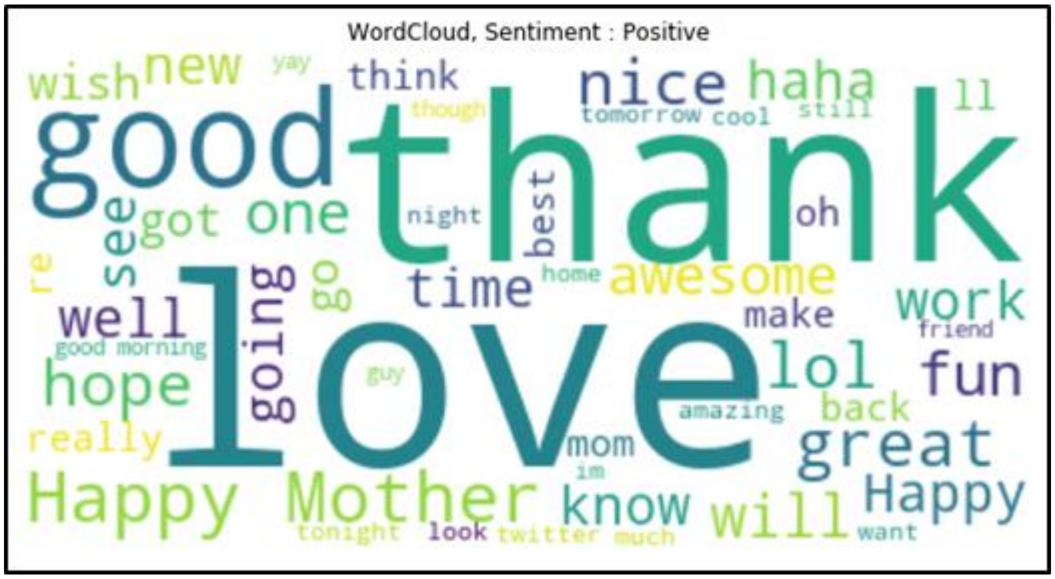
However, we see that for neutral sentiment, we see a lot of data points on the 45-degree curve, suggesting that in those cases, the entire text of the tweet is selected to capture the sentiment. In addition, we see a few data points where the selected data has more characters than the main tweet. These correspond to the tweets with URL's embedded in the text body. For these tweets, the selected text seems to have included the URL"s. Because we removed the URL's from the main text but not from the selected text, we see this artifact of the selected tweet having more characters than the main tweet. This is no longer the case, as we can see, when the character count of main tweets is extracted without removing the url.

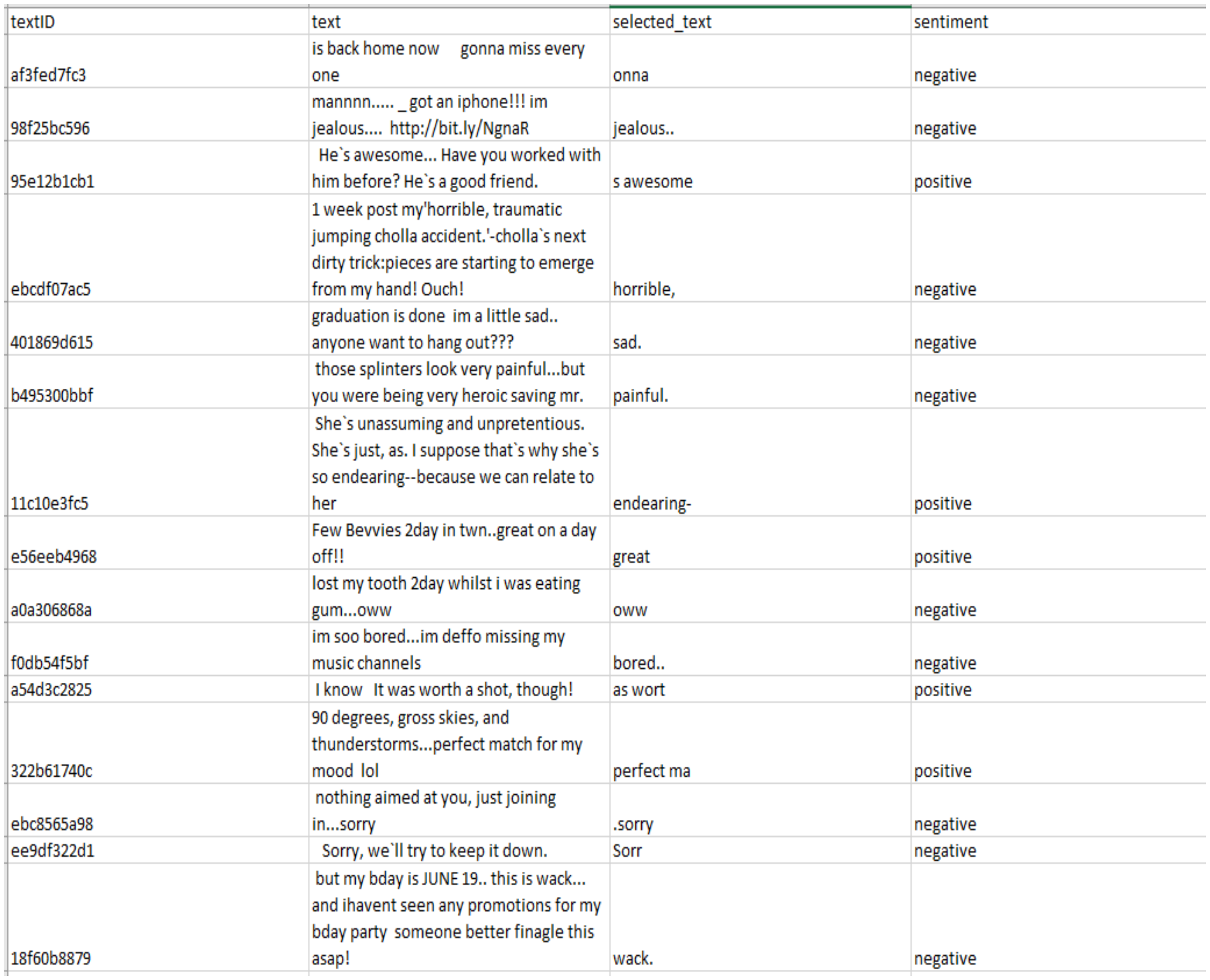
Finally, we expect the most common words associated with each of the sentiments to be different. To visualize this, we plot word clouds for each of the sentiments in the talks. We find that the words in tweets associated with positive sentiment have 'happy' words more often and tweets with negative tweets have more occurrences of words with negative connotations. For example, the most common words associated with positive sentiment are ***love, thank, good, great, happy, fun, haha, hope, etc***. On the other hand, common words in the negative tweets are ***word, miss, sad, sorry, hate, etc***. Word *work* also commonly appears in neutral tweets.

Other common words in neutral tweets are ***lol, time, going, etc.*** The word clouds for the sentiments are shown below









textID	text	selected_text	sentiment
af3fed7fc3	is back home now gonna miss every one	onna	negative
98f25bc596	mannnn..... _ got an iphone!!! im jealous.... http://bit.ly/NgnaR	jealous..	negative
95e12b1cb1	He`s awesome... Have you worked with him before? He`s a good friend.	s awesome	positive
ebcdf07ac5	1 week post my'horrible, traumatic jumping cholla accident.'-cholla`s next dirty trick:pieces are starting to emerge from my hand! Ouch!	horrible,	negative
401869d615	graduation is done im a little sad.. anyone want to hang out???	sad.	negative
b495300bbf	those splinters look very painful...but you were being very heroic saving mr.	painful.	negative
11c10e3fc5	She`s unassuming and unpretentious. She`s just, as. I suppose that`s why she`s so endearing--because we can relate to her	endearing-	positive
e56eeb4968	Few Bevvies 2day in twn..great on a day off!!	great	positive
a0a306868a	lost my tooth 2day whilst i was eating gum...oww	oww	negative
f0db54f5bf	im soo bored...im deffo missing my music channels	bored..	negative
a54d3c2825	I know It was worth a shot, though!	as wort	positive
322b61740c	90 degrees, gross skies, and thunderstorms...perfect match for my mood lol	perfect ma	positive
ebc8565a98	nothing aimed at you, just joining in...sorry	.sorry	negative
ee9df322d1	Sorry, we`ll try to keep it down.	Sorr	negative
18f60b8879	but my bday is JUNE 19.. this is wack... and ihavent seen any promotions for my bday party someone better finagle this asap!	wack.	negative

Figure Sample of dataset

Note About Dataset

We do notice that the raw dataset is not of the best quality, especially the field for the selected_text. We find several problems

1. The words in selected text are sometimes parts of words of the main text. This is often times not accurate. For example, in first example in the dataset example, “onna”(mis-spelled from *gonna*) is considered to be the negative word. However, we would expect it to be “miss every one”.
2. Sometimes, punctuation causes the jaccard score to be zero as the accuracy metric provided does not remove punctuation before jaccard is calculated.

Based on these observations the takeaway is that we should not do any cleaning of the text before sending it through the models. However, as the selected text is often times few words, there is potential to train this using bag-of-words approach and using a carefully designed post processor.

Model Development: Bag-of-Words Approach

The first method to getting the selected text is to treat the tweets as use bag-of-words approach to identify the best word predictive of the sentiment in the tweet and use rule-based post processing to select the text indicative of the sentiment. We follow the following steps to get the selected text

1. Train a classifier model to predict the sentiment of tweets given the text based on a bag-of-words approach
2. Obtain the probabilities of each word in the tweet given the sentiment using the *predict_proba* method of the classifier
3. Get the word in the tweet with highest probability for the sentiment as the selected word
4. Use rule based approach to expand the word to a portion of text to improve average jaccard score between predicted and selected texts on training dataset

Training Classifier and getting best predictive words for each class

We use a logistic regression classifier on the sentiment of the tweets using a feature vector obtained from TF-IDF vectorization of the tweets. The text is not cleaned before vectorization other than stripping whitespace from ends and converting all text into lower case. The best hyperparameters were selected by grid search cross-validation and is found to have a ROC-AUC score of 0.835 on the test set. The complete classification report is reproduced below

Classification Report of Logistic Regression Sentiment Classification Model

```
          precision    recall  f1-score   support

      0       0.68      0.66      0.67      2313
      1       0.65      0.68      0.67      3339
```

```
           2       0.76      0.73      0.74      2592

    accuracy                           0.69      8244
   macro avg       0.70      0.69      0.69      8244
weighted avg       0.69      0.69      0.69      8244
```

The confusion matrix of the same model is shown in the figure.

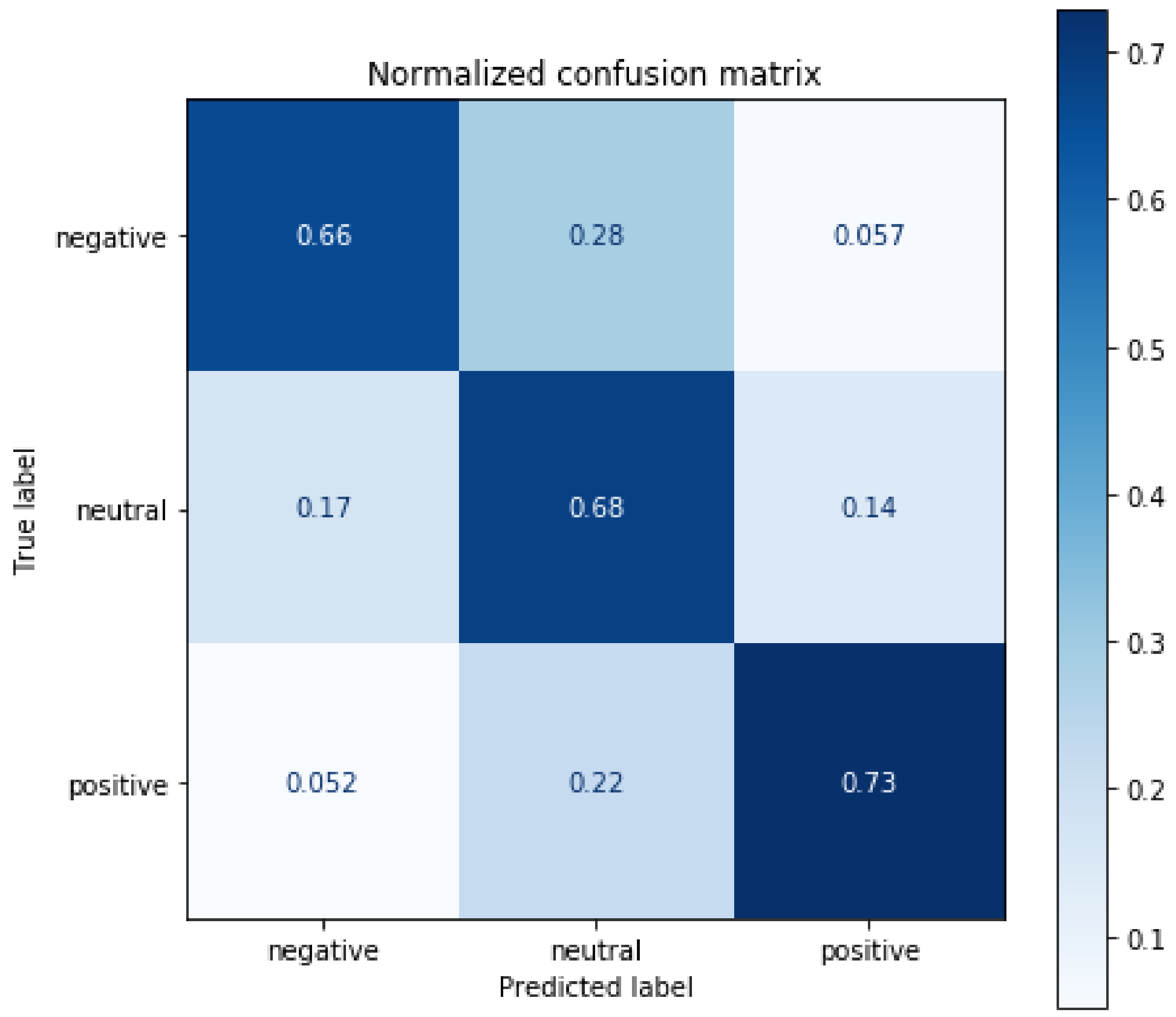


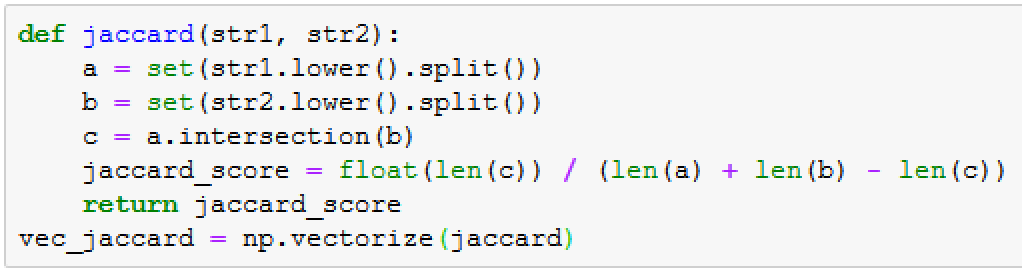
Figure 5 Confusion Matrix of the Logistic Regression Sentiment Classifier

From this, it is possible to predict the probability of each class for every word in the vocabulary. In this way, we can obtain the words which have highest probability for each particular class of the classifier. Highest and lowest predictive words for each of the classes, obtained from our classification model is shown in appendix A. As expected, we find that the words with strongest probability in positive class have positive connotations and vice versa for negatie class. For example, strongest predictors of positive class are *good, awesome, love, happy, thanks, etc.* Similarly, strongest predictors of the negative class are *hate, fail, miss, stupid, and sucks.*

Finally, strongest predictors of neutral class turn out to be *catching, airport, jump, private, invite, etc.*

Post Processing and Prediction of selected text

The Kaggle competition being solved uses word level Jaccard score to evaluate the closeness between selected text and prediction. Jaccard score is defined based on following code snippet. As can be seen, highest Jaccard score is 1 when all the words in predicted text exactly match those in the prediction. Having fewer or higher number of words in prediction brings down the Jaccard score.



```
def jaccard(str1, str2):
    a = set(str1.lower().split())
    b = set(str2.lower().split())
    c = a.intersection(b)
    jaccard_score = float(len(c)) / (len(a) + len(b) - len(c))
    return jaccard_score
vec_jaccard = np.vectorize(jaccard)
```

Figure 6 Definition of word-level Jaccard score

The mean Jaccard score of the training dataset for the trivial case, when the entire tweet is returned as selected text is 0.589. This is relatively high because for neutral case, most of the text is returned as selected text.

In the first iteration of constructing a text selector, we will return the strongest predictive word for the sentiment in the tweet. For example, for the following tweet, we can get predict the strongest word in this way

```
test tweet:                    'going to bed its late and I have headache'
sentiment:                     'negative'
prediction (most likely word: 'headache'
selected text (ground truth): 'headache'
```

However, the mean jaccard score for the training dataset (all sentiments) between the known selection and predicted word in this way is only 0.240. This is because most of the dataset has more than one word selected to denote the sentiment. To improve the performance, we add the following rule-based approach

1. If sentiment is ‘neutral’, return the entire text as prediction

Else, return the best word as before. This drastically improves the mean Jaccard score of predictions to 0.593, which is already greater than the trivial case. We also notice that for ‘positive’ and ‘negative’ sentiments, we often see that the selection is entire tweet when the original tweet consists of less than 6 words. We also add this rule to the prediction in version 3 of the text selector, which improves the mean jaccard score to 0.605.

Version	Approach	Mean Jaccard score

Trivial score	Return entire tweet as prediction	0.589
1	Select best word in each tweet for the sentiment	**0.240**
2	For ‘neutral’ sentiment, return full tweet. Return best predictive word for rest	**0.593**
3	Return complete tweet for ‘neutral’ sentiment or if original tweet contains fewer than 6 words	0.605

In order to improve the prediction capability of our model for ‘positive’ and negative’ ratings, we make use of parse trees. After parsing the sentence, we return the noun_chunk containing the highest predictive word. However, with this approach, the average jaccard score drops to 0.584. Looking at the predictions, we see that this performs badly on tweets where extracted text is only a word, but does better in cases where the text is larger. However, as the selected text in our training data is usually few words, we do not get as good of a performance using noun chunk parsing.

Version	Approach	Mean Jaccard score
Trivial score	Return entire tweet as prediction	0.589
4	Use parse trees to select the noun chunk containing the best predictive word and return it as predicted selection	**0.584**
5	Return a 5 word interval around the best predictive word as predicted selection.	**0.589**

As can be seen, the bag-of-words approach and parsed tree approach can only show nominal improvement over the trivial case. In the bag-of-words approach, this is due to the inherent inability of the bag of words approach to capture the phrases. On the other hand, parsing based methods, as used in this analysis seem to be failing due to the lack of a particular grammatical rule in selecting the text of interest in the tweet. Whether the selection is a word or a phrase is not always clear.

Therefore, we turn to deep learning-based methods to get better performance on the dataset.

Model Development: Deep Learning Approach

Over the past decade, several problems in NLP have seen drastic improvement in their benchmark performance scores using Deep Learning methods. Even within deep learning, NLP saw several improvements over the past few years, starting with Recurrent Neural Networks(RNN), Long Short Term Memory (LSTM) and transformer based methods like BERT.

BERT, which stands for Bidirectional Encoder Representations from Transformers is a language model developed at Google AI in 2019 which shows improved performance on multiple classes of NLP tasks like Named Entity Recognition (NER), Question Answering(QA) and Multi-Genre Natural Language Inference (MNLI). While training the deep learning model from scratch would require an extremely large dataset, a pre-trained language model like BERT can be used for transfer learning for different types of tasks.

Even with transfer learning, models like BERT may still be a challenge to run in applications with limited computational resources. One way to make a model smaller is to train a smaller/ shallower deep learning model and adjusting the weights such that is response in intermediate layers matches that of intermediate layers of a larger model. This approach is called 'knowledge distillation'. In our model, we use a language model called distilBERT, released by HuggingFace in March, 2020 and available in OpenTransformers library. The source paper for distilBERTR claims that it retains 97% of language understanding of BERT while being 40% smaller. More information on distilbert and the approach of finding distilled representations of larger modes is discussed more in the following paper by huggingface.
https://arxiv.org/pdf/1910.01108.pdf

In our case, a pretrained distilBERT model is used to train on our dataset. We apply transfer learning approach where we import a pre-trained model and train it on the current dataset. For this, we utilize a pre-trained model trained on the Stanford Question Answering Dataset (SQuAD). The model used is *'distilbert-base-uncased-distilled-squad'* model imported from the ***simpletransformers*** library. Link to the library is below:
https://github.com/huggingface/transformers/tree/master/examples/distillation

https://www.kaggle.com/doomdiskday/full-tutorial-eda-to-dnns-all-you-need/comments#Baseline-Models

Pre-Processing

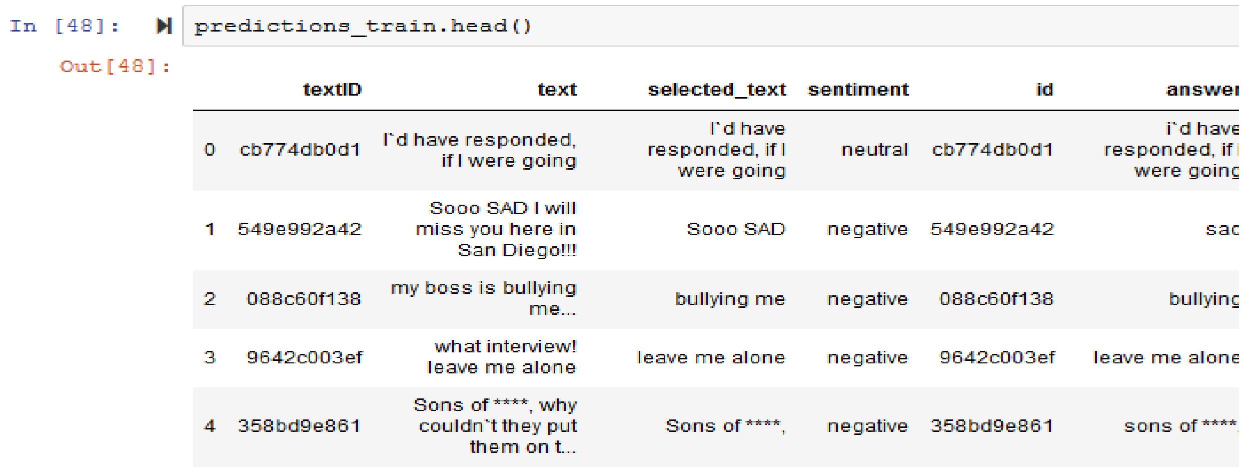
The selected text in the training data often includes html tags, punctuations, misspellings and url's from the original tweet. Therefore, we do not perform any text cleaning before feeding the data to the model. Only text cleaning operation is converting the text into lower case. In pre-processing the 'csv' file containing the data is converted into a json file where each row is converted into following format.

```
{
"context": " i`d have responded, if i were going",
 "qas": [{
     "question": "neutral",
     "id": "cb774db0d1",
     "is_impossible": false,
     "answers": [{
               "answer_start": 1,
               "text": "i`d have responded, if i were going"
               }]
     }]
}
```

In treating this as a question answering problem, we case the problem where we input a question and text (context) to the model and ask it to select a subset of context such that the question is answered using selected text. In this problem, looking at above example, the “context” key is the original tweet. The sentiment and selected text are in “qas”. The “question” is treated to be the sentiment. The answer text and start index of answer in context are stored in “answer” . “qas” also has addition detail, “is_impossible” which will be true if answer to question is not available in the context. In our case, this is always “False”.

Model Training and Results

The question answering model is trained with training data for three epochs. The predictions are again converted back to a dataframe and jaccard score is calculated. The training data has a jaccard score of 0.88 using the deep learning approach. The sample responses predicted by the model are shown in figure. In post processing, we still return complete text of the tweet for “neutral” sentiment or when the original tweet contains less than three words.



```
In [48]:  predictions_train.head()
Out[48]:
```

	textID	text	selected_text	sentiment	id	answe
0	cb774db0d1	I`d have responded, if I were going	I`d have responded, if I were going	neutral	cb774db0d1	i`d have responded, if were goin
1	549e992a42	Sooo SAD I will miss you here in San Diego!!!	Sooo SAD	negative	549e992a42	sa
2	088c60f138	my boss is bullying me...	bullying me	negative	088c60f138	bullyin
3	9642c003ef	what interview! leave me alone	leave me alone	negative	9642c003ef	leave me alone
4	358bd9e861	Sons of ****, why couldn`t they put them on t...	Sons of ****,	negative	358bd9e861	sons of ****

Figure 7 Sample predictions of distilbert model showing very good match with the selected text

Conclusions

In the current problem, we find that the deep learning model outperforms the traditional approaches, whether the bag-of-words or parse trees approach. The summary of the performance using different approaches is shown below.

For future work, we would try to extend the training for more epochs and also try other deep learning approaches to compare and hopefully improve the behavior.

Approach	Details	Mean Jaccard score
Trivial score	Return entire tweet as prediction	0.589
Bag-of-words	Return complete tweet for ‘neutral’ sentiment or if original tweet contains fewer than 6 words (ver 3)	0.605
Parse-trees	Return a 5 word interval around the best predictive word as predicted selection. (ver 5)	**0.589**

DistilBERT	*distilbert-base-uncased-distilled-squad’*	0.88 * best performance

Appendix A

Best and worst predictive words for each class

```
Predictive words        P(negative | word)
                hate     0.97
                fail     0.97
                miss     0.98
              stupid     0.98
               sucks     0.98
Non-Predictive words          P(negative | word)
                love     0.00
                glad     0.00
              thanks     0.00
             awesome     0.00
                hope     0.00
Predictive words        P(neutral | word)
            catching     0.87
             airport     0.87
                jump     0.87
             private     0.90
              invite     0.91
Non-Predictive words          P(neutral | word)
               happy     0.00
                good     0.00
              thanks     0.00
                love     0.01
             awesome     0.01
Predictive words        P(positive | word)
                good     0.99
             awesome     0.99
                love     0.99
               happy     1.00
              thanks     1.00
Non-Predictive words          P(positive | word)
                 sad     0.00
                miss     0.00
               sorry     0.00
                hate     0.00
               sucks     0.00
```