Emotion Detection From President Donald Trump's Tweets

Xiaohan Jiang
CMPE Software Engineering
San Jose State University
San Jose, USA
xiaohan.jiang01@sjsu.edu

Swapnil Avinash Parihar
CMPE Software Engineering
San Jose State University
San Jose, USA
swapnilavinash.parihar@sjsu.edu

Pranav Karmalkar
CMPE Software Engineering
San Jose State University
San Jose, USA
pranav.karmalkar@sjsu.edu

***Abstract*—In this paper, we apply emotions detection and quantification to the US president Donald Trump's tweets texts. President Trump updates his twitter daily and presents opinions on various social topics. We are curious about the underlying emotions of his tweets. Analyzing and classifying text on the basis of emotions is a big challenge and can be considered as an advanced form of Sentiment Analysis. This paper proposes a method to classify text into five different Emotion-Categories: Happy, Angry, Surprise, Sad, and Fear. Python's text2emotion library provides a similar functionality but proved ineffective in handling negation and utilizing person specific vocabulary-emotion connection. Using this library as a fundamental algorithm, an improved algorithm was designed to achieve higher accuracy and faster running time. We customized an emotion-word dictionary for Trump tweets analysis goals, implemented negation checking and selective word replacement. We have also successfully devised a method to visualize emotion distribution based on topic modeling. Moreover, we have tested our model in many circumstances : Trump's retweet / deleted / election days tweets. All these testings showed that our model provides significant accuracy in classifying tweets taken from Trump's tweets.**

***Index Terms*—Emotion Detection, Sentiment Analysis, Data mining, Natural Language Processing, Topic Modeling, Named Entity Recognition, BERT, Time Series Analysis**

I. Introduction

Emotion analysis is a branch of natural language processing that deals with the clustering or classification of text data according to the emotions present. It can be understood as a multinomial generalization of sentiment analysis. The primary motivation behind this analysis lies in automating the task of manually going through a large text corpus to understand the sentiments of the author. The analysis can be exclusive or non-exclusive. A non-exclusive clustering allows the items in the test set to be clustered in more than one class. Various factors impact the selection of an algorithm for clustering. The size of the text corpus and whether it is structured can significantly impact the choice of algorithm. Another factor that decides the choice between a classification and a clustering algorithm is whether the dataset is supervised or unsupervised.

This project deals with a large, unstructured, and unsupervised dataset of over 55,000 President Trump tweets. A tweet is a short text of not more than 280 characters used to share remarks, news, or emotions by an individual. The word limit of the tweet makes emotion analysis particularly challenging as the information present in a single tweet can be just a few or even a single word. President Trump is quite active on the Twitter web platform and often posts his opinion on various topics including but not limited to politics, sports, cinema, and foreign matters. As one of the most powerful people currently on the planet, his emotion about these matters can impact future policies. His tweets give us an insight into his likes and dislikes and the decision-making process. While the President presents an array of emotions in his tweets we limited the classification of text to these five emotions: happy, angry, surprised, sad, and fear. Here we deal with multinomial classification with the resultant classes fixed. To achieve this, the algorithm should not only cluster the text into non-exclusive grouping but also identify the "emotion" associated with the cluster. To further complicate the task, since the dataset does not have a target variable standard measures of accuracy do not apply to this project.

There was a need to understand the emotion variability in his tweets based on time-series analysis. So, for this NER is used to improve the evaluation metrics and understand various trends in this project. BERT algorithm was used to evaluate the predictions done by our algorithm. Interesting insights were formed from this analysis, which will be discussed further in the report.

The next subsection provides a preview of previous related works in the field. Section II dives into the system design & implementation details. Section III presents various experiments done on the data and shines a light on the different data pre-processing steps involved during the model generation. Section IV discusses the results of the paper and concludes all the findings.

II. Related Work

This problem was initially considered in the most primitive form possible i.e. to first classify tweets as either positive or negative. To implement this approach we first used word2vec on the dataset and then implemented KMeans on this dataset. We were able to classify words from the tweets into positive and negative sentiments. The next step was to form clusters that identify emotions. But when the number of clusters was

Special thanks for Professor Dr. Magdalini Eirinaki.

increased to accommodate emotions into various clusters, we were not able to cluster the words properly. This caused us to change our approach and go with a custom prepared similarity model, which is an improvement on the already existing text2emotion library. Text2emotion python library source code was a good starting point to have an approach to this problem in a very unique way.

III. System Design & Implementation Details

A. Word2Vec and KMeans

The problem of associating emotions to a tweet or a text corpus, in general, can be treated using a variety of clustering approaches like K-means and word2vec discussed earlier. Our initial approach to this problem was to apply word2vec after lemmatization and removal of stopwords. After extracting word2vec we applied the KMeans algorithm to it and detected the clusters. The main problem we faced here was we were able to successfully extract the positive and negative sentiment clusters but could not extract emotions from it. Finally, after this stage, we went ahead and applied TF-IDF and analyzed the closeness and sentiment coefficients. We also tested Python's text2emotion library which performs the task of emotion classification but based on the testing found the results to be below our expectations. An improved version of the algorithm was created and is discussed in the next section.

B. New Improved Custom Algorithm

Based on the limitations observed of the previous attempts, the primary requirements of the algorithm could be summarised as follows:

1) The classification process should result in a non-exclusive solution where the text should belong to the 5 predefined categories.
2) The algorithm should be better at handling negation.
3) Due to the lack of training data, the algorithm should have some pre-training about the different output categories.
4) An algorithm that could take the advantage of a "Trump" specific dictionary could be a big plus.

A lexicon-based approach satisfies all these requirements. The primary logic of a lexicon-based approach lies in using a pre-labeled dictionary. Each word in the dictionary has an emotion associated with it. The more inclusive(having more words associated with correct meaning) a dictionary the better the algorithm will be able to identify emotions. When a text input is fed to the algorithm for each word in the text if it is present in the lexicon-based dictionary adds its emotion to the final value. If the word is not present in the dictionary it will be ignored. A percentile-based addition of all the emotions can give a good approximation of the emotions attached to the text. This approach forms the base algorithm of this project. The tweet text though cannot be directly used with text processing. Key components of the text processing include removing stopwords, punctuation, and hyperlinks, replacing shortcut words with their full form, and handling negation in a sentence. After text processing is completed the cleaned text can be fed the algorithm that gives a dictionary containing all relative values of all the emotions in the text. Fig. 1 shows the flow of the algorithm.

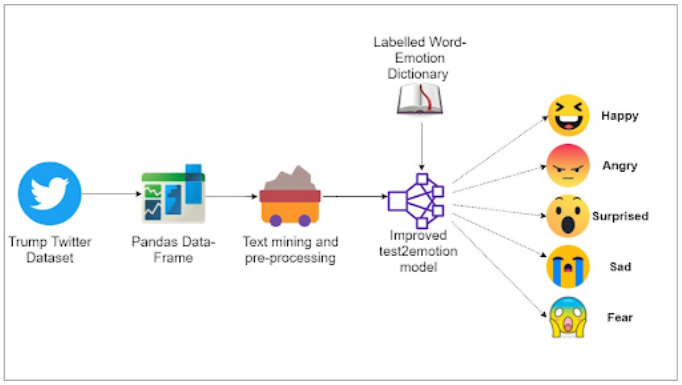

Fig. 1. Architecture diagram of Algorithm's Flow

The project is developed in Python and uses the NLTK library for performing operations like lemmatization and removing stopwords, pandas library is used for manipulating the input data, regular expressions are used for text handling and pillow and matplotlib is used for visualization. Text2emotion library is used as a comparison algorithm to the one we have developed as it works on a similar principle but our enhancements result in improved performance as well as better accuracy. The pillow library is an image manipulation library that helps in creating an official tweet type of representation for all the tweet texts. Fig. 2 shows the results of this representation.

Fig. 2. Emotion Visualization of One tweet

C. BERT NER

BERT model uses a masked LM algorithm to categorize various words in sentences into various categories like locations, organizations, etc. For this particular part of the problem, we used the hugging face BERT model to categorize the words. It is a pre-trained model trained on CoNLL-2003 Named Entity Recognition dataset. As in the dataset, each token will be classified as one of the following classes: O—Outside of a named entity, B-MIS —Beginning of a miscellaneous entity right after another miscellaneous entity I-MIS. B-PER —Beginning of a person's name right after another person's name I-PER. B-ORG —Beginning of an organization right after another organization I-ORG. B-LOC —Beginning of a location right after another location I-LOC.

Fig. 3. NER on product and organization dataset

IV. Experiments / Proof of concept evaluation

A. Data Pre-processing

The text imported directly from tweets contains many hyperlinks, the reference to other tweet handlers, stopwords, and punctuations. This data cannot be directly compared with the dictionary as it will result in many errors. Another important issue with the language used in Twitter is due to the short word limit users often use shortcuts. An example of such an issue is the common short form "grt". It is a short form of the word great. As the word "great" indicated an emotion corresponding to happiness it should not be ignored by the dictionary. An example of a tweet with this issue can be "Grt book by the author". Thus replacing shortcuts with their full meaning is one of the first key operations involved in data pre-processing.

Another important issue in text data emotion extraction is negation handling. Negation handling is the method to handle words that change the meaning of the word it is associated with. An example of this issue can be seen in the following text "I am not happy". Here the word happy will generate a "Happy" emotion-based on its value in the lexicon dictionary. Now if the "not" word is not handled the algorithm will show an incorrect result. There are many techniques to negate this issue. One method that is most commonly used is checking the bigrams containing the negation words. So if we have a dictionary of all the bigrams containing a negation word we can change the meaning of the word coming after it. One advantage of using bigrams is if we have a list of all the applicable bigrams this process of identification can be done in O(1) time using the dictionary data type. This technique is used in the text2emotion library but it suffers from a serious drawback. Consider the text "I am not even happy."; The bigram in this text will be "not even" which does not give the algorithm enough evidence to predict the emotions of the text. So to counter-attack this issue we came up with an innovative solution. It was observed that more often than not, negation words are meant to change the "emotion" corresponding to words that follow it. So "not good", "not that good" and "not even that good" all have the same meaning since most of the words that occur in between the word "not" and "good" do not have any emotions attached to them. Taking advantage of this observation, we created an algorithm that would traverse a text until it finds a word with emotion associated with it or finds the end of a sentence. If a word-emotion pair is found in the dictionary, its emotion is reversed. If it is not found and we reach the end of the sentence then that negation is not processed. The second part is important as it avoids a very serious pitfall as it will result in changing the meaning of the next sentence in the text. We often use negation to emphasize the same emotion as "I like you. You know that don't you. I have always loved you". Here the shortcut "don't" which can be expanded to "do not" does not change the meaning of any word. Such pitfalls are also avoided by stopping the search after the end of the sentence has been reached. Using this technique we were able to reduce the number of unhandled negation words from a whopping 10% to 3% for our dataset. While this method required O(n) time complexity, it is a small price to pay for the improved accuracy achieved. To take advantage of bigram time complexity both the algorithms are used in tandem to provide improved accuracy with minimum time complexity.

The data processing model follows these steps:

1) Word is converted to lowercase.
2) All hyperlinks and Twitter handles are removed.
3) Contradictions like don't and never are converted to not for better processing.
4) All known bigrams with negations are processed and converted.
5) Shortcut words are replaced.
6) The new negation handling algorithm is deployed.
7) All the punctuations are removed and the data is fed to a lemmatizer.
8) This corpus of words is then finally processed by the algorithm.

Based on these experiments and improved text processing, we observed a remarkable improvement in the algorithm's processing time. The original text2emotion library took 70 seconds to generate the results for 1,000 tweets. Our improved algorithm completed the task in less than 0.5 seconds on the same hardware. As can be seen, a massive improvement is seen in the results partly due to the use of proper data structure and partly due to more refined data pre-processing. To provide even more freedom to the user to handle pre-existing biases in the text we also added an optional parameter that can add weights to the input data which the experimenter can determine after analyzing the data.

B. Topic Modeling

We utilize topic modeling, an unsupervised method, for clustering the main topics of Trump's tweets. This step enables us to get a general idea of the contents of massive tweets. Considering the large amount of political terms and politician names in Trump's tweets, we decided to use unigram and bigram for bagging of words. Also, to focus on emotion detection, we specify the parameter n_component (number of topics) as a relatively small number : 5. After setting up topic modeling, we extract 10 mostly used terms from each topic. Find out original tweets that contain these 10 terms and throw them into our emotion detector(get_emotion()). The get_emotion() function will return the accumulation results of 5 emotions. By combining previous steps, we create plot_topic_emotion() which asks two parameters, topic name and topic information data frame. It plots a pie chart of the top 10 terms and a bar chart of emotion distribution of the topic. The visualization allows us to present the general emotions of

Trump when he talks about specific topics or terms. Fig. 4 - Fig. 6 shows the topic modeling visualization results.

```
plot_topic_emotion('Default',trump_topic_info)
```

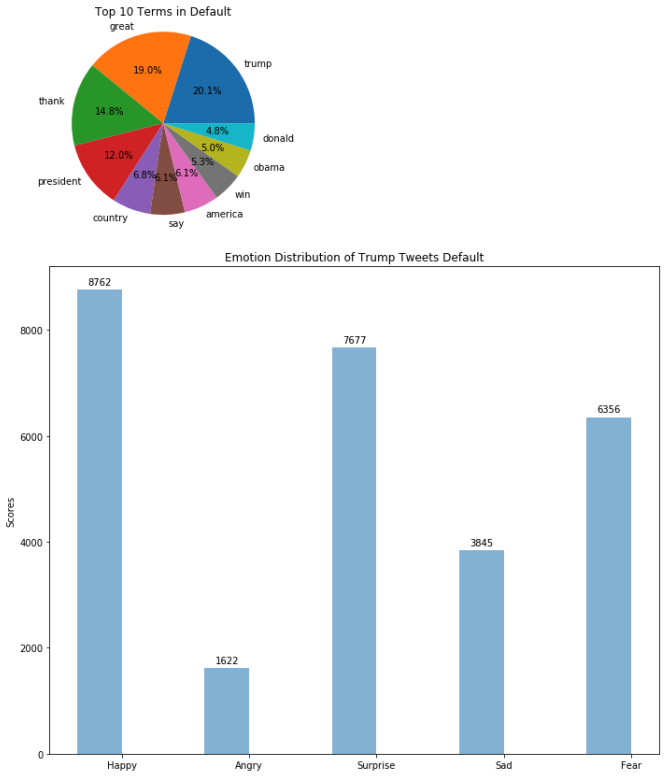

Fig. 4. Emotion Detection In General Topic

In general, the emotions present in Trump's tweets are similar, he usually tweets with happy, surprise and fear emotion! He mentioned a lot about himself, and MAGA(Make America Great Again). We couldn't see the obvious variation of emotion from topic to topic. It is because the most frequent terms in every topic are quite similar(trump, greate, donald, president...), the frequency of these words exceed others by a large amount. So we decide to choose specific words (especially nouns in the topic). And detect the emotions regarding these words.

C. Associated words emotion detection

In this section, we focus on the analysis of words and related tweets' emotion detection. To realize pretty printing of visualization, we modified part of previous code. We defined function plot_words_emotion() which takes two parameters: words list and clean tweets list. Words list contain three words that we are curious about, for example: ['china','russia','america']. We want to know how many Trump tweets about these countries and his emotions regarding different countries. We can manually change words list, but for standard visualization, we can only insert exact 3 words in the list.Fig.7 - Fig. 8 shows different words list emotion detection and visualization. These words are chosen from Trump's most frequently used words.

```
plot_topic_emotion('Topic1',trump_topic_info)
```

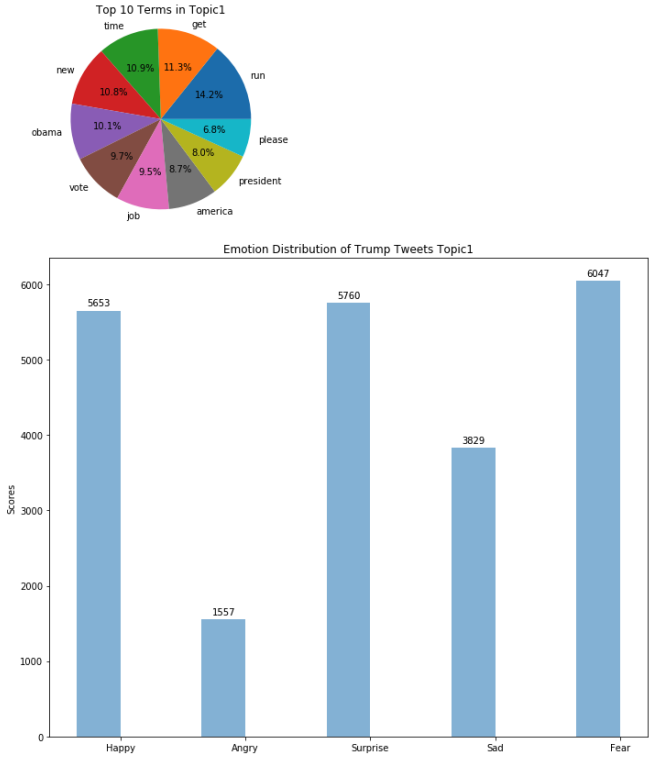

Fig. 5. Emotion Detection of Topic1

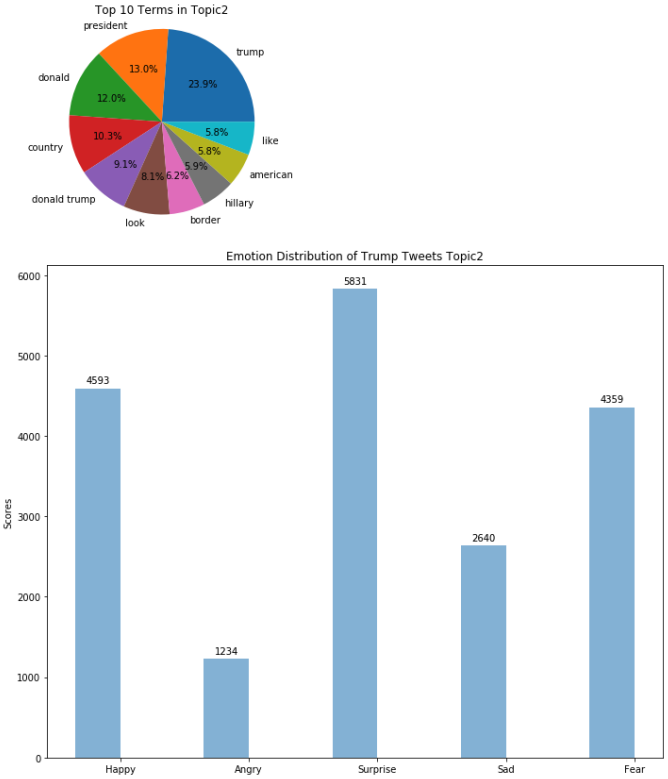

Fig. 6. Emotion Detection of Topic2

```
list3 = ['obama','obamacare','barackobama']
plot_words_emotion(list3,clean_tw)
```

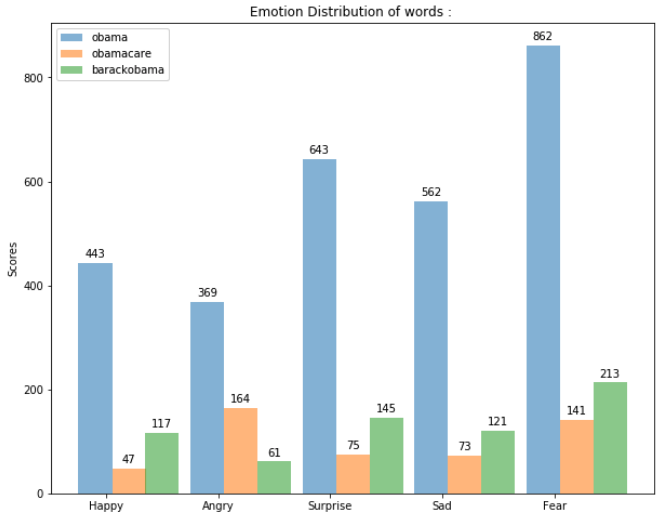

Fig. 7. Emotion Detection of Words Regarding 'Obama'

```
list1 = ['china','russia','america']
plot_words_emotion(list1,clean_tw)
```

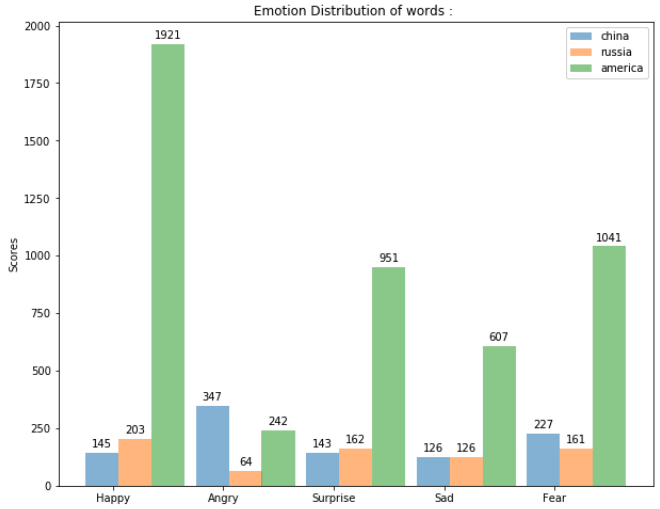

Fig. 8. Emotion Detection of Words Regarding Countries

D. Retweet content analysis

We are very curious about the contents of Trump's retweets. So we select the retweets and use topic modeling to cluster Trump's retweet topic. We find out that in Trump's retweet, he mentioned 6 politicians many times. These 6 politicians come from different parties and president Trump's emotions varied when mentioned or retweeted from them.

Brief introduction of three politicians in Fig. 9 :

- Dan Scavino : Assistant to the President and Deputy Chief of Staff for Communications.
- Tom Fitton is known for pro-Trump commentary.
- Jim Jordan : A close ally of President Donald Trump, Jordan is a founding member of the Freedom Caucus and chaired the Caucus from its establishment in 2015 until 2017.

```
retw1 = ['danscavino','tomfitton','jim']
plot_words_emotion(retw1)
```

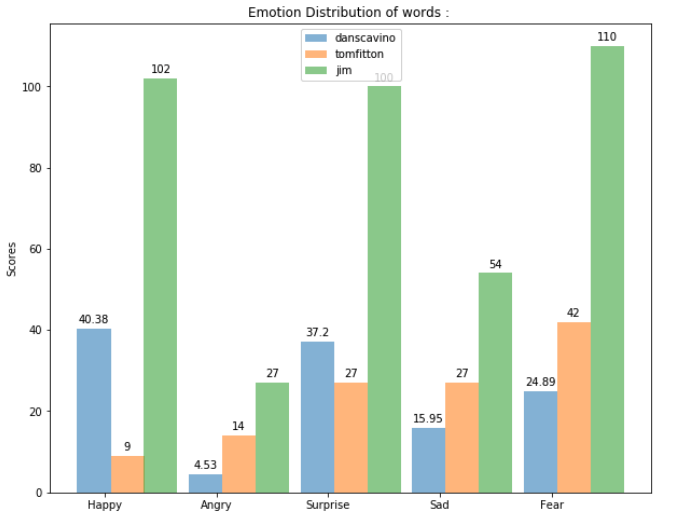

Fig. 9. Emotion Detection of Trump Team's Politicians

Brief introduction of three politicians in Fig. 10 :

- Nancy Pelosi - Democratic.
- Adam Schiff - Democratic.
- Mike Bloomberg - a candidate for the 2020 Democratic Party nomination for President of the United States.

```
retw2 = ['pelosi','mike bloomberg','schiff']
plot_words_emotion(retw2,clean_retw)
```

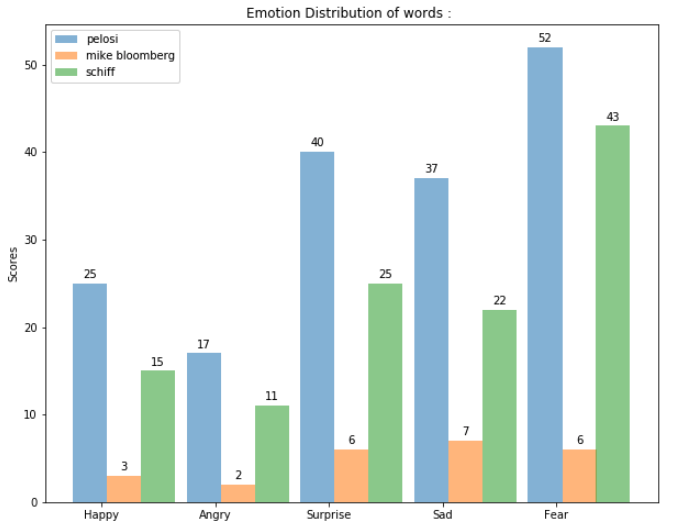

Fig. 10. Emotion Detection of Democrat Politicians.

The results seem to make a lot of sense. President Trump has more positive emotions regarding republicans.

E. What he deleted?!

We dive deeper into the data and find out there are 1040 tweets were deleted by Trump, which makes us curious about

what content he removed and what's the emotions of those removed tweets. We repeat similar steps as above and find out the top terms and topic he removed are pretty similar to the tweets in general. But we finally found some new terms. Trump deleted a lot of tweets regarding 'fake news', 'coronavirus' and 'witch hunt'. So we analyze the emotion of his removed tweets and current tweets. Fig.11 and Fig.12 shows his attitudes towards these topics.

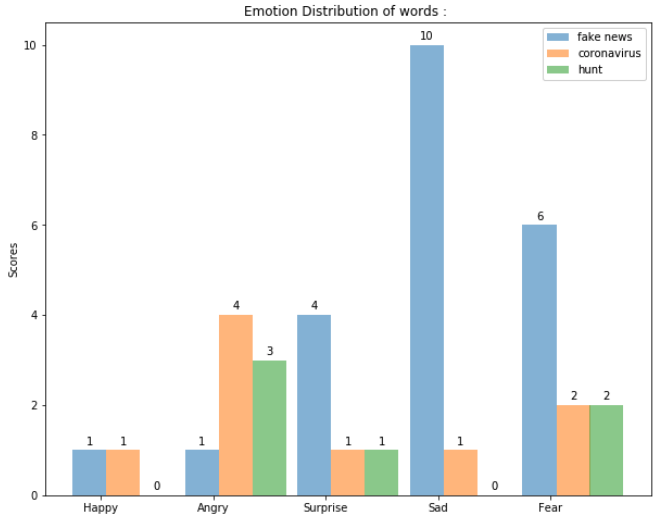

Fig. 11. Emotion Distribution in deleted tweets regarding 'fake news', 'coronavirus' and 'witch hunt'

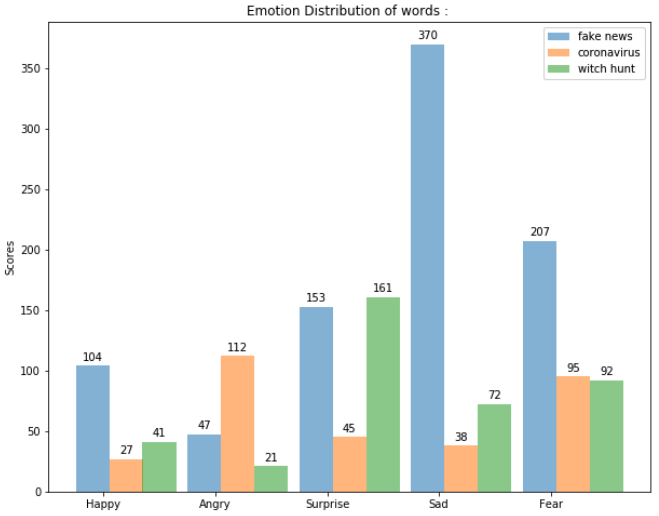

Fig. 12. Emotion Distribution in current tweets regarding 'fake news', 'coronavirus' and 'witch hunt'

It is easy to tell that president Trump is very upset about fake news, he put a lot more attention on fake news than coronavirus, the deleted tweets topic is the only place that coronavirus show as top term. Trump is very mad about coronavirus since the angry score the highest in both graphs, and exceed the other two terms. 'Witch hunt' : Trump implied he's been persecuted like a witch(impeachment). "He claimed those accused in the city's infamous 17th century witch trials received more due process than he has as he faces impeachment."

F. Election days' tweets

We collect the tweets from the 11/02 to 11/06 and analyze the topic and emotions of these texts.

President Trump is very angry when he talks about his competitor Joe Biden. He mentioned many times of vote and win but very little emotions are captured from these tweets, maybe because of the uncertainty of the election and after 11/03 he started to lose.

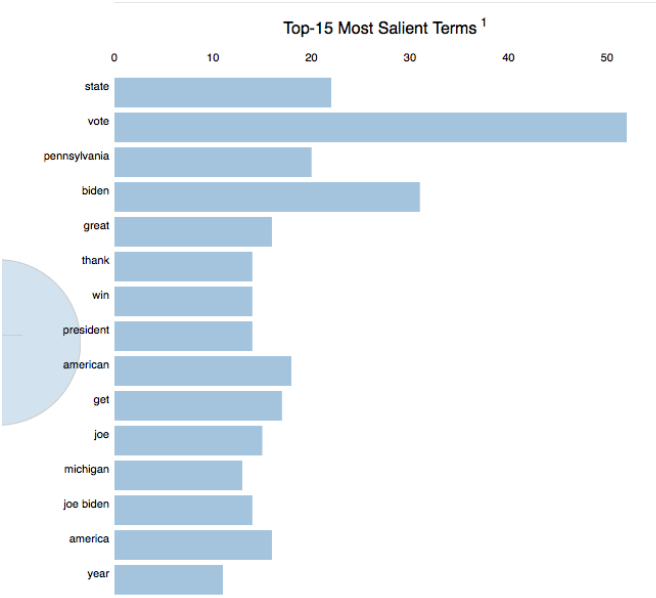

Fig. 13. Top Term During Election Days

```
e_list = ['biden','vote','win']
plot_words_emotion(e_list,ele_tw)
```

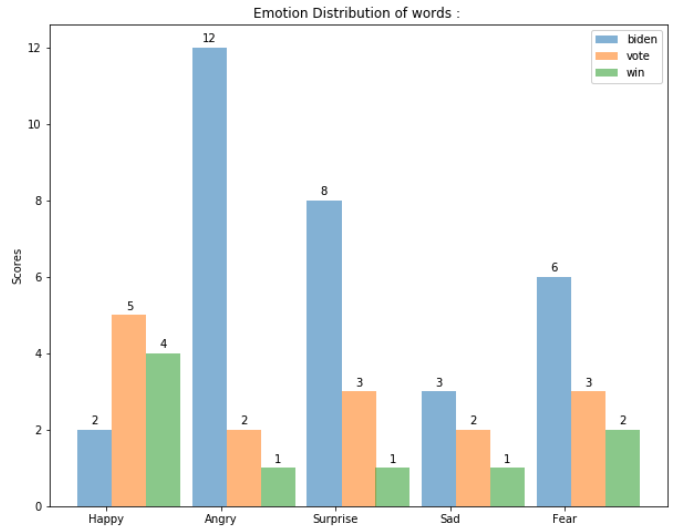

Fig. 14. Emotion Detection of Election Top words

G. BERT NER Data Mining

Mining data for BERT is a unique process as regular methods used for text processing, like stop words removal, punctuation removal, lemmatization, standardization, or lower case handling cannot be applied. Also, BERT NER does not recognize words in lower or upper cases, there is a need to convert these words into title case for recognition. Stop words were carefully removed to make sure important words were not omitted. The biggest problem faced was to join the locations as BERT predicts, United States of America as 'United'—B-LOC, 'States'—I-LOC, 'America'—I-LOC. So, the stack data structure was used to join these words and make meaning out of them. Further locations like 'North America' and 'America' were recognized separately, so major locations were needed to match and group together. So, partial matching string algorithms were used to match these words as the same.

H. NER Time Series Analysis

We used BERT NER to do a time-series analysis of various locations in the dataset. We use NER to extract locations from different tweets and do one hot-encoding of the tweets to include all the locations from every tweet. For this, we created a new dataframe where we added locations to this dataframe. After this, we analyzed new tweets and compared the extracted emotions in a time-series graph.

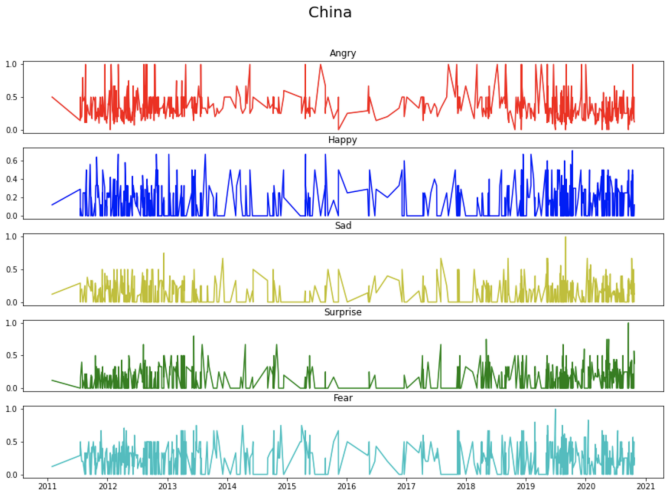

Fig. 15. Time series analysis of Trump tweets containing the keyword China

Fig. 15 analysis was done on: "European Countries are sadly getting clobbered by the China Virus. The Fake News does not like reporting this!" and China was the location extracted by NER. After doing time series analysis it can be concluded that Trump was mostly angry in most of his Tweets. Also, one more thing can be extracted that he rarely feared China in most of his tweets.

Fig. 16 analysis was done on: "...owned Dominion Voting Systems, turned down by Texas and many others because it was not good or secure, those responsible for the safeguarding of our Constitution cannot allow the Fake results of the 2020 Mail-In Election to stand. The World is watching!" and Texas

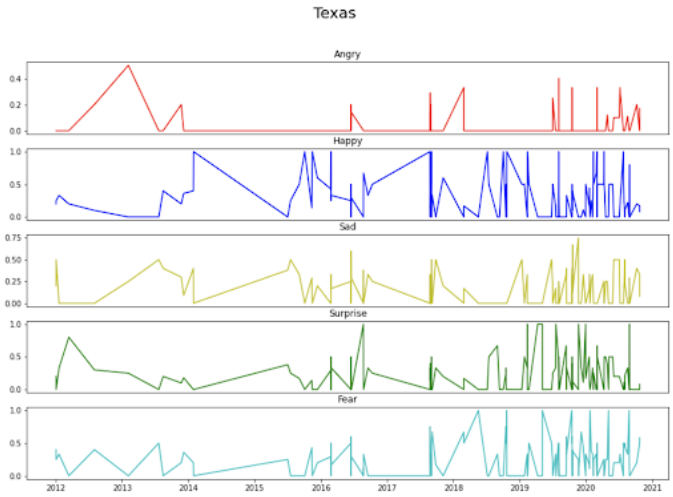

Fig. 16. Time series analysis of Trump tweets containing the keyword Texas

the location extracted by NER. After doing time series analysis it can be concluded that Trump mostly tweeted about Texas during elections and he was rarely angry about Texas.

V. Discussion & Conclusions

As discussed before we experimented with Word2Vec and KMeans algorithms to form clusters of different emotions but due to the no cohesive nature of words we were only able to extract positive and negative sentiment of different words. This approach though being very generalized lacked the sophistication needed to cluster words into emotions.

A new algorithm was designed for the task. The project shifted from a list-based word-dictionary to a dictionary data structure in python. This resulted in a significant improvement in the response time of the algorithm. With an improved negation handling algorithm, the number of unprocessed negations for the dataset was reduced from 10% to 3%. The project also improved upon the data pre-processing techniques to further increase the algorithm's response. A dataset targeted dictionary of word-emotion pairs was created using topic modeling tools. The new model generated using this algorithm outperformed other contenders in terms of accuracy. An analysis of the deleted and retweeted tweets provided some very useful insights about their content. Image processing and topic modeling processes helped with data analysis and served as an attractive visualization tool. The project also provided an option to adjust the weights of all the emotions extracted. While techniques like lemmatization reduced the required word processing sometimes resulted in unexpected outcomes. Many of such surprises were identified and corrected during the validation and testing phase of the project. It should also be stated that while the project's "move forward and check" approach to negation handling improves the results significantly it does not fully eradicate the issue.

In the Associated words' emotions detection section, the selection of words is very important. I was planning to use association rule mining to find words that are highly associated

and use them as parameters of plot_words_emotion(). But the computation of association rule mining is too expensive which leads to OOM problems several times. We tried to split the data into smaller pieces and only do association rule mining on very small data such as deleted tweets, but none of them work out. We still keep the code block at the end of our notebook.

The most important issue we faced while using BERT NER was the detection of locations like “USA”, “America”, “US” and “United States” which is nothing but the same location but detected separately. Multiple algorithms and custom algorithms were written to try and merge these words but the advanced study of NER and collection of similar words dictionary is required to improve the detection. Also, using the same dataset to train NER would greatly improve the prediction accuracy. BERT detects locations that are in title case so specialized text preprocessing was required to improve the detections. Stop words removal and high-frequency word removal mining techniques fail to improve this model. Finally, standardization and lemmatization are not good techniques to use because “American” which is not reduced to “America” is not a location but used to describe a person. So, custom data mining algorithms were written to improve the accuracy of the detection of words.

VI. Project Plan / Task Distribution

Table 1 shows task distribution:

TABLE I
TASK DISTRIBUTION

Task	**Assign To**	**Status**
Processing shortcut/frequent words not in the dictionary	Katy	Complete
Topic Modeling	Katy	Complete
Retweet / deleted / election days tweets emotion analysis	Katy	Complete
Negation Logic	Swapnil	Complete
Final Algorithm	Swapnil	Complete
Visualization of emotion	Swapnil	Complete
NER	Pranav	Complete
Timeseries Visualization	Pranav	Complete
Word2Vec + KMeans Limitations	Pranav	Complete

References

[1] Our data resource comes from https://www.thetrumparchive.com/ Download it from the FAQ page. At the time we implemented our final solution, we used the data that was updated on Nov 06,2020.

[2] text2emotion 0.0.5, accessed on Nov 01,2020, https://pypi.org/project/text2emotion/.

[3] B. Liu. Sentiment Analysis and Subjectivity. Handbook of Natural Language Processing, Second Edition, (editors: N. Indurkhya and F. J. Damerau), 2010.

[4] Melville, Wojciech Gryc, Sentiment Analysis of Blogs by Combining Lexical Knowledge with Text Classification, KDD09, June 28–July 1, 2009, Paris, France.Copyright 2009 ACM 978-1-60558-495-9/09/06.

GitHub Repository

https://github.com/pmvkirock/unsupervised-emotion-detection