Detecting Fearmongering Language in Comparison to Sympathetic Language in President Trump's Tweets

Yohamy Polanco
The Graduate Center, CUNY
365 Fifth Ave, New York, NY USA
ypolanco@gradcenter.cuny.edu

Abstract

I present a program that can detect fearmongering language from lists that contain words that are correlated with fearmongering. This is in comparison to sympathetic language that also is generated in a list with words that are correlated to sympathetic language. These lists are then used to be able to detect language that are possibly fearmongering or sympathetic language. There is not much work to be found on fearmongering language and detection, this is still a considerably new term that I assume will become more prevalent in future work.

1. Introduction

Fearmongering is a fairly new word that has entered in our common day vocabulary. This term can be defined as "*raising or exciting alarms especially needlessly"*[1]*.* A synonym of fearmonger is scaremonger or an alarmist. Scaremongering was seen before in history during The Cold War and The Red Scare, when the government instilled fear to Americans regarding communist Russian spies being amongst them, and the possibility of a nuclear bomb being used to attack the United States. We are now seeing similar tactics and language being used in today's world. The difference is that during the Red Scare, most of the fearmongering language was used as Propaganda on either posters or television commercials especially during the 50's when McCarthyism was on the rise. Currently, at the height of social media and the ability to be able to send something out to the world in an instant for example *Twitter*, fearmongering has become something that cannot be avoided.

2. Data

President Trump is notoriously known for his constant use of Twitter. This makes him a prime candidate to test fearmongering language based on his twitter feed.

The data that was used was obtained from President Trump's Twitter feed. To obtain the data, a pure Python wrapper called Twython was used for Twitter's API. A total of 2,700 tweets were used for fearmongering and sympathetic language detection of which do not include Retweets. The tweets date from December 18, 2017 to November 28, 2018. In order to obtain the "full text" portion of each tweet,

[1] https://www.merriam-webster.com/dictionary/scaremonger

excluding all other details that were not pertinent to this project, the module functools[2] was used, which returns only the "full text" of each tweet.

The fearmongering and sympathetic word related terms were obtained from Word Associations Network[3]. The terms that were used to generate these lists were *fear* and *sympathetic*. I generated two lists of the exact length of 111 terms.

3. General Approach

Once the data was obtained, NLTK toolkits *SnowballStemmer, Tokenizer* were used to tokenize each word in the data and the lists. Then each word had to be stemmed to its root form. This was done so that bound morphemes do not cause any problems since we are testing for words without any regards to its bound morphemes. The stemmer that was used was not optimal but generally it worked for this project. The next step was to take each list and compare each item in the list to those inside all the tweets and count the frequency of times each word appeared in the tweets. This is done by using an if statement embedded inside a for loop. Each match was then appended to a list along with its frequency count.

Table 1 shows the ten most common fearmongering related terms found in President Trump's tweets. Table 2 shows the ten most common sympathetic related terms found in President Trump's Tweets.

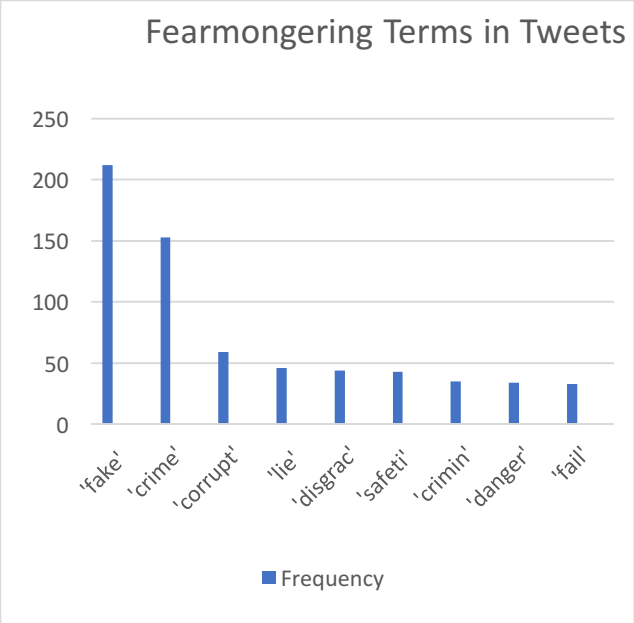

Table 1

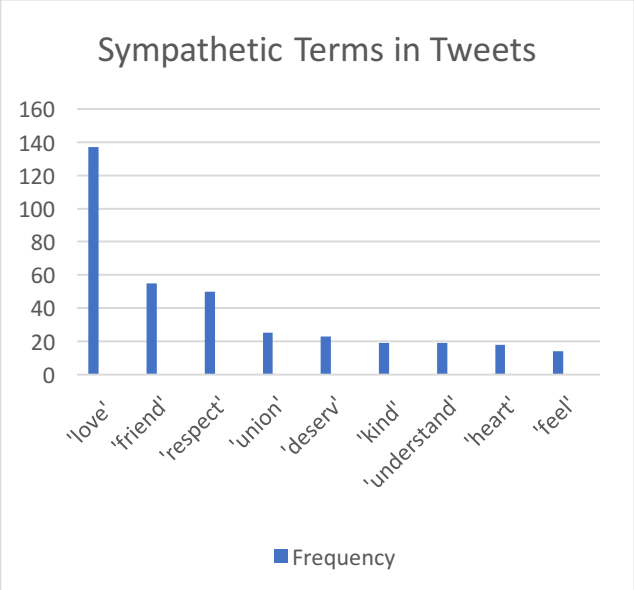

Table 2

[2] Functools https://docs.python.org/3/library/functools.html

[3] Word Associations Network https://wordassociations.net/en/

4. Results

From the 2,700 tweets, there were 212 instances where the term *fake* occurred. This is the term that has the most instances across all the tweets. From the sympathetic list, *love* is the term that has the most occurrences. The average from the ten terms shown in Table 1, is 66 meanwhile the average for the ten terms shown in Table 2 is 36. What this shows is that for every sympathetic word found in the data, there are almost twice as many fearmongering words. These results support the notion that President Trump instills fear using Twitter but there are a few shortcomings in this project.

5. Shortcomings

There are some shortcomings that this project fails to appropriately handle. For one, the data to that is used to determine fearmongering speech is not accurate. The list that was used was a list for words related to *fear* but not *fearmongering*. The specificity of fearmongering requires there to be a more precise list of related words.

Another shortcoming is the failure to consider context. In general, words have different meanings depending on their environment or the surrounding words, depending on single words, is not accurate enough for this project. In future work, bigrams would be used to narrow down the specific environments of both fearmongering and sympathetic language.

It was difficult to be able to gather information about fearmongering and using NLTK for detection, since fearmongering is still a fairly new term. An assumption can be made that the process of detecting fearmongering language, may be similar to that of hate speech. This would also have to be taken into consideration onto future work.

6. Conclusion

I present a general way to detect fearmongering and sympathetic language based on Twitter. In future work, I plan to further develop this project and start considering the shortcomings and how to make this program more efficient.

7. Code

```
def stemming(text):
        tweet_tokens = []
        tweet_stems = []
        for word in text:
                word.casefold()
                token = nltk.word_tokenize(word)
                tweet_tokens += token

        for token in tweet_tokens:
                stem = [stemmer.stem(token)]
                tweet_stems += stem
                return tweet_stems
```

```
def frequency(text, sublist):
	freq = Counter()
	for z in text:
		if z in sublist:
			freq[z] += 1
	return freq



with open("twitter.json", "r") as source:
	trump_data = json.load(source)
	from string import punctuation

	get_values = functools.partial(map, operator.itemgetter('full_text'))
	tweets = list(map(operator.itemgetter("full_text"), trump_data))

	trumplist= []
	trumplist = stemming(tweets)

	strp = Counter(word.strip(punctuation) for line in trumplist for word in line.split())
	count = Counter(strp).most_common(60)

with open("fear_words.txt", "r") as source:
	fear = stemming(source)

with open("sympathetic.txt", "r") as source:
	sympathetic_words = stemming(source)

for i in fear:
	if i in sympathetic_words:
		fear.remove(i)
		sympathetic_words.remove(i)


fear_freq = frequency(trumplist, fear)
sym_freq = frequency(trumplist, sympathetic_words)
```