Safe Space

Andreas Zurhaar, Sandu Zuza [1]

May 27, 2019

Abstract

The purpose of this paper is to show how negative or hurtful comments can be censored and rewritten into positive ones using natural language processing techniques.Three different techniques of increasing complexity are explored and their resulting censorship capabilities are measured. This includes a "Simple Censorship," "Naive Vectorization," and "Smart Vectorization." Experiments were performed to fine tune these processes, and the conclusions based on this research are delved into. However, Smart Vectorization had the best overall results compared with the other two techniques. Possible improvements on our research are also included.

1 Introduction

Peoples interactions with negative comments have increased over the years, especially in a world where everyone is on social media. In 2014, a Pew Research Center study found that about 66 percent of internet users who have experienced online harassment said their most recent incident occurred on a social networking site or app [3]. A lot of these comments could be automatically filtered out and altered to be positive in order to ensure that people have a more positive experience online. That is the inspiration for this research, and this is why we have produced natural language processing techniques in order to mark negative comments and then transform them into positive comments.

In this paper these techniques are introduced and explained. Following this, results of the various experiments conducted on these techniques will be showcased and examined. These experiments revolve around how changes to the censorship techniques can improve or degrade the accuracy of the outputted censored comment. A section that includes ways to further the research is also included, finally ending with a conclusion of the results of everything discussed.

[1]Universiteit Maastricht, Department of Computer Science, P.O. Box 616, 6200 MD Maastricht, The Netherlands

2 Corpora

In order to train any machine learning algorithms or perform any sentiment analysis, a few corpora (databases of words) need to be used for both labeling and for examples to train. For this project three different corpora were used. One is a database of all swear words [4] for help in labeling bad words. Another corpus is just examples of positive words [2] that is used for labeling positive words. The final corpus is from a Stanford project [1] and has 1.6 million sentiment labeled example tweets to extract and use.

3 Vectorizing

In order to represent our words as vectors the gensim Word2Vec model is used. The magnitude of the sum of all vectorized words in a sentence is used to represent that sentences value.

4 Training

In order to train the sentiment analyzing algorithm, we used the vector value of the sentences from the Stanford corpus. There were a few machine learning classification options to try. First Naive Bayes was attempted, but it was over fitting because the Stanford corpus had an uneven 80 percent negative examples to 20 percent positive examples. There was no option to weight them accordingly. So instead Random Forest classifier is used because there is an option to weight the examples when training the vectors. Random forest has an accuracy of 65 percent when determining a positive or negative sentence.

5 Censorship Techniques

This section explains the mechanics and goals of each algorithm used for censoring negative comments and replacing them with positive ones.

5.1 Simple Censorship

This module takes in a sentence and it replaces the swear words with "*". This is accomplished with the help of a database populated with negative and offensive words. Every word in the sentence is cross referenced with the offensive words and if there is a match that offensive word is replaced.

5.2 Naive Vectorization

Naive Vectorization takes in a sentence as an argument and locates all negative words. A bigram that is labeled ”negative” is made from the found negative word and the word before it in the sentence. The bigram is vectorized. This ”negative” bigram's magnitude is compared to all other vectorized bigrams compared to all bigrams created using the good words found in the stanford corpus and is replaced with the bigram that has the closest values. Thus the negative words are removed with a positive bigram as a replacement.

5.3 Smart Vectorization

Smart Vectorization module is similar to Naive Vectorization, as it also takes in a sentence compares two vectorized bigrams. The ”negative” bigram from the inputted sentence starts with the word before the negative word. This ”negative” bigram is then cross referenced with ”positive” bigrams created from the source corpus that also start with the same word. Then the magnitude of the second words of both bigrams are compared. The ”positive” bigram whose second word has the closest magnitude to the second word of the ”negative” bigram replaces the ”negative” bigram. This helps the censored sentence keep some semblance of meaning from the orignial sentence.

6 Experiments

To see how the different implemented censorship techniques can be improved, different experiments were ran. All of them were designed to help the overall improvement of censorship, maintaining a similar context, and creating grammatically correct replacements of the uncensored comments. Also as to gain insight as to why they work the way they do.

6.1 Naive Vectorization vs Smart Vectorization

This test is designed to see the difference in capabilities in keeping a proper syntactical English sentence. If an algorithm correctly replaces uncensored sentence with a grammatically correct one, it is given a point. Below you will see how the Naive and Smart Vectorization techniques compare with one another on two sets of ten sentences. The first set is an ”easier” set, which means the sentences are less ambiguous in terms of their sentiment. The second has less cut and dry questions which makes them harder to classify.

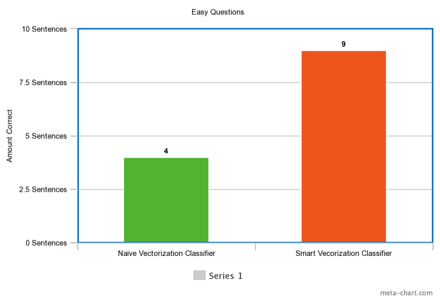

Figure 2:

Here is an example of a sentence and the outputted censored sentence of both classifiers

Naive
sample sentence - ”You are a prick”
changed sentence- ”You are as effectively”
Smart
sample sentence - ”You are a prick”
changed sentence- ”You are a rockstar”

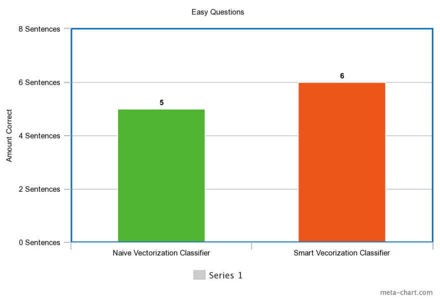

Figure 2:

Here is an example of a sentence and the outputted censored sentence of both classifiers for the ”harder” questions

Naive
sample sentence - ”This mother fucker stole my fucking wallet”
changed sentence- ”They won stole to win wallet”
Smart
sample sentence - ”This mother fucker stole my fucking wallet”
changed sentence- ”This blossom stole my comfy wallet”

6.2 Sentiment analysis

This test was designed to see how our algorithm can analyze the sentiment of a difficult and easy sentence. Difficult sentences included negative words, however in the context is still positive.

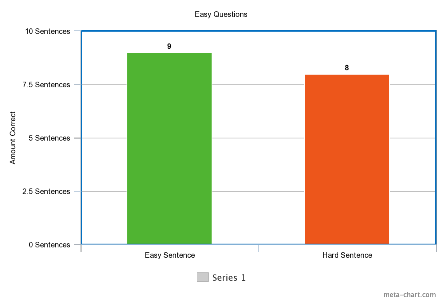

Figure 2:

Here is an example of a difficult sentence that was classified

Difficult Question
"This is fucking brilliant"

7 Conclusions and Insights

Naive Bayes over fitted the data, so Random Forest classification was used instead because it was better suited for the goal. Random Forest is better suited because we can accurately weigh the positive and negative sentences when training. For the testing, the results of our Naive vs Smart vectorization had expected results for the easier sentences. The Naive was only able to get 40 percent accuracy. While the Smart algorithm was able to achieve 90 percent accuracy. These results are expected because smart vectorization was designed to keep proper sentence syntax. However, when the difficulty of the sentences rose the Naive algorithm was able to keep up with the Smart algorithm and only miss-classifying one more sentence. A possible reason for this could be that the longer and more complicated a sentence gets, the more words you have to take into consideration when constructing a proper syntactical sentence. When the sentences as small as social media comments usually are, the algorithm is a viable solution to censoring swear words.

8 Further Research

For further research someone could create bigger n-grams consisting of the original bigrams, plus the word after or before the swear word. Another avenue would be implementing a convolution neural network instead of Random Forest for classifying. Another way to improve our research is to keep a dictionary of all words except swear words, and then replace the swear word with the most similar word that has the closest vector distance.

References

[1] Bhayani, R. and Huang, L (2009). Twitter sentiment classification using distant supervision. *Stanford*, Vol. 1, p. 12.

[2] Bing Liu, Junsheng Cheng, Minqing Hu (2005). Opinion observer: Analyzing and comparing opinions on the web. *International World Wide Web conference*.

[3] Duggan, Maeve (2014). Online harassment. *Pew Research Center*, Vol. 1, p. 1.

[4] patch (2014). List-of-Dirty-Naughty-Obscene-and-Otherwise-Bad-Words.