Classifying Movie Reviews through Sentiment Analysis

David Brehm
Fall 2021
https://github.com/dtbrehm/DSC680

Business Problem

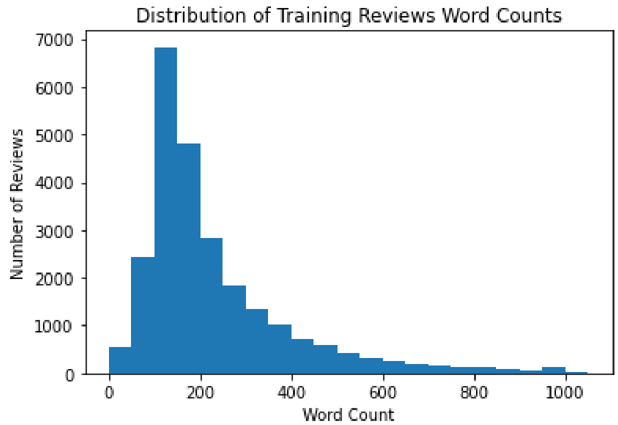
The goal of this project was to examine a set of reviews to perform sentiment analysis. Natural Language Processing is a topic I wanted to gain more experience in as it's something I might be interesting in pursuing professionally. I decided to look at IMDB movie reviews since that is a very tangible example that would be easier to interpret. There was a good dataset available online with 50,000 reviews, 25,000 training reviews that were composed of 12,500 positive and 12,500 reviews, along with another 25,000 reviews to test with.

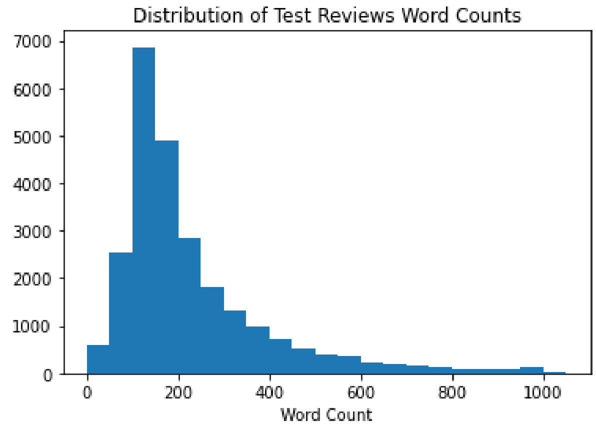
Exploratory Data Analysis

The data was already pretty well structured in that there weren't missing values and there were an equal number of positive and negative reviews in the data set. The typical text cleaning needed to be done here. Setting the text to lowercase, removing punctuation, and in this case removing line break tags as well. Below is a review example before being cleaned.

```
Homelessness (or Houselessness as George Carlin stated) has been an
issue for years but never a plan to help those on the street that were
once considered human who did everything from going to school, work, or
vote for the matter. Most people think of the homeless as just a lost
cause while worrying about things such as racism, the war on Iraq,
pressuring kids to succeed, technology, the elections, inflation, or
worrying if they'll be next to end up on the streets.<br /><br />But
what if you were given a bet to live on the streets for a month without
the luxuries you once had from a home, the entertainment sets, a
bathroom, pictures on the wall, a computer, and everything you once
treasure to see what it's like to be homeless? That is Goddard Bolt's
lesson.<br /><br />Mel Brooks (who directs) who stars as Bolt plays a
rich man who has everything in the world until deciding to make a bet
with a sissy rival (Jeffery Tambor) to see if he can live in the streets
for thirty days without the luxuries; if Bolt succeeds, he can do what
```

Something else that I was curious about was the length of these reviews. The result of this was quite a bit larger than I was expecting, with the mean review length being 233 words and a median length of 174 words. Below are histograms of the training and test dataset word counts.





Modelling

Unigrams

In order to transform the list of reviews into a format that can be used for machine learning, the reviews need to be vectorized. This can be accomplished using CountVectorizer from sklearn.feature_extraction. From here I decided to examine a logistic regression model using unigrams. The first step was finding the best inverse lambda to fit the model with. Below are accuracies for various C values.

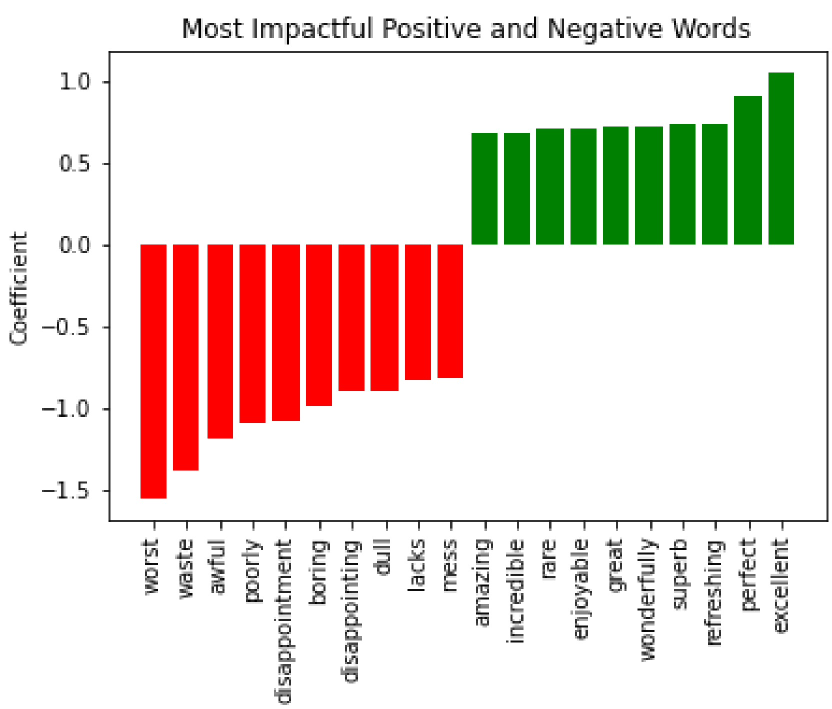
```
Accuracy for C = 0.001: 0.838
Accuracy for C = 0.01: 0.87
Accuracy for C = 0.025: 0.876
Accuracy for C = 0.05: 0.878
Accuracy for C = 0.1: 0.879
Accuracy for C = 0.2: 0.877
```

The most accurate C value for this model was 0.1. Running the logistic regression model with this C value on the test data resulted in an accuracy of 0.876.

```
Accuracy for C = 0.1: 0.876
```

Another interesting thing to examine with this type of problem is looking at which words had the largest impact on the model. Below are the coefficients for the 20 most positive and negative words, as well as a chart of the 10 most positive and negative words. None of the words in these lists are surprising at all, which gives more confidence in the model.

Word	Coef	Word	Coef
worst	-1.54547	excellent	1.04841
waste	-1.38191	perfect	0.899503
awful	-1.18223	refreshing	0.736993
poorly	-1.09556	superb	0.734325
disappointment	-1.0723	wonderfully	0.720155
boring	-0.981767	great	0.714538
disappointing	-0.895419	enjoyable	0.705139
dull	-0.893737	rare	0.701286
lacks	-0.829379	incredible	0.682931
mess	-0.812027	amazing	0.681652
avoid	-0.805743	funniest	0.670984
terrible	-0.800358	wonderful	0.66548
poor	-0.779959	loved	0.661513
fails	-0.77816	surprisingly	0.631078
bad	-0.769084	favorite	0.628813
horrible	-0.761945	enjoyed	0.62192
worse	-0.756897	perfectly	0.620734
laughable	-0.745431	gem	0.620403
save	-0.710512	today	0.618587
badly	-0.707038	fantastic	0.60854



Bigrams

After looking at this model for single words, I was curious to see if a logistic regression model performed better or worse with bigrams. The same steps were executed here with the added parameter of ngram_range=(2,2) to the CountVectorizer. Below are the accuracies for various C values on the bigram model.

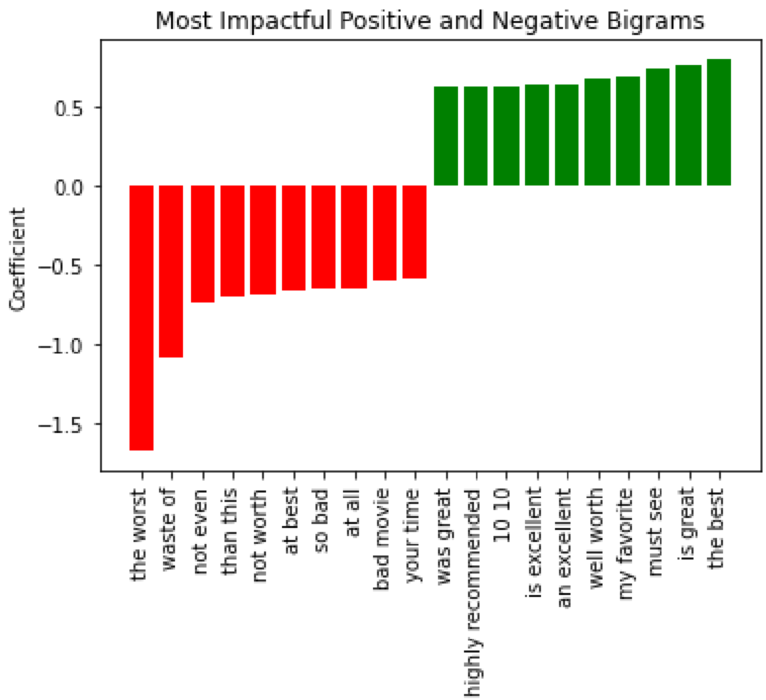
```
Accuracy for C = 0.001: 0.82
Accuracy for C = 0.01: 0.863
Accuracy for C = 0.025: 0.869
Accuracy for C = 0.05: 0.873
Accuracy for C = 0.1: 0.874
Accuracy for C = 0.2: 0.873
```

The most accurate C value for this model was 0.1 again. Running the logistic regression model with this C value on the test data resulted in an accuracy of 0.872.

```
Accuracy for C = 0.1: 0.872
```

Below are the coefficients for the 20 most positive and negative bigrams again, as well as the same chart of the 10 most positive and negative bigrams.

Word	Coef	Word	Coef
the worst	-1.67167	the best	0.801327
waste of	-1.08942	is great	0.764393
not even	-0.73852	must see	0.740755
than this	-0.702103	my favorite	0.697141
not worth	-0.685337	well worth	0.683301
at best	-0.653191	an excellent	0.640112
so bad	-0.6529	is excellent	0.636447
at all	-0.649736	10 10	0.630191
bad movie	-0.601302	highly recommended	0.625442
your time	-0.583505	was great	0.623896
not good	-0.582582	loved this	0.619108
boring and	-0.576712	definitely worth	0.602738
bad acting	-0.573722	loved it	0.596271
worst movie	-0.564118	highly recommend	0.587587
waste your	-0.557024	love this	0.5834
supposed to	-0.548217	very good	0.580102
sit through	-0.528494	enjoyed this	0.579978
the original	-0.522896	very well	0.56658
how bad	-0.51782	great job	0.541093
none of	-0.513195	on dvd	0.540354

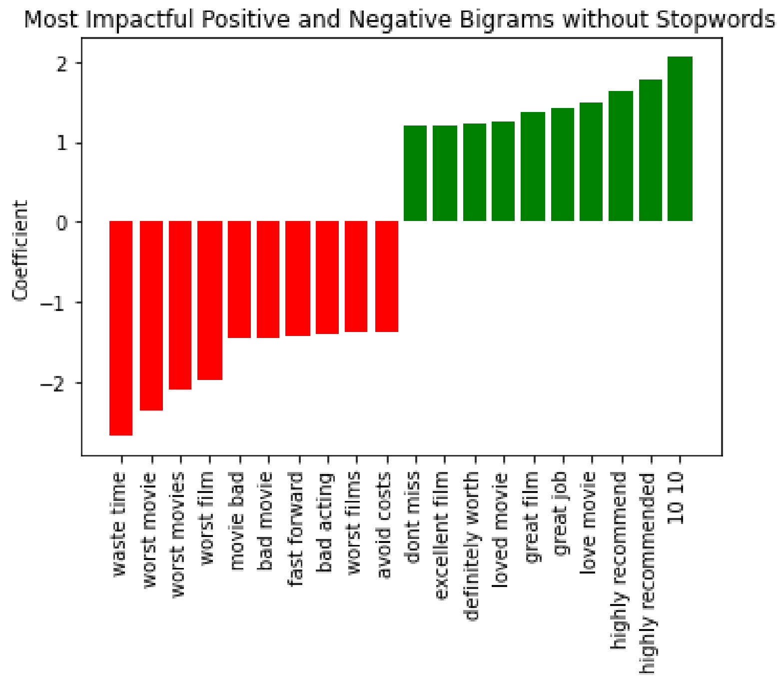


Bigrams without Stopwords

These bigrams largely contain stopwords. Will the performance improve here if stopwords were removed? I thought yes, but results said otherwise. Below are the C training values, the final accuracy of 0.821, and the top 10 positive and negative bigrams. The final accuracy was the lowest of the three different models. I thought it was interesting that "10 10" went from the 8th most positive bigram with stopwords to the most impactful without it. The top and bottom coefficients were also larger than the model that kept stopwords.

```
Accuracy for C = 0.001: 0.765
Accuracy for C = 0.01: 0.806
Accuracy for C = 0.025: 0.818
Accuracy for C = 0.05: 0.824
Accuracy for C = 0.1: 0.828
Accuracy for C = 0.2: 0.831
Accuracy for C = 0.5: 0.833
Accuracy for C = 1: 0.834
Accuracy for C = 10: 0.834
Accuracy for C = 100: 0.834
```

```
Accuracy for C = 1: 0.821
```



Results

The logistic regression model using bigrams performed slightly worse than the model with single words. This was a bit surprising at first until I noticed that this still included stowords.

I thought the overall positive and negative words were decently insightful. They were basically what one would expect if they had to describe positive and negative sentiment. I suppose it is interesting how even with this seeming good categorizations, the model was still less than 88% accurate.

Another interesting note was that removing stopwords for the bigram model required a higher C value and was less accurate overall. This trend of unigram being more accurate than bigram, and bigram being more accurate than bigram without stopwords is basically the opposite of what I was expecting.

Overall though, I thought the results here were promising. This type of analysis could be used to study which words tend to be the most positive or negative. It could also be used to compare langauge between different platforms. For example comparing positive movie review words to positive words for restaurants or consumer products.

Questions

1. Would other models have performed better than logistic regression?
2. Would narrower steps on C values discovered a higher accuracy model?
3. How well do these most impactful words carry over to reviews on different sites?
4. How would using higher n-grams perform?
5. How would using a range of n-grams such as both unigrams and bigrams perform?
6. How would stemming affect the accuracy of this model?
7. How does TfidfVectorizer perform compared to CountVectorizer?
8. What other steps could be done to improve the accuracy of this model?
9. Why do different C values result in different accuracies?
10. What words were neutral in this dataset and did not have an impact on the model?

References

IMDB Movie Reviews Dataset:

Maas, A., Daly, R., Pham, P., Huang, D., Ng, A., & Potts, C. (2011). *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics.