Brandon Butler

Troy Carloni

Dr. Rahul Singh

Multimedia Systems CSC 664 - 01

13 Dec 2022

**Assigning Scores to Movie Reviews by Sentiment Analysis – Project Report**

**Introduction**

Team 2 was tasked with making a program that is capable of sentiment analysis. This program would be tasked with looking over a Imdb movie review and determine the sentiment of the review and scale it between 1 – 10.

Sentiment analysis is becoming increasing important and used in our society. From brands and celebrities using it to get a better understanding of their audience and how they feel when talking about them on social media; to politicians doing the same to figure out the increasingly volatile political climate, sentiment analysis is a valuable task.

This report will go over how we planned to handle this project, a high-level description of our work, our experiences before during and after the project, and an experimental evaluation on how our project ultimately worked.

**Proposed Problem Formulation**

For the project we want to be able to determine if a review is positive or negative using just code. In order to accomplish this, we need to first make a program that can look over an entire review and separate the words from the punctuation and meaningless words. Next, we need to find a way to figure out if these words are positive or negative. Lastly we need to make culminate all of this information to determine if the review as a whole is either positive or negative. These three steps are critical to our project since they are how we get all of our data, breakdown its meaning, and come to our conclusion.

**High-Level Description of Your Idea**

To accomplish this, we determined that we would need a large sample of positive and negative reviews that we would use to compare with the review in question in order to see what similarities they have. From there we would use our result to get our positve or negative score, 1 being very negative and 10 being very positive.

To do this we used a data pool of Imdb reviews gather by Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts of Stanford University. With their data set we had over 25’000 positive and negative reviews respectful.

We then used the rake\_nltk library to extract keywords and phrases from both the data set and the review in question. When looking at the data set rake would look over these words and phrases and see how many documents it appears in. Based on this, it would assign a valued to these words and rank them on how positve and negative they are. With this it would then look over the review in question and looks for these words and phrases. Reviews with more positve verbiage and less negative verbiage would be scored as positive and once it complete, they would be put on a scale from 1 – 10.

**Prior Work in the Area**

Brandon Butler – Personally, I had never worked on anything like this before. My only experience with sentiment analysis came from what we had been taught in class. To make things worse this class was my first-time coding in python as well. While those made my first steps on this problem feel insurmountable, I got to work reading some articles and documentation to learn as much as I could as fast as I was able to.

One of the first things I looked us was some documentation on how to work within a python environment. This made me realize that python was no where near as dauting as I once thought. All of my past experiences in coding had made this endeavor a relatively painless one.

To gain a better understanding of how we would make this program I read documentation by Abinash Tripathy, Ankit Agrawal, and Santanu Kumar Rath, in ther article called “Classification of sentiment reviews using n-gram machine learning approach”. While parts of their article were more directed toward machine learning we got the idea of getting a large pool of data that we would use to compare with our review. This was similar to what were taught in class.

Troy Carloni

As far as the topic goes, I have never explored text analysis. It was an interesting venture, I didn’t have an understanding of how powerful text analysis can be, after going through this project I can see how useful it is. I do have a fair amount of experience with manipulating strings, and this helped a lot when it came to formatting the collection of text/data into useful data to analyze.

**Description of Your Work**

Open and neglist (Troy)

Text

Description automatically generated

Opens reviews from the data set and puts them into an array. This is so the words and phrases can be seperated from one another and eventually counted and ranked on how negative they are.

poslist (Troy)

Text

Description automatically generated

Similar to neglist, this is used to put all of the positve words into an array to eventually be counted and ranked.

testList (Troy)

Text

Description automatically generated

Within the “test\_corpus/” folder are our defined positive and negative reviews along with one written one. This assigns all those reviews within that folder as testList to be used for testing along with just general sentiment analysis.

Analysis open and rake (Brandon)

Text

Description automatically generated

This is where rake extracts the keywords and phrases from the documents. Along with this it also gets the frequency of these works and phrases. From there we assigned these as to the identifies of test and phrases.

key in test (Troy, Brandon)

Text

Description automatically generated

While looking over a file the number of negative and positive keywords are counted using this.

phrase in phrases (Troy, Brandon)

Text

Description automatically generated

Similar to the previous lines of code, this is used to count the number of positve, and negative phrases a given file may have.

Calculations (Troy)

Text

Description automatically generated

Given all of the past calculations, we take all of the words and phrases and assign a wordScore to them based on their frequency and overlap in other positve and negative reviews that also use them.

write (Troy)

Text

Description automatically generated

sum and score (Troy)

Text

Description automatically generated

Once all of the scoring of words and phrases is complete, the document as a whole gets its positive to negative rating. This is done by looking at the file as a whole and counting up all of these words and phrases. Depending on how many of these used and a score is set from 1 to 10, 1 being very negative and 10 being very positive.

Box Chart (Troy)

Text

Description automatically generated

Based on the suggestion from the professor, Troy made it so each time the code is run it will also produce a graph of the results so that people can more easily examine the data.

Data Collection and Creation (Brandon)

Graphical user interface, application

Description automatically generated

Based on the suggestion from the professor, Brandon selected three negative and three positve reviews to be used for testing. This is so we have a clearly defined test on known negative and known positive reviews. In addition to this, Brandon wrote his own review of the movie “Barbarian” and tried to be as middle of the road on the movie to see if the program would see it accordingly.

**Experimental Evaluations**

When we initially made this program, we ran into issues with getting a clear sentiment analysis for about 33% of our sample size. If we ran 6 tests only 4 would come back with our desired results. We also noticed that this only happened when looking at negative reviews. After our 2nd phase presentation, the professor suggested that our issue might be that we weren’t accounting for some verbiage or nuances in the English language. For example, if someone were to say a movie was “good” our old program would register that as a positive statement due to good showing up in many positive reviews when describing movies, the reviewer enjoyed. However, if we came across a review that said a movie was “not good” then they would just cancel each other out seeing how “not” is typically a negative term used in negative reviews and the aforementioned “good” is positve. Despite the statement “not good” meaning something negative our old version of our program never accounted for this, thus our negative reviews weren’t very accurate.

That was when we discovered rake and found a better way to approach this issue. Rake was able to look over the words and even phrases a lot faster and accurately than before. Upon testing it after this we got more and more accurate results. To test this, we took some more advice the professor suggested and wrote our own reviews. In these reviews we wrote some extremely negative ones, along with some glowingly positive ones. Lucky, the program was able to accurately determine these reviews as positve and negative.

To further test this however, we decided to try and write a review that was very middle of the road on its feelings for a movie. If our program was able to score this within the range of 4.5-5.5, we would know that our program was completely accurate. This is where we ran into some issues. For the times we tested this middling review Brandon wrote it testing within that range 80% of the time. While that was a pretty good result, we wondered why this was the case.

An idea was the use of slang within the document confused the code. It was possible that the reviews within the data set didn’t use the same type of slang words and phrases or disproportionately used them in positive or negative reviews thus skewing our scoring.

While an 80% is a decent amount of accuracy when looking at ambiguous reviews, it does imply that our method wasn’t sound and can be worked around assuming someone was purposefully trying to write a review to get past our code. While at first, we believed this to be an unlikely thing to happen in the real world, we remembered just how chaotic our world is.

After the final presentation, the professor suggested that we compare what we grade the reviews on a scale from 1- 10 as we would see it and then compare them to what our program got. To accomplish this, we chose 25 random reviews each and read through them all. We think decided how positive or negative these were before putting all of our scores into a list. Once we ran our code it produced a graph showing our ratings and what the program believed.

Chart, bar chart, histogram

Description automatically generated

The blue lines are what we believed the reviews were and the red are what the program got. When looking at a negative review what we thought was a 1 the program thought was around 2.93 or in more extreme cases what we thought was a 3 the program believed was an 8.5. While this was surprising generally, we and program agreed that a positive review was positive, and a negative was negative 77% of the time.

This differed from our original written test and upon further reflection we believe we can answer why. In terms of why our scores varied even when agreeing that a review was positve or negative has to come with language and how we interpret some words. While we might see phase like “worst movie ever” extremely negatively to the point that we give it a 1 the program might see that as not that serious. Even among each other we graded some of some movies differently based on our biases.

In terms of why we disagreed with our program 23% of the time could be a number of reasons. In our final presentation, the professor suggested that our program might be skewed due to some words cancelling each other out. While we didn’t fully understand or underplayed this then we now see that this might be at the root of that 23%. Another reason could just be the same biases that altered our scores when we agreed. Upon rereading some of the reviews our opinions changed for better and for worse.

**Conclusions**

Now that we have reached the end of the semester, we can look back at the work we have done and can stand proud. We were tasked with making a program that can determine if a review is positive and negative and we accomplished that by taking all of the words and phrases in a document and comparing them to a data set of defined positive and negative reviews. From there we were able to rank and count these words and phrases and give a score to the document as a whole to determine if it was positive or negative.

While this is impressive, moving forward we need to account for people trying to be ambiguous with their reviews due to our program struggling to more accurately determine where these reviews lie. While it might be an impossible battle to combat bad actors in this ever-growing field, it is a necessary battle to combat the spread of misinformation across sites dedicated to reviewing movies and other media to social media accounts of world leaders and those impersonating them.

This has been a very insightful experience, that we won’t soon forget.

**References**

# Bibliography

Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., & Potts, C. (2011). *Large Movie Review Dataset v1.0*. Retrieved from Large Movie Review Dataset: https://ai.stanford.edu/~amaas/data/sentiment/

Martineau, J., & Finin, T. (2009). Delta TFIDF: An improved feature space for sentiment analysis. *3rd International AAAI conferece on weblogs and social media*.

Tripathy, A., Agrawal, A., & Kumar Rath, S. (2016). Classicfication of sentiment reviews using n-gram machine learning approach. *Elsevier*, 117 - 126.