Moviereview.py – A tool for the lazy movie review reader

By: Steve Bárdi

Assignment #3 – Text Mining and Analysis

Project Overview

I used IMDB as a source for this project. Instead of choosing a static source, such as a movie that is already specified within the code, I used IMDB's search feature to allow the user to search for a movie. I also used Natural Language Processing (nltk.org) to check for sentiment within the five best reviews (best as defined by IMDB). The goal of this project was to create a tool for people who don't want to rely on mere ratings but are also too lazy to actually read reviews (I may also be this type of person).

Implementation

For this project I used the following libraries: Imdb and nltk. NLTK required the 'vader_lexicon' package to be downloaded as well. As such an extra line of code was added to ensure that this package is downloaded as part of the code. In terms of implementation, I wanted to take advantage of imdb's search_for_title() function. The concept behind the search is that the code asks for a user input which is then submitted to the Imdb package. The package then returns a dictionary that has a title, year, and an imdb_id(used later as a reference). The user then can either verify the search as having found the correct movie that he/she has been looking for OR can retry by typing 'no' (invalid inputs are also handled).

Once, the search() function has established which movie the user wants analyzed, we use the NLTK package to analyze the sentiment within the 5 best reviews. Here, I found that it is easier to let Imdb deal with what it considers best. This was a design decision because I am confident that their algorithms will be more accurate than mine (were I to develop one). Note: best does not mean positive. It means that they are the most 'valuable'. Another design decision I made was that instead of processing each review individually and then averaging their scores, I just combined all 5 reviews as part of one large string. This allows us to run a single SentimentIntensityAnalyzer() on the string (Simple is always better).

The scores are then stored in a dictionary which we then use to read the negative and positive sentiment scores out of, using the .get() function. If the positives outweigh the negative, we call the average sentiment within the 5 best reviews positive and vice versa.

Results

My design was meant to be simple on the output end. The goal of this project was to create a simple/rapid output. As such, there are no fancy graphs that are generated. The output of the analysis performed by my code is one of these:

- The top 5 reviews for the movie: [TITLE]. Has been generally positive.
- The top 5 reviews for the movie: [TITLE]. Has been generally negative.
- The top 5 reviews for the movie: [TITLE]. Has been generally neutral.

A neat design consideration is that the program does not quit, unless the user responds 'no' to "Would you like to check another movie?". This allows the user to check multiple movies without having to restart the program. I decided to intentionally not display (print) the dictionary with the sentiment scores, with the purpose of keeping the output as simple as possible!

Examples (**bold** characters are user input):

Please Enter a Title ➔ **Jackass** ➔ {'title': 'Jackass: The Movie', 'year': '2002', 'imdb_id': 'tt0322802'} Does the above match your selection? If not, you will asked to enter the title again. PLEASE MAKE SURE TO ONLY TYPE EITHER 'yes' or 'no' ➔ **yes** ➔ Your selection has been confirmed! ➔ The top 5 reviews for the movie: Jackass: The Movie . Has been generally positive. ➔ Would you like to check another movie? 'yes' or 'no' only ➔ **no**

Jackass: The Movie = Generally positive sentiment

Please Enter a Title ➔ **Emoji** ➔ {'title': 'The Emoji Movie', 'year': '2017', 'imdb_id': 'tt4877122'} Does the above match your selection? If not, you will asked to enter the title again. PLEASE MAKE SURE TO ONLY TYPE EITHER 'yes' or 'no' ➔ **yes** ➔ Your selection has been confirmed! ➔ The top 5 reviews for the movie: The Emoji Movie . Has been generally negative. ➔ Would you like to check another movie? 'yes' or 'no' only ➔ **no**

The Emoji Movie = Generally negative sentiment

--- At this point the process is evident, I will list a couple other movies and their sentiments ---

Avengers (2012) = Positive

Taxi Driver (1975) = Negative

The Godfather (1972) = Positive (no surprises here)

Fifty Shades of Grey (2015) = Negative

Overall, I am very satisfied with the results. I did a good couple rounds of testing and the sentiment analysis generally agrees with the public's consensus. However, there are scenarios where a sample size of 5 reviews can falsely identify a movie as positive. However, it should be noted that we are analyzing the reviews sentiment. Therefore, it is possible that in some cases even a review criticizing a film may come off as including more positive sentiment than negative sentiment.

Reflection

I really should have managed my time better. I had a small issue with the NLTK package that was holding me back from completing this assignment and I left it to the last minute to troubleshoot. Luckily, I figured out what the issue was (vader_lexicon missing) and it was smooth sailing from then on. What I have come to realize is that perhaps linking the 5 reviews together as a single string may not be the most accurate way to judge average sentiment. This is because a long review has more impact on the average sentiment than a shorter one (basically creating a false weighted average). This means that my design choice makes the analysis perhaps less precise. I've made that choice to maintain simplicity, but perhaps the accuracy of the results could be improved by individually assessing the sentiments first and then averaging the sentiment scores together. In terms of testing, I built-in a repeat function from the start. Therefore, it was relatively easy for me to check portions of the code rapidly. I used intermittent (and temporary) print functions all throughout the development stage to see where to code gets stuck. This troubleshooting method really helped me out. This assignment really showed me how I could use packages in a way that solves a real-world issue that I have. That is being too lazy to read reviews. I wish I would've known about the vader_lexicon package before I started because that really messed up my time management.

I chose to work alone on this project to ensure that I do all the work. This way I was able to learn a lot through troubleshooting. While the vader_lexicon error was super frustrating, it also taught me how to troubleshoot on my own. I used the nltk.org website and the installation guide within the website to troubleshoot and I figured out that the specific package still needed to be installed. If I could do anything differently next time, I would start troubleshooting any issues I have much earlier. This error was nothing too serious, but it was enough for me to say, I will deal with this later. Then I managed to push it off until the last minute, which also almost cost me missing the deadline. If anything, start troubleshooting earlier.

Honor Pledge

I, István Bárdi, pledge my honor that I have neither received nor provided any unauthorized assistance during the completion of this work.