CS410 Project Progress Report: Sentiment Analysis of Letterboxd
RMK Group (Rui Mao, Matt Malitz, Karl Vosatka)
Team Captain: Karl Vosatka
Prof. Cheng Zhai
11/14/2022

We are building a tool that will perform sentiment analysis on reviews scraped from letterboxd.com, a social media website for film reviews. The project consists of a web scraper component to collect reviews, a sentiment analysis model trained on movie reviews, and an app to enable users to interact with the results of analysis.

The review scraper is constructed and works as desired. Given the URL for a movie listed on letterboxd.com, the scraper will acquire the full-text review data for the 60 most popular reviews of that movie for analysis. It can also easily be adjusted to grab reviews sorted on the different sorting options available on letterboxd.com, including high/low numerical rating or recently posted reviews. Reviews are stored as part of a JSON file, along with other metadata for each movie such as the title, cast and crew, and, importantly, the average numerical rating of that movie. At first, it was difficult to pull certain reviews, because all reviews past a certain length are partially hidden by a “more” button that requires input from a website user, which is incompatible with the BeautifulSoup toolset. However, we resolved this problem by finding and following hidden links that reliably yield full-text reviews. Going forward, we hope to build a function that will run the scraper on different movies in order to develop a dataset for proper sentiment analysis model training.

A basic sentiment analysis model is implemented based on NTLK’s built-in data set - Cornell university’s polarity dataset v2.0. The dataset includes 2000 positive or negative movie reviews and is split to 9:1 as training and testing sets separately. The model was trained by using the SVM algorithm. A feature vector is constructed by selecting the top N (N is 500 or 5000 for now) frequently used words in reviews. Stop words are removed during this process. The model achieved 89% testing accuracy, and a pre-trained model was saved for the

sentiment analyzer. A prototype sentiment analyzer is implemented. The input is a text file that includes the test movie reviews. The input also takes a feature vector generated from the above model. The pre-trained model will be loaded to run the sentiment analysis and provide a result for each review as either positive or negative.

The major challenge faced is that the accuracy of the predicted sentiment analysis results is much lower than expected. One reason might be that the training dataset was collected before 2004, and many internet languages used nowadays are not included in these reviews. The future task will be working on the genuine letterboxd reviews and JSON input format, trying different training algorithms, and increasing the feature vector size. We may also use a different training dataset (also from letterboxd) for a better result if time allows. There are two main places where we could use some guidance from TAs or other students about our model. First, to get an accurate prediction or result, how large of a dataset should we use in terms of number of reviews? Second, if we are able to implement a model trained on scraped letterboxd reviews, how can we randomly sample reviews and movies to build our model? To what extent should we concern ourselves with random selection for building the model?

The Java applet backend is completely finished, but the front end still needs to be started. The backend at this moment has the ability to launch a python program within Java using the ProcessBuilder package. The frontend will require that I write some sort of way for the user to select a review which they would like to receive sentiment analysis for, and also allow the user to review the results of said analysis. This will likely result in numerous bugs having to do with race conditions since the front end needs to wait for the python sentiment analysis to complete before letting the user select another review. I will solve this problem using synchronized blocks along with using the wait() method when running fork exec functions. Another major problem that I am expecting is OS inconsistencies resulting in errors when forking and execing. The problem is that different OS's have different terminal commands which means that "python3 main.py" might result in an error in some OS's while executing the program

flawlessly in other OS's. Because of this I will have to do some extensive testing to make sure that the code I write is platform independent.

Based on our work so far, we are planning to scale back our application for now and focus on building a solid working model that is able to perform sentiment analysis that is somewhat accurate when analyzing letterboxd data. Since letterboxd reviews are influenced by social media communication styles, we may consider using letterboxd-scraped reviews to train our model, even though it may result in over-fitting. If possible within the time left to work on the project, we hope to build a model that is more accurate to the varied ways that people approach discussing topics they love in a social media context. We will assess the model's accuracy based on comparison of sentiment analysis output to the numerical rating associated with a review (i.e. we expect a written review that is scored as positive to have a score higher than 3). We hope to then collect sentiment analysis results of a number of the most popular reviews for the movie, and compare those results to the score that the movie received overall.