Movie-Reviewer Final Report

Zephren de la Cerda
Kadyn Marshall

Abstract

We created a program that generates a review based off of the user reviews on IMDb. The user gives the program the name of the movie, and whether they would like a positive or negative review. The program will generate language models from the user reviews and generate a new review for each of the three models we used; trigram, markov chains and machine learning. As a result, the movies that had a higher review counter produced higher quality reviews. Between the models, with the same data set, the trigram model was one of the faster models with reviews that almost sounded like they could be written by a human. The machine learning and markov model produced readable but nonsensical reviews.

Introduction/Motivation

Our motivation for this project was to see if we could generate a movie review that fit the following three criteria: a review that made sense, described specific details about the movie, and gave a positive or negative connotation. We believed if we could generate a movie review that met these goals, then it would be comparable to real movie reviews written by real people.

Related Work

A lot of similar work has been done on fake text generation. Some examples include fake amazon product reviews and fake news generation. The models that are

commonly used are N-grams, markov models, and recently neural networks have attracted a lot of attention in this field. After researching and reading several articles that describe these methods in detail, we decided to try using all three of these models for our review generation. For the N-gram model, we used our implementation of tri-grams from assignment 2. For the Markov model we used an implementation from an online article. For the Neural network we used a package called textgenrnn. These sources can be found in the references section.

Recurrent Neural Networks (RNNs) have attracted a lot of attention for their ability to solve complex machine learning problems. RNNs fundamentally differ from other types of neural networks, such as simple feed-forward networks or Convolutional Neural Networks, because the current state of the network is “remembered” and fed in as an input to the next time step.

System Description

Data

When deciding on a dataset to use for our movie generator, we first looked at a few free sets online that had millions of reviews in them, but they were limited to a smaller set of movies because the reviews were only collected between a limited set of years. We thought about all of the online sources with expansive collections of movie data and reviews, such as Rotten Tomatoes, Metacritic, and IMDb. This led us to think: If this is where millions of people go to write and read reviews, then why can’t we just use those as our dataset? After some research, we decided to create our own dataset by scraping reviews straight from IMDb’s user reviews web page on the spot when the program is run. This allows for a more dynamic dataset that grows with time as movies are released and more people write reviews.

Since Movie-Reviewer fetches the dataset at runtime, the dataset and corpora are not static. Therefore we cannot simply provide a link to the dataset. Instead, we will

provide an example of what the data looks like, and then expand on how our program retrieves the data in the implementation section.

Here is an example of one of the thousands of reviews we pull straight from IMDb. Each review has a title, and a rating along with it. This rating is what we used to determine whether the review was negative or positive. Having this rating associated with each review allowed us to divide the reviews into three corpora: one for just the negative reviews, one for the positive, and one that contains all of the reviews.

Implementation and Algorithm

The first step that the program takes is to collect input from the user. It prompts the user for the movie title, the type of review they want (negative, positive, or average), and which models they would like Movie-Reviewer to use. The program then uses an API called OMDb (see references) retrieve IMDb's movie ID for the user provided movie title. Once we had the movie's ID, we just had to append it to a specific URL like so:

'https://www.imdb.com/title/' + imdbID + '/reviews/'

Once we figured out how to programmatically retrieve the URL, we had to figure out how to parse the HTML and extract the reviews. This part was mostly about figuring out how the html was structured, but the real challenge came when we discovered that the site only loads 25 reviews at a time and requires the user to click the “Load More” button to load the next 25. We knew 25 reviews wouldn’t be nearly enough for generating our models, and if we weren’t able to figure out how to load all of them then we would have to look for data elsewhere.

By tracking the network flow when clicking on the load more button, we noticed that it was actually making a request to a different URL. After more digging in the HTML we notice that the load button has an attribute called data-key that contains the key to the next 25 reviews. Appending the key to the ajaxurl, also provided by an attribute in the load more button, gives the URL to retrieve the next 25 reviews. Here is the html tag for the load more button, which includes its attributes:

After this discovery, we wrote the code to read 25 reviews, fetch the next 25, then read those and so on until all reviews were read and we had our corpus. From there we were able to separate the reviews into two separate corpora, and feed them into the models that the user specified should be used.

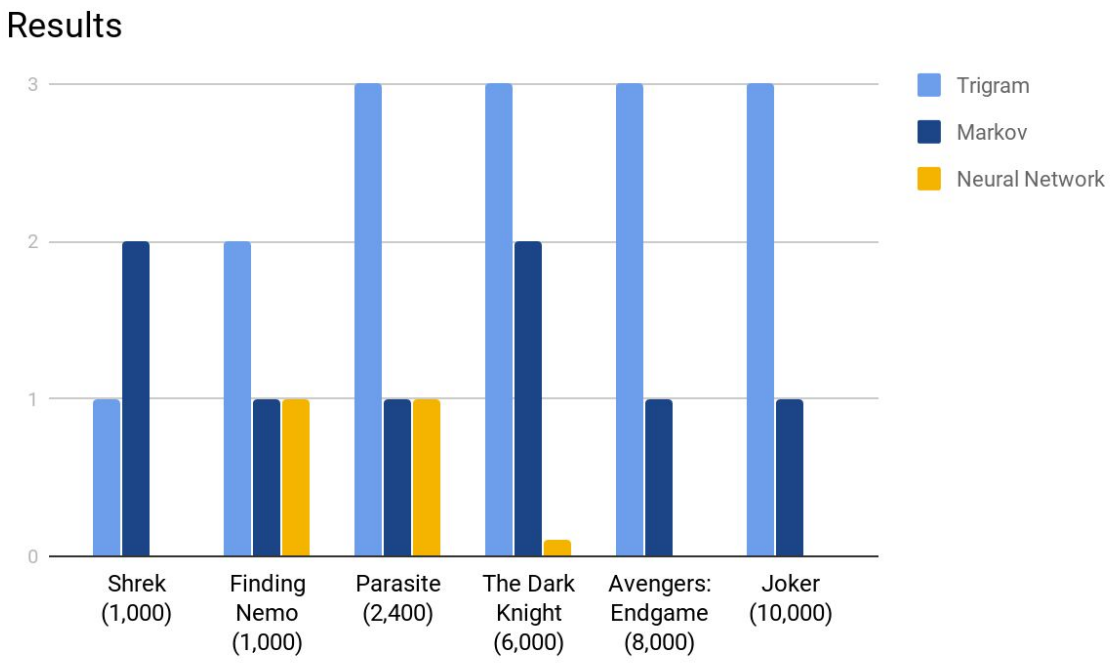
Trigram model tracks three tokens at a time, and uses the previous two tokens to predict the next token. It chooses the 5 most likely tokens to come next, and then picks a random one. The markov chain uses a similar approach, in that it uses the current word to predict the next word, but there are more probabilities involved that determine the most likely next word to be used (see reference for more information). The implementations of the Trigram model and Markov model were modified so that it produces a review that is the average length of all the reviews. The Neural Network model was not modified as it is a complex package that we could only use right out of

the box.

Experiment setup

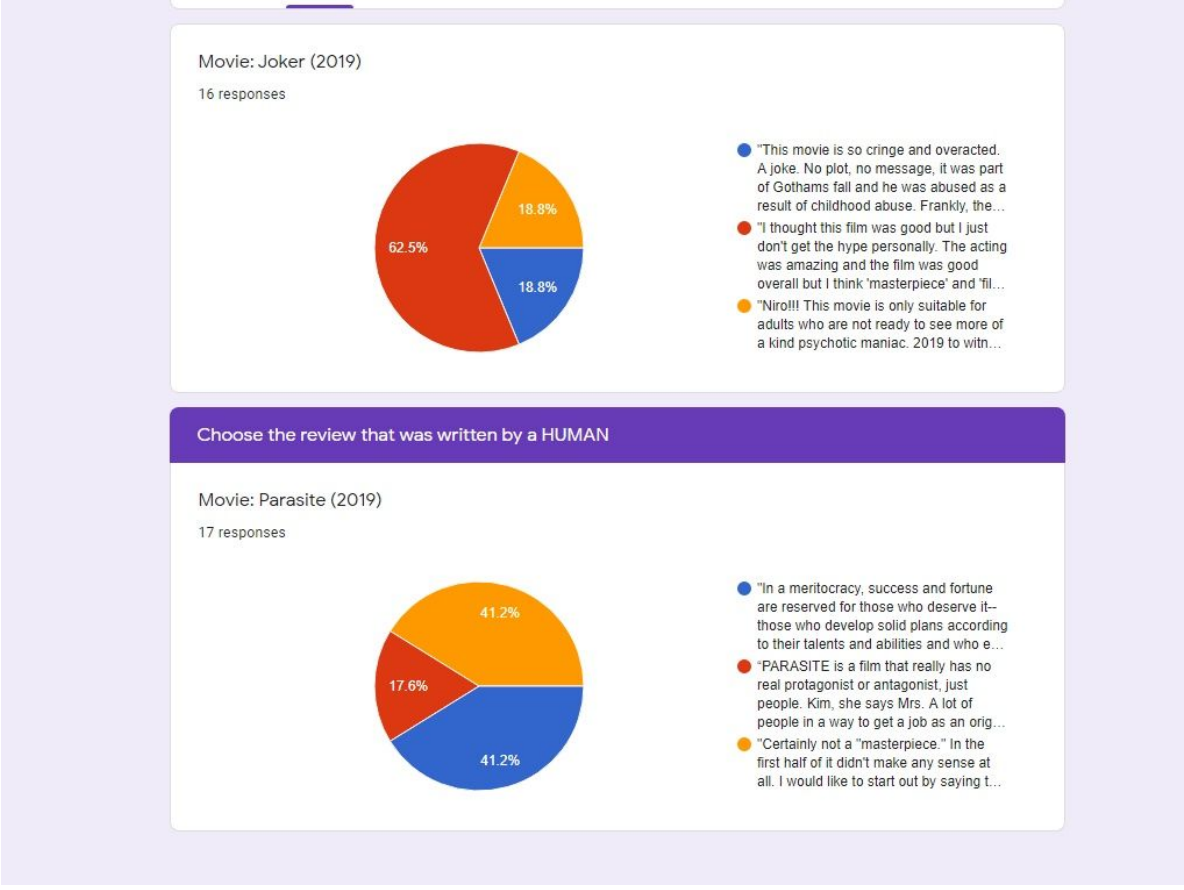
To test the accuracy of models, we ran two types of experiments. The first one was using movies that varied from release date, genre, and number of written reviews on IMDb. We wanted to see if our generated reviews would be higher quality if we have the models more words to train on, and if the reviews would be more mature or immature depending on the movie it was writing. We also wanted to see if we could trick people into believing that our generated reviews were written by a real human.

Evaluation and Discussion



The results of our experiment were quite surprising. We expected the Neural Network to produce the highest quality movie reviews compared to the Trigram and

Markov model, although this was not the case. The machine learning model took hours to run, as the other two models only took minutes, and the results were very disappointing for the first few movies we ran it on. The Markov model was also a bit disappointing, some of the reviews were readable but did not make any sense. The trigram model was our winner, we noticed there was a threshold for the number of reviews needed for an output that made sense, and once that threshold was met (around 2,000 reviews), every review it generated was very close to a human review. We were even able to fool some of our classmates that believed our computer generated reviews were real. For our first test, we used one real review that was posted on IMDB (red), and two fake reviews generated by our trigram model (blue and orange). Only 62.5% of the class guessed correctly. For the second test, the review written by a human was blue, so only 41.2% of the class guessed correctly.



Conclusion and Future Work

The Movie-Reviewer project was a success. It uses a dynamic dataset, it creates reviews that make sense and express sentiment, and it was able to fool people into thinking its reviews were written by a human. We are proud that we were able to pull it together in a few week period, and that we overcame the challenges involved in using a non-static dataset. The results of the different models were unexpected, but satisfactory nonetheless. The Trigram model exceeded our expectations, and the Neural Network model produced disappointing reviews. We suspect that this has to do with the amount of time that we let the model train on the data before producing the review. We left it at the minimum while conducting our experiment, because it still would take over an hour to run for the movies with thousands of reviews. In the future, we would like to let the model take its time to train longer and see if the results improve.

Aside from trying to improve the models, we would like to make a few other changes that would improve the program as a whole. We would like to also feed the review titles to the models, so that fake review titles are generated along with the fake reviews. We would also like to improve the movie search feature, because as of now it is impossible for the user to know which movie is being reviewed when they enter in a title such as “Batman” because there are many batman movies on IMDb. Finally, once we have made all of the discussed improvements, we would like to host the application on a web server and create a simple UI so that the project is accessible to anyone online.

References

Minimaxir. (2020, April 28). Minimaxir/textgenrnn. Retrieved May 27, 2020, from https://github.com/minimaxir/textgenrnn

Build a Markov Chain Sentence Generator in 20 lines of Python - jeffcarp. (n.d.). Retrieved May 27, 2020, from https://www.jeffcarp.com/posts/2019/markov-chain-python/

Shaver, B. (2017, December 29). Simulating Text With Markov Chains in Python. Retrieved May 27, 2020, from https://towardsdatascience.com/simulating-text-with-markov-chains-in-python-1a27e6d13fc6

OMDb API. (n.d.). Retrieved May 27, 2020, from http://www.omdbapi.com/