Decoding the Critic: A Semantically Aware Movie Recommender System

Grant Doan, Prayash Joshi, Reagan Orth, Ved Patel

November 16, 2025

1 Abstract

Creating a movie recommendation system remains a challenge, as many existing ones need help with issues such as cold start, data sparsity, scalability, the filter bubble issue, and Sentiment Consistency and Ambiguity. Our project aims to create a recommendation system that accommodates the entirety of a user's interest by creating a model that focuses on the similarities between different users. By creating such a system, we will counter the common issue of recommendations centered around only one movie. Additionally, as users input more reviews into our system, recommendations can reliably improve. The approach for our system consists of two scraped datasets, one of which consists of all data associated with a particular movie and another which consists of significant reviews. Using natural language processing, we can isolate specific film elements that users prefer, such as actors or directors. Next, we created a graph neural network that used the movie dataset to create movie similarity mappings. Finally, we created a recommendation algorithm that used different metrics to recommend movies.

2 Technical Description

2.1 Introduction

2.1.1 Problem Description

Movie recommendation systems always have biases, and a successful system will eliminate as many of these as possible. A very prominent example is popularity, because people trying to see the best films of all time tend to watch many of the same, unrelated movies. In the IMDb recommended selections for Mr. Smith Goes to Washington (a 1939 comedy/drama about a boy scout leader who becomes a moral idealist US senator), we can find the film Wages of Fear (a tense 1955 French suspense thriller about hired drivers transporting dangerous explosives across a hazardous road). Both are highly rated films, but they have very little do with each other, and there is little reason to believe that a user who likes one will like the other beyond the fact that both are old and filmed in black and white. A more modern example is that Lord of the Rings (a fantasy journey film), Joker (a dystopian antihero film) and Interstellar (a huge-production sci-fi space travel film) are recommended on the page for The Shawshank Redemption (a relatively slow-paced, atmosphere- and feeling-driven prison film). None of these three recommended films have a single prison scene, they don't share starring actors or a director, they don't share the slow pace, and as a whole are very different films that appeal to very different audiences.

Another frequent issue is grouping films with other films of similar ratings. When looking at a film of rating 8.0, the recommended films tend to be in the high 7's and 8's. When looking at films of rating 5.0, the recommended films tend to be high 4's and low 5's. However, clearly every user will rate films differently, and users will not base their ratings on the IMDb average user rating. Therefore, we believe that although rating-based recommendation has some limited merits, it is more harmful than helpful overall.

2.1.2 Motivation

The goal of this project is to create a movie recommendation system that removes common biases and issues with the popular systems today. Movie preferences are very complex, and it's often fairly difficult as a viewer to know what kinds of films you enjoy. Even after you know, some themes or genres are easier to find, some are harder. In all cases we're confronted with the problem that we do not know how much we will like a movie until we see it, which costs us between one and four hours for each one. Online lists can be very helpful, reading reviews and watching trailers might sway one's opinion or give a little indication of how good the film will be, but there is no way to ever know for sure.

In order to solve this problem, we have created a recommendation system that takes in commentary from users who have already seen the movies, and uses that information to better understand if it might apply to the user in question. In this manner, we have been able to predict how much the user will enjoy a movie to some degree of accuracy, which in turn will allow the user to watch more movies that they enjoy and fewer that they dislike.

2.2 Literature Review

General overviews of the recommendation system research space decompose filtering methods into 3 categories, each with their own frameworks and methodologies of execution: Content based filtering, collaborative filtering, and hybrid filtering. Each of these filtering approaches are addressed along with recent executions in the literature space.

2.2.1 Content Based Filtering

Content based filtering takes the approach of finding items that share characteristics motivated by the idea that if a user enjoys item A, and item A is similar to item B, then the user in focus will enjoy item B. As a user, it is easy to find similarities between movies through making the connections between genres, actors or actresses, directors, and other qualitative factors, but for machines, quantifying these things precisely and efficiently is the focus of nearly all of the current landscape of recommendation systems. A content based recommendation system approach supplied by Pujahari and Sisodia in 2022 propose matrix factorization-based feature refinement using textual data. Term Frequency Inverse Document Frequency (TF-IDF) extracts the essential words or phrases in the text documents and displays it in a vector space, weighing highly reoccurring words as essential. In order to combat the sparsity issue that plagues vectorization of text data, the authors use a factorization technique that attempts to lower the dimension finding factors that correlations that are redundant and optimizing feature refinement. The item feature matrix F is subject to reduction based on projecting the feature matrix on a vector W where that vector is the combination that either includes or doesn't include a certain quality in the similarity measure. This is upheld by the constraint that $W^t * W = I$. Therefore, by finding optimal matrix of weights for our feature space, sparseness and redundancy between factors is minimized. Furthermore, finding optimal values for W allows them to select the most important features for consideration by the system, which is a trade off if there is a limitation on existing computing power. They then propose an iterative optimization process for W and another factor K which is the coefficient matrix used for projecting the original item's feature space to the selected feature space.

2.2.2 Collaborative Filtering

Collaborative filtering removes the item from consideration and is motivated only by the idea that a user with similar taste gives a better idea of items a user may enjoy. This space takes advantage of potentially recommending items that are outside of what the user considers their "taste", but may still be in line with something they enjoy. In The Adaptive Web, by Ben Schafer, Dan Frankowski, Jon Herlocker, and Shilad Sen, a primary technique to accomplish the task of finding neighbours to a target users uses k-Nearest-Neighbors to classify a target profile with historical and current data of other users in an effort to classify based on similar preferences [2]q. Similarity between users to classify into a neighbor or not a neighbor

has a variety of approaches. One such framework supplied by Bobadilla and colleagues use Mean square different between 2 users x and y, given similar movies they have rated together. Given these 2 user's meet a similarity threshold, the system begins predicting creating a weight for how similar a user is to another user and considers k neighbors for the predictive rating of a user onto an item. Average weighted sum and adjusted weighted aggregation are both used as predictions for ratings, and a normalization factor is applied to these aggregated sums based on the similarity of the user to the user that whose rating is being predicted. This way, the nearest neighbor's contribution to the rating in question is considered as more important to the prediction than a neighbor who is further away. The authors explore a variety of similarity measures such as Constrained Pearson Correlation, Pearson Correlation, Spearman rank Correlation, cosine similarity, and Mean Squared Differences to their neighbor classification systems and reject MSD as out performing more conventional measures of similarity.

2.2.3 Hybrid Filtering

Hybrid filtering alleviates the filtering system from focusing significantly on the user and significantly on the item, combining both to yield recommendations of items both similar to the taste of the user and of the taste of similar users. In Goyani's survey, outputs between content based methodologies and collaborative filtering based methodologies are used as input for recommendation. Deng and colleague's work in hybrid recommendation combined used popular user-item rating matrices as well as static feature's pulled from the Movielen's database to create dynamic and static features that are then hybridized to calculate how a user may be interested in a certain movie. This is then used to generate similarity matrices and neighborsets of both movies and users, respectively, and combined again with the user's item rating matrix to generate a prediction for the item. These feature matrix calculations are sparse matrices that indicate a user's interest in a certain feature, so for a user u and feature k, a user can be interested(1) or not interested(0). The features comprise of characteristics about the movie itself being things like genre, movie tags, and other relevant information users associate with movie content that are general across many movies and genres. This worked in combination with kNN to predict values for movies yet to be rated by the user in focus.

2.3 Limitations of Current Approaches

Each of these filtering methods come with common challenges that are limited by a variety of factors. *Cold Start* : For collaborative and hybrid filtering systems, effectiveness only comes when there are an adequate amount of users in the system to find neighbors and, therefore, similarity. Moreover, actually being able to rate 2 users as similar requires even greater specificity between the users that are engaged with the system. Therefore, there is trade-off between the performance of the neighboring classification and the users that are in the system as recommenders can attempt to begin their classification with little users, but the users that are classified as neighbors to a particular user may not be similar at all. This comes as a result of not having nearly enough users to find similarities between, throwing off the accuracy of item recommendation. *Scalability* In order for our systems to create valuable insight into recommending items to users, it requires we have lots of data for the model to learn from and recommend. As a result, the more users, movies, and other items we have added on to the model for consideration can become a computational burden due to the expense of processing. This is the tradeoff that is expected when attempting to tune high functioning models that can succeed across a wide variety of users. Paired with the issue of data being sparse, issues arise with the model being exposed to data that is of high quantity but varying quality.

2.4 Proposed Approach

2.4.1 Problem Definition

Our task can be condensed into two core mathematical problems: edge prediction for our graph [5], and sentiment/feature extrac-

tion for user reviews. For the graph, we will input film information features from our movie dataset to gauge similarity between movies, which we can then use for similarity measurements. For the review feature extraction, this is a matter of pulling out names, titles, genres, and plot elements that are mentioned in the review, as well as noting whether the review is positive or negative overall.

2.4.2 Data Pre-Processing/Feature Engineering

Preprocessing and feature selection are a critical part of any machine learning model, and even more so in such a complex problem as this. There are many important decisions to make in order to prioritize different selection optimizations, and it is therefore imperative to include information that we believe to be critically important to recommendations, and omit information that might cause the problems we intend to address.

Surrounding popularity bias, we will not be taking into account overall ratings or number of watches/reviews. Additionally, we will try to mitigate the effect of popularity by strategically dropping reviews in an inversely proportional manner to their popularity, such that uncommon movies will keep all of their reviews and extremely popular movies will only have a fraction of their body of reviews.

Our data was scraped from the IMDb site, and therefore had a number of formatting issues and type inconsistencies. To clean this dataset, we fixed all of these formatting issues, and subsequently assigned each director, actor, and genre to an index in order to make a binary matrix of 0s and 1s. We also added one feature for each decade, as individual years are far sparser and do not tend to give us much additional information. These features were chosen from our own experience searching for movies, and therefore are the factors that we believe should have the largest impact on what kind of movie each one is and which films it may be similar to. We initially intended to make use of the plot synopsis, but as we have achieved meaningful results without it, we have decided not to overly complicate the model with thousands of NLP features or a condensed representation thereof.

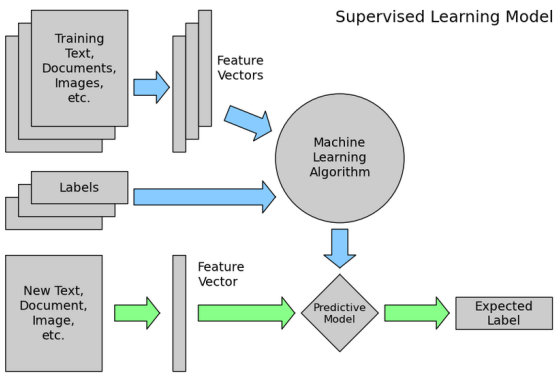
Our NLP step utilizes a pretrained classification model known as BERT. BERT is a text encoding transformer that is characterized by deep learning bidirectional text representations. The model is pretrained on general purpose texts such as wikipedia and has been pretrained to perform tasks such as sentiment analysis, emotion categorizations, speech to text translation, and many others [3].

For this project, we explore aspect-based sentiment analysis on a pretrained bert model by classifying statements into positive, negative, or neutral categories, then use nltk to parse out figures of speech that are the subject under review. As a result of the use of a pretrained BERT model, we leave our text as a whole when putting into our BERT model, but lemmatize and tag parts of speech to find our relevant subjects.

2.4.3 Workflow/pipeline/system architecture diagram

Our final deployment product is a movie recommender for online users. By processing a few reviews of movies they've watched, we will use a classification and recommendation model that can find something they could enjoy. Our model needs to go through a series of pipelines in order to take the user's input, launch the model, and display the result to the user. [1]

Our project's first phase is classification. The classification model is trained on a fixed dataset of IMDB movie reviews. The feature vectors and labels are fed into our GNN classification model. We include a workflow pipeline that takes the user input as new text, processes its feature vectors, and runs it through a predictive model to generate the expected labels. [4]



2.5 Experimental Evaluation

2.5.1 Dataset Exploration and Patterns

The Movie dataset is a dataset of IMDb's top 1000 movies. We have most of the critical information for each movie, such as name, year, director, starring actors, plot synopsis, MPAA rating, length, and genres. The dataset is inherently limited to popular movies, but we believe 1000 movies will still allow us to make meaningful recommendations. There are not many patterns to this dataset, as the range of movies that people love is so broad and encompasses most facets of the film industry. However, it should be noted that certain echo chambers exist within this dataset, such as the inclusion of many of the most famous foreign-language movies and few others, or directors that modern audiences love such as Christopher Nolan or Quentin Tarantino, or popular franchises such as The Avengers or The Lord of the Rings. The dataset includes a mix of popular films, classics, and hidden gems, and does not include the 1000 most highly rated movies, but rather uses a closed-source selection algorithm. We do know that it is largely based on ratings that have been built up over the past twenty-five years, and therefore certain films naturally have more attention and certain ones have less.

An additional dataset used in our sentiment analysis comes from user reviews of these same top 1000 movies. We scrape information such as the user that gave the review, the rating of the movie, and the proportion of users that found that movie review helpful. The review that is scraped is the most relevant piece of information as the content of review as this review is used to the strengths and weaknesses of a movie as well as what a user enjoys about movies.

2.6 Predictive Model

Our predictive model comes in three parts. The first is with sentiment analysis, where we take in user-written reviews and predict sentiment on individual sentences and the aspects that are given those reviews. We do this in an effort to investigate which parts of a movie are important to its ratings. For example, common strengths from a movie can be its aesthetics, characters, story, acting, etc. and we aim to tag each of these movies with common strengths. We then map these strengths for the movies we have reviews for. Furthermore, these tags can be characterize a reviewer as well. Seeing these sentiments appear commonly throughout a user's reviews gives a general idea of the particular film elements a user enjoys.

The second part is the forming of the graph, where the model predicts links between movies. This is done using a Graph Neural Network with two SAGEConv layers, taking in an n x m matrix of True/False features, where n is the number of movies and m is the number of features, and outputting an n x n matrix of similarities. We used a similarity function in order to train the network, which compares features between movies and outputs similarity score. In our trained model, we can clearly see that movies with similar features tend to be rated very similarly to each other. For example, the most similar film to The Godfather is The Godfather Part II, the most similar to The Dark Knight is the two other Batman movies in that series, and the Lord of the Rings movies all feature the other two as the most similar.

The third part comes in the form of using our graph to predict movies to recommend. One fault of modern recommendation systems is that all of the recommendations tend to be very similar to each other, so if one is not a good recommendation then most or all of them will not be. Therefore, we choose to use several different algorithms, and therefore can report a more diverse list of recommendations. For some, we check individual movie similarity, for others we check joint similarity, and for still others we use the NLP-extracted features as applicable in order to give us our output list.

2.6.1 Final results

In the Graph Neural Network section, we have been able to build the similarities graph. As this is an unsupervised problem (we do not have labels in advance), we cannot compare to any objective standard, but upon inspection we have been fairly happy with the results from our custom similarity-based loss function. By our subjective evaluations, the movies that

the graph finds similar do typically tend to be quite similar to each other.

Fine tuning our pretrained BERT model to our movie reviews was a simple task in adding additional information to continue training. Our BERT model currently does a good job of evaluating overall sentiment from movie reviews in totality and individual statements with an accuracy of about .88 and our loss converging on a value of about .20. However, extracting the topics that are the subject of such reviews has been a much more difficult task as parsing out exactly which nouns contribute or don't contribute to the label has been difficult. There is certainly more to be added on to tune our BERT model in neutrality detection and aspect retrieval, but an elementary classification of positive and negative has been obtained and works with confidence.

An additional lead that was pursued in our project was the modeling of common topics in an effort to find film elements the user enjoyed. We discovered that the aspect-based sentiment analysis proposed as part of the PyABSA can be fine tuned to include movie reviews well, and can lead to better synthesizing of common elements in movie reviews [6]. Leveraging named entity recognition and sentiment analysis in tandem yielded outcomes that allowed for parsed results from positive statements to be linked with a users interest, creating a more robust characterization of the user's preferences for our model. Results of this were observed on situational cases where actors, plot elements, and IMDB genre's were parsed out by fuzzy matching. Given that this matching required specific terminology, the model would find positive film elements if these elements were spelled almost correctly.

2.7 Deploying to Production

2.7.0.1 Automation

One of the team goals were to replicate how applications are developed, deployed and put into production in industry right now. We've adopted That meant following a rigorous Crisp-DM methodology of problem approach. That meant understanding the business or problem, understanding our existing data, preparing the data, modeling it, and evaluating it. The model is only ready for deployment when its evaluation metrics meet our criteria. Otherwise, the process returns to business comprehension and iterates until we are ready to deploy. Automation was a crutial aspect of our production line. We utilitzed Gitlab as our propriatary version control. However, due to academic institution restrictions, we had to transferred our workflow to github. Using verison control allowed members to work ascyncronously and avoid conflicts with codebases.

Every time a commit was sent to github, it preformed a series of actions to test the code, check for compilation errors, and pipeline the application to Google Cloud Run for hosting. Note, the automation is in place such that if there were any errors in the code, the pipeline to website deployment would immediatly be halted util one of the engineers on our team resolved the issue.

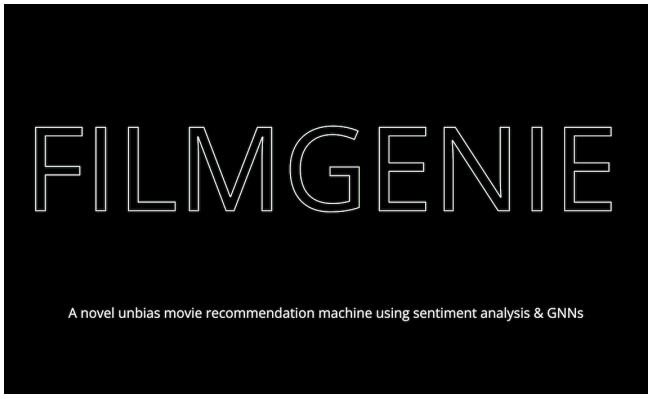
2.7.0.2 Online Model

Our model is setup to collect user data and provide recommendation to the user(with varying results based on the quality of the review). This gave us an opportunity to obtain more data to feed into our recommendation model, specifically the sentiment analysis. Currently, the data is being stored in a real-time database but we plan to redirect those reviews to our NLP model to train on online data coming from the website along with our existing IMDB reviews dataset.

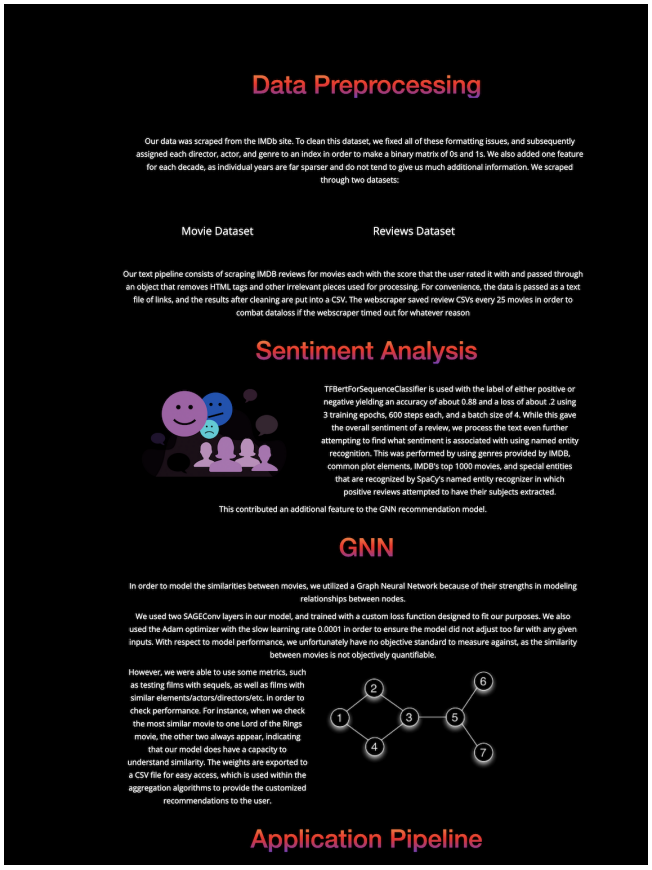
2.7.0.3 Webapp

Our dynamic web app is using a flask backend for routing and a simple HTML,CSS and JavaScript for the front-end. It also incorporates some of bootstrap's css styling for forms and containers. The UI is very user friendly. There is a bi-directional flow of information(up and down). As the user scrolls down, we summarize how our model works in different parts and the recommender is the last thing. Users have the ability to get personalized movie recommendation when they insert at least 2 movies reviews. Their inputs are sent via a JSON file to the recommender, which returns a list of movies. The movies are then displayed along with their respective

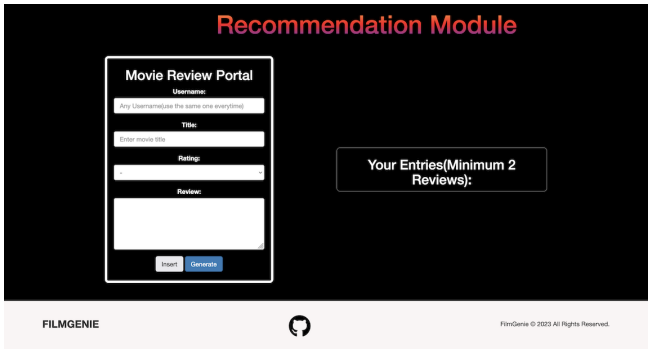
year, genre, IMDb rating, and a movie poster that links to the IMDb site. It is hosted on Google Cloud. The reality is that our website's front-end and back-end are under the same docker build. Thus, when we build and run our website on Google Cloud, it has a 2 minute cold start if the website has not been used recently(in the last 5 minutes). This is largely due to tensorflow loading its packages when the website is initially warmed up. Below are the preliminary website and database review structure respectively:



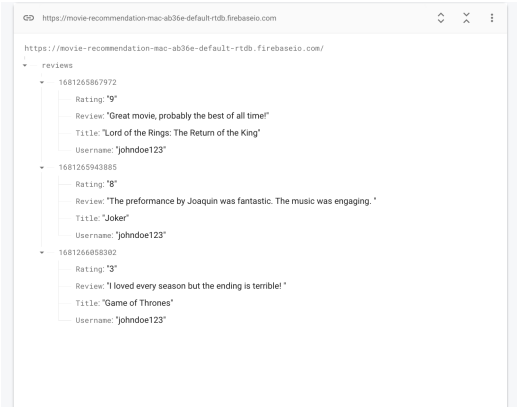
(a) Landing Page



(b) Content



(c) Review Module



2.8 Future Work

For future recommendations, wa can work on expanding the dataset, as we currently only have a thousand movies; and far less than that considering that serious movie fans will likely have seen a large percentage of these already. Additionally, we can improve our recommendation algorithms and similarity functions to give more importance to successful characteristics and less to the harmful or irrelevant ones. In particular, we could utilize the review-extracted film features to check which films are more about their plots, and which genres people generally associate it with rather than the accepting the IMDb-assigned ones as ground truth. For review processing, we would like to improve our model's ability to extract important features, and find a way to assign individual sentiment to them. Many times reviews will not be entirely positive or negative, and it might greatly improve our model's expressive power if we could capture the nuances in sentiment. As far as displaying results, we would like to include a drop-down list and/or have string matching in order to reduce the need for perfect spelling in movie titles. The website cold starts would be on the list of future work. The possible techniques to solve the issue are listed above. An NLP challenge faced by the team involved the quality of training data the model was tuned on. While results were promising with the accuracy and loss specified above, it is important to recognize that there was significant ambiguities and inconsistencies when reviewers gave scores to movies and what the score reflected. The BERT model was trained on labels between a 1-10 scale which took anything above a 5 as positive and 5 and below as negative. Yet, we would often observe some reviewers giving what would be considered to be a harsh review but still giving a positive score. While this is a small piece of the datasets reviews, it is still an worthwhile future change to pivot into.

Regarding our cold start issue, one possible solution is to isolate the front-end and back-end. Our users will spend between 1 to 4 minutes to insert reviews in our website. Thus, displaying the front-end would have no consequences on how our online model is run.

Another possible solution is to narrow down the modules causing this long load-time, and removing unnecessary local imports in our application. This is not a guaranteed fix but it could optimize not only cold starts but also the long build times.

2.9 Conclusion

As mentioned in our initial paper, reddit is currently the most popular place to get movie recommendations. Because it involves people making recommendations, this is highly biased. We want to deploy a smart automated machine learning model that can classify a user's review of a given movie and make a real-time recommendation. Rather than simply looking at ratings or genres, our goal is to create a bias-free system based on the user's preferences.

We want to base our recommendations on what is said about a specific movie because it provides the most insightful information. We will create a graph with a GNN using our web-scraped IMDb data, including summaries and plot synopses, and then use the graph and user reviews to create our recommendation model. Content-based filtering, collaborative filtering, and hybrid filtering have all been investigated. We used the first stages of our TF/IDF NLP preprocessing scripts after gathering a collection of movies to web-scrape.

Sentiment analysis upon use of a pre-trained BERT model supplied by HuggingFace has yielded great success upon the additional tuning including the movie reviews we scrape from IMDb. Despite its binary classification, it is still able to classify individual statements with high levels of accuracy of .90 and a converging loss of .20. We hope to transition build atop this model with multi-class aspect sentiment analysis to find specific film elements to connect to movies and users as to create a more personalized character for the recommendation system to consider. We hope to build upon the filtering systems as mentioned in the section 1.2 by utilizing the topics that have positive associations in user reviews.

This model will eventually help us recommend movies to users. The next step is to expand our GNN, improve rating mappings, improve sentiment analysis and experiment with various other techniques.

2.10 Contributions

- Grant Doan - Data cleaning, Sentiment Analysis, Review Feature Extraction
- Prayash Joshi- Docker and Cloud Integration, Dynamic Webapp, Full-Stack
- Reagan Orth - Data cleaning, GNN, Aggregated Recommendation Algorithm
- Ved Patel - Web Scraping, Project Poster

References

[1] Satvik Garg, Pradyumn Pundir, Geetanjali Rathee, P.K. Gupta, Somya Garg, and Saransh Ahlawat. On continuous integration / continuous delivery for automated deployment of machine learning models using mlops. 2021.

[2] Woon-hae Jeong, Se-jun Kim, Doo-soon Park, and Jin Kwak. Performance improvement of a movie recommendation system based on personal propensity and secure collaborative filtering. *Journal of Information Processing Systems*, 9(1):157–172, 2013.

[3] M. V. Koroteev. Bert: A review of applications in natural language processing and understanding, 2021.

[4] Dan Ofer. Machine learning for protein function. 03 2016.

[5] Shiwen Wu, Fei Sun, Wentao Zhang, Xu Xie, and Bin Cui. Graph neural networks in recommender systems: A survey, 2020.

[6] Heng Yang and Ke Li. A modularized framework for reproducible aspect-based sentiment analysis. *CoRR*, abs/2208.01368, 2022.

2