Initial Paper Draft

Grant Doan, Prayash Joshi, Reagan Orth, Ved Patel

March 22, 2023

1 Introduction

1.1 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 intend to create a recommendation system that takes in commentary from users who have already seen the movies, and use that information to better understand if it might apply to the user in question. In this manner, we believe we will be 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.

1.2 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.

1.3 Challenges and Solutions

Our primary focus is creating a recommendation system that does not contain these two common and harmful biases. The premise of our solution is that the things that are said about a movie can provide rich insight into what might appeal to viewers and what turns them away. We therefore

intend to use IMDB's film summary and featured plot synopsis in conjunction with other critical film information in order to create an undirected, unweighted graph, with nodes representing movies and edges representing similarities. After this, we will use IMDB's user reviews and ratings in order to develop a rating prediction model using our graph with its basis on the individual user. With a large enough dataset, we should learn useful similarities and preferences, such that a user who liked certain movies will be recommended movies with similar casts or genres or plot elements, or other factors that we may not think to mention. Even more important for our model than recommending perfect films is not recommending dissimilar films.

2 Related Work [Maybe Grant can do this, since you did most of the lit review?]

2.1 Literature Survey

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.

2.1.1 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 teh 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. An important issue tackled by their approach is data sparsity, which is a common limiting factor for evaluating text based features (reviews, plot synopsis, tags, etc.) as the feature vectors that are considered are large in dimension. Using smaller vectors that do not include textual information is an approach, but often suffers from being blanket generalizations or categorizations of the information if we use the information just supplied by what can be found on a search. For example, saying to action movies are similar because of their genre is a true statement, but user preferences tend to be more specialized than simply liking only action movies, therefore a system using only this consideration may not have the best performance on recommending something specifically to a user's taste.

2.1.2 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. An important part of filtering with this method is

creating relationships between users that link them based on similarity. A survey by Goyani calls these "neighbours" but other terms such as trust have been used to describe the relationship. Ratings given by the user and other users form a user-item matrix and is a large catalog of items rated by the user. In The Adaptive Web, by Ben Schafer, Dan Frankowski, Jon Herlocker, 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. 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. K hear can be any number of neighbors so long as efficiency of computing the values for similarities is inline with computation limitations. For enterprise systems like Netflix and Hulu users exist in the millions, therefore computation must be scaled adequately to provide for efficient prediction and recommendations. This will be further explained in limitations.

2.1.3 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.

2.2 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. *Data Sparsity* : Following Cold start, data sparsity comes commonly with the matricization of items into feature matrices to use for similarity. Consider matrices composed of a user and their ratings. Collaborative filtering dictates that users can have connections drawn between them by rating similar items, but it is common that a user will have ratings for movies or items that do not exist for the focused user. As we scale, we have matrices of user's ratings for movies that are empty but are still taking up space on the system. This same issue is paralleled in content based filtering where similarity between tags is often compared using NLP methods. As a

result, comparisons between corpuses of text involve creating matrices where words in one synopsis, for example, do not show up for another synopsis. We have this issue of the data that is being fed into our model being mostly empty and taking up space on what could otherwise be used for data with value. While this is not an issue on small systems, a final issue of scalability presents itself. *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.

3 Proposed Approach

3.1 Problem Definition

Our task can be condensed into two core mathematical problems: edge prediction for our graph[6], and rating prediction for individual users. For the graph, we will input our features and obtain the predicted edges, which we can then use for similarity measurements: the shortest path from any movie to any other movie is the similarity measurement. We will likely not need extensive graph traversal calculations, as we will generally be using films that are very similar to other films on our graph. For the user rating prediction, we will be using the graph and inputting the user's preprocessed reviews and film ratings and obtain an approximate rating prediction out of ten.

3.2 Data Preprocessing and 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.

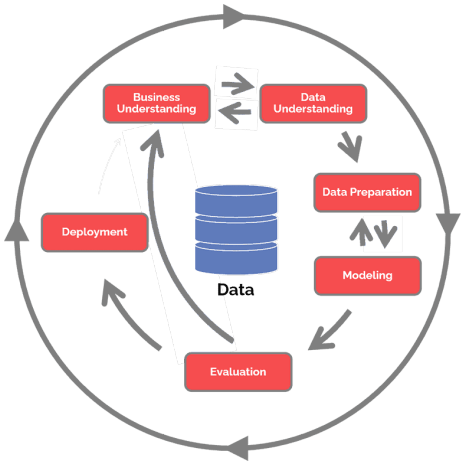
In order to build our graph, we will use the plot description, plot synopsis, year released, director name, names of starring actors, genres, and possibly length and MPAA rating, as these are the features that we believe should have the largest impact on what kind of movie each one is, and which film it may be similar to. For the recommendation aspect, we will be building user profiles from their ratings and any reviews they have on the site.

To preprocess the text we gather, we will use TF/IDF NLP preprocessing [1], with a modification to emphasize words with capital letters that are not at the start of sentences. For example, a user writing "BEST MOVIE EVER" is much more emphatic than its lowercase counterpart, which may at times give us better insight into the user's opinions. Within the capitalization importance correction, we intend to count actor, director, and film names as a single word and give them heavy emphasis, as these contributors are often a large reason for a user to watch (or stay away from) a film. Preprocessing for all other features will largely depend on the web-scraped formatting, such as removing newlines and replacing or removing ill-formatted characters.

3.3 Workflow, Pipeline and System Architecture

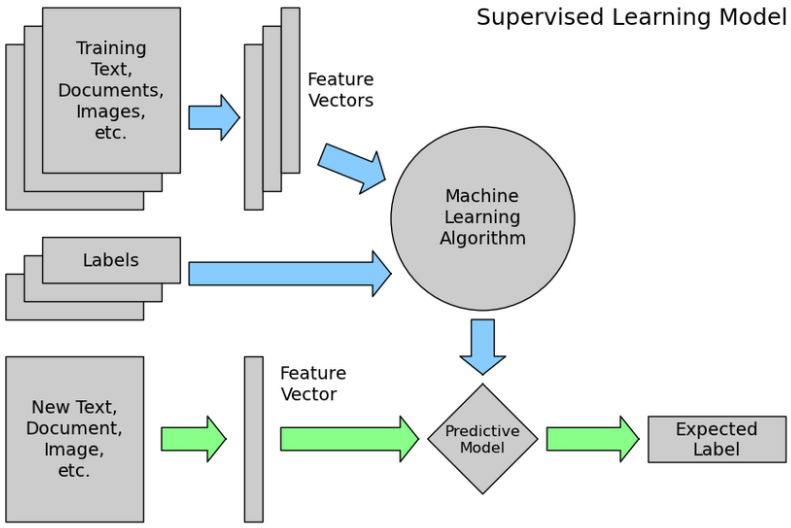
We discovered more flexibility and adaptability in going back and forth with datasets and models in the CRISP-DM approach after a thorough review of both SEMMA and CRISP-DM methodologies. As a result, we are employing it as our primary project management methodology. To summarize, CRISP-DM is an iterative process that involves 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. Refer to the diagram below: .[4]

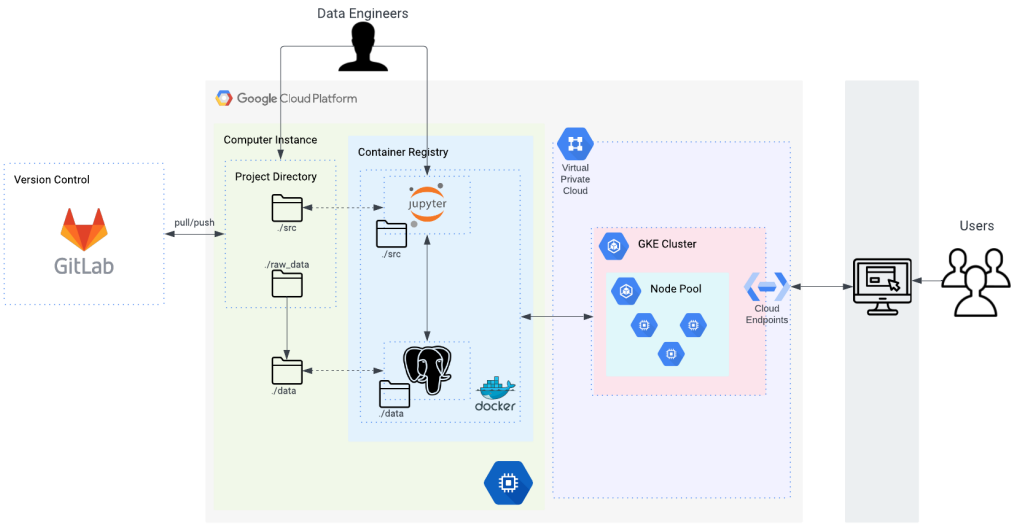


Our final deployment product is a movie recommender for online users. By processing a few reviews of movies they've watched, we will use a classification and recommendation model that can find something they could enjoy. Our model needs to go through a series of pipelines in order to take the user's input, launch the model, and display the result to the user.[2]

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..[5]



Below is a sample of a possible system architecture on Google Cloud Platform using its Compute Engines: [3]



3.4 Team Meetings And Collaborations

Our team has chosen to use a Discord channel to share interesting and useful literature, send progress updates, and discuss meeting notes, among other things. We meet briefly in person during class to discuss the necessary steps to stay on track. Otherwise, the majority of our meetings are asynchronous and online. Our team meets through a virtual discord call every other week. Detailed notes of the things discussed during the meetings are recorded and uploaded to a shared repository. Though, our most popular form of meeting up is through text on Discord. Thus, sharing progress updates on a regular basis via Discord messages has been a huge success for us.

We have a Gitlab repo for collaboration, which has been shared with all admin users. For version control, we use Gitlab. We use VS Code's liveshare feature to collaborate virtually on a Python notebook file. We can easily see changes made to the document in real time. This method is being used because everyone is familiar with the user interface. As the project progresses, we will use notebooks with GPUs hosted on a cloud server to process our results more quickly.

4 Experimental Evaluation

4.1 Dataset Exploration

Our dataset consists of two separate datasets. The first is simply the movie description, including the name, year, director, plot summary, and more. The second one is the reviews that are associated with each movie. Although separate, they will be used in unison and fed into our graph neural network. The First dataset is relatively straightforward and does not have much preprocessing required. However, as previously mentioned, the first dataset will have undergone sentiment analysis to extract words that hold more significance than others. For instance, we decided to split this up further into a positive and negative set. The positive includes reviews from 160 movies with more words with a positive connotation. This was derived from the aforementioned TF/IDF NLP preprocessing.

4.2 Predictive Model

Once we have created our different datasets, we can feed them into our Graph Neural Network to create a predictive model. We will use a large, undirected, and unweighted graph where the nodes will be the movie titles. Now the main reason we chose to use a Graph Neural Network was its ability to model relationships. We will take advantage of this by using the graph's edges to represent similarities between movies. Nodes that have small edges connecting them indicate that the movies are similar. Initially, our model will check a user's preferences by considering their ratings and reviews. Next, the model will check which movies are the most similar to the ones the user already prefers. This will

be relatively easy as the graph is designed to handle this task. After that is completed, all of the descriptive words used in the user reviews, such as the preference for specific actors or directors, will combine movie preferences.

4.3 Initial Result Analysis

We have yet to enter user preferences to test our graph neural network. However, we plan to complete this step as soon as the graph is constructed. We have collected the datasets for our project, including those for specific connotations.

5 Future Work

Since the data collection phase of our project is complete, all we have to do next is to construct the graph neural network. The first step will be relatively straightforward as we will create an adjacency matrix using our dataset of IMDB's summaries and plot synopsis. The rows and columns of the matrix will be a particular movie, and the matrix values will be the "similarity score." Next, we can use a variety of python libraries to create our graph from the adjacency matrix. Once the graph is created, we can select our GNN model and feed in user reviews to create personalized recommendations.

6 Conclusion

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.

We are committed to using the CRISP-DM project methodology as a team because its agile characteristics are ideal for our team workflow. Once our machine learning algorithm has been trained, we can pipeline new data from online users and obtain the expected label for our classification model. Our team is still thinking about a cloud server approach to the workflow, but our top priority right now is to begin validating our classification model.

This model will eventually help us recommend movies to users. The next step is to validate our classification model, test weighted filters for specific words, and experiment with various other techniques until we are satisfied. Meanwhile, some team members can start working on a website that takes user movie title, review, and rating(out of 10) and post them to an online database for us to pipeline and process in our model. These tasks will help us get closer to our goal of creating a recommendation model.

References

[1] José Camacho-Collados and Mohammad Taher Pilehvar. On the role of text preprocessing in neural network architectures: An evaluation study on text categorization and sentiment analysis. *CoRR*, abs/1707.01780, 2017.

[2] 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.

[3] Google. Mlops: Continuous delivery and automation pipelines in machine learning nbsp;—nbsp; cloud architecture center nbsp;—nbsp; google cloud, Feb 2023.

[4] Ioannis Karamitsos, Saeed Albarhami, and Charalampos Apostolopoulos. Applying devops practices of continuous automation for machine learning. volume 11, page 363, 2020.

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

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