Netflix Recommendation Engines

Won Joon, Choi
choix471@umn.edu
4766633

Tim Schaefer
scha0164@umn.edu
1682728

Aparajita Kar
kar00004@umn.edu
5539822

Sumukh Nitundila
nitun001@umn.edu
5663068

***Abstract*—Using the "Netflix Data Latest 2021" dataset, structure and implement a DBMS system to support direct queries from multiple datasets and the data analytics engine and generate recommendations based on previous view shows, and customer demographics. We expect to learn how to extract, load, transform (ELT) and manage the process and structure of the system. A recommendation engine is a type of data filtering tool using machine learning algorithms to recommend the most relevant items to a particular user or customer. It operates on the principle of finding patterns in consumer behavior data. In our case, the data is based on Netflix movies and shows. The idea is to recommend to users the movies that they would like to watch depending upon certain features like genres, language, availability of shows, IMDB Score, rating, etc. Our team has decided to apply four different approaches to build the movie recommender system.**

***Index Terms*—Netflix, Data Mining, Recommendation Engine, Content Filtering, Collaboration Filtering, Data Reduction, Data Materialization**

I. Introduction

Recommendation engines have become very popular in this era. Companies such as Amazon, Youtube, Netflix, Doordash, Walmart and Instacart uses recommendation engines to provide suggestions to their users regarding the products that they use. The aim is to attract customers to improve their visibility and user interactions like in Netflix, Amazon Prime Video and Hulu. Walmart, Amazon, Instacart uses it to recommend items to customers which they would like to buy. This project aims to understand various recommendation algorithms to suggest users the movies they would like to watch.

The team has implemented Recommendation Engines that take the Netflix database [1] (9425 entries X 29 attributes) and generates recommendations based on user interests, and user historical views. The data provide a multitude of fields (29) however, a majority of those fields did not have bearing on the user selection. The focus was on (Country Availability, Language, Genre, Tags, IMDb Score, Hidden Gem Score, Directors, ratings, etc.) that were identified as user influences. This data was cleaned and conditioned to enable effective inclusion into a recommendation engine. The team implemented Content Based Filtering, K-Nearest Neighbor using Reduced Dimensionality, and Decision Tree to address the different user modes of interaction with the recommendation engine.

II. Data Description

A. Data Exploration

The dataset used in this project was the Netflix 2021 Dataset[1] obtained from Kaggle at the below reference. The dataset has 29 columns and 9425 records. It has information about the movies including language, genre,actors, directors, tags, Maturity Rating, country availability of the movie,various ratings provide by IMDb, Metacritic, Rotten Tomatoes etc. It also includes information about the awards received, trailers, summary of the movie and the release data of the movie.

A strong correlation is present between various ratings provided by different organizations like IMDb, Rotten Tomatoes, Metacritic Score and Hidden Gem Scores. Below is a comparison showing the correlation between these variables.

	Hidden Gem Score	IMDb Score	Rotten Tomatoes Score	Metacritic Score
Hidden Gem Score	1.000000	0.395314	0.581481	0.602252
IMDb Score	0.395314	1.000000	0.654538	0.695574
Rotten Tomatoes Score	0.581481	0.654538	1.000000	0.900020
Metacritic Score	0.602252	0.695574	0.900020	1.000000

Fig. 1. Correlation Plot

Owing to the correlation between the variables, the team decided to use only IMDb rating as a feature representing movie ratings. Data discovery was conducted on the data to identify the key attributes that would influence the user selection and the development of a user model. Some of the prominent attributes include genre, language, tags, Maturity Rating and country availability. The data was then cleaned to address any inconsistencies present. Examples of this include the multi-dimensional dictionary mapping of the key attributes, elimination of punctuation, removal of stop words, reduction of some categorical attributes to numerical attributes, and removal of extraneous fields.

III. Methods

A. Content Based Recommendation Algorithm

Content Based Recommendation Algorithm uses the similarity of features based on user defined attributes. The most overlapping features with the user input features are recommended. This method compares the interests of a user such as genre of the movie, language, summary of the movie and the various tags with the attributes of other available movies in the dataset. [2] The algorithm lets in the user decide a movie, the language, genre of the movies, country the user wants to watch in, IMDb score and the release year of the movies. The dataset is then filtered based on these given inputs by the user. The idea is to provide top 10 recommendations to the user by computing a similarity matrix between these given

user interests and available features in the dataset. Hence, the user provides a sample movie title from the list of all available movies. The similarity is then computed using the given title by the user. The measure of similarity is generally computed using cosine similarity or Pearson similarity. For the purpose of this project, the team decided to use the cosine similarity. Cosine similarity between two vectors x and y is defined by:

$$sim(x, y) = \frac{x \cdot y}{||x|| \times ||y||}$$

The filtered data is then processed by removing the stop words, the words which are very frequent in the English language. The stop word list used is the stop word list for English text processing tool related to python. A *bag of words* is created using all the words from the features: Country, Genre, Tags, Summary, languages, view rating and actors. A TF-Idf matrix is created for all movies. It is then used to compute the cosine similarity score. Once it is computed the system recommends the top 10 most similar movies.

B. Modified Content Based Recommendation Algorithm

Content based recommendation system as mentioned in the previous section can be used to get better results with slightly different preprocessing techniques. First modification we can make is *impute* some of the missing data with data from other columns or a default value. If data for tags is missing, we can try imputing that column with data from genre. If data for movie summary is missing, we can impute it with movie title. The movie language is set to *English* by default. English has different forms of the same word. So a stemming technique like the *porter stemmer*[3] to remove common morphological and inflectional endings from words. Once it is stemmed, all relevant columns can be merged. To figure out weather columns merged are relevant, we can use Tf-Idf on this merged column, and apply a dimensionality reduction technique like *t-distributed stochastic neighbor embedding* and try to create clusters[4] from this data as shown in Figure 2.

Singular Value Decomposition (SVD) is involves factorizing of a matrix of size $m \times n$ to a product of a unitary matrix of size $m \times m$, a diagonal matrix of size $m \times n$ and another unitary matrix of size $n \times n$:

$$M = U\Sigma V^*$$

We used SVD on this dataset to see if we can decompose this data into more meaningful attributes and see if that gives any better results. But when plotted to a graph and clustered Figure 3, it gave a lot more overlap among attributes. This deterioration could also be observed when we used this to generate recommendations. So this method was not used in our recommendation. A good selection of columns will have clusters with respectable separation and less overlap. So different combination of columns were tried before picking the columns *Title, Summary, Director, Writer, Genre and Tags*.

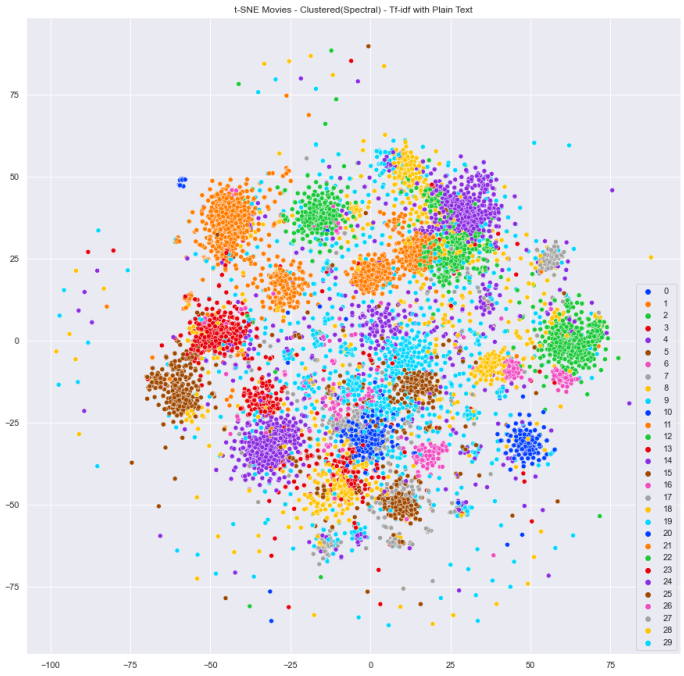

Fig. 2. Spectral clustering with 30 clusters

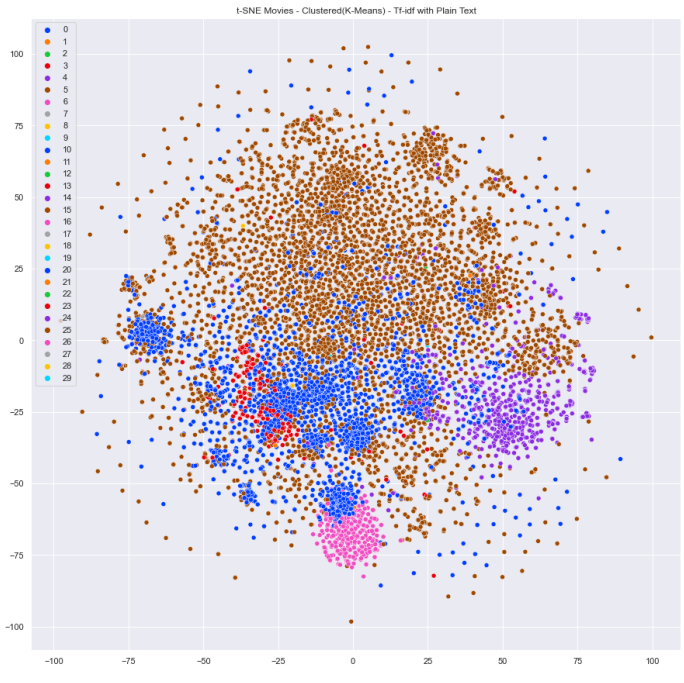

Fig. 3. Spectral clustering of attributes after SVD

Pairwise cosine similarity is calculated and the most similar 10 data points are recommended.

C. K-Nearest Neighbor Using Reduced Dimensionality

Our second method of approaching recommender system for Netflix data was using K-Nearest Neighbors(KNN) using reduced dimension. First step was to choose which variables to use. Our team agreed on using title, genre, tags, languages,

country availability, view Rating, IMDb Score, and release date as our features of interest. However, genre, languages, country availability, and IMDb Score were used for KNN approach. We wanted to explore what recommender system would turn out when we reduced the dimension of our data. This led us to blindly recommending based on the reduced dimension. Before using dimensionality reduction technique, we first one-hot encoded the categorical variables which are genre, languages, country availability. One-hot encoding is commonly used in machine learning to transform the data so that features that are not in order do not get trained with order. Then, we ended up with 361 columns. With the transformed data, it is common to use dimensionality reduction techniques for efficiency and avoiding multicollinearity issues. We wanted to observe how dimensionality reduction technique works in the recommender systems. An auto encoder was used to reduce the dimension using only the encoded layer. Auto encoder is a special type of neural network that tries to copy its input to its output. [5] As we can see from Figure 4, auto encoder first tries to capture important features from the input. Based on those extracted features, it reconstructs the data so that it is similar to its input. In our case, we are using encoded data which is in the latent space to capture the important features from our data. There were five different layers used as we can see from Figure 5.

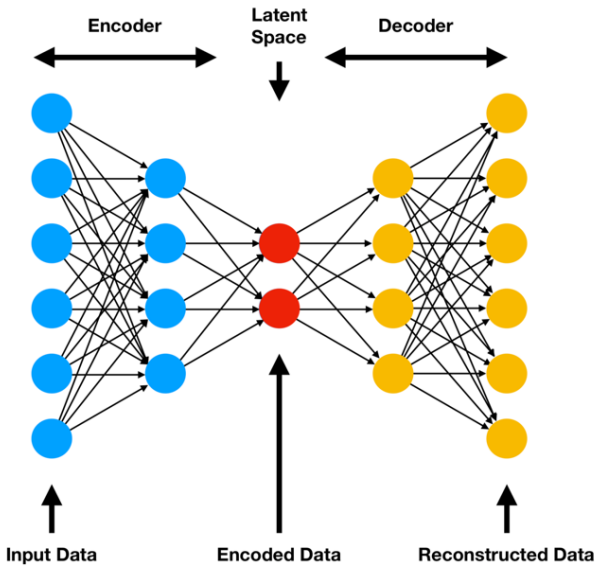

Fig. 4. Example of a simple auto encoder

After training for 20 epochs, encoded data was used for KNN to find the closest distance between data points using Minkowski distance metric.

D. Decision Tree Recommendation Engine

A decision tree recommendation engine was implemented [6]. An eight layer / branched tree was implemented to allow for decisions based on the user preferences. The decision

```
class DimensionReduction(Model):
    def __init__(self):
        super(DimensionReduction, self).__init__()
        self.encoder = tf.keras.Sequential([
            layers.Dense(64, activation="relu"),
            layers.Dense(32, activation="relu"),
        ])

        self.decoder = tf.keras.Sequential([
            layers.Dense(32, activation="relu"),
            layers.Dense(64, activation="relu"),
            layers.Dense(361, activation="sigmoid")])
    def call(self, x):
        encoded = self.encoder(x)
        decoded = self.decoder(encoded)
        return decoded
```

Fig. 5. Code snippet of the architecture of autoencoder used

points were made based on criteria extracted from either a user viewing history, or from user preference selections, and a ”User Model” is generated. The only assumption made was the “Country Availability”, since this can be determined from the user IP address. The language was determined from the most common occurring language. Sorting of the user genre, and tags was done to identify the most common and these are used to influence the recommendation. Numeric values were assigned to the ratings (“Ratings” ex G, PG, R) to allow for accommodation of mixed rating and determine an average user rating. A tolerance window was assigned to the rating to ensure any recommendations would align with historic performance, this also allows for a level of recommendation control (i.e. “NC-17” movies would not be recommended to a “G” or “PG” user). The scores (“IMDb Score” and “Hidden Gem Score”) and release year were averaged. All of the numerical values were assigned tolerances of either a window (+/- X points or years). When conducting the recommendation, at each decision layer, a check is conducted to ensure that shows exist, if they do not, the recommendation engine goes up one decision layer (auto prunes the tree). For example, if the “Available Country” is Brazil, “Languages” is Cantonese, and “Genre” is Action, there will be no selections beyond the third decision layer, in which case the engine goes up one layers (layer 2 – 6 shows available), and makes a random selection of the shows available in that layer. Figure 6 shows architecture of the Decision Tree Recommendation Engine.

IV. Results and Analysis

A. Content Based Recommendation Algorithm

Content Based Recommendation Algorithm requires an input from the user. As it is based on the movie features, the algorithm takes in key values and compares it with the key values of the available movies in the dataset. Some of the results are shown below. Figure 7 shows top 10 recommended movies for the following input from the user. Title→The Dark Knight, Genre→Action, Country Availability →United States,

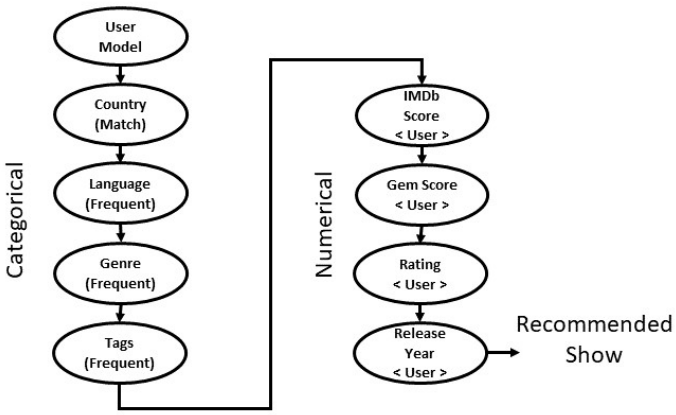

Fig. 6. Decision Tree

IMDb Score→5 and above, Language →English, Release Year →2005 and beyond. These movies are likely to be watched by the user.

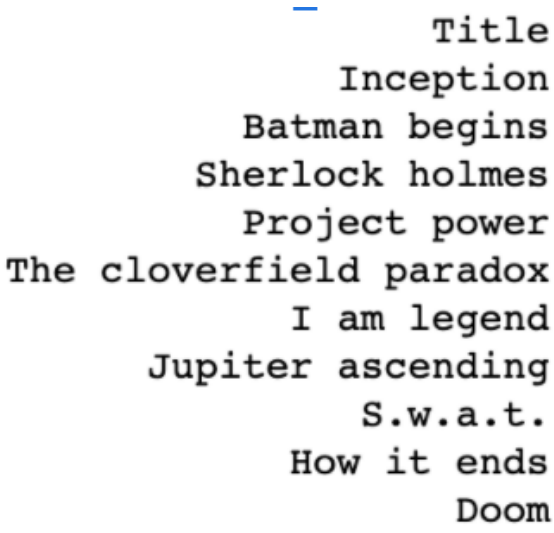

Fig. 7. Top 10 recommended movies

The content based algorithm is a simple algorithm. It is less cumbersome as it takes fewer inputs from a user. It makes it relatively easy for a small sample space to build a model and provide recommendations. It is user independent as it only takes into consideration the input from a user. It is reliable as it works well with a dataset that has a fewer user profiles.

It is a user input based unlike collaborative filtering which takes into consideration the items liked by another user. If new movies are present in a dataset, most likely it wouldn't be watched by many user. Hence, an important aspect of this algorithm is that these new movies may also be recommended to a user since it is feature based and doesn't depend on the ratings provided by other users. If the movie attributes are similar to the new movie, it is likely to be recommended.

However, some of the drawbacks of this algorithm are that it is overly generalized. Hence, users may expect surprising results. It requires a great deal of domain knowledge because feature representation of the items are hand-engineered. Hence, the model can perform well based on the characteristics that were hand engineered.

In context to Netflix recommending movies, it does both content based and collaborative filtering to provide accurate suggestions to its customers.

B. KNN

Unlike other recommender systems that we built, this method blindly recommends the movies based on the reduced dimension of our data. Analyzing the recommended movies from Figure 8, it is not the ideal way to approach the recommendation systems. This is because content based systems require the knowledge of the features. If we use dimension reduction, the model that we create does not know about the features of each movie thus resulting in blind recommendation of each movie. The approach was taken to explore different ways of creating recommender systems.

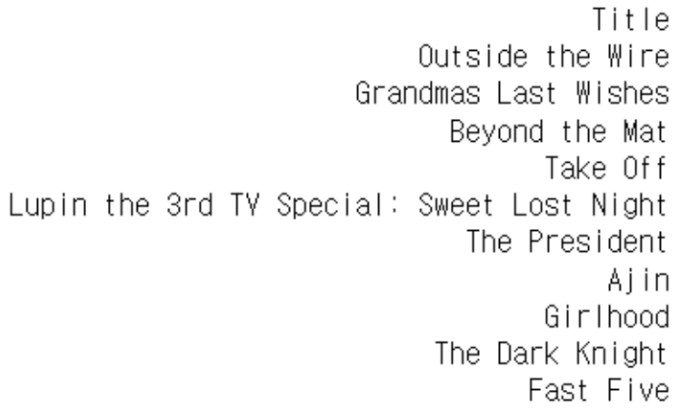

Fig. 8. Top 10 recommended movies for the movie from KNN - The Dark Knight

C. Modified Content Based Recommendation Algorithm

This modified content based recommendation system only takes in one input, the movie title. It recommends 10 movies based on only that one attribute. Although there can an optional filtering based on the language, it is not a mandatory field since people can be interested in foreign language movies too.

This gives fairly good result as we can see in Figure 9

D. Decision Tree Recommendation Engine

Figure 13 shows the reduction in the movie selection as the "User Model" or profile is moved through the decision layers. The final selection is randomly selected from the last "non-zero" show decision layer. Depending on the output objective function, one to N recommendations can be generated. The decision tree is lacking relationship between the genre and

Title
The Dark Knight Rises
Batman Begins
The Dark Knight
Inception
Batman Returns
Batman: Mask of the Phantasm
Gotham
Batman Ninja
Batman v Superman: Dawn of Justice
Batman Forever

Fig. 9. Top 10 recommended movies for the movie - The Dark Knight

tags, which tends to ”over filter” the selection (there tends to be significant overlap between genre and tags - which should be further investigated). Shows with ”empty” ratings are excluded from selection as part of a parental control mechanism. Ratings need to be included to ensure that the recommendation is suitable for the ”user”. Data mining techniques can be applied to the generation of a rating inference to allow for inclusion of unrated shows in the decision tree. The output for the Decision Tree Recommendation Engine using ”The Dark Knight” as a single input is shown in Figure 10. The decision tree implementation also provided a platform for generating recommendations on historical views (ex. the last 10 shows or 20 shows viewed). This provides a mechanism for focusing the recommendations based on the user model. The user model is generated using both categorical and numeric representations, and provides bounding of these fields to ensure that the recommendations are closely match to the user model. An example input list shown in Figure 11 is used to generate a user model, and generate recommendations. Numeric parameters are averaged and assigned a tolerance, and categorical features are selected based on the most frequent. An example output is show in Figure 12

```
Recommended Movies by Title
Ghost Rider
Alexander: Theatrical Cut
Sherlock Holmes
Doom
The Dark Knight
Batman Begins
I Am Legend
Defiance
Paul Blart: Mall Cop
S.W.A.T.
```

Fig. 10. Decision Tree - Top 10 Recommended Movies

```
testMovie[0] = 8516 - 2 # The Dark Night
testMovie[1] = 101 - 2 #The Replacement Killers
testMovie[2] = 236 - 2 # Lupin
testMovie[3] = 270 - 2 # Running Man
testMovie[4] = 324 - 2 # Rogue
testMovie[5] = 515 - 2 # Ava
testMovie[6] = 696 - 2 # The Outpost
testMovie[7] = 1454 - 2 # The Other
testMovie[8] = 1890 - 2 # Eye For An Eye
testMovie[9] = 2014 - 2 # Agent
```

Fig. 11. Decision Tree - Historical - Input Movies

```
Recommended Movies by Title
Cirque du Freak: The Vampires Assistant
Ghost Rider
The Dark Knight
I Am Legend
Defiance
S.W.A.T.
```

Fig. 12. Decision Tree - Historical - Recommendations

V. Comparison of the results from four methods

In this section, we will discuss the outcome of the recommended movies of The Dark Knight. All our approaches were content based since we used features of each movie for creating four different models. The intersections / overlaps of each of the techniques is shown in Figure 14. All three models except KNN approach had overlaps from recommended movies. Batman Begins had the highest overlap while other recommended movies had two models overlapping. Also, the

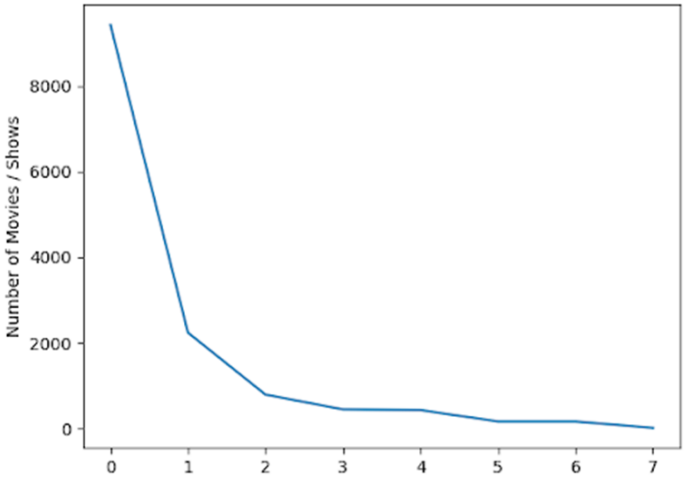

Fig. 13. Decision Tree - Reduction Transversing Decision Layers

outcome of the recommended movies aligned with The Dark Knight since these movies share similar characteristics. The implementation of four different techniques allowed for evaluation of the performance from a comparison perspective. One of the hardest components is how to determine the accuracy / precision of the recommendation. In recommender systems, user based ratings can be evaluated by comparing it with the future ratings provided by the user. However, we do not have any user ratings available for evaluation.

The use of combinatorial techniques (bagging / boosting / federated learning) would reduce the level of error and increase both the accuracy and precision with increases in processing cost.

Title	Number Overlaps
Batman Begins	3
Doom	2
Project Power	2
S.W.A.T.	2
I am Legend	2
Sherlock Holmes	2

Fig. 14. Show Overlaps across the Techniques

VI. Conclusion

The team implemented four different Recommendation Engines to the Netflix Dataset that we have obtained from Kaggle. The primary aim of this project is to understand different algorithms that can accurately provide movie suggestions to users. The team decided to implement Content Based Recommendation Engine, Modified Content Based Recommendation, KNN Using Reduced Dimensions and Decision Tree Approach. All of these models have their own limitations and advantages. The unsupervised nature of the algorithms makes it difficult to evaluate the model performance. There is no way to understand (within the scope of this project) whether the results truly reflect the movies that would be liked by the particular user. Unlike supervised algorithms, wherein we can improve model performance by tuning parameters, the only viable suggestion we can make is to work on some feature engineering and try capturing important attributes based on the inputs from the user.

The content based algorithm gave a fairly good result. Since, a majority of the similarity of words come from the summary of the movie, the algorithm depends primarily on the content of the summary. For movies, which do not have a good summary, the algorithm may not provide good results. However, it is an easy approach that is based on a single user input. It doesn't have much computations and hence very fast when implemented.

The KNN approach using reduced dimension did not have an ideal recommendation as previously mentioned. Features have to be used without reduced dimension to recommend movies since reduced dimension is commonly used for collinearity issues. Collinearity issues arise when there are redundant features for the prediction of the outcome. However, since the data that we have needs to be preserved the way it is our KNN approach had poor result. As it is something that we wanted to explore, it was good observation that we identified and confirmed the feasibility of this approach.

The modified content based algorithm performed relatively well since we picked an asymmetric distance measure like cosine to actually compute the movies most similar to the ones that is being viewed. Porter Stemming used to convert the words to base form, made the similarity measure much more reliable. But for some of the movies, the genre or tags are a lot more fuzzy. There is no grading scale of how much of the movie is dedicated to one genre when compared to the other. This results in some recommendations where the movie recommended is not as close. Another drawback for this approach is the fact that sometimes movie title and summary might be misleading. For example, *Good Will Hunting* is a movie about a janitor named Will Hunting. But the algorithm takes good, will and hunting as different tokens and starts recommending movies about hunting. So it cannot meaningfully differentiate between the noun Hunting and the verb hunt. Even with these limitations on the fringes, it serves as a great entry starting point when user data is not available to improve the engine based on feedback.

The decision tree approach provided a logical / efficient method to generate recommendations. It provided a foundation for historical recommendations (XX number of previous shows ex. 10 previous:10 recommendations), similarity recommenda-

tion (ex 1 previous:10 recommendations) and filtering based on the different features using a simple GUI. The implementation of a intelligent agent (IA) would allow for modifications of branches to improve both performance and increase relevance of the recommendation. The IA would look at the inputs, outputs, and appropriately adjust the feature filters.

VII. Future Work

The dataset had a lot of information regarding other movie attributes such as directors, actors, writers, runtime which may provide valuable information towards improving model performance. Secondly, there may be a lot of interactions present between these variables. The development of relationships between the Genre and Tags, Actors and Directors has the potential of further refinement of the recommendations. There also needs to be a better way to incorporate some context when tokenizing the different attributes. The application of a boosting / bagging or federated learning can be used to integrate the different approaches in the recommendation engine. The use of inference (based on genre, tags, etc) can be applied to shows that do not have ratings to increase the recommendation options. Access to user demographics would also significantly improve the recommendation accuracy (collaborative based filtering), however, we were unable to identify a dataset to support this. Reverse inference - building a user demographic model based on historical shows should allow for decoupling the reliance on obtaining demographic information and should improve the recommendation engine outputs. A hybrid filtering method with that of content-based and collaborative filtering can be useful to provide suggestions. This would take care of the limitations of both models. It is a mix of both user profile and content based,

VIII. Acknowledgements

We want to acknowledge the feedback from Professor Jaideep Srivastava and teaching assistant Jiacheng Liu to enable this to be a successful project.

References

[1] S. Mubarak, "Netflix dataset latest 2021," Aug 2021. [Online]. Available: https://www.kaggle.com/syedmubarak/netflix-dataset-latest-2021

[2] U. Mishra, "What is a content-based recommendation system in machine learning?" [Online]. Available: https://www.analyticssteps.com/blogs/what-content-based-recommendation-system-machine-learning

[3] "Introduction to stemming." [Online]. Available: https://www.geeksforgeeks.org/introduction-to-stemming

[4] "Clustering." [Online]. Available: https://scikit-learn.org/stable/modules/clustering.html

[5] J. Jordan, "Introduction to autoencoders." Mar 2018. [Online]. Available: https://www.jeremyjordan.me/autoencoders/

[6] J. Wu, "Building a collaborative filtering model with decision trees," May 2019. [Online]. Available: https://medium.com/@jwu2/building-a-collaborative-filtering-model-with-decision-trees-56256b95