Predictive movie recommendations using NLP content-based and user-based approaches

Tomas Hrdlovics
thrdlovi@purdue.edu

Anton Nagy
nagy18@purdue.edu

Andrea Zapata
azapatad@purdue.edu

1. ABSTRACT

Choosing a movie to watch is always a hard choice, and though someone may have previous information about the characteristics they enjoy in a movie, the enormous number of choices can still be overwhelming for a user making a conscious decision. That's the reason why most streaming platforms are currently using recommendation systems to help the users to choose the next movie, and keep them engaged with their services. There are many approaches to give a recommendation, in this paper we are going to cover a user-based collaborative filtering approach, a content-based filtering approach and a hybrid approach to determine whether an user would like a movie that they have not previously ranked using Kaggle's "The Movie Dataset" [1]. Our goal is to compare the accuracy between the different approaches and see in which scenarios each of the approaches performs better.

2. INTRODUCTION

As choices for consumers become more and more numerous, it becomes increasingly difficult to make an informed decision about what to invest in. That is why there are systems in place almost everywhere to recommend content and goods to consumers. These recommendation systems can be as basic as simply telling the consumer what the most popular or least expensive brand of an item is, or be as complicated as giving personalized recommendations based on extensive knowledge of the consumers tastes.

One prevalent example of a need for a good recommendation system is the problem of choosing a movie to watch. Because there are so many movies in existence, people have far too many options to choose from when selecting a movie they have not seen before to watch. Movies, and other forms of media content, are inherently difficult to recommend to consumers because people's preferences of them are subjective. Any given movie, irregardless of how popular it is, will be liked by some and disliked by others. There are some people who dislike the movie, "The Godfather," even though the general consensus of the public and critics is that it is one of the greatest films of all time. Because of this, perhaps simply recommending the most popular movies is not enough here. After all, people do not want to waste three hours of their limited time consuming a movie that they will not enjoy. It is imperative to narrow down the consumer's options so they can make a more informed decision about what movie they will enjoy.

Most platforms for streaming movies, such as Netflix, Amazon Prime Video, Disney+, etc. employ recommendation systems to attempt to maximize their users enjoyment in order to keep them engaged and increase the likelihood that the user will continue to consume content on that platform. Netflix's CPO has claimed that over 80% of the content that is consumed via Netflix come through recommendations; if a streaming service recommends shows and movies that users do not enjoy or are not interested in, there is a direct drop in content consumption and therefor a drop in subscription renewals [7]. There are many different approaches to creating recommendation systems, some better than others, and all have costs and benefits.

In this paper, we will compare multiple different movie recommendation systems to find a model or combination of models that has the greatest accuracy when classifying whether a given user will enjoy a given movie or not. We use a user-based approach, a basic content-based approach, a content-based approach using natural language processing, and then a hybrid model to classify movie enjoyment by a given user. For doing so, we used Kaggle's "The Movie Dataset", a dataset containing metadata on over $45,000$ movies and 26 million ratings from 270,000 users for all 45,000 movies. We compared the performance between the models by their accuracy on a validation set and then combined the tuned models into an ensemble to predict on test data, which reached around 78% accuracy.

3. RECOMMENDATION SYSTEMS

Recommendation Systems aim to provide an user with relevant options, for example recommending movies, songs, books, etc. There are many ways to approach this problem as collaborative filtering, content-based filtering and hybrid approaches.

3.1 User-based collaborative filtering

The user-based model we implemented was a user-based collaborative filtering algorithm. There are other forms of collaborative filtering, such as item-based, but a user-based collaborative filtering approach only requires information about the user's past preferences and other user's preferences. No additional information about the movie is required to formulate predictions.

User-based collaborative filtering operates under the principle that users with similar tastes will enjoy any given movie a similar amount. Using this assumption, we can recommend movies for users by predicting the user's enjoyment of any given movie. For example, if user X has rated movies A, B, and C highly, and user Y has rated movies A, B, C and D highly, we can predict user X will enjoy movie D.

The advantages of a user-based collaborative filtering approach is it only needs data on the users, not on the movies itself. It is fairly easy to implement and if needed, explaining why a recommendation was made is also simple; the answer is that similar users liked the movie. Unfortunately, this method of user-based collaborative struggles with sparse datasets. Because there are so many movies, most users have only seen a very small proportion of those movies. The number of actual reviews a user has given movies is a small percent of the total number of movies in the dataset, which results in a very sparse dataset. As there are more movies and more users added to the dataset, the cost to run a KNN algorithm, a common approach to user-based collaborative filtering, also increases. Additionally, if a new user is added to the database, they likely will not have seen and reviewed many movies, so there will not be sufficient data about their preferences to get accurate similarity measures to the other users. Similarly, when new movies come out, until many users see and review that movie, the movie will not have sufficient data to accurately recommend it to the users. Underrepresented movies, or movies that are small budget or that very few people have seen for whatever reason, have the same problem as the cold start problem of completely new movies or new users. Since few people have reviewed them, it is challenging to recommend them in a purely user-based approach.

To combat the problem of data sparsity in the user-based collaborative filtering approach, there is a technique called matrix factorization. Because not all users have rated all movies, our matrix will be sparse, meaning there will be numerous undefined values in the matrix. To combat this, one can decompose a sparse matrix to its component parts. These new low-dimension matrices will be less sparse.

Item based collaborative filtering, on the other hand, is based on recommending items that are similar to other items a particular user has rated highly. For item-based collaborative filtering, an item-item similarity matrix is first computed. For a movie recommendation system, we could use this movie-movie similarity matrix to determine what movies other users who have liked the given movie also liked, and recommend those movies. The main benefit an item-based approach to collaborative filtering has over a user-based approach is the item-item matrix can be pre-computed offline. During computation, this item-item matrix, however, can have dramatic computational overhead [6].

3.2 Content-based filtering

Instead of relying solely on the user's ratings of movies and using similar users to predict scores for movies that a user has not seen, a content-based filtering approach relies on actual features of movies. These movie features, such as genre, director, budget, plot, and more will provide insight on what type of movie a user enjoys. A content-based filtering approach use previous reviews of an user in order to find similar movies. For predicting the label for a movie A, it will look at the previous ratings that the same user gave similar movies to A and use those ratings to make a prediction.

An advantage of a content-based approach over a user-based collaborative filtering approach is the improved scalability. The user-based approach increases in computational complexity as the number of users and movies increases. While the content-based approach will only have that problem for whenever new movies are entered into the system,

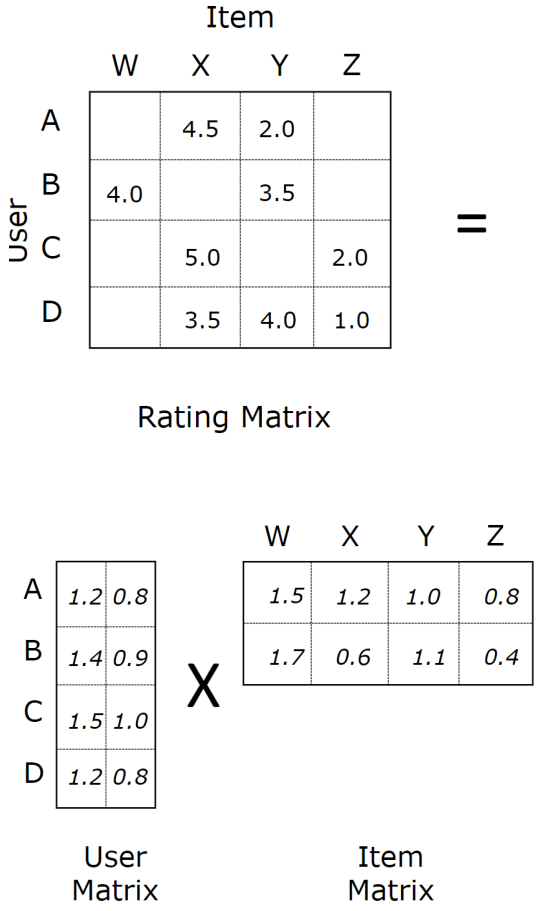


Figure 1: Matrix factorization for User-based collaborative filtering [3]

the user-based approach will change as the user-base grows and when new ratings appear.

Movie metadata has different types of features, we separated continuous and categorical features from natural language features for preprocessing and posterior model learning. For categorical features, we used Label encoding and one hot encoding to convert them to continuous features so that this problem could be treated as a binary classification problem and therefore standard machine learning algorithms as Logistic Regression, Support Vector Machines or Decision trees can be used.

3.2.1 Natural Language Features

One of the features that we want to use for this approach is the overview of the movie, which is a short description about the movie plot. We consider that this feature should be very important for defining the similarity between movies, because it is the only feature that gives us a hint about the actual content of the movie rather than standard information. As we are are dealing with Natural Language, there are different approaches for feature extraction as Term Frequency – Inverse Document Frequency (TF-IDF). With this approach the idea is to measure how important word is on a bag of words, to do so, we define the Term Frequency (TF) of a word t on a document d as the number of times that

the word appears on the document $f_{t,d}$. In Figure 2, the different variations of this metric are shown, some of these variations are used to account for the length of the text or other scaling terms.

Term frequency	
n (natural)	$\mathrm{tf}_{t,d}$
l (logarithm)	$1 + \log(\mathrm{tf}_{t,d})$
a (augmented)	$0.5 + \frac{0.5 \times \mathrm{tf}_{t,d}}{\max_t(\mathrm{tf}_{t,d})}$
b (boolean)	$\begin{cases} 1 & \text{if } \mathrm{tf}_{t,d} > 0 \\ 0 & \text{otherwise} \end{cases}$
L (log ave)	$\frac{1+\log(\mathrm{tf}_{t,d})}{1+\log(\mathrm{ave}_{t \in d}(\mathrm{tf}_{t,d}))}$

Figure 2: Term frequency variations

The inverse document frequency (IDF) measures how much information the appearance of certain term provides, for example, common stop words as "the" might have a large TF score, but they do not provide too much information as they will appear in most of the documents. In Figure 3 we can see the formulas for both IDF where N is the total number of documents and d_{ft} the number of documents in the collection that contain the term t. Then we can see, that the Term Frequency - Inverse Document Frequency (TF-IDF) is just the multiplication of both indexes. [5]

$$\mathrm{idf}_t = \log \frac{N}{\mathrm{df}_t}.$$

$$\text{tf-idf}_{t,d} = \mathrm{tf}_{t,d} \times \mathrm{idf}_t.$$

Figure 3: Inverse Document Frequency (IDF) and Term Frequency – Inverse Document Frequency (TF-IDF) formulas

3.3 Hybrid approach

As we mentioned before, user-based and content-based recommendation use different aspects of a dataset to come up with a label of their own. As each model could perform better for a particular kind of dataset, using a hybrid approach could help us find a balance that could perform better overall. The idea behind this approach is to use the outputs of user-based and content-based models as input for a linear regression model, so that we could learn the weights to output a final label.

4. DATA PREPROCESSING

The dataset we will use for our project is Kaggle's "The Movie Dataset." The dataset contains metadata for over 45,000 movies, including budget, revenue, cast, crew, plot information, etc. The dataset also has rating information from almost 300,000 users spanning all 45,000 movies [1].

4.1 Rating Data

For using a collaborative approach, we need to ensure that we have enough information of the users and movies that we are going to include in our training data. Having a user with few reviews might lead to bad performance in our content-based approach as we will use the movies that the user had previously ranked to find their similarity with the new movie. Also, for our user based approach, we will consider the reviews from different users to a given movie, so it is precise to consider movies that have been reviewed from more than one user. For this project, we will set a threshold of 50 for both cases, meaning that we are going to ignore reviews coming from users with less than 50 reviews and movies with less than 50 reviews.

In real life, there are different types of users when it comes of how they rate their movies. Some users might give higher scores to all the movies they like, while other users tend to be more critical so they never rank a movie with high scores. Then using a universal threshold to separate movies in categories like / dislike is not a good idea as different users have an inherent different scale for their rankings. In [4], a standardized metric to transform ratings from users to a binary label is proposed. Our rankings take values between [0.5, 5.0] with step of 0.5, so for each user we find the expected rating for all the movies that they had previously ranked using:

$$E = \sum_{c=0.5}^{5.0} c \cdot P(c),$$

where c is each possible ranking and P(c) is the proportion of user-rankings that take value c. After finding the expected rating for all users, we set the label like = 1 to samples whose rating is larger than the expected ranking for the user and like = 0 otherwise.

4.2 Movie Metadata

For this file, we first reviewed the attributes that were included in the dataset. Figure 4 contains a description of the features and its types. Initially, we deleted homepage, imdb id and poster path as the information does not relate to movie's popularity. For list features, we selected the most popular value for belongs-to-collection, production-companies, production-countries and spoken-languages and treated them as categorical features; for genres, we converted the column to a one-hot vector. Most of the continuous features had NaN values, that were transformed to zero in most cases, but due to the high number on NaN values we additionally deleted the features budget and tagline.

The attributes seem to be with few exceptions uncorrelated, in Figure 5 we present part of the correlation matrix.

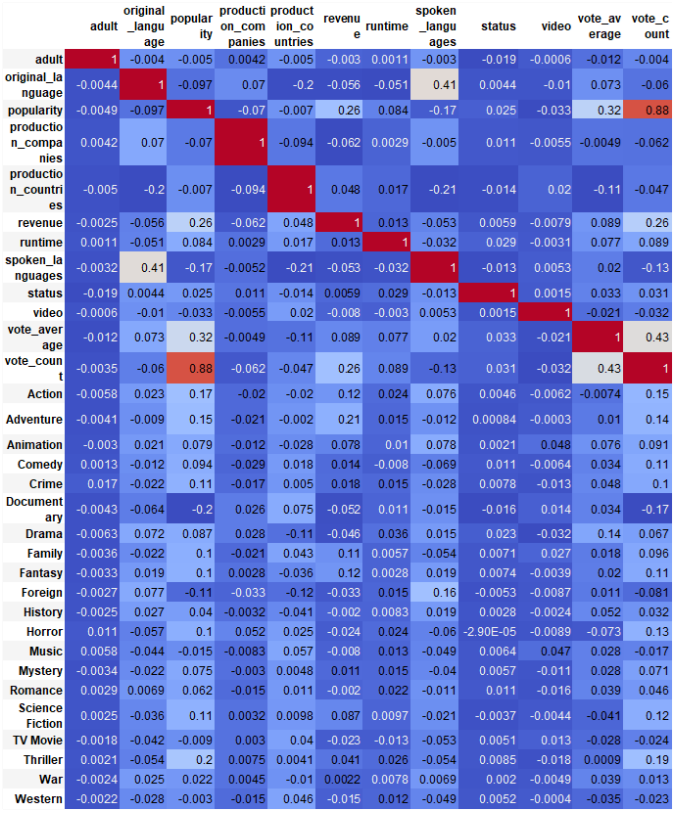
5. MODELS

5.1 Training, Validation and Testing

Because of the abundance of users, while at the same time having only relatively few reviews of movies per user (below 200 for most of the cases), we have decided to choose a different pattern than splitting user reviews into 3 parts (training testing and validation).

Feature	Type
adult	bool
belongs_to_collection	list
budget	float
genres	list
homepage	url
imdb_id	string
original_language	categorical string
original_title	string
overview	string
popularity	float
poster_path	string
production_companies	list
production_countries	list
release_date	date
revenue	float
runtime	float
spoken_languages	list
status	categorical string
tagline	string
title	string
video	bool
vote_average	float
vote_count	float

Figure 4: Feature Description



	adult	original_language	popularity	production_companies	production_countries	revenue	runtime	spoken_languages	status	video	vote_average	vote_count
adult	1	-0.004	-0.005	0.0042	-0.005	-0.003	0.0011	-0.003	-0.019	-0.0006	-0.012	-0.004
original_language	-0.0044	1	-0.097	0.07	-0.2	-0.056	-0.051	0.41	0.0044	-0.01	0.073	-0.06
popularity	-0.0049	-0.097	1	-0.07	-0.007	0.26	0.084	-0.17	0.025	-0.033	0.32	0.88
production_companies	0.0042	0.07	-0.07	1	-0.094	-0.062	0.0029	-0.005	0.011	-0.0055	-0.0049	-0.062
production_countries	-0.005	-0.2	-0.007	-0.094	1	0.048	0.017	-0.21	-0.014	0.02	-0.11	-0.047
revenue	-0.0025	-0.056	0.26	-0.062	0.048	1	0.013	-0.053	0.0059	-0.0079	0.089	0.26
runtime	0.0011	-0.051	0.084	0.0029	0.017	0.013	1	-0.032	0.029	-0.0031	0.077	0.089
spoken_languages	-0.0032	0.41	-0.17	-0.0052	-0.21	-0.053	-0.032	1	-0.013	0.0053	0.02	-0.13
status	-0.019	0.0044	0.025	0.011	-0.014	0.0059	0.029	-0.013	1	0.0015	0.033	0.031
video	-0.0006	-0.01	-0.033	-0.0055	0.02	-0.008	-0.003	0.0053	0.0015	1	-0.021	-0.032
vote_average	-0.012	0.073	0.32	-0.0049	-0.11	0.089	0.077	0.02	0.033	-0.021	1	0.43
vote_count	-0.0035	-0.06	0.88	-0.062	-0.047	0.26	0.089	-0.13	0.031	-0.032	0.43	1
Action	-0.0058	0.023	0.17	-0.02	-0.02	0.12	0.024	0.076	0.0046	-0.0062	-0.0074	0.15
Adventure	-0.0041	-0.009	0.15	-0.021	-0.002	0.21	0.015	-0.012	0.00084	-0.0003	0.01	0.14
Animation	-0.003	0.021	0.079	-0.012	-0.028	0.078	0.01	0.078	0.0021	0.048	0.076	0.091
Comedy	0.0013	-0.012	0.094	-0.029	0.018	0.014	-0.008	-0.069	0.011	-0.0064	0.034	0.11
Crime	0.017	-0.022	0.11	-0.017	0.005	0.018	0.015	-0.028	0.0078	-0.013	0.048	0.1
Documentary	-0.0043	-0.064	-0.2	0.026	0.075	-0.052	0.011	-0.015	-0.016	0.014	0.034	-0.17
Drama	-0.0063	0.072	0.087	0.028	-0.11	-0.046	0.036	0.015	0.023	-0.032	0.14	0.067
Family	-0.0036	-0.022	0.1	-0.021	0.043	0.11	0.0057	-0.054	0.0071	0.027	0.018	0.096
Fantasy	-0.0033	0.019	0.1	0.0028	-0.036	0.12	0.0028	0.019	0.0074	-0.0039	0.02	0.11
Foreign	-0.0027	0.077	-0.11	-0.033	-0.12	-0.033	0.015	0.16	-0.0053	-0.0087	0.011	-0.081
History	-0.0025	0.027	0.04	-0.0032	-0.041	-0.002	0.0083	0.019	0.0028	-0.0024	0.052	0.032
Horror	0.011	-0.057	0.1	0.052	0.025	-0.024	0.024	-0.06	-2.90E-05	-0.0089	-0.073	0.13
Music	0.0058	-0.044	-0.015	-0.0083	0.057	-0.008	0.013	-0.049	0.0064	0.047	0.028	-0.017
Mystery	-0.0034	-0.022	0.075	-0.003	0.0048	0.011	0.015	-0.04	0.0057	-0.011	0.028	0.071
Romance	0.0029	0.0069	0.062	-0.015	0.011	-0.002	0.022	-0.011	0.011	-0.016	0.039	0.046
Science Fiction	0.0025	-0.036	0.11	0.0032	0.0098	0.087	0.0097	-0.021	-0.0037	-0.0044	-0.041	0.12
TV Movie	-0.0018	-0.042	-0.009	0.003	0.04	-0.023	-0.013	-0.053	0.0051	0.013	-0.028	-0.024
Thriller	0.0021	-0.054	0.2	0.0075	0.0041	0.041	0.026	-0.054	0.0085	-0.018	0.0009	0.19
War	-0.0024	0.025	0.022	0.0045	-0.01	0.0022	0.0078	0.0069	0.002	-0.0049	0.039	0.013
Western	-0.0022	-0.028	-0.003	-0.015	0.046	-0.015	0.012	-0.049	0.0052	-0.0004	-0.035	-0.023

Figure 5: Correlation Matrix on Movie Metadata Features

We have decided to divide the dataset of users and their reviews into 2 parts. The first three quarters of users are used for testing and validation purposes and the remaining quarter is used for training a model and give final testing accuracy. This achieves the goal of tuning the model without the exposure to the testing dataset while at the same time keeping the number of reviews per user and data split relatively low. The distribution of all parts is following: User splitting: -75% training and validation, 25% training and testing, Split of reviews per user: 75% training, 25% validation or testing

```
Train   ....75%....|Validate   ..25%..
Train   ....75%....|Validate   ..25%..
Train   ....75%....|Validate   ..25%..
Train   ....75%....|Test       ..25%..
```

5.2 Content-Based filtering without NLP

Content based filtering is based on information we have about the movies. The predictive models for this part predicts data based on numerical attributes. Categorical values were transformed using one hot encoding. This however leads to a huge dimensionality problem known as the curse of dimensionality. For instance in our dataset we have more than 200 production companies. To fight with it we have decided to use PCA (principal component analysis) in order to remove columns that are linearly dependant on other columns or don't provide us useful information (for instance almost all values being 0).

Using *Scikit-learn* we train variety of models and combine the results of models into one vote and final prediction. This approach is known as an ensemble method and is supposed to minimize the noise, bias and variance. The models we used are as follows:

- **Random Forest Regressor** - Bagging technique that uses many trees for classification. In our case 200.
- **Decision Tree Classifier** - For the best results we limit our tree depth to 4 to prevent over-fitting
- **K-NN** - we use majority vote of 3 closest movies
- **SVC** - polynomial kernel
- **Linear Regression**
- **AdaBoost Classifier** - 150 estimators
- **GaussianNB** - in order to counter 0 probability problem, we use Laplace's smoothing with the parameter set to 1e-09

The accuracy of these models were then linearly combined and output one prediction. Such prediction is marked as Combination in the figure (8). The best accuracy was achieved when we prioritized prediction of the 2 models that performs the best (*Linear Regression* and *Random Forest*) - giving them vote of weight 4 and 3 respectively - while giving last two models just a vote with single weight. The rest has a vote of weight 2. Such combination outperforms all single models although it is fair to say that only by a tiny margin. Test error for combination model is 66.7%, test error for linear regression is 66.6%. When we used simple majority voting where all models has the same weight the testing error is 6 1.4%.

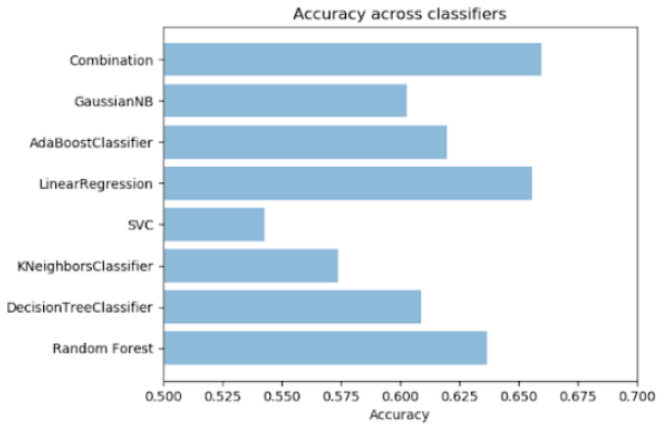


Figure 6: Accuracy across weak learners

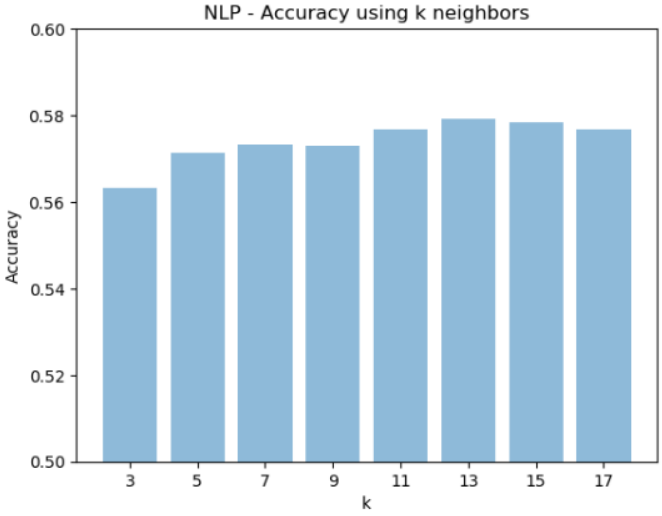


Figure 7: NLP Based Accuracy by number of neighbors

5.3 Content-Based filtering using NLP

In order to use the overview feature as part of our content based approach, we used NLP techniques for feature extraction. As we mentioned before, we used TF-IDF implementation in sklearn TfidfVectorizer with standard attributes removing stop words for English. We filtered the dataset so that we will train individual models for each user, so the TF-IDF matrix will only consist on movies that the user has previously rated. For implementation purposes, we also grouped all the movies in the testing dataset for a given user, so that we only calculate this matrix once, and find the similarity for each movie on the user test set.

In order to find the similarity, we computed the lineal kernel between the test movie and every other movie that the user previously rated and found the movies with the highest scores. In Figure 7 we can see the accuracy on the validation set for different values of k, we choose k=13 for our test model. Overall, similarity scores where in 1e-2 magnitude, but we noticed that for movies within the same franchise, for example, The Chronicles of Narnia: Prince Caspian and The Chronicles of Narnia: The Lion, the Witch and the Wardrobe, the similarity score was around 1e-1. Because of that, we decided to use a weighted label, instead of majority voting as we did for other models. We multiply each similarity score for their corresponding label and found the mean value dividing it by the sum of the k similarity scores. With this approach we were able to achieve an accuracy of around 0.57%.

5.4 User Based Recommendation

To implement the user-based collaborative filtering approach, we use the K Nearest Neighbors algorithm, which finds the k most similar users to the user in question. The KNN algorithm uses a similarity function to calculate how similar any two given users are, and so this metric can give us the k most similar users to any given user. There are a plethora of distance metrics that could work in the KNN algorithm including Pearson Correlation, and Hamming Distance (for binary classification problems), but we employ one of the most common similarity functions, Cosine Similarity, given by the following formula.

$$\text{Cosine Similarity} : Sim(u_i, u_k) = \frac{r_i * r_k}{|r_i||r_k} = \frac{\sum_{j=1}^{m} r_{ij} r_{kj}}{\sqrt{\sum_{j=1}^{m} r_{ij}^2 \sum_{j=1}^{m} r_{kj}^2}}$$

To calculate cosine similarity between user A and user B, we simply use the *Scikit-learn* library to calculate the cosine of the angle formed by the vector made up of user A's movie preferences and the vector made up of user B's movie preferences. To do this, we transform our data comprised of user IDs, movie IDs, and binary "like" classification to a 2-dimensional matrix with the users on one axis and the movies on another and the classification as the values. Now that we have our user-movie matrix, we calculate the cosine similarity between our given user and all of the other users who have seen the movie that we are trying to predict if the given user will enjoy. We then find the k users with the highest cosine similarities to the user in question. We then use majority voting with these k users to determine our predicted "like" classification for the given user and the given movie. A similar process for predicting user ratings is described in the paper, "Design and Implementation of Movie Recommendation System Based on Knn Collaborative Filtering Algorithm," although the algorithm is modified because the specific problem we are solving is a binary classification problem [2].

After training the user-based collaborative filtering algorithm on the training set and testing on the validation set with different values of k, we found the optimal k value is 13. With a k value of 13 for our KNN algorithm, we get a final accuracy of 77%.

5.5 Hybrid Based Recommendation

Hybrid based recommendation combines both user and content based recommendations. This combination finds a balance between user's distinct taste and preference of movies (content based) and the similarity of preferences among other users. Hybrid classification is fairly similar to the con-

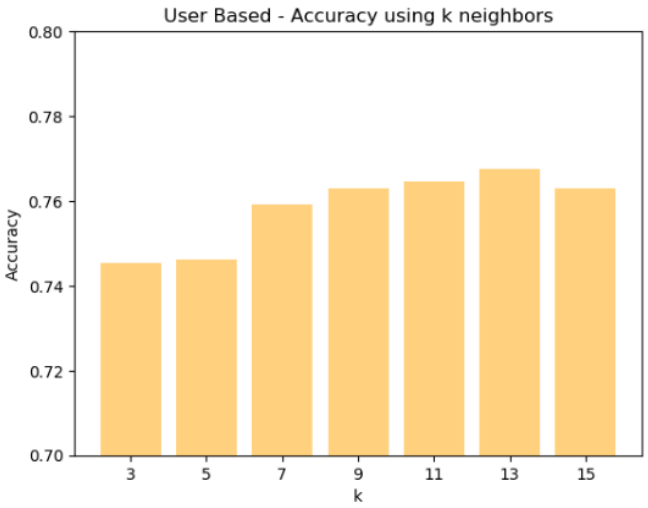


Figure 8: User Based Accuracy by number of neighbors

tent based filtering that combines different models. This time 2 other models are added. NLP and User Based Recommendation. Because of a huge difference in accuracy between User Based and Content Based recommendations, decision has been made that if user based prediction has higher confidence than 70%, the final prediction is automatically assigned based on the user prediction. In other cases user based recommendation is just part of a majority voting where different classifiers have the same vote. In this scenario weights of the new models (*User Based*, *NLP*) are 9 and 3 respectively. This results in a performance which is close to 78% - 77.8%.

6. CONCLUSIONS

Recommendation systems present a major challenge for modern systems, as they will need to interpret all the collected data of their users to make good recommendations. With the launch of new streaming providers, having a clear idea of what a user would like or not, can made give advantage to companies not only for recommendations but also for the development of new content. In this project, we found that user based recommendation is a strong model compared with its content based equivalents. As we mentioned earlier, our dataset contained more than 20 million reviews which provides a great input for recommendations. Nevertheless, a user based recommendation system will present some challenges in terms of scalability, as with a growing number of user-user interactions in the data, the predictions are volatile. For our testing, we just considered users with more than 50 ratings, which seems like a large number considering that some users do not rate the movies that they see or would need some time to achieve that number of rating. Because of this, relying solely on user based recommendations would not be a good idea.

Neither of the content-based recommendation models achieved an accuracy as high as the user-based model. We think that this might have to do with the quality of the features included for the model. As we mentioned in the Data Preprocessing section, we had to remove important features as budget because they had many null values. We think that a better collection of features could improve the accuracy of this type of models. Also, we noticed that applying dimension reduction helped us to get better results, especially for features like genre, where a movie could have multiple values. For other features that had a similar structure, we decided to pick the most representative value instead of applying dimension reduction. Perhaps using the complete set of values for this attributes could expose some patterns that would be very helpful for specific users. Regarding the overview feature, we think that a longer description of the movies could give us more insight to achieve larger similarity scores. So far, our model is just able to predict very clear plot similarities, as the ones found between movies that share character names or places. This was probably caused by choosing TF-IDF as the feature extraction mechanism, which would output a larger score for movies that share specific terms.

Finally, as we expected, combining multiple models improved our accuracy. As we mentioned earlier, each model is able to capture some essential insights to approach this problem, therefore, an ensemble would be the best option overall. Once each individual model get tuned to achieve their best performance, finding a balanced distribution between each model prediction would guarantee an overall performance.

7. FUTURE WORK

After conducting a thorough exploration of recommendation systems, there are a few areas we would like to explore further in the future. Because our user-based approach worked better with our data, we would first examine whether a different dataset would lend itself to other models, perhaps a dataset with fewer user reviews in comparison with movie attributes. We are curious whether different subsets of movies could be more accurately recommended with certain approaches; for example, within the same genre or franchise, our NLP content-based approach functioned very well. Our hypothesis is that user-based recommendation approaches span different genres and franchises better than content-based approaches because the recommendations are not based on specific features of movies. Movies in the same genre and franchise will share more movie-specific features such as plot and key-words. Our user-based approach does not account for these movie-specific features, so we would like to specifically explore if a user-based recommendation system gives better cross-genre recommendations.

Additionally, our natural language processing component of our content-based approach dealt with a plot description attribute of each movie. Given more time, we would like to find addition data containing text-based user reviews of movies. We would like to see if using some sentiment-analysis NLP techniques on user reviews would improve the accuracy of our models.

8. REFERENCES

[1] R. Banik. The movies dataset, Nov 2017.

[2] B.-B. CUI. Design and implementation of movie recommendation system based on knn collaborative filtering algorithm. *ITM Web of Conferences*, 12, 2017.

[3] S. Ghosh. Simple matrix factorization example on the movielens dataset using pyspark, 2018.

[4] S. B. K. Jieun Son. Content-based filtering for recommendation systems using multiattribute networks. *Expert Systems with Applications*, 89:404–412, 2017.

[5] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008.

[6] B. M. Sarwar, G. Karypis, J. A. Konstan, J. Riedl, et al. Item-based collaborative filtering recommendation algorithms. *Www*, 1:285–295, 2001.

[7] B. Smith and G. Linden. Two decades of recommender systems at amazon. com. *Ieee internet computing*, 21(3):12–18, 2017.