Hybrid Music Recommender System

Course project for CSE 6240: Web Search and Text Mining, Spring 2020

Pranshu Trivedi
Georgia Institute of Technology
ptrivedi31@gatech.edu

Sung Jae Hong
Georgia Institute of Technology
shong313@gatech.edu

Wyndham Hudson
Georgia Institute of Technology
whudson9@gatech.edu

ABSTRACT

In this project, we aim to develop a hybrid music recommender system that combines collaborative filtering methods and content based filtering methods to improve accuracy and relevance of the songs recommended. For this project, we created two baseline models that use collaborative filtering to recommend songs users might like based on their history. We found that the recommender system that uses singular value decomposition performed better than the system that uses k-nearest neighbors with smaller root mean square error values. A novel hybrid model developed mitigated the problem of cold start users at a cost of increased root mean square error.

1 INTRODUCTION

With more people on the web and with more services for commercial and industrial uses, recommender systems have become an essential feature of web applications. From e-commerce where they predict user interests to online advertisements where they suggest to users relevant advertisements to increase probability of interaction, recommender systems have become very significant for both businesses and consumers. With recently increasing competition amongst companies to attract and retain their users, businesses spend a lot more time, money, and resources into these engines. However, a big challenge companies face is on creating an algorithm that performs well on their unique dataset.

In this project, we focus on developing a hybrid music recommendation engine that combines the advantages of both methods using the Taste Profile User Dataset and the Million Songs Dataset. We will first analyze the datasets and perform collaborative filtering using Singular Value Decomposition and K-Nearest Neighbors as our baseline. We will then do content-based filtering using the Million Songs Dataset and also analyze different algorithms on collaborative filtering for hyperparameter tuning. With these methods, we create a hybrid recommendation engine and compare the performance with our baselines. Hybrid recommender engines have been known to perform better than recommender engines solely using collaborative filtering or content-based filtering. The developed hyrbid recommender engine performs worse than the baseline approaches with respect to the measured root mean square error, however, it does mitigate the cold start problem [3]. Because no hybrid recommender engines have been previously created for the Million Songs Dataset, we hope that our project will be used as a baseline for future studies on music recommendation engines and improve performance on music recommendation.

2 LITERATURE SURVEY

Bobadilla et. al provide a comprehensive survey of recommendation system technologies [3]. In their paper, they first set forth the foundations of recommendation systems and enumerate the factors that can determine a recommendation system's performance. The first and crucial factor is the type of data available. Whether user information, user-item ratings or feature data of items is available can impact the types of filtering techniques and models that can be feasible and robust. The filtering technique dictates how the recommendation system makes its recommendations. The most notable filtering techniques are content-based and collaborative filtering. Content-based filtering recommends based on the features describing an object. On the other hand, collaborative filtering uses information about other user preferences to make recommendations about a specific user. Another notable technique is hybrid filtering which combines other multiple techniques to form a more robust recommender. Another factor of recommendation systems is the level of sparsity of the database. For example, a collaborative filtering technique may use a very sparse user-item rating matrix, which can waste a lot of resources if not stored properly.

Previous content based models have been developed for music recommendation. Many content based models use the acoustic qualities of the music itself often with limited success [4]. Bu et al. attempted to use social media data on Last.fm to generate high-order features based on Last.fm's objects of tracks, users, tags, groups, and artists. Bu et al. created a hypergraph to combine and rank the multiple relationships between the different objects on Last.fm. Additionally, Bu et al. encorporated tradional acoustic-based concepts into an overarching model. Bu et al. evaluated their proposed algorithm based on the generated hypergraph by predicting whether a set of test songs would appear in a listener's recommended songs given the hypergraph without the test songs. The proposed hypergraph model outperformed acoustic-based models, other graph based models and collaborative filtering models in terms of Mean Average Precision. Precision was defined as the number of correctly recommended items divided by total recommended items. It is important to consider Bu et al.'s hypergraph was based on a smaller dataset and only contained around 16,000 tracks and 371 artists in total. While this model could be extrapolated to a larger dataset, it would be potentially computationally expensive with a large hypergraph and performance could be diminished compared to traditional collaborative filtering techniques. Additionally, Bu et al. stated that they saw a very sparse user-track matrix in collaborative filtering that could have led to poor performance, and did not try to optimize this model.

Traditionally, recommender systems perform well using collaborative filtering methods, disregarding the cold-start issue for new users [5]. Generally, there are two methods: rating-oriented and ranking-oriented. Rating-oriented Collaborative Filtering is a pointwise method of predicting a user's ratings on unrated items based

on similar users' ratings. Ranking-oriented Collaborative Filtering, however, directly predicts the preference ranking of items for each user. Huang et al. propose a ranking-oriented model which directly predicts a list of items ranked by probability for each user. This algorithm, called ListCF, reduces computational complexity of training and prediction procedures by returning a list of unrated items to a user based on other similar users, instead of returning a single unrated item. Huang et al. evaluated ListCF by comparing the results between other popular memory-based CF algorithms on the Movielens-1M, EachMovie, and Netflix recommender datasets. The metric used for comparison was the Normalized Discounted Cumulative Gain. While ListCF was not as efficient in run-time as other Collaborative Filtering algorithms, Huang et al. claim the rank accuracy on both the Movielens-1M and EachMovie datasets was greater than the other common algorithms examined. While the ListCF algorithm works well on the test datasets evaluated in the paper, it showed some degradation in performance when the user-item matrix was sparse. This implies that ListCF may see lower accuracy on a music recommendation dataset, as user ratings across different genres may be very sparse (if users only listen to few types of music).

3 DATASET DESCRIPTION

3.1 Data preparation

3.1.1 Source. The source of data that was decided upon was the Million Songs Dataset [2](referred to as the features dataset) and one of its complementary user dataset, the Taste Profile subset. These two datasets are available for direct download from the Millions Songs Dataset website.

3.1.2 Data preprocessing. A random sample of 10000 songs from the features dataset was used for analysis because the whole dataset is 240GB. The features dataset was provided in a HDF5 format in nested directories that needed to be extracted to a csv file for analysis. After this extraction, the data was loaded as a DataFrame in pandas and duplicates were removed. Irrelevant feature columns were dropped. Rows with missing values for any columns were dropped since there were not many missing values (only 4 rows). The feature dataset was then scaled using sklearn's RobustScaler to improve the ability to compare features in analysis. The user dataset was provided in a text file in a space delimited format. This was also converted into a csv file. No duplicates or missing values were found. This dataset is sufficient for our purpose because the features dataset contains a million songs and the accompanying user dataset contains more than a million unique users. With both of these datasets, we have fulfilled the data requirements of content-based, of collaborative filtering, and of a hybrid recommender, which is the goal of this project.

3.2 Raw data statistics

3.2.1 Features Dataset. The complete features dataset contains more than a million datapoints(songs) however, for analysis a random subset of 10000 datapoints was used. The following are the relevant features used:

- hotttness (float): refers to how popular the artist of the song was at the recorded time.
- familiarity (float): refers to how familiar users are with the artist of the song.
- duration (float): total duration of the song
- loudness (float): general loudness of the song
- tempo (float): measures the beats per minute of the song.
- key (integer): refers to the key of the song. This refers to what scales the song is built around.
- mode (integer): refers to the mode of the song. Major(1) or minor(0) scale.
- time signature (integer): refers to the number of beats per bar.

The summary statistics of these relevant features are shown in Figure 1. Key, mode and time signature are both categorical variables that have been encoded into integers.

3.2.2 User Dataset. The user dataset that is part of the Million Songs Dataset is in the form of triplets as *(user, song, play count)*. These are some relevant statistics about it:

- Number of users: 1,019,318 unique users
- Number of songs: 384,546 unique songs
- Number of datapoints/triplets: 48,372,586 *(user, song, play count)* triplets.

	hotttness	familiarity	duration	loudness	tempo	key	mode	time_signature
count	9996.000000	9996.000000	9996.000000	9996.000000	9996.000000	9996.000000	9996.000000	9996.000000
mean	0.385706	0.565456	238.518765	-10.488092	122.910529	5.276611	0.690976	3.564626
std	0.143469	0.160161	114.157658	5.399337	35.183146	3.554199	0.462114	1.266462
min	0.000000	0.000000	1.044440	-51.643000	0.000000	0.000000	0.000000	0.000000
25%	0.325506	0.467611	176.006080	-13.166250	96.965750	2.000000	0.000000	3.000000
50%	0.380756	0.563666	223.059140	-9.382000	120.161000	5.000000	1.000000	4.000000
75%	0.453930	0.668020	276.375060	-6.535250	144.006750	8.000000	1.000000	4.000000
max	1.082503	1.000000	1819.767710	0.566000	262.828000	11.000000	1.000000	7.000000

Figure 1: Descriptive statistics of relevant features.

3.3 Data Analysis

3.3.1 Correlations. The Pearson correlation matrix was computed for all relevant features introduced previously. The heatmap of this correlation is shown in Figure 2. The heatmap indicates that most features do not linearly correlate with each other. However, hotttness and familiarity are highly linearly correlated. This makes sense because they both essentially measure the popularity of the artist of the song.

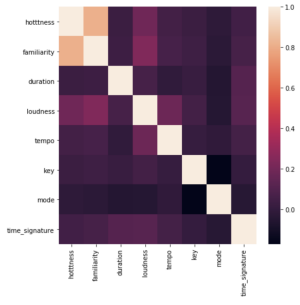
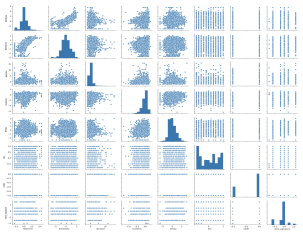


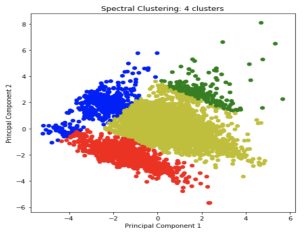
Figure 2: Heatmap of the Pearson correlation coefficient matrix.

3.3.2 Pairwise Distributions. Pairwise distributions of features and feature distributions were plotted to gain additional insights. These are shown in Figure 3a. The pairwise plot confirms the linear correlation of hotttness and familiarity as their plot is fairly linear. Key, mode and time signature are also confirmed to be categorical variables as their distributions with other features has discrete spaces. It can also be seen here there is an heavy imbalance of the mode feature where there are more than twice as many songs with a major(1) mode than the minor(0) mode. The familiarity feature also appears to be normally distributed, whereas both duration and loudness seem to have large tails in their distributions. This indicates that most songs are typically of the same duration and loudness.

3.3.3 Dimensionality Reduction and Clustering. Using Principle Component Analysis, the relevant features were reduced to two principal components to enhance the visualization of the dataset and various clustering methods were tried but spectral clustering[7] seemed to work the best to cluster the data into four groups. The clusters from this method are shown in Figure 3b. The yellow cluster seems to include some of green cluster's datapoints. The blue cluster also includes some datapoints from the red cluster.



(a) Pairwise distribution of relevant features.



(b) Four clusters found by spectral clustering.

Figure 3: Clustering and Distribution of Dataset

4 EXPERIMENTAL SETTINGS

We use two collaborative filtering models, singular value decomposition and k-nearest neighbors, to train and build models to predict the top songs for a user based on the million song dataset. We separate the dataset into training and testing. The models are trained to predict the number of times a user will have played a song. We used a train-test split of 75% and 25%, respectively. The data is cross-validated using a 5-fold split. We evaluate the models on the million song dataset using the Root Mean Square Error, defined as

$$RootMeanSquareError = \sqrt{\frac{1}{|\hat{R}|}\sum_{\hat{r}_{ui}\in\hat{R}}(r_{ui}-\hat{r}_{ui})^2}$$

where $\hat{r}_{ui}$ is the model prediction of the number of times a user u plays the song i, and $\hat{R}$ is the collective set of predictions. The ground truth value for the number of plays a user u has for a given song i is r_{ui}. The ground truth value r_{ui} is taken from the dataset directly.

After training the models, the songs with the highest predicted number of plays can be returned for each user to serve as a base recommender system.

The system settings used to run the code are 16GB RAM and an Intel i7-9750H CPU.

5 BASELINES

5.1 Baseline 1: Singular Value Decomposition

Our first baseline model uses Singular Value Decomposition in the collaborative filtering method-based recommender system. This model was based on a paper by Raghuveer et al. [1]. We use the same model from the Surprise Recommendation Kit [6], a library the paper referenced and used, and the Root Mean Square Error error metric in which the authors denote to be a better error metric than Mean Absolute Error as it penalizes larger error more. Singular Value Decomposition is a matrix factorization technique that attempts to factorize matrix A into 3 matrices such that $A = USV^T$ where U and V are orthogonal matrices with eigenvectors from AA^T and A^TA and S is a diagonal matrix with roots of the positive eigenvalues of AA^T and A^TA.

The model then creates a prediction $\hat{r}_{ui}$ for a given user u and item i using

$$\hat{r}_{ui} = \mu + b_u + b_i + {q_i}^T p_u$$

where b_u is the bias for user u, b_i is bias for item i, q_i is the vector associated with item i, and p_u is the vector associated with user u [6]. To reduce the error, the following regularized square error is reduced:

$$\sum_{r_{ui}\in R_{train}}(r_{ui}-\hat{r}_{ui})^2 + \lambda(b_i^2 + b_u^2 + ||q_i||^2 + ||p_u||^2)$$

where r_i is the ground truth label. The error is then minimized using stochastic gradient descent [6].

As for our hyper-parameters, the model was trained for 20 epochs with a regularization constant of 0.02. The learning rate for the stochastic gradient descent was 0.02. The number of instances in the dataset was kept at 10000. Singular Value Decomposition is suitable for this problem because it is able to discover important features underlying interaction between two different kinds of entities and predict ratings in collaborative filtering in order recommend items that a user might like but is not directly given.

5.2 Baseline 2: K-Nearest Neighbors

The K-Nearest Neighbors method used is based from the same paper by Raghuveer et al. [1]. While the paper uses a different dataset, the collaborative filtering method is relevant to our specific problem. The Surprise Recommendation Kit[6] was used to create a basic K-Nearest Neighbors model. In this dataset, each song is not rated by a user, instead the number of plays are used in place of a rating. A user-song matrix with the number of plays is created from the million song dataset, forming the ground truth set of predictions.

The K-Nearest Neighbors model then creates a corresponding prediction $\hat{r}_{ui}$ for a given user u and item i using

$$\hat{r}_{ui} = \frac{\sum_{v\in N_i^k(u)} sim(u,v)\cdot r_{vi}}{\sum_{v\in N_i^k(u)} sim(u,v)}$$

where $sim(u,v)$ is a similarity measure between two users u and v. This similarity measure was defined to be the cosine similarity between all the common song play values between the two users.

The number of clusters used was five. The cosine similarity is given below

$$sim(u,v) = \frac{\sum_{i \in I_{uv}} r_{ui} \cdot r_{vi}}{\sum_{i \in I_{uv}} r_{ui}^2 \cdot \sum_{i \in I_{uv}} r_{vi}^2}$$

The K-Nearest Neighbors model makes sense for our data. K-Nearest Neighbors in this application finds the closest users based on similarity in number of plays given a set of shared songs. It follows logically that if users share commonalities in the songs they listen to most, the users could have similar taste in music. Therefore if one user listens to a song that the other similar user has not listened to yet, it might be suitable to predict this song due to their shared taste. This forms the basis for new music discovery which is desirable in music recommendation engines.

5.3 Baseline Results and Discussion

As discussed, the two baselines were evaluated using the Root Mean Square Error. The bar chart in Figure 4 depicts the values for each baseline. With a Root Mean Square Error value of 4.17, Singular Value Decomposition performed better than K-Nearest Neighbors which scored 4.69. Singular Value Decomposition performed better because the user-item matrix is very sparse.Due to the nature of the dataset, there are not many songs users have listened to compared to the number of songs available. This makes K-Nearest Neighbors more difficult to provide good performance as the clusters will be scattered. As there are more data available, this problem of sparsity will affect K-Nearest Neighbors more than Singular Value Decomposition. Indeed, as we varied the size of the dataset to train on and increased the size of the dataset, Root Mean Square Error for K-Nearest Neighbors increased whereas the recommender system using Singular Value Decomposition remained mostly stable. Our performance was similar to that of Kumar et al. where our model with Singular Value Decomposition scored 4.1 whereas the paper reported a Root Mean Square Error of 3.6.

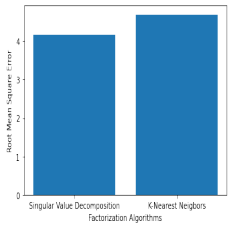


Figure 4: Baseline results for SVD and K-Nearest Neighbors.

6 PROPOSED METHOD

6.1 Content Based Filtering Method

While collaborative filtering models are traditionally used with great success in recommender applications, collaborative filtering fails when a user has little or no data associated with them. In these cold-start cases, a content based filtering method has been generated as suitable replacement for recommendations. Therefore in order to improve over the baseline methods, we propose a hybrid method integrating a content based model.

In our content based model, we use a standard application of K-Nearest Neighbors to associate songs based on the different features from the Million Songs Dataset. In order to pick the best parameters for the content based model, an experimental set up was used where 1000 random users were selected. For each user, 75% of their songs were sampled and input into the K-Nearest Neighbors model, and 25% was used as a test set. We evaluated our content based model using average recall per user and average precision per user. It should be noted that for most songs, the model did not return any songs that intersected with the test set, however since the content based model is objectively trying to cast a wide net to a user without much data, the highest average recall and average precision will be the guiding principles for all users. The number of predicted songs was another variable that could be manipulated and therefore average recall was the main measure used for evaluation.

The parameters that create a K-Nearest Neighbors model which outputs the most songs that users actually had listened to, based on the average recall per user, were established through experimentation described in section 7.1 and used in the hybrid model described below. The parameters tested were the features used for K-Nearest Neighbors calculation ('hotttness', 'familiarity', 'loudness', 'tempo', 'key', 'mode'), the distance metric ('Manhattan' , 'Euclidean', 'Chebyshev', 'Minkowski'), and the number of songs returned (or K in K-Nearest Neighbors). The ideal parameters were 'hotttness' and 'familiarity' for the parameter of feature space, 'Manhattan' for the parameter of distance metric, and the number of songs returned were 50, excluding input songs whose distances were zero.

6.2 Collaborative Filtering

On the other hand, a drawback of the content-based filtering approach is that the method can only recommend songs that are very similar to the input songs. Collaborative filtering looks into preferences of similar users and can recommend diverse songs a user has not listened to but might like based on other users. By incorporating the strengths of content-based filtering and collaborative-filtering, hybrid recommender systems have been studied to perform better than recommender systems solely relying on collaborative-filtering and content-based filtering.

For hyperparameter testing, we tried different collaborative-filtering algorithms and chose the algorithm that returned the lowest prediction error on our validation set. We used 5-fold cross validation and used root mean square error as our performance metric to compare our results with our baseline models. The algorithms that were tested, in addition to singular value decomposition and K-Nearest Neighbors, are Slope One, Co-Clustering, and Non-negative Matrix Factorization. Similar to our baseline models, each model was trained for 20 epochs with a regularization constant of 0.02. The learning rate for the stochastic gradient descent was 0.02. The collaborative filtering models were implemented using the Surprise Recommendation Kit. For each model, given the User-Song dataset, the model returns K songs a user might like. For testing purposes, K was kept to 10.

6.3 Hybrid Model

The hybrid model is the combination of the content based model and the collaborative filtering model with the hyperparameters of the individuals set as previously discussed in their respective sections. To construct the final model, the top n recommendations from both the content-based model and the collaborative filtering for each user are calculated. The hybrid model tunes the ratio of recommendations from each of the models based on the input user's history. This property of the model combats the cold start problem of a new user [3]. If the input user has not played many distinct songs, the user is regarded as having a short user history, thus the content-based model is more relied upon for the final output recommendations of the hybrid model. On the other hand if the user does have a significant user history, the collaborative filtering model is more relied upon. A user is said to have a significant user history if they have played more than the average number of songs played per user. If a user's history is more than 50, the top n recommendations of the collaborative filtering model are used. If a user's history is less than 50 distinct songs, then this distinct songs metric is scaled to the range of $[0,n]$, where n is top n recommendations for each of the previous models. The scaled metric, s, determines the total number of collaborative filtering recommendations. The rest of the recommendations n - s come from the top n recommendations of the content-based model. Thus, the final output of recommendations is a combination of the top n recommendations of both the content-based model and the collaborative filtering model, in which the ratio of recommendations from these two models is linearly tuned according to the user history of the input user.

7 EXPERIMENTS

7.1 Content Based Filtering Method

In order to determine how many neighbors may be optimal to return in the content based K-Nearest Neighbors algorithm, we evaluated the average recall and precision as seen in Figure 5. As expected, as more songs were returned, the average recall increased and the average precision decreased. However, it would be unreasonable to return a recommendation list of 400 songs to a user, and as a result, we settled that 50 songs would be used as the benchmark for the remaining evaluations.

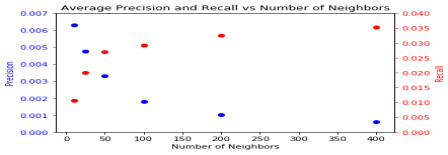
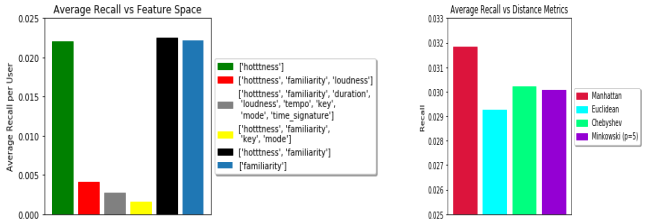


Figure 5: Average Recall and Precision with increasing K-neighbors.

Additionally, different features were evaluated to find which may be the most useful to maximize average recall per user, while keeping all other parameters the same. The results in Figure 6a show that a combination of 'hotttness' and 'familiarity' generate



(a) Average Recall against Different Feature Spaces

(b) Average Recall against Different Distance Metrics

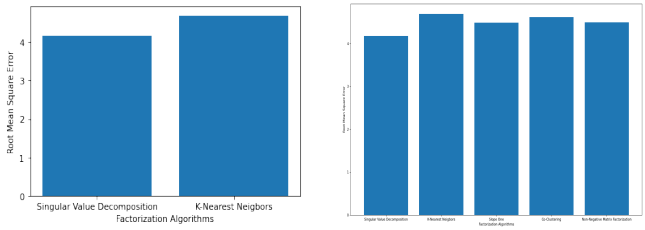
Figure 6: Hyperparameter Testing for Content-based Filtering

the highest average recall. Adding more features seems to diminish the average recall.

It was discovered that the distance metric used in the K-Nearest Neighbors algorithm could result in differences in average recall. When keeping all other factors the same, different distance metrics were tested. The results, shown in Figure 6b shows that the Manhattan distance results in the best average recall per user.

7.2 Collaborative Filtering

To ensure our experiment with collaborative filtering models are consistent with the baseline models, we kept several hyperparameters the same. Each model was trained for 20 epochs with a regularization constant of 0.02. The learning rate for the stochastic gradient descent was 0.02. The collaborative filtering models were implemented using the Surprise Recommendation Kit. For reference, the baseline results for Singular Value Decomposition and K-Nearest Neighbors are shown below in Figure 7a. The root mean square error for singular value decomposition and K-Nearest Neighbors are 4.17 and 4.69 respectively.



(a) Baseline Results for Singular Value Decomposition and K-Nearest Neighbors

(b) Performances of Multiple Collaborative Filtering Algorithms

Figure 7: Experimentation of Different Collaborative Filtering Algorithms

The results for the performances of other collaborative filtering algorithms are shown below in Figure 7b. As we can see, our results were consistent with our baseline in that the singular value decomposition performed the best in terms of having the lowest error in root mean square error. Our other approaches performed better

than K-Nearest Neighbors, but still performed worse than Singular Value Decomposition. An explanation for this is that singular value decomposition is an effective matrix factorization technique that is able to learn the inference of the relations between the user matrix and the item matrix and calculate the probability that a user will like a new item based on the user's history and other user's preferences effectively.

7.3 Hybrid Model

7.3.1 Experimental Setting. To compare the hybrid model with the baseline approaches, the root mean square error metric must be computed. However, since the content-based model does not predict the plays for a user-song pair, the plays for each content-based recommendation were assigned to the mean of all plays in the dataset. This would simulate the penalty that a recommendation from the content-based model may not necessarily align with the user data. To evaluate the root mean square error of the hybrid model, n was set to be 10. The hybrid model was then tested with a 5-fold cross validation, where the training and testing set consisted of unique user ids from the user dataset. All of the features data was used for training on the content-based side. For each input user, the top n recommendations and their estimated plays were generated by the hybrid model. The estimated plays were then used to calculate the root mean square error on the testing set in each fold. The mean root mean square error was then calculated.

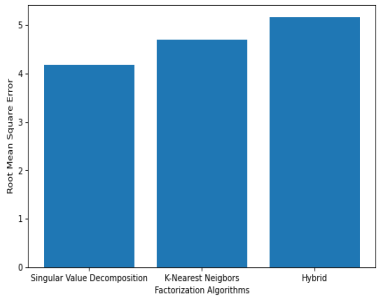


Figure 8: Baseline comparison with the Hybrid model

7.3.2 Comparative Analysis. Figure 8 shows the comparison of the hybrid model's root mean square error with the root mean square error of the baseline approaches. The root mean square error of the hybrid model is 5.16, which is larger than the root mean square error of both the baseline approaches. This makes sense because the hybrid model is penalized with respect to the user dataset for input users with a short user history. Thus, the root mean square error is expected to be higher than the root mean square error of the collaborative filtering baseline approaches. However, the hybrid model solves the problem of the cold start, which is useful, when a music streaming platform is constantly adding new users or is relatively new. Using the collaborative filtering model is preferable to using the content-based model because it uses user data to make recommendations. However, these recommendations can become inaccurate for users with short histories or impossible with new users. Thus, the hybrid model mitigates the cold start problem at a cost of root mean square error.

8 CONCLUSION

Through this project, we built a hybrid recommender engine that combines the strengths of content-based recommender engines and collaborative-filtering based recommender engines. One main weakness of collaborative filtering is cold start where we cannot make recommendations to users whom we have very little data on. This can be addressed through content-based filtering since these engines would not need data on other users. On the other hand, content-based filtering methods cannot recommend items outside of a user's interest whereas collaborative-filtering can by using other users' similar interests. The developed hybrid model mitigated the problem of the cold start and harnessed the strength of the collaborative filtering approaches. However, it was evaluated to be having a worse root mean square error than the two baseline approaches. This suggests that a trade-off between solving the cold start problem and the accuracy of the hybrid model must be made. The novelty of this project comes from the fact that there are currently no hybrid recommendation engines for the Million Songs Dataset (only content-based or collaborative filtering).

9 CONTRIBUTION

All team members have contributed a similar amount of effort.

REFERENCES

[1] Raghuveer K Ananth G S. 2020. A Movie and Book Recommender System using Surprise Recommendation Kit. *International Conference on Innovative Advancement in Engineering and Technology (IAET-2020)* (2020).

[2] Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere. 2011. The Million Song Dataset. In *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011).*

[3] Jesús Bobadilla, Fernando Ortega, Antonio Hernando, and Abraham Gutiérrez. 2013. Recommender systems survey. *Knowledge-based systems* 46 (2013), 109–132.

[4] Jiajun Bu, Shulong Tan, Chun Chen, Can Wang, Hao Wu, Lijun Zhang, and Xiaofei He. 2010. Music Recommendation by Unified Hypergraph: Combining Social Media Information and Music Content. In *Proceedings of the 18th ACM International Conference on Multimedia.* Association for Computing Machinery, New York, NY, USA, 391–400.

[5] Shanshan Huang, Shuaiqiang Wang, Tie-Yan Liu, Jun Ma, Zhumin Chen, and Jari Veijalainen. 2015. Listwise Collaborative Filtering. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Santiago, Chile) *(SIGIR '15).* Association for Computing Machinery, New York, NY, USA, 343–352. https://doi.org/10.1145/2766462.2767693

[6] Nicolas Hug. 2015. *Welcome to Surprise' Documentation.* https://surprise.readthedocs.io/en/stable/index.html

[7] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.