# Music Recommender System Application

## Final Project Report


***The word cloud was generated by https://www.jasondavies.com/wordcloud/ using all the text from this report.**

**Kaiqi Du, Zeying Yu,** and **Cheng Yu**

DSC 478
Professor Velkoski
DePaul University
June 11, 2019

**Abstract**

Recommender systems are widely used in the modern society, such as entertainment, online shopping and financial services. Since the information boomed on the Internet, especially music, people always need music but they may be fed up with their current playlist in their smartphones and want to listen to something nice and new. In this final project, we built a smart and multi-functioned music recommender system to address such daily problem of people nowadays. We researched intensively on models and algorithms of recommender systems, and built our own application from scratch in Python using several open-source libraries. We used content-based, collaborative-based, and popularity-based models for recommender systems, and incorporated various algorithms including cosine similarity and k-nearest neighbors into those models. We selected a pair of datasets that was based on real-world users and their ratings of songs to execute and evaluate our models of the application. It turned out that our music recommender system application was not only highly-interactive and user-friendly that could generate different output results of songs based on user's needs and preferences but also had fairly high accuracy based on RMSE evaluation. We compared the RMSE values of the content-based and collaborative-based models, and found that collaborative-based models had a lower RMSE, so that users can trust our application, especially the most sophisticated and smart collaborative-based functions. In our future work, we could try more approaches to evaluate our application and further optimize our models and algorithms, and we could also construct a more comprehensive and up-to-date song dataset to cater to Generation Z today!

**Introduction**

Many people on this planet love music, and some of them are even obsessed with it. Our team of three are definitely parts of the music obsessors, because we even made a music recommender system as our final project to express not only our expertise in computer programming and machine learning but also our passion for music, of course, in a geeky way. We found a daily life problem that we all had the times that we would like to listen to something nice and new, but we could not figure out a fast and efficient way to get some recommended music what we really wanted. Music radios and Billboards provide music that are either cherry picked by DJs or voted by the public in general, but they may not be customized to us with unique tastes of music. Also, our families and friends may be like us enough in many aspects, but

we may still have different playlists of music in our smartphones. That's why we were highly excited to build our own application of a music recommender system to provide educated guess of our preferred songs.

## Background

Recommender systems are nowadays widely used in the business world on the Internet, such as Netflix is using recommender systems for movies, and Amazon is using recommender systems for products. The general idea of recommender systems is to predict the future preferences of a set of items for a user, and recommend the top items. In particular, music recommender system is a machine learning system which learns from a user's listening history and then recommends the user songs that he or she may prefer in the future. From learning to recommendation, there is a magic black box in between. What we were doing in our final project was to construct our own black box. The black box of our music recommender system contained different models to implement our various tasks of music recommendations. They were content-based, collaborative-based, and popularity-based models of recommender systems, which were the widely-used models in both research and industry. In order to implement those models, we used cosine similarity, k-nearest neighbors and other machine learning algorithms. We also used a pair of real-world datasets to execute and evaluate the models of our music recommendation system.

## Methods and Tools

### Programming Languages

We built our music recommender system using Python 3.7.3. Noticeably, the application must be executed in Python 3 environment.

### Libraries

We imported various open-source libraries in this application, comprising NumPy, SciPy and Scikit-learn.

### Datasets

We used two corresponding datasets from the Million Song Dataset. The Million Song Dataset was a bundle of music datasets that was open to the public for free. All the datasets were based on real-world data from one of the largest online music websites, Last.fm, and its users. We selected the datasets from the Million Song Dataset because it was real, comprehensive and research-friendly. Among all the datasets that it provided, we picked a smaller version with 10000 selected song IDs, their corresponding listener IDs and their ratings of the songs based on listening times (10000.txt) (**Figure 1**). The smaller version, although lack of some users and songs, was good enough for our recommender system and could enhance the speed of performance of the application execution.

```
b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOAKIMP12A8C130995	1
b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOBBMDR12A8C13253B	2
b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOBXHDL12A81C204C0	1
b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOBYHAJ12A6701BF1D	1
b80344d063b5ccb3212f76538f3d9e43d87dca9e	SODACBL12A8C13C273	1
b80344d063b5ccb3212f76538f3d9e43d87dca9e	SODDNQT12A6D4F5F7E	5
b80344d063b5ccb3212f76538f3d9e43d87dca9e	SODXRTY12AB0180F3B	1
b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOFGUAY12AB017B0A8	1
b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOFRQTD12A81C233C0	1
b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOHQWYZ12A6D4FA701	1
b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOIYTOA12A6D4F9A23	1
b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOIZAZL12A6701C53B	5
b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOJNNUA12A8AE48C7A	1
```

**Figure 2**. Small portion of 10000.txt dataset, comprising user_id, song_id and ratings (from left column to right).

In addition, since the importance of 10000.txt was to provide ratings of each song by its listeners, we also needed another more comprehensive dataset for song data (song_data.csv), which comprised not only song IDs, but also their corresponding real song names, their artists and other information about the songs (**Figure 2**) (also see **Appendix I** for Data Analysis and Detailed Descriptions of the Datasets).

```
In [4]: song_data.head()
```

Out[4]:

| | song_id | title | release | artist_name | year |
|---|---|---|---|---|---|
| 0 | SOQMMHC12AB0180CB8 | Silent Night | Monster Ballads X-Mas | Faster Pussy cat | 2003 |
| 1 | SOVFVAK12A8C1350D9 | Tanssi vaan | Karkuteillä | Karkkiautomaatti | 1995 |
| 2 | SOGTUKN12AB017F4F1 | No One Could Ever | Butter | Hudson Mohawke | 2006 |
| 3 | SOBNYVR12A8C13558C | Si Vos Querés | De Culo | Yerba Brava | 2003 |
| 4 | SOHSBXH12A8C13B0DF | Tangle Of Aspens | Rene Ablaze Presents Winter Sessions | Der Mystic | 0 |

**Figure 2**. Head of song_data.csv, comprising song_id, title, release album, artitst_name and year.

In combination of this pair of datasets, each dataset shared the feature of song IDs, which connected both datasets, so that we could get both song ratings/users and the information of their songs at once from both datasets altogether.

This pair of datasets was used to execute the music recommender system, and made our tasks of music recommendation available to real-world users. Furthermore, the datasets were also used for evaluation of our system to demonstrate its correctness, functionality and accuracy.

Models and Algorithms

**Content-based Model**. Content-based recommendation, also called item-based model, was an approach of recommender system to match users to the content or items they have liked before. In this model, the attributes of the users and the items are important. In particular of our music recommender system, the users and the songs were essential to find out the most similar songs. Unlike many literature indicated, we did not rely on apparent music features, such as music genres, for the content-based recommendation. Given the nature of datasets, we found that each song had a number of users who gave high ratings, and we simply call it as "list of fans". Obviously among different songs, their "lists of fans" may or may not share the same users, and if so, the number of shared users may be little or large. Based on the differences, we assumed that if two "lists of fans" of two songs shared more common users, then the two songs would be more similar to each other.

There were various methods to address similarity between contents, and we used cosine similarity to calculate the distance between the selected song and each of the other songs in the song dataset. The cosine similarity was computed between the user vector and the item vector (**Figure 3**). We computed the cosine similarity between two "lists of fans" of two songs, and the songs that were most similar are the ones that the system would be highly recommended. (see **Appendix II** - MRS.py for detailed Python code of **def centered_cosine** that used cosine similarity in content-based model).

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}},$$

**Figure 3**. The formula of cosine similarity.

**Collaborative Filtering-based Model**. Collaborative filtering-based recommendation, also called user-based model, was another important approach of recommender system. The rational of this model was that users with similar characteristics shared similar tastes. For example, if I like tennis and basketball as my favorite sports, Professor Velkoski also likes tennis and basketball as your favorite sports, and another random person likes baseball and soccer as his or her sports, then we could tell that Professor Velkoski and I shared similar preferences, then the other favorite sports of mine, such as taekwondo, that Professor Velkoski may have never tried may be still interesting to Professor Velkoski. In particular of our music recommender system, we used the same idea. By doing that, we used the k-nearest neighbor to implement such collaborative filtering-based model. K-nearest neighbor algorithms (kNN) was to find clusters of similar users based on similar song ratings, and then make predictions using the average ratings of top-k nearest neighbors. (see **Appendix II** - MRS.py for detailed Python code of **def knn** that used k-nearest neighbor in collaborative filtering-based model).

**Popularity-based Model**. Popularity-based recommendation was the most straightforward method. It worked with the trend, and basically used the items which were in trend by the rankings or ratings. For example, we did not know what books to buy on Amazon, so we could simply see the rankings of the best sellers to pick some books, and such rankings would be generated from popularity-based models. In particular of our music recommender system, a user's song list based on the user's ratings would use the popularity-based model. We made a couple of simpler functions based on this model in order to provide more options for users of the music recommender system application (see **Appendix II** - MRS.py for detailed Python code of the rest of functions that used popularity-based model).

**Results**

We spent a lot of time on building and testing the highly smart and multi-functioned music recommender system by doing extensive research and tedious programming, so our most significant result was surely the application itself! (**Figure 4**). From the welcome page of the application, users could interact with the system and choose whatever functions they wanted to play with. "1. Find similar music" was based on content-based recommendation; The

collaborative filtering-based recommendation spanned from Option 2 to 6; and Option 7 to 10 were all based on popularity-based recommendation.

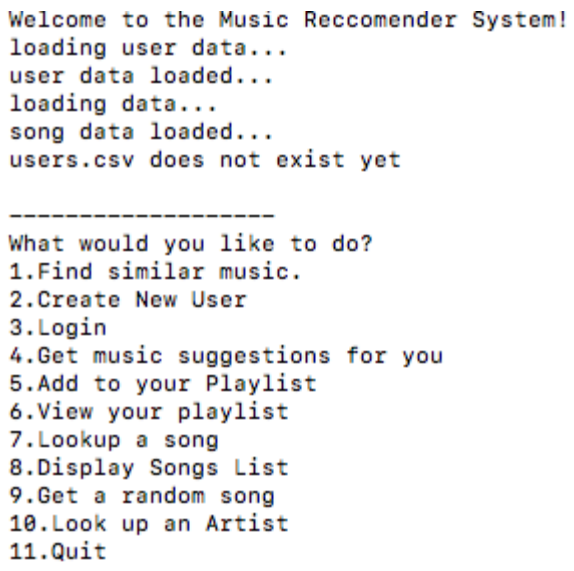


**Figure 4**. Welcome page of the application.

The sample results of recommended songs through different recommender system models were provided in the following corresponding sections.

Content-based Recommended Songs:

By selecting "1. Find similar music", the application asks "*Please give the id of the song you would find similar music to:*", then referring to the song id from the dataset, for example, we could find our favorite song *The Old Saloon*'s ID in the dataset was *SOLLGNU12AF72A4D4F*. After we typed in the song ID, it will ask how many similar songs do we want. We could type any reasonable numbers to see the number of songs that are most similar to *The Old Saloon*! (**Figure 5**).

```
1
Please give the id of the song you would to find similar music to: SOLLGNU12AF72A4D4F
Please enter the number of similar songs you would like: 10

********Similar Songs*********
Song ID       Song Title
SOHQWYZ12A6D4FA701                  "Heaven's gonna burn your eyes"
SODDNQT12A6D4F5F7E                  "Apuesta Por El Rock 'N' Roll"
SOBYHAJ12A6701BF1D                  Constellations
SOAKIMP12A8C130995                  "The Cove"
SOFGUAY12AB017B0A8                  "Stacked Actors"
SOBBMDR12A8C13253B                  "Entre Dos Aguas"
SODXRTY12AB0180F3B                  "Paper Gangsta"
SOBXHDL12A81C204C0                  Stronger
SODACBL12A8C13C273                  "Learn To Fly"
SOFRQTD12A81C233C0                  "Sehr kosmisch"
```

**Figure 5**. Content-based recommendation output results page.

Collaborative Filtering-based Recommended Songs:

Collaborative filtering-based recommendation has the most smart and interactive functions in this application.

By selecting "2. Create New User", any users of our application could create a user account, login through "3. Login", and then in "5. Add to your Playlist" the user could either choose like or dislike several randomly popped up songs by the application, and the songs that the user liked will automatically added to the user's playlist in the application for his or her future reference. In addition, the user can select "4. Get music suggestions for you" to see 10 suggested songs by the collaborative filtering-based model, and the user can also add those songs to his or her own playlist in the application. Next time when the user login back to this application, the user could even see his or her own playlist by selecting "6. View your playlist." (see **Appendix III** - complete sample runs with descriptions and **Demo Video** on https://youtu.be/ZyRJpn-L_PI).

Popularity-based Recommended Songs:

The following functions will generate different results by different approaches of popularity-based recommendation.

In "7. Lookup a song", the user can enter a song ID and then the application will give a song list by all the users who liked the song that you selected, and the songs in the list will be ranked by ratings; moreover, in "8. Display Songs List", this option will give a list of songs

according to total ratings by the users; additionally, in "9. Get a random song", a randomly selected song from the song dataset will pop up, and if you happen to like this song, you could choose to print all the songs in the dataset by the song's artist, printing in the order of high to low ratings; furthermore, in "10. Look-up an Artist", the user can give a name of an artist, for example "Beyonce", if the artist's name is in the dataset (of course Beyonce is), then the application will print all the songs by Beyonce in the dataset from high to low ratings (see **Appendix III** - complete sample runs with descriptions and **Demo Video**).

**Discussion and Conclusion**

Evaluation

If our recommended songs have a high similarity to the user's preferences, it proves that our music recommender system works well; then, the users will have a higher chance of continuing to use our application. However, if our recommended songs have a low similarity to each user's musical tastes, users are not likely to be satisfied with the application. In order to evaluate our music recommender system, we examined users' musical preferences and affinity to our recommended songs using an evaluation metric, root mean squared error (RMSE). RMSE was used to measure the differences between the values predicted by a model and the values observed (**Figure 6**). A lower RMSE value means that the method was doing a better job to match recommended songs to users' preferences.

**Root mean squared error (RMSE):**

$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(d_i - \hat{d}_i)^2}$$

**Figure 6**. RMSE formula. $\hat{d}_i$ is the predicted ratings by users, and $d_i$ is the observed ratings.

We randomly split the data from 10000.txt into the test set (20%) and the training set (80%). We then applied the two predictive content-based and collaborative filtering-based models to the training sets. Next, we made predictions on the testing data and computed the values of RMSE to see the average difference between our predicted ratings and the observed

ratings (**Figure 7**). We compared the RMSE values of content-based recommendation and collaborative filtering-based recommendation.

| | Content-based model using Cosine | Collaborative filtering-based model using kNN |
|---|---|---|
| RMSE | 0.477104 | 0.296031 |

**Figure 7**. Comparison of RMSE values of algorithms in content-based and collaborative filtering-based recommendation.

The evaluation showed that both models had relatively low values RMSE, indicating relative high accuracy. In particular, the collaborative filtering-based model has achieved the even lower RMSE value, so this model had higher accuracy among all the models in the applications. We therefore concluded that the collaborative filtering-based model was the first choice method for our music recommender system application.

Future Works

Our music recommender system application still has room to polish and improve furthermore. For our future work, we could use different approaches of evaluation metrics for recommender systems to perform a more well-rounded evaluation of our application, such as mean absolute error (MAE), mean average precision (MAP) and normalized discounted cumulative gain (NDCG). Then, we might compare and contrast different evaluation metrics to find out where and how we could better optimize our application.

In addition, we could try different datasets, not only limited to the selected datasets in this application. We could execute the application that could be applied to various kinds of music datasets and then evaluate the models in the application using different datasets. We believe that the results would be more interesting. Our recommendation system did not use the feature, year, instead using user and song similarities. We could further our recommendation algorithm by considering year of release. This may or may not improve user satisfaction, possibly depending on the age range of the users in the test set. If the age range of the users is large, the release year may prove to be a useful feature in recommending songs. Therefore, we could ask users to input their ages when creating an account to help us give better song recommendations. Last but not

least, we may need more songs in our music datasets, especially the up-to-date ones, in order to attract young kids who are picky and capricious.

**Acknowledgment**

Many thanks to DSC 478 course and Professor Velkoski for providing this valuable opportunity of final project, so that our team could utilize our computer programming skills on practical machine learning problem through our ambitious application of the music recommender system. We truly learned not only a lot of machine learning applications in general in the classroom but also dug intensively into a specific area through research on this final project.

**References**

Deng H. (2019) Recommender Systems in Practice. Towards Data Science. https://towardsdatascience.com/recommender-systems-in-practice-cef9033bb23a

Doshi N. (2018) Recommendation Systems — Models and Evaluation. Towards Data Science. https://towardsdatascience.com/recommendation-systems-models-and-evaluation-84944a84fb8e

Jain A. (2016) Quick Guide to Build a Recommendation Engine in Python. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2016/06/quick-guide-build-recommendation-engine-python/

Le E. (2017) How to build a simple song recommender system. Towards Data Science. https://towardsdatascience.com/how-to-build-a-simple-song-recommender-296fcbc8c85

Leskovec J., Rajaraman, A., Ullman, J.A. (2014) Mining of Massive Datasets. Chap. 9 Recommendation Systems. Cambridge University Press, Cambridge, UK. http://infolab.stanford.edu/~ullman/mmds/ch9.pdf

Li S. (2017) Building A Book Recommender System – The Basics, kNN and Matrix Factorization. Datascienceplus.com. https://datascienceplus.com/building-a-book-recommender-system-the-basics-knn-and-matrix-factorization/

Longo C. (2018) Evaluation Metrics for Recommender Systems. Towards Data Science. https://towardsdatascience.com/evaluation-metrics-for-recommender-systems-df56c6611093

Mall R. (2019) Recommender System. Towards Data Science. https://towardsdatascience.com/recommender-system-a1e4595fc0f0

Mayeesha T. (2018) Recommending Animes Using Nearest Neighbors. Medium.com. https://medium.com/learning-machine-learning/recommending-animes-using-nearest-neighbors-61320a1a5934

Polamuri S. (2015) COLLABORATIVE FILTERING RECOMMENDATION ENGINE IMPLEMENTATION IN PYTHON. Dataaspirant. http://dataaspirant.com/2015/05/25/collaborative-filtering-recommendation-engine-implementation-in-python/

Rehorek T. (2016) Evaluating Recommender Systems: Choosing the best one for your business. Medium.com. https://medium.com/recombee-blog/evaluating-recommender-systems-choosing-the-best-one-for-your-business-c688ab781a35

Schedl M., Knees P., McFee B., Bogdanov D., Kaminskas M. (2015) Music Recommender Systems. In: Ricci F., Rokach L., Shapira B. (eds) Recommender Systems Handbook. Springer, Boston, MA

Shakirova E. (2017) Collaborative filtering for music recommender system. 2017 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus).

Sharma P. (2018) Comprehensive Guide to build a Recommendation Engine from scratch (in Python). Analytics Vidhya. https://www.analyticsvidhya.com/blog/2018/06/comprehensive-guide-recommendation-engine-python/

Vyash H. (2018) Code Your Own Popularity Based Recommendation System WITHOUT a Library in Python. Hackernoon.com. https://hackernoon.com/popularity-based-song-recommendation-system-without-any-library-in-python-12a4fbfd825e