# T-RECSYS+: An Improved Music Recommendation System

Anonymous Author(s)

## ABSTRACT

A recommendation system is a type of filtering system that predicts a user's preferences for a specific item. Its purpose is to suggest items that the user might find appealing. In our research, we build a music recommendation system to make prediction of users' listening preference. Our system extends the previous T-RECSYS algorithm which uses a hybrid of content-based and collaborative filtering as input to a deep learning classification model. We enhance the performance of this algorithm by incorporating the latest Spotify API, which provides access to 11 music features including danceability, liveness, tempo and so forth. Additionally, we leverage more advanced deep learning models to achieve a higher level of precision and accuracy in our recommendations. In detail, we promote the precision scores from the original 88% to the current over 95%. Our code is available at https://github.com/zcj0125/T-RECSYS-An-Improved-Music-Recommendation-System

## CCS CONCEPTS

• **Information systems** → **Collaborative filtering**; • **Human-centered computing** → **Collaborative and social computing systems and tools**.

## KEYWORDS

music recommendation system, deep learning, collaborative filtering, Spotify API

**ACM Reference Format:**

Anonymous Author(s). 2023. T-RECSYS+: An Improved Music Recommendation System. In *Proceedings of ACM Conference (Conference'23).* ACM, Taipei, TaiwanA, 5 pages. https://doi.org/XXXXXXX.XXXXXXX

## 1 INTRODUCTION

Music Recommendation Systems are becoming an increasingly important part of our daily lives. In today's fast-paced world, where we have instant access to millions of songs, the idea of discovering new music that aligns with our preferences can seem daunting. Fortunately, music recommendation systems provide a solution to this challenge.

We want to create a more accurate music recommendation model, and we noticed a paper published in 2019, which created a novel algorithm for recommending music to users. The paper introduces a new system called Tunes Recommendation System (T-RECSYS). It scores each song in a database according to user preference by utilizing a learned hybridization of content-based and collaborative filtering, returning the top-k scoring songs [3].

After studying this paper and the algorithmic idea in depth, we found that this paper may leave room for further innovation. We intend to improve this model in the following aspects to make the recommendation system more accurate.

- To enable the content-based filtering in T-RECSYS, the original paper considers six categories of metadata: genre, artist type, artist era, mood, tempo, and release year. Considering that three of the categorical features (artist era, tempo, and release year) are unorderable and the deep learning model only accepts the numeric values input vector, one-hot encoding is necessary. This will increase the number of dimensions in the input data, the memory usage and the computational cost. In our experiment, we utilize the Spotify API [10] to get 11 numeric features, and each feature value can be used to input the deep learning model directly. Also, some features are more accurate and more intuitive. (For instance, in the tempo part, we have the specific beats per minute, instead of 'slow', 'medium' and 'fast').
- The original paper gave the description of the T-RECSYS and the general implementation idea. In our paper, we clearly illustrate the whole data processing procedures to make sure the readers can more easily understand the algorithm and the algorithmic process in practice. In addition, with the help of the Uniform Resource Identifier (URI) of each track, we can easily find the frequency of two songs appearing together in a dataset of playlists, and calculate the similarity of two song pairs (collaborative filtering implementation). Adding the similarity features into the input layer, our system can easily know whether one song is part of the playlist and provide more accurate predictions.

## 2 RELATED RESEARCH

Early in 2012, a survey of recommendation systems was done by Song et al. They tried to survey a general framework and state-of-art approaches in recommending music [9]. In this paper, they identify three key components in music recommendation - user modelling, item profiling, and match algorithms. At that time, the question is mainly about how to organize and manage the millions of music titles available, so in the first step, they modeled the user profile and the music item profile. Then, some state-of-art approaches in music recommendation were introduced including Metadata Information Retrieval, Collaborative Filtering, Content-based Music Information Retrieval, Emotion-based Model and Context-based Information Retrieval.

In 2017, Fang et al. note that "Music has been shown to have a motivational effect that can encourage people to exercise more strenuously or for longer periods of time" [2]. To address the problem that much audio is likely to be a poor match to any specific user, they propose to use a User Profiling (UP) step. They chose the Million Song Dataset (MSD) as the database and applied questionnaire



*Conference'23, July 2023, Taipei, Taiwan*



ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/XXXXXXX.XXXXXXX

to prune the initial 384,500 songs to 1,000 songs the user is more likely to enjoy in order to make the recommendation problem more tractable [2]. After the questionnaires were filled in, they employed clustering technique to divide users into different groups based on their listening taste. Finally, they obtained the group's profile by averaging and rounding the user's ratings for each of the features listed by the users. Although the paper compared the group-based profiles results to the individualized profile results and proved its validity, it may be too subjective if their group-based results were completely derived from the questionnaires. The outcome of the survey could be inaccurate and also, the result of exercise music recommendation is not suitable for everyone.

In 2021, Fathollahi et al. proposed a music recommendation system based on a similarity system. For this measurement, they considered cosine similarity and Euclidean distances between feature vectors [8]. Innovatively, they found they achieved a better accuracy by combining acoustic and visual features. To analyze the acoustic features, they utilized chroma, a feature which groups frequencies into the associated chromatic class independent of octave. Chroma relationships are useful for analyzing music by key and identifying harmonic relationships. Also, they introduced Mel-spectrogram to analyze the power spectral density of a sound. In the experiment, they designed two models. The first model had only a single input feature while the second model had three features: Mel-spectrogram, chroma stft, and spectral contrast (a powerful feature for classification and similarity detection). By feeding different inputs into the input layer, they could prove the system valid by expecting a better result on the output of the second model. However, according to the results, it could be concluded that databases containing short-time music pieces have not yet achieved high accuracy for tasks like similarity measurement and classification [8].

In 2019, the T-RECSYS was introduced by Fessahaye et al. [3]. They combined both collaborative filtering and content-based filtering to build a classification deep learning model. By setting different levels of discrimination thresholds, their system was able to maintain fairly high precision in each trial. They also calculated the sample standard deviation between all the trials and proved the system stable. However, as we describe as above, the system still has large room to be improved. Our main task is to recreate the T-RECSYS, make the whole process more clear and get a higher precision in 50% threshold by default.

# 3 DATA

## 3.1 Dataset

We employ the Spotify Million Playlist Challenge Dataset as our research dataset. The dataset contains 1,000,000 playlists, including playlist titles and track titles, created by users on the Spotify platform between January 2010 and October 2017 [1]. In addition, this dataset also provides detailed descriptions of each playlist and the URI information of the corresponding tracks. With the help of specific API platforms, we can obtain elaborate track features as data input to the machine learning system.

## 3.2 API Platform

In our experimental implementation, we accessed the Spotify API by the way of the spotipy python library [5]. The Spotify API is a great public tool, allowing the use of Spotify's wealth of data on music to build many kinds of systems [10]. By inputting the URI information of each track into the Spotify API, we were able to retrieve 11 metadata attributes such as danceability, energy, key, loudness, mode, speechiness, acousticness, instrumentalness, liveness, valence and tempo. Given that all metadata values are numerical and ordinal, we utilized them as the features of each track and fed them directly into our deep learning system as the input vectors.

Here is an example of one set of 11 feature values:

```
'danceability': 0.732,
'energy': 0.75,
'key': 11,
'loudness': -6.366,
'mode': 0,
'speechiness': 0.231,
'acousticness': 0.00264,
'instrumentalness': 0,
'liveness': 0.109,
'valence': 0.401,
'tempo': 155.096,
```

In our research, extracting features from the contained songs proves to be valuable, as it enables us to gain a deeper understanding of the interrelationships between songs. This allows us to effectively perform clustering and construct a personalized music recommendation system[10].

# 4 ALGORITHM

## 4.1 Overview

Our algorithm is inspired by T-RECSYS[3]. It can be outlined as follows:

- For every playlist with over 10 tracks, we randomly select 9 songs (since these tracks are collected in one playlist, there is a high probability that these tracks will be loved by users) from the playlist and an additional 10th song (either from the same playlist, representing a *positive pair*, or randomly chosen from other playlists, representing a *negative pair*) that the user may or may not like.
- The features of the 10 tracks are then inputted into our deep learning model, which returns a probability value to predict the probability that the user would enjoy the 10th song based on the first 9.

Through the training process, the network learns to analyze the user's musical preferences based on their existing playlists and continuously recommends new songs that the user is likely to enjoy.

## 4.2 Collaborative Filtering

Collaborative filtering systems predict a person's affinity for items or information by connecting that person's recorded interests with the recorded interests of a community of people and sharing ratings between like-minded persons [4]. In our implementation, the collaborative filtering measures the similarity between the first 9

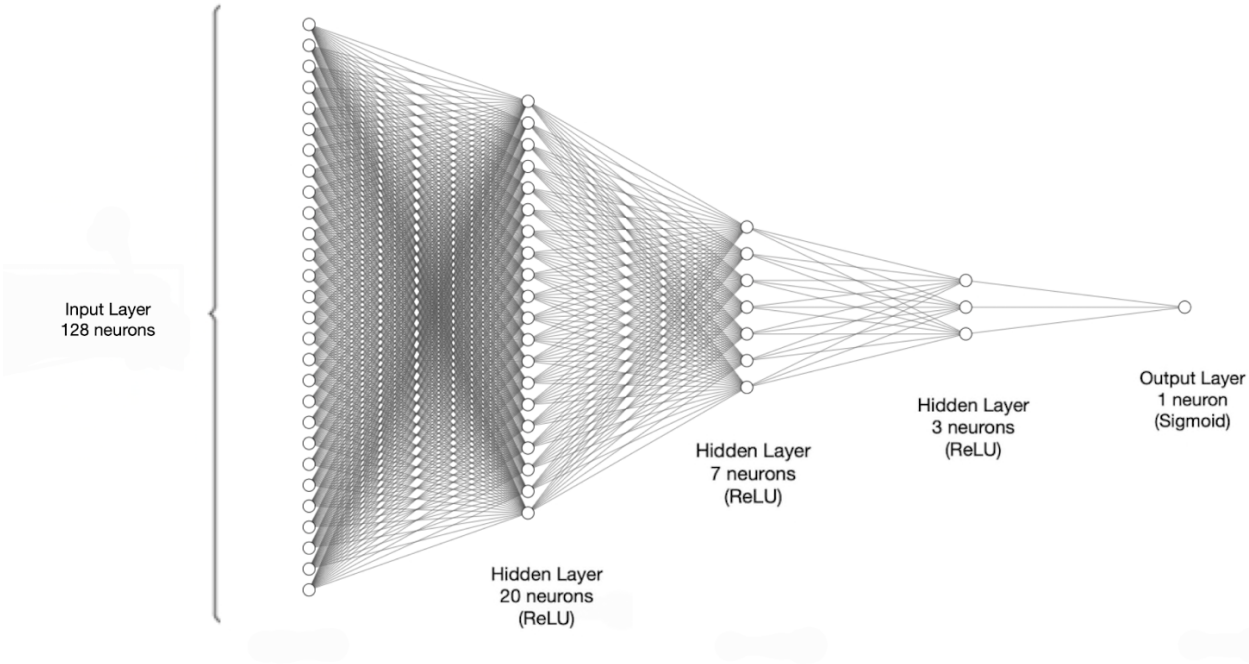


**Figure 1: T-RECSYS+ neural network. The input to the network is of size 128 = (10 songs in one sample) ×(11 features per song) + 9 Sorenson indices + 9 volume measures.**

songs and the tenth song to determine whether the user would love the tenth one or not. In the Spotify Million Playlist Challenge Dataset, The URI of the song is the only credential of each song (because song renaming is very common). Therefore, the similarity can be defined as the frequency with which the two songs' URI appear together in a dataset of playlists[3].

We calculate the Sorenson index for each pair:

$$\frac{2\,|X \cap Y|}{|X| + |Y|}$$

where X and Y represent the set of playlists that the first song and second song in the pair appear in respectively [3].

The Sorenson index is a vital feature which determines whether the tenth song is included in this playlist. By adding this index into our input data pairs, it can significantly improve the operational efficiency and prediction accuracy of the whole system.

## 4.3 Data Processing

According to the T-RECSYS described above, our full dataset consists of positive pairs and negative pairs. The positive pairs represent all the ten songs enjoyed by the user, while the negative pairs represent that the first 9 songs enjoyed by the user and a random 10th song. In positive pairs, we randomly sample 10 tracks from their 11 features in one playlist, which means we can obtain a (10, 11) size matrix in each playlist. The negative pairs are similar to the positive pairs, but the tenth song in each matrix is replaced with a randomly selected song from another playlist that is not found in the original playlist. After that, we can concatenate the 11 features from the 10 songs with the Sorenson index features and volume features to get a complete sample to feed forward in the network.

It should be noted that if the variance of a certain feature (like tempo) is several orders of magnitude larger than that of other features, it will dominate in the learning algorithm, hindering the learner from learning from other features, thus reducing accuracy. In order to make the features the standard normally distributed data (mean of 0 and variance of 1), we utilize the StandardScaler class in the sklearn to scale our data [7].

## 4.4 Deep Learning Model

After meticulously collecting and processing the dataset, it can be utilized as input for a deep neural network to train the model. Subsequently, thorough testing will be performed to ensure the model's ability to provide precise music recommendations.

Some hyperparameters used in the process are described in Table 1.

We utilize PyTorch as our deep learning framework [6]. T-RECSYS+ has four full connection layers with ReLU activation, three of which use dropout during training, a final sigmoid activation function. The number of nodes in the full connection layer is in order: 128 -> 20 -> 7 -> 3 -> 1. The dropout rate in each layer is 0.2.

In our implementation, 100,000 playlists were used. Among them, 96,428 playlists contain over 10 songs, which means that we have 96,428 postive pairs and 96,428 negative pairs. We randomly spilt our full dataset into 80% training set and 20% testing set. We use

**Table 1: Hyperparameters for Training T-RECSYS+**

| | |
|---|---|
| batch size | 16 |
| learning rate | 0.001 |
| dropout ratio | 0.2 |
| number of epoch | 20 |
| training threshold | 0.5 |
| testing threshold | 0.5 |

cross-entropy loss for the classification task, and compute an average accuracy performance metric.

## 5 RESULTS

We evaluate the performance of the machine learning model by calculating the test accuracy of the model on the testing set. The input data is fed through the model in a forward direction and generates the output. We compare the each output with the testing threshold and get a binary prediction value. In our implementation. we set the testing threshold as 0.5, which means that it will be predicted as a positive recommendation if the confidence score exceeds 50%. After the operation on the testing set, the average accuracy can be defined as the ratio of the number of correctly predicted samples and the total number of tested samples. As depicted in the Figure 2, we have found that our average accuracy values consistently remain above 97%, even in repeated trials with different random data samples. Hence, we can consider the system to be very efficient, reaching and sustaining excellent performance on the test set after only a couple epochs of training. We think this is largely due to the use of the Sorenson index in training; when training without Sorenson index, the test data reaches only approximately 70% performance, but with the Sorenson index included, performance is consistently above 97%.

Our final result can be shown as a confusion matrix. The matrix is defined as

$$\begin{bmatrix} TN & FP \\ FN & TP \end{bmatrix}$$

where the TP indicates the true positive, the FN indicates the false negative, the FP indicates the false positive and the FP indicates the false positive.

Here is the confusion matrix of our 4th (last) trial, after finding the preceding 3 trials to be consistently performing:

$$\begin{bmatrix} 18370 & 956 \\ 60 & 19186 \end{bmatrix}$$

To finally evaluate our model and analyze the result, we use the precision metric:

$$precision = \frac{TP}{TP + FP}$$

The precision value can be regarded as the proportion of actually positive samples in the positive samples predicted by the classifier. In music recommendation system, this value can be described as how often a recommended song is actually enjoyed by the user. In our implementation, our precision value can reach above 95.3%, higher than 88% in the original paper. This result may suggest that in addition to strong predictive capabilities on playlist matches, an extended version of this system could give fairly accurate recommendation to the user by analyzing their individual listening tracks.

## 6 CONCLUDING REMARKS

In this experiment, we designed a deep learning model to extract million music features for the improvement of the recommendation system, T-RECSYS, which we refer to as T-RECSYS+. We introduce a new API platform and facilitate the neural network model and finally get remarkable results. It is also notable that there are some possible avenues for the future research. One of them is the data processing issue. When sampling the 10th song from other playlists into the negative pairs, there is no way to know whether the user would or would not like the 10th song. There is possibility that the user would fairly enjoy this song and even the song just exists in the original playlist. Also, considering about the spatial complexity, we choose 10 as our data size. For further research, more songs could be extracted from one playlist to obtain higher precision.

## REFERENCES


[1] Ching-Wei Chen, Paul Lamere, Markus Schedl, and Hamed Zamani. 2018. Recsys Challenge 2018: Automatic Music Playlist Continuation. In *Proceedings of the 12th ACM Conference on Recommender Systems* (Vancouver, British Columbia, Canada) *(RecSys '18)*. Association for Computing Machinery, New York, NY, USA, 527–528. https://doi.org/10.1145/3240323.3240342

[2] Jiakun Fang, David Grunberg, Simon Lui, and Ye Wang. 2017. Development of a music recommendation system for motivating exercise. In *2017 International Conference on Orange Technologies (ICOT)*. 83–86. https://doi.org/10.1109/ICOT.2017.8336094

[3] Ferdos Fessahaye, Luis Perez, Tiffany Zhan, Raymond Zhang, Calais Fossier, Robyn Markarian, Carter Chiu, Justin Zhan, Laxmi Gewali, and Paul Oh. 2019. T-RECSYS: A Novel Music Recommendation System Using Deep Learning. In *2019 IEEE International Conference on Consumer Electronics (ICCE)*. 1–6. https://doi.org/10.1109/ICCE.2019.8662028

[4] Jonathan L. Herlocker, Joseph A. Konstan, and John Riedl. 2000. Explaining Collaborative Filtering Recommendations. In *Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work* (Philadelphia, Pennsylvania, USA) *(CSCW '00)*. Association for Computing Machinery, New York, NY, USA, 241–250. https://doi.org/10.1145/358916.358995

[5] Paul Lamere. 2023. spotify.py. *Spotipy* https://pypi.org/project/spotify/ (2023).

[6] Shen Li, Yanli Zhao, Rohan Varma, Omkar Salpekar, Pieter Noordhuis, Teng Li, Adam Paszke, Jeff Smith, Brian Vaughan, Pritam Damania, and Soumith Chintala. 2020. PyTorch Distributed: Experiences on Accelerating Data Parallel Training. https://doi.org/10.48550/ARXIV.2006.15704

[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.

[8] Mohamadreza Sheikh Fathollahi and Farbod Razzazi. 2021. Music similarity measurement and recommendation system using convolutional neural networks. *International Journal of Multimedia Information Retrieval* 10 (2021), 43–53.

[9] Yading Song, Simon Dixon, and Marcus Pearce. 2012. A survey of music recommendation systems and future perspectives. In *9th international symposium on computer music modeling and retrieval*, Vol. 4. Citeseer, 395–410.

[10] Spotify. 2023. Spotify for Developers API. *Spotify* https://developer.spotify.com/documentation/web-api/ (2023).

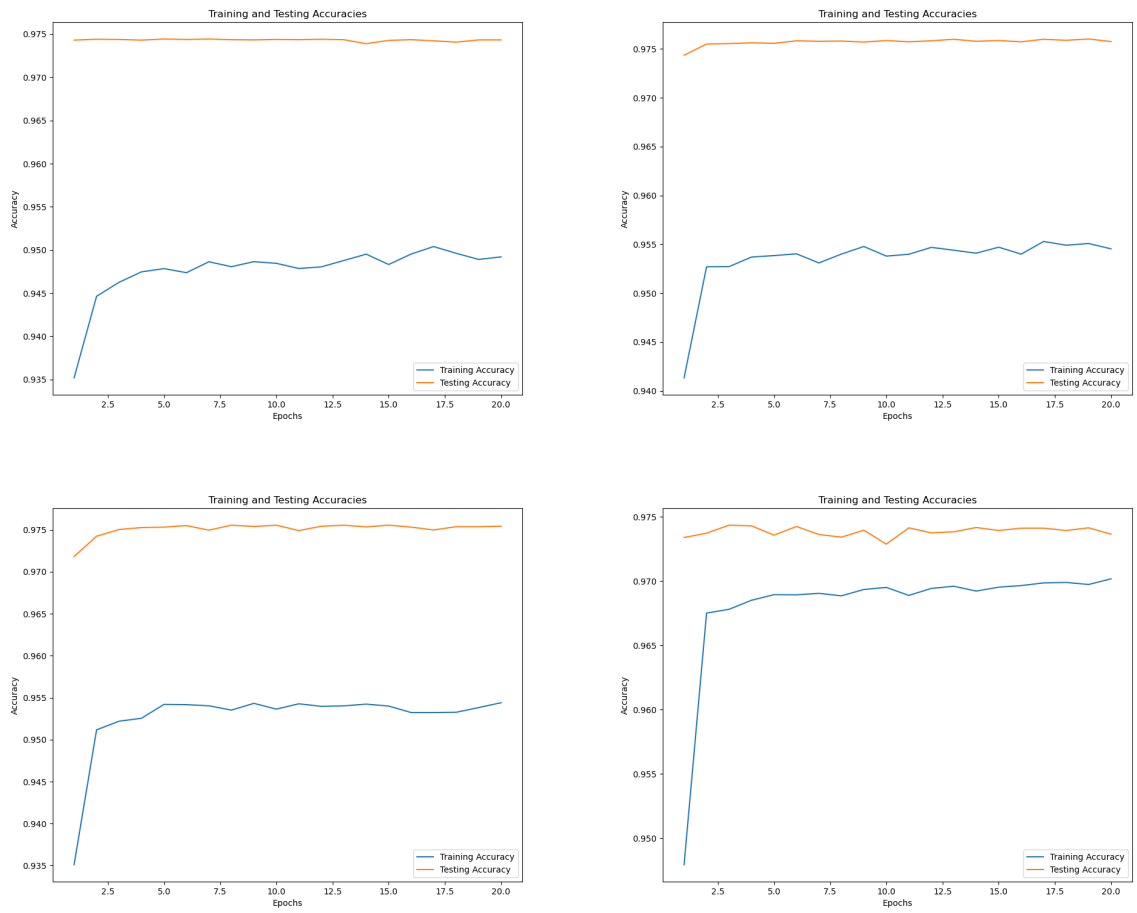


**Figure 2: Training and Testing Accuracies for 4 Independent Training Instances of T-RECSYS+**