Collaborative Filtering Using Probabilistic Matrix Factorization and a Hierarchical Bayesian Model

Nurudeen Sherif
Department of Computer Science
Nanjing University of Science and Tech
Jiangsu Province, China
sherif@njust.edu.cn

Gongxuan Zhang
Department of Computer Science
Nanjing University of Science and Tech
Jiangsu Province, China
gongxuan@nedu.cn

Project information

The implementation code is available on GitHub at:

https://github.com/sherifmavericks/CF

A manual is also available on GitHub at:

https://github.com/sherifmavericks/CF#user-content-collaborative-filtering-project-using-deep-learning

1 Introduction

Nowadays, web enabled online services like e-stores and streaming platforms which offer an increasing amount of different products have been integrating Recommender Systems (RS) into their platforms aiming to make a more effective use of information. Recommender Systems play an important role by assisting customers to find products according to their taste over those platforms. Different approaches have been developed to provide accurate recommendations to users, however there has been a lack of data and research on the scope of music recommending systems.

Conventional methods use ratings given to items by users as the only source of recommendation. The prediction accuracy drops significantly when the ratings are very sparse. Moreover, they cannot be used for recommending new products which have yet to receive rating information from users. This is known as the ”cold start” problem. Content information may be utilized, but loosely coupled methods process the auxiliary information once and then use it to provide features for the CF models (one-way interaction). Tightly coupled methods allow two-way interaction.

2 Aim

The aim of this group project is to develop and test a recommending system using two approaches: a pure probabilistic matrix factorization and a hierarchical Bayesian model using deep learning approach by employing the ratings (feedback) matrix and the related content information for learning stronger features. The recommender system will be evaluated in order to analyze how tightly coupled methods can automatically learn features from the auxiliary information and naturally balance the influence of the rating and the content information. The evaluation results of the implemented model will be compared with the baseline approach.

3 Background & Related work

3.1 Collaborative Topic Modeling for Recommending Scientific Articles

Wang and Blei [1] developed an algorithm to recommend scientific articles to users of an online community. The algorithm combines the merits of traditional collaborative filtering and probabilistic topic modeling. It provides an interpretable latent structure for users and items, and can form recommendations about both existing and newly published articles, therefore tackling the cold start problem. The algorithm was evaluated against a large subset of data from CiteULike, a bibliography sharing service, which showed that the algorithm proposed provides a more effective recommender system than traditional collaborative filtering.

3.2 A Hybrid GA-based Collaborative Filtering Model for Online Recommenders

Ho et. al [2] created a Hybrid Genetic Algorithm-based Collborative Filtering MOdel applied on the MovieLens dataset. The model implements collaborative and content-based methods separately and then combines their predictions. It incorporates some content-based characteristics into a collaborative approach and some collaborative characteristics into a content-based approach. It also constructs a general unifying model that incorporates both content-based and collaborative characteristics. The model tackles the cold start problem occurring during the early period of usage, by adding an additional ’’preference” component which consists of features such as the user’s preferred film type and language. This means that, during the early stage, the preference component will dominate the search function in the GA operation;

3.3 Music recommendations using cover images

A blog post by Eric Bernhardsson [3] described an experiment which considered cover images as a means for music recommendations. The first attempt at recommendation involved resizing all image to 16x16, converting them to grayscale, subtracting the mean and normalizing by the variance, which resulted in fairly poor recommendations. A further attempt involved using pyleargist, a wrapper around a C library, which takes an image and outputs an image descriptor, which improved the recommendation results. According to Eric, the experiment’s future work would involve learning a mapping from image space to collaborative filtering space and venturing into the realms of deep learning.

3.4 Combining Spotify and Twitter Data for Generating a Recent and Public Dataset for Music Recommendation

Pichl et al.’s research [4] is based on the #nowplaying dataset, the same dataset used in this project. Its aims were to implement and evaluate a pure collaborative filtering based recommender system. They found that for a high number of recommendations the recommender system was rather limited. During the evaluation phase, they discovered that further work was required. This included gathering of further context based information that is available such as time stamp or geolocation and implementing hybrid recommender system utilizing this additional context based information.

3.5 Recommending music on Spotify with deep learning

Sander Dieleman, a Spotify intern, wrote a blog post [5] discussing his work on music recommendations using deep learning at Spotify. Dieleman’s work focused on content-based music recommendation using convolutional neural networks. It involved training a model to predict the latent representations of songs that were obtained from a collaborative filtering model. This enabled the ability to predict the representation of a song in the collaborative filtering space, even if no usage data was available. If two songs are close together in this space, they are probably similar. If a song is close to a user, it is probably a good recommendation for that user, considering that they haven’t listened to it yet. If predicting the position of a song in this space from audio is achieved, recommending the song to the right audience without having to rely on historical usage data can also be achieved.

3.6 Collaborative Deep Learning for Recommender Systems

Wang et al. [6] propose a a hierarchical Bayesian model called collaborative deep learning (CDL). CDL jointly performs deep representation learning for the content information and collaborative filtering for the ratings (feedback) matrix by using an stacked deep autoencoder model which is tightly coupled with a Collaborative Topic Model (CTR) in order to automatically learn features and balance them with the influence of ratings. Extensive experiments on three real-world datasets from different domains showed that CDL can significantly perform better than the state of the art.

3.7 Types of Collaborative Filtering [2]

Content-based filtering

As the name suggests, content-based filtering focuses on content. In a music-related context, a song will be recommended to a user if in the past the user has listened to another that shares the same genre as the song being recommended. Content-based filtering selects items based on the correlation between the content of the items and user preferences. For example, current search engines are based on automatic analysis of the content of documents and the content of users query. This type of filtering also bring disadvantages as the system can only recommend items scoring highly against the user profile. The user is restricted to see the items similar to those already rated and new items will never be recommended due to the working on an individual user.

Item-based Collaborative Filtering

Item-based Collaborative Filtering focuses on the similarity between the currently active user and other users. It aims to find new items the active user has never seen before but are predicted they would be interested in, based on other users' interest in the items. For example, users A, B and C all like Item 1. Two of the users, A and B also like Item 3. Item 3 would then be recommended to user C based on user A and B's interest in the item. The similarity can either be measured by the same item which is known as item-based CF or by the same type of user, which is known as user- based CF. This method is quite common for current recommendation systems, which have been wildly used by MovieLens and Yahoo. Item-based techniques first analyze a user-item matrix to identify relationships between different items, and then use these relationships to indirectly compute recommendations for users.

User-based Collaborative Filtering

User-based Collaborative Filtering is similar to item-based CF with the difference being that it looks at similarities between users rather than items. For example, User A likes Item 1, 2 and 4. User B likes Item 1 and 2. Items 1 and 2 are common between User A and B. Therefore, Item 4 will be recommended to User B as User A and B are deemed as like-minded users.

4 Data

For this project, we use as main data sources the #nowplaying dataset [7] and the public Mu- sicBrainz[1] music encyclopedia for music metadata. The former is a social media-based dataset which combines Twitter and Spotify data in order to create a relationship indicating music listening behaviour of Twitter users. The authors attempted to find a match between the user's Twitter and Spotify user names. If the user names were identical, a match was found and that user's data was kept in the dataset. In the situation where a match was not found, the data was discarded in order to preserve data quality and accuracy. The dataset also provides Artist and Track IDs for the MusicBrainz database. Using these identifiers provided, we are able to extract further artist and track-related features which boosts recommendation accuracy.

[1]`https://musicbrainz.org/`

4.1 Dataset analysis

Table 1 presents number of tweets, users, tracks and artists in the #nowplaying dataset dump downloaded on 06/04/2016.

Table 1: Number of tweets, users, tracks and artists in the dataset as of 06/04/2016.

Tweets	Users	Artists	Tracks
88,745,248	5,497,005	151,440	772,485

Table 2 showcases the top 5 most popular sources used by the users in the #nowplaying dataset as of 06/04/2016.

Securenet Systems Radio Playlist Update 13,326,642

Table 2: Top 5 most popular sources as of 06/04/2016.

Source	Times used
Spotify	7,132,339
Web	4,139,372
SAM Broadcaster Song Info	3,439,207
BigUrl	3,170,726

Table 3 presents the top 5 most popular tracks listened to by the users in the #nowplaying dataset as of 06/04/2016.

Table 3: Top 5 most popular tracks as of 06/04/2016.

Artist	Track	Times played
ON	Generations	934,021
NP	B4	597,004
Rihanna	Should I?	331,434
Coldplay	Swallowed in the Sea	327,967
Taylor Swift	Crazier	264,918

4.2 Sample Extraction

During the data analysis phase of the project, we discovered that the data was too large for the computational power we had available. This lead to the decision of extracting a sample from the data set. For this, we used the following workflow for extracting a sample data set for training/validating our models:

1. Aggregate the data and calculate each users tweet count (No. of songs published)
2. Filter users with 3 or more tweets
3. Extract a sample of 100.000 records using reservoir sampling
4. Calculate basic statistics such as number of users and tracks, and mean and standard deviation of the number of publications (items) per user.

Table 4 presents stats of the extracted sample such as number of users and tracks, mean and standard deviation. Further explanation regarding to the sampling method used for this task is exposed in the following section.

Table 4: Stats of the extracted sample data set.

Users	Tracks	Mean	Std. Deviation
35,122	49,679	2.8	16.1

4.3 Reservoir Sampling

Reservoir Sampling [8] is a collection of randomized algorithms used to choose a random sample of k items from a list S containing n items, in the context where n is a very large or unknown number. Typically, n is large enough for it to not fit into main memory. The pseudo-code for Reservoir Sampling:

- Keep the first item in memory.
- When the i-th item arrives (for $i > 1$):
- with probability 1/i, keep the new item instead of the current item;
- with probability 1 - 1/i, keep the current item and discard the new item.

So:

- when there is only one item, it is kept with probability 1;
- when there are 2 items, each of them is kept with probability 1/2;
- when there are 3 items, the third item is kept with probability 1/3, and each of the previous 2 items is also kept with probability (1/2)(1 - 1/3) = (1/2)(2/3) = 1/3;
- by induction, it is easy to prove that when there are *n* items, each item is kept with probability 1/n.

Figure 1 shows the transformation schema used in Pentaho to getting a sample from the now playing data set that has users with 3+ published items.

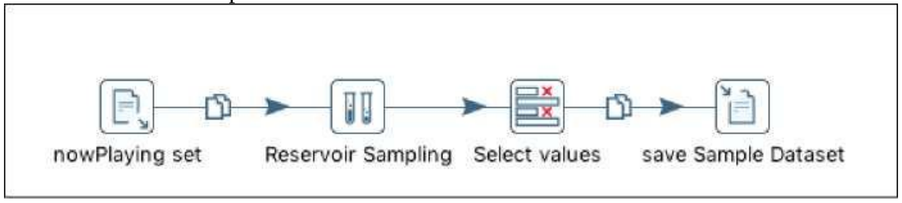

Figure 1: Reservoir Sampling in Pentaho: Data Integration Platform.

4.4 Data Integration for content extraction

In order to integrate the nowplaying sample set with the content information of each track from MusicBrainz, another transformation was built in Pentaho to obtain the list of user-tracks ids. Figure 2 outlines the flow diagram of the transformation, where only the track-based output was used in our project as it can be also used to obtained the artist content information. The obtained output file was used to obtain the content information from the MusicBrainz Postgres Database, as well as the required files to generate the rating matrix for the CTR matrix factorization model. The generated files can be found under the data folder in project repository.

4.5 Ratings Matrix generation

A ratings matrix was generated, consisting of the entire collection of items in the sample data set to give a J by S matrix X. We generated a binary matrix whereby if the user has listened to a track we assign a value of 1, while if they haven't listened to the track, we assign 0. Two files were generated,

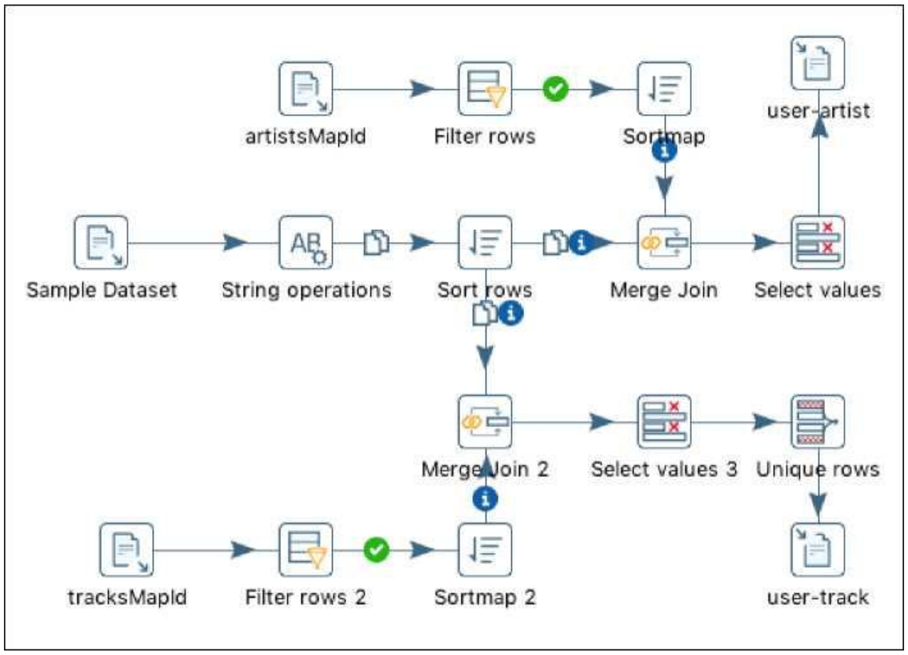

Figure 2: Mapping of nowPlaying/MusicBrainz Ids for content extraction and rating matrix construction

output_by tracks and output_by_user. Each file aggregates each item and associates the user_id or track_id in the corresponding row. Tables 5 and 6 present an example of the format of each file generated.

Table 5: Aggregated tracks played per user.

userJd	count	tracked	track_id
693afc22f9b838a38...	2	013d1e71f6a43adc...	dcca7fcbaa1736d5...
ba6b4825ca63356b0...	2	006cddf752f33eb8...	001c124ec6753b0c...

Table 6: Aggregated users played per track.

track_id	count	userid	userid
114fd187dfed3fcd...	2	662ed5aa9df25a7b43...	663jd5aa9df25a3b...
063ed02c53d83fd2...	2	aae7d3b35d7fca7be0...	762ed56aa9d65a7b...

5 Technological Stack

5.1 Google: TensorFlow™

”TensorFlow™is an open source software library for numerical computation using data flow graphs. Nodes in the graph represent mathematical operations, while the graph edges represent the multidimensional data arrays (tensors) communicated between them. The flexible architecture allows you to deploy computation to one or more CPUs or GPUs in a desktop, server, or mobile device with a single API.” [9]

5.2 Dato: GraphLab Create

”GraphLab Create is an extensible machine learning framework that enables developers and data scientists to easily build and deploy intelligent applications and services at scale. It includes distributed data structures and rich libraries for data transformation and manipulation, scalable task-oriented machine learning toolkits for creating, evaluating, and improving machine learning models, data and model visualization for all aspects of development, and a client to define and deploy both distributed batch jobs to Dato Distributed as well as real-time machine learning services to Dato Predictive Services. It is designed for end-to-end developer productivity, scale, and the variety and complexity of real-world data.” [10]

5.3 Pentaho: Data Integration

”Pentaho Data Integration prepares and blends data to create a complete picture of your business that drives actionable insights. The platform delivers accurate, analytics-ready data to end users from any source. With visual tools to eliminate coding and complexity, Pentaho puts big data and all data sources at the fingertips of business and IT users.” [11]

6 Methodology

6.1 Item Similarity Model

The item similarity model calculates the similarity between items using the historical data representing the interaction between the users and the tracks. Based on the similarity between track *a* and b, it recommends a track c for user u using a weighted average of the user’s previous interactions.

Because the #nowplaying dataset does not consist of rating information, the item similarity model uses Jaccard similarity to calculate the similarity between two tracks, which is an ideal choice when the information available only indicates whether the user listened to a song or not and not the rating a user awarded a song.

Jaccard similarity can be used to measure the similarity between two sets of tracks. In this model, calculating the Jaccard similarity between two tracks is achieved as follows:

$$JS(a, b) = \frac{|U_a \cap U_b|}{|U_a \cup U_b|} \qquad (1)$$

where U_i is the set of users who listened to track i.

6.2 Popularity-based Recommender Model

The Popularity-based Recommender Model determines a track’s overall popularity and ranks the track based on it. The a track is recommended to a user, the track is scored based on the number of times it appears in the dataset. In this model, the track scores are identical for all users, which means the Popularity Recommender Model does not provide tailored recommendations. However, it can be used as a model for new users in an attempt to tackle the ”cold start” problem.

6.3 Factorization Recommender for Ranking Model

The Factorization Recommender for Ranking model aims to learn latent factors for each user and track and uses the factors to rank recommendations in terms of the likelihood of observing (user, track) interactions.

6.4 Collaborative Deep Learning (CDL) Model

The Collaborative Deep Learning model is a novel tightly coupled method for Recommending Systems based on the Stacked Denoising Auto-Encoder (SDAE) deep learning model. Its basic version uses a bag-of-words representation for content features, where the CDL can simultaneously extract an effective deep feature representation from content and capture the similarity and relationship between items and users. Additionally, the deep autoencoder can be easily extended to incorporate other auxiliary information to further boost the performance.

Figure 3 describes each section of the proposed model. Our implementation is based on the Matlab code created by Wang et. al. [6] and the deep autoencoder implementation in Tensorflow presented by Green [12]. First, an unsupervised pre-training is performed one layer at a time by minimizing the (cross-entrophy) error in reconstructing the input. Then, the middle layer $X_L/_2$ serves as a bridge between the ratings and content information, being the key that enables CDL to simultaneously learn an effective feature representation and capture the similarity items. Finally, once all layers are pretrained, the network goes through a second stage of training or supervised-fine-tuning, where the prediction error is to be minimized.

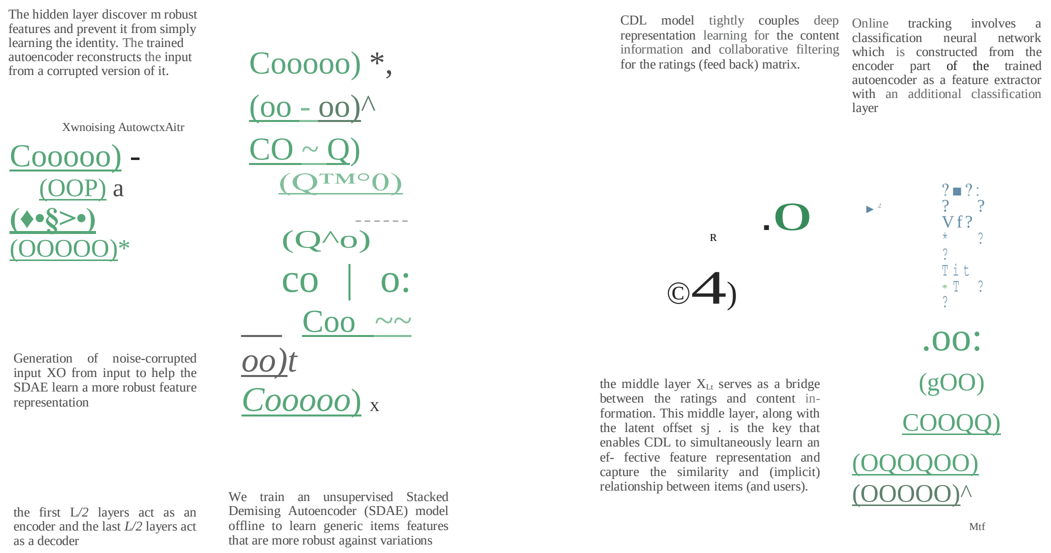
The hidden layer discover m robust features and prevent it from simply learning the identity. The trained autoencoder reconstructs the input from a corrupted version of it.
Xwnoising AutowctxAitr
Generation of noise-corrupted input XO from input to help the SDAE learn a more robust feature representation
the first L/2 layers act as an encoder and the last L/2 layers act as a decoder
We train an unsupervised Stacked Demising Autoencoder (SDAE) model offline to learn generic items features that are more robust against variations
CDL model tightly couples deep representation learning for the content information and collaborative filtering for the ratings (feed back) matrix.
Online tracking involves a classification neural network which is constructed from the encoder part of the trained autoencoder as a feature extractor with an additional classification layer
the middle layer $X_{L/2}$ serves as a bridge between the ratings and content in-formation. This middle layer, along with the latent offset sj . is the key that enables CDL to simultaneously learn an ef- fective feature representation and capture the similarity and (implicit) relationship between items (and users).
Mtf

Figure 3: Collaborative Deep Learning Model workflow.

7 Evaluation

For evaluating our baseline recommender systems, we select 3/4 of the extracted sample data and set it as the training set while the remaining data is set as the test set. We use recall as the performance measure as the rating information is in the form of implicit feedback. Figure 4 presents the evaluation of recall over the top K items with K=10, and where results are averaged over all users. The rank factorization model is considered the most stable and better model as the recall increases with the number of top recommendations needed. Althought the popularity model performs better for K>6 but lower it is unstable for lower values of K as the recall value fluctuates. Additionally, the mean average precision was also evaluated for each model. Figure 5 outlines once again the the factorization model is more stable with increasing K.

8 Results

8.1 Item Similarity Model

When run, the Item Similarity Model recommends 10 tracks for each user. Additionally, using the model, similar tracks can also be retrieved. Table 7 presents an example of 5 track recommendations generated by the Item Similarity Model, with Rock being a predominant genre. Table 8 presents an example of 5 similar tracks generated by the Item Similarity Model, indicating poor similarity accuracy.

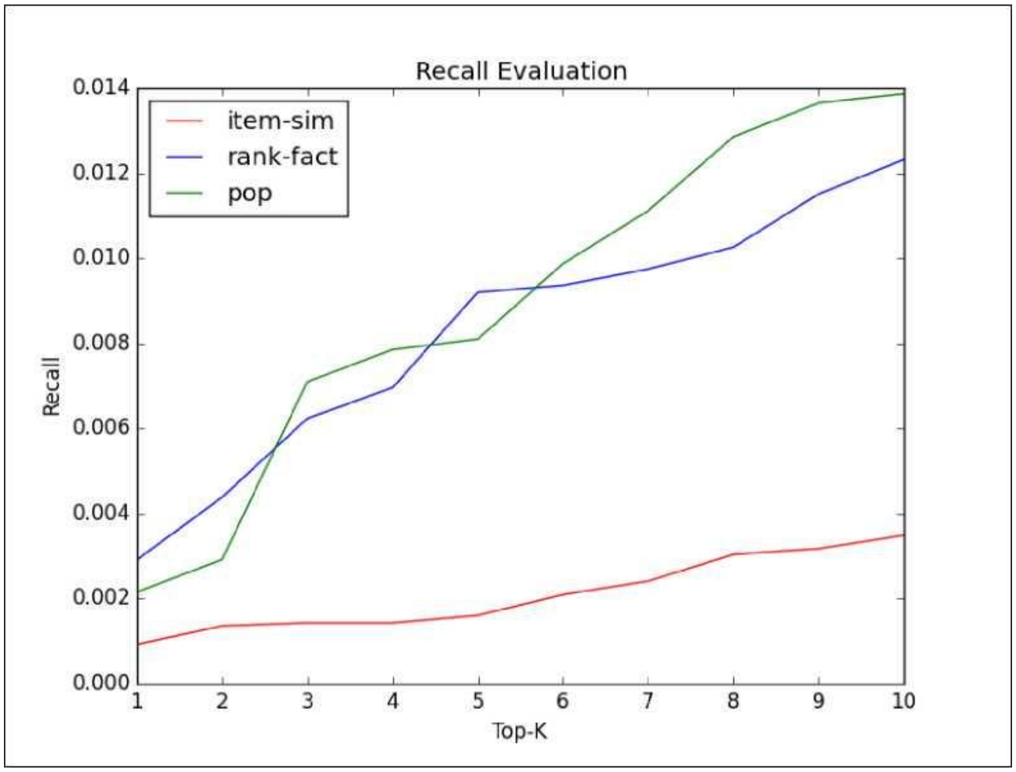

Figure 4: Recall Evaluation over Top-K items.

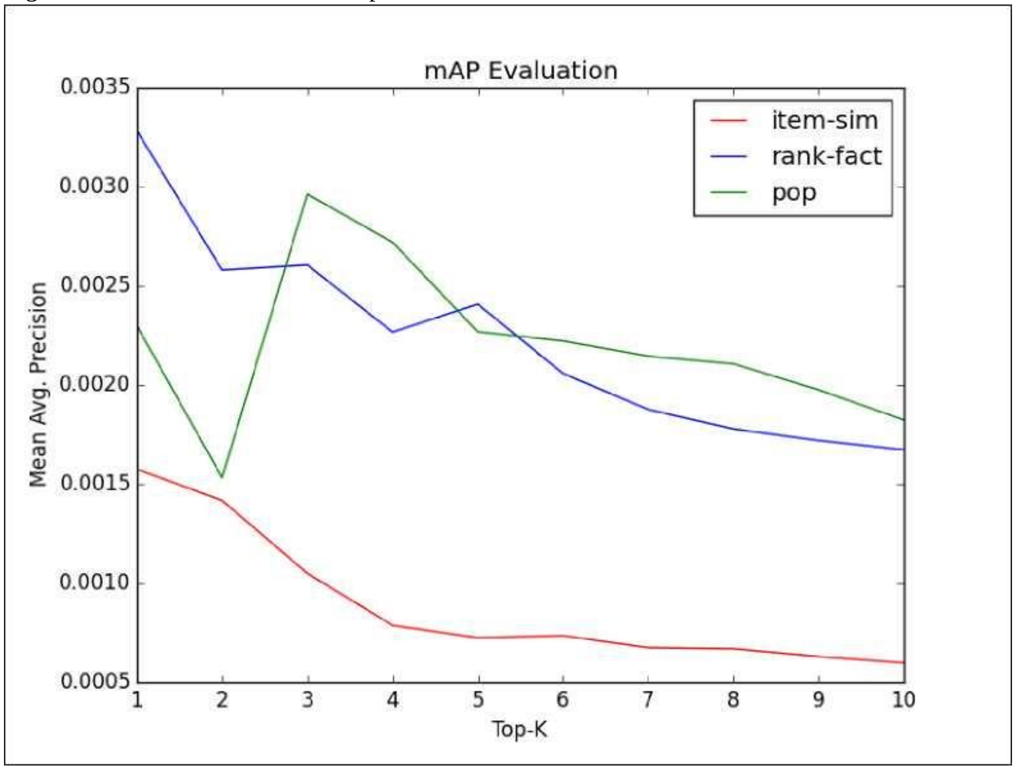

Figure 5: Mean Average Precision Evaluation over Top-K items.

Table 7: Example of 5 track recommendations generated by the Item Similarity Model.

Artist **Track** **Genre**

Table 8: Example of 5 similar tracks generated by the Item Similarity Model.

Track	Similar Track	Track Genre	Similar Track Genre
Iris by Iris	Rockstar by Nickelback	Hard Rock	Southern Rock
Iris by Iris	Antenna by Fuse ODG	Hard Rock	Afrobeats, Dance
Iris by Iris	All I Wanna Do by Sheryl Crow	Hard Rock	Pop
Iris by Iris	Vertigo by Vertigo	Hard Rock	Alternative rock
Iris by Iris	Boomerang by Boomerang	Hard Rock	Electronic, Latin, Funk/Soul, Pop

8.2 Popularity-based Recommender Model

When run, the Popularity-based Recommender Model recommends 10 tracks for each user. Additionally, using the model, similar tracks can also be retrieved. Table 9 presents an example of 5 track recommendations generated by the Popularity-based Recommender Model, with Electronic being a predominant genre. Table 10 presents an example of 5 similar tracks generated by the Popularity- based Recommender Model, indicating significantly poor similarity accuracy.

Table 9: Example of 5 track recommendations generated by the Popularity-based Recommender Model.

Table 10: Example of 5 similar tracks generated by the Popularity-based Recommender Model.

ON I Electronic, Rock

Track	Similar Track	Track Genre	Similar Track Genre
Times by Ann Lee	Io non sono qui by Ushas	Eurodance	Hard Rock
Times by Ann Lee	Cinta Jangan Kau Pergi by Sheila Majid	Eurodance	Jazz, Funk/Soul, Pop
Times by Ann Lee	Seal Your Fate by Obituary	Eurodance	Rock
Times by Ann Lee	No Man Is an Island by Tenth Avenue North	Eurodance	Rock, Pop
Times by Ann Lee	My Life by Richie Spice	Eurodance	Reggae

Artist **Track** **Genre**

2
2

2
2

8.3 Factorization Recommender for Ranking Model

When run, the Factorization Recommender for Ranking Model recommends 10 tracks for each user. Additionally, using the model, similar tracks can also be retrieved. Table 11 presents an example of 5 track recommendations generated by the Factorization Recommender for Ranking Model, with Rock being somehow a predominant genre. Table 12 presents an example of 5 similar tracks generated by the Factorization Recommender for Ranking Model, indicating poor similarity accuracy.

Table 11: Example of 5 track recommendations generated by the Factorization Recommender for Ranking Model.

Artist	Track	Genre
A	A	Alternative Rock

Table 12: Example of 5 similar tracks generated by the Factorization Recommender for Ranking Model.

Track	Similar Track	Track Genre	Similar Track Genre
Just Say Yes by Snow Patrol	It's Raining Men by The Weather Girls	Synthpop, Electronica	Hi-NRG, Post-disco
Just Say Yes by Snow Patrol	Catfish Blues by Robert Petway	Synthpop, Electronica	Blues
Just Say Yes by Snow Patrol	Amor Depredador by The Mills	Synthpop, Electronica	Latin Rock, Alternative rock
Just Say Yes by Snow Patrol	Damn It Feels Good to Be a Gangsta by Geto Boys	Synthpop, Electronica	Gangsta Rap, Southern Hip Hop, Horrorcore
Just Say Yes by Snow Patrol	Walking On The Sun by Smash Mouth	Synthpop, Electronica	Ska Pop, Psychedelic Rock

8.4 Collaborative Deep Learning (CDL) Model

Because of the limited time period we had available to finish this assignment, we were not able to finish the integration of the deep autoencoder implementation with the rating matrix factorization (the latter is an implementation of Wang et. al.[6]). However, in our GitHub repository, we include the developed code using TensorFlow in the cdl folder, where we set up a Stacked Denoising Autoencoder configured with 2 hidden layers of 200 and 50 neurons, a dropout module to learn stronger features, and a Gradient Descent optimizer, even though a Momentum optimizer would be a better algorithm (easy to use in TensorFlow) that can be used to alleviate the local optimum problem.

9 Future Work

As CDL is actually a more general framework which can also admit other deep learning models, a further investigation could be done on integrating the current solution with existing models such as Boltzmann machines, Recurrent Neural Networks, and Convolutional Neural Networks. Additionally, refining the feature representation by adding audio signals from track previews to boost the prediction accuracy, as applied by Spotify [13] is another potential future focus.

References

[1] Chong Wang and David M Blei. "Collaborative topic modeling for recommending scientific articles". In: *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining.* ACM. 2011, pp. 448-456.

[2] Yvonne Ho, Simon Fong, and Zhuang Yan. "A Hybrid GA-based Collaborative Filtering Model for Online Recommenders." In: *ICE-B.* 2007, pp. 200-203.

[3] Erik Bernhardsson. *Music recommendations using cover images.* 2014. URL: `http : / / erikbern . com / 2014/04/01/ music - recommendations - using - cover - images-part-1/` (visited on 08/03/2016).

[4] Martin Pichl, Eva Zangerle, and GUnther Specht. "Combining Spotify and Twitter Data for Generating a Recent and Public Dataset for Music Recommendation." In: *Grundlagen von Datenbanken.* 2014, pp. 35-40.

[5] Sander Dieleman. *Recommending music on Spotify with deep learning.* 2014. URL: `http : / / benanne . github .io/ 2014/08/05 / spotify - cnns . html` (visited on 08/03/2016).

[6] Hao Wang, Naiyan Wang, and Dit-Yan Yeung. "Collaborative deep learning for recommender systems". In: *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* ACM. 2015, pp. 1235-1244.

[7] Eva Zangerle et al. "# nowplaying Music Dataset: Extracting Listening Behavior from Twitter". In: *Proceedings of the First International Workshop on Internet-Scale Multimedia Management.* ACM. 2014, pp. 21-26.

[8] Wikipedia. *Reservoir sampling — Wikipedia, The Free Encyclopedia.* URL: `https ://en. wikipedia.org/wiki/Reservoir_sampling` (visited on 02/04/2016).

[9] Google: TensorFlow™. *TensorFlow is an Open Source Software Library for Machine Intelligence.* URL: `https://www.tensorflow.org/` (visited on 02/04/2016).

[10] Dato: GraphLab Create. *Sophisticated machine learning as easy as "Hello, World!".* URL: `https://dato.com/products/create/` (visited on 02/04/2016).

[11] Pentaho: Data Integration. *Data Integration \Pentaho Data Integration Platform.* URL: `http : / / www . pentaho . com / product / data - integration` (visited on 04/04/2016).

[12] Christofer Green. *Diving Into TensorFlow With Stacked Autoencoders.* URL: `http : / / cmgreen . io/ 2016 / 01 / 04 /tensorflow_deep_autoencoder . html` (visited on 04/01/2016).

[13] Aaron Van den Oord, Sander Dieleman, and Benjamin Schrauwen. "Deep content-based music recommendation". In: *Advances in Neural Information Processing Systems.* 2013, pp. 2643-2651.