TOP-N MUSIC RECOMMENDATION SYSTEM

A thesis submitted in the partial fulfillment of the requirements for the award of degree of

B. Tech

In

Computer Science And Engineering

By

Paritosh Yadav (106114061)

Ritul Jain(106114081)

Vishal Singh(106114092)

COMPUTER SCIENCE AND ENGINEERING

NATIONAL INSTITUTE OF TECHNOLOGY

TIRUCHIRAPPALLI – 620015

MAY 2018

BONAFIDE CERTIFICATE

This is to certify that the project titled **TOP-N MUSIC RECOMMENDATION SYSTEM** is a bonafide record of the work done by

Paritosh Yadav (106114061)

Ritul Jain(106114081)

Vishal Singh(106114092)

In partial fulfilment of the requirements for the award of the degree of **Bachelors of Technology** in **Computer Science And Engineering** of the **NATIONAL INSTITUTE OF TECHNOLOGY, TIRUCHARAPPALLLI,** during the year 2017-2018.

Dr. M.Sridevi	**Dr.Leela Velusamy**
Project Guide	Head of Department

Project viva-voce held on ______________

INTERNAL EXAMINER	**EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

We are grateful to the department of Computer Science And Engineering, NIT Trichy for providing us the opportunity to do **Top-N Recommendation Music Recommendation System** as our final year project.

We wish to extend our heartiest thanks to **Dr. M.Sridevi,** Professor, NIT Trichy, for the stepwise guidance that she provided. Her constant encouragement and faith in us enabled us to complete the project work efficiently in the stipulated time interval.

We would also like to express our gratitude to **Dr. LeelaVelusamy**, Head of the Department, Computer Science And Engineering for taking time amidst her work, to help us for successful completion of the project.

Our sincere Thanks to **Dr. Mini Shaji Thomas**, Director NIT Tiruchirapalli for having provided all facilities for carrying out this project work.

Last but not the least, our sincere thanks to one and all whom we have missed to acknowledge individually.

Paritosh Yadav

Ritul Jain

Vishal Singh

LIST OF FIGURES

CHAPTER 1: INTRODUCTION ………………..…………………………………1

1.1. Recommendation Phases …..........…......……………………………………......3

1.2. Recommendation Techniques…..........…......……………………………………....3

1.3. Collaborative filtering process …..........…......……………………………......…....5

CHAPTER 3: DATASETS ………………………………………………..………...14

3.1. Some Similar Artists according to Echo Nest - Table.……………………………......15

3.2. A snap of triplets_file …... ………………………………....16

3.3. A snap of metadata_file ….. ……………….......….....16

3.4. Combining the datasets…... ……………….......….....16

3.5. After merging the datasets .... ……………………...........…17

CHAPTER 4: METHODOLOGY…………………………………...…………..18

4.1. Code snippet …...……………………….......18

4.2. Dataset after transformation ….................................... ……………….......…...19

4.3. Top 5 recommendations for the given id .... ……………….......….....19

4.4. FISM Algorithm …... ……………….......…....23

CHAPTER 5: RESULTS & DISCUSSIONS………………………………..……...27

5.1. Effect of varying alpha - Graph .... ……………………....…28

5.2. Effect of varying the dimensionality of factor matrices – Graph ……….......…....28

5.3. Varying number of iterations - Graph …..................... …………………............…29

5.4. Varying the learning rate - Graph ....……………….................29

5.5. Varying N in Top-N Recommendations - Graph ....……………………......30

5.6. Comparing for Top 5 Recommendations – Graph ....………........…....30

5.7. Comparing for Top 10 Recommendations - Graph …....…………………............…31

5.8. Comparing for Top 15 Recommendations - Graph ....……………….......…....31

TABLE OF CONTENTS

BONAFIDE CERTIFICATE……………………………………………………………...i

ACKNOWLEDGEMENT…………………………………………………………...ii

ABSTRACT ………………………………………………………………….......iii

LIST OF FIGURES ……………………………….…………………………..iv

CHAPTER 1: INTRODUCTION ………………..…………………………………1

1.1 Phase of Recommendation system …....……………………………………………1

1.1.1. Information collection phase…………….…………………………………....1

1.1.2. Learning phase……………………….……………………………….....2

1.1.3. Prediction/recommendation phase…………….……………………….......2

1.2 Recommendation filtering system………………………………………………...3

1.2.1. Content-based filtering………………………….………………………….3

1.2.2. Pros and Cons of content-based filtering techniques…………………..………4

1.2.3. Collaborative filtering……………......…………………………………...…..…4

1.2.3.1. Memory based techniques…………………………………………..…..5

1.2.3.2. Model-based techniques………………………………………..….6

1.2.4. Pros and Cons of collaborative filtering techniques……………………………...6

1.2.4.1. Cold-start problem………………………………………………………7

1.2.4.2. Data sparsity problem……………………………………………….....7

1.2.4.3. Scalability…………………………………………………..……....7

1.2.5 Hybrid filtering……………………………………………………………………7

1.2.5.1 Weighted hybridization……………………………………….....……...8

1.2.5.2. Switching hybridization…………………………………………………… 8

1.2.5.3 Cascade hybridization……………………………………………………. 9

1.2.5.4. Mixed hybridization……………………………………………………...9

1.2.5.5. Feature-combination……………………………………………….…………..9

1.2.5.6 Feature-augmentation……………………………………….……………...10

1.2.5.7 Meta-level…………………………………………………...…………...10

CHAPTER 2: LITERATURE REVIEW ……………………………...……..…11

2.1 Introduction……………………………………………………………….…...11

2.2 Existing Collaborative Filtering Techniques……………………………..…….11

2.3 Existing Content-Based Techniques……………………………………..….…11

2.4 Hybrid Approaches…………………………………………………….…....12

2.4.1 Limitations of Existing Methods……………………………….……...12

2.4.2 Hybrid Methods…………………………………………………….……..12

CHAPTER 3: DATASETS …………………………………………...………..14

3.1 Million Song Dataset…………………………………………………………..14

3.1.1 Introduction………………………………………………………………..14

3.1.2 Usage……………………………………………………...………14

3.1.3 Dataset Used In the project..16

CHAPTER 4: METHODOLOGY……………………………...…………..18

4.1 Popularity Based Recommender…………………………………….………...18

4.2 Item Similarity based Collaborative filtering model-based Recommender……......20

4.2.1 ALGORITHM…………………………………………………………….…20

4.3 Factored Item Similarity Model (FISM) based recommender……………….…22

4.3.1 Introduction……………………………………………………………..…22

4.3.2 Motivation……………………………………………………………....22

4.3.3 FISM Algorithm………………………………………………………….23

4.3.4 Explanation…………………………………………………………….....24

4.3.5 Algorithm…………………………………………………………….……25

CHAPTER 5: RESULTS & DISCUSSIONS…………………………….…...27

5.1 Metrics Used…………………………………………………………….…….27

5.2 Graphs………………………………………………………………...………..28

5.2.1 On varying various parameters used in FISM…………………………...……..28

5.2.2 Comparing FISM with other algorithms……...……………………...………..30

CHAPTER 6: CONCLUSION…………………………………………………………..32

REFERENCES……………………………………………………………...…..33

CHAPTER 1: INTRODUCTION

The explosive growth in the amount of available digital information and the number of visitors to the Internet have created a potential challenge of information overload which hinders timely access to items of interest on the Internet. Information retrieval systems, such as Google, DevilFinder and Altavista have partially solved this problem but prioritization and personalization (where a system maps available content to user's interests and preferences) of information were absent. This has increased the demand for recommender systems more than ever before. Recommender systems are information filtering systems that deal with the problem of information overload by filtering vital information fragment out of large amount of dynamically generated information according to user's preferences, interest, or observed behavior about item. Recommender system has the ability to predict whether a particular user would prefer an item or not based on the user's profile. Recommender systems are beneficial to both service providers and users. They reduce transaction costs of finding and selecting items in an online shopping environment. Recommendation systems have also proved to improve decision making process and quality. In e-commerce setting, recommender systems enhance revenues, for the fact that they are effective means of selling more products. In scientific libraries, recommender systems support users by allowing them to move beyond catalog searches. Therefore, the need to use efficient and accurate recommendation techniques within a system that will provide relevant and dependable recommendations for users cannot be over-emphasized.

1.1: PHASES OF RECOMMENDATION SYSTEM

1.1.1. INFORMATION COLLECTION PHASE

This collects relevant information of users to generate a user profile or model for the prediction tasks including user's attribute, behaviors or content of the resources the user accesses. A recommendation agent cannot function accurately until the user profile/model has been well constructed. The system needs to know as much as possible from the user in order to provide reasonable recommendation right from the onset. Recommender systems rely on different types of input such as the most convenient high quality explicit feedback, which includes explicit input by users regarding their interest in item or implicit feedback by inferring user preferences indirectly through observing user behavior. Hybrid feedback can also be obtained through the combination of both explicit and implicit feedback. In E-learning platform, a user profile is a collection of personal information associated with a specific user. This information includes cognitive skills, intellectual abilities, learning styles, interest, preferences and interaction with the system. The user profile is normally used to retrieve the needed information to build up a model of the user. Thus, a user profile describes a simple user model. The success of any recommendation system depends largely on its ability to represent user's current interests. Accurate models are indispensable for obtaining relevant and accurate recommendations from any prediction techniques.

1.1.2. LEARNING PHASE

It applies a learning algorithm to filter and exploit the user's features from the feedback gathered in information collection phase.

1.1.3. PREDICTION/RECOMMENDATION PHASE

It recommends or predicts what kind of items the user may prefer. This can be made either directly based on the dataset collected in information collection phase which could be memory based or model based or through the system's observed activities of the user. Fig. 1 highlights the recommendation phases.

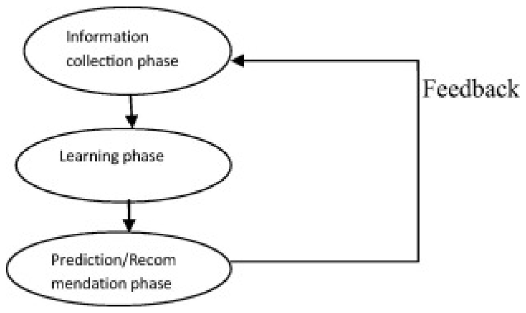

Information collection phase
Learning phase
Prediction/Recom mendation phase
Feedback

Figure 1.1 . Recommendation phases.

1.2: RECOMMENDATION FILTERING SYSTEM

The use of efficient and accurate recommendation techniques is very important for a system that will provide good and useful recommendation to its individual users. This explains the importance of understanding the features and potentials of different recommendation techniques. Fig. 2 shows the anatomy of different recommendation filtering techniques.

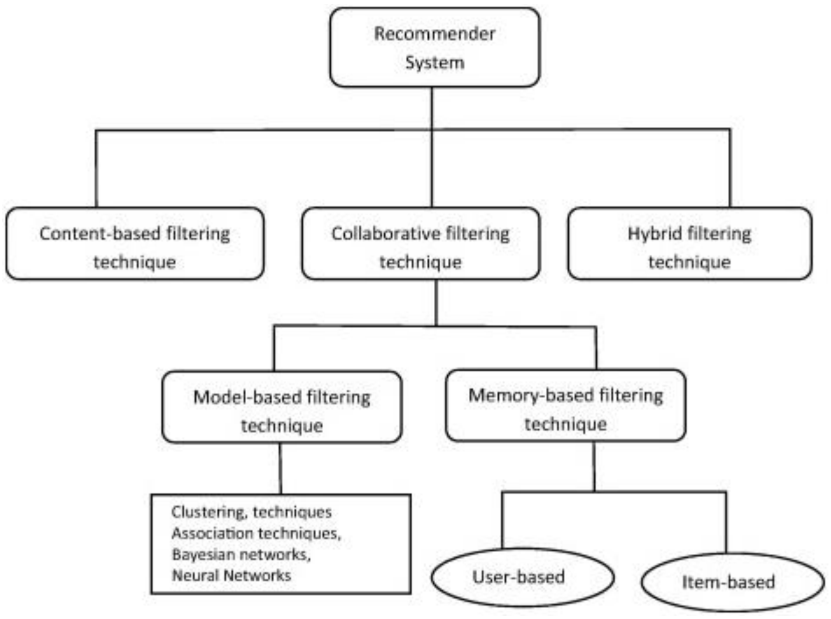

Recommender System
Content-based filtering technique
Collaborative filtering technique
Hybrid filtering technique
Model-based filtering technique
Memory-based filtering technique
Clustering, techniques
Association techniques,
Bayesian networks,
Neural Networks
User-based
Item-based

Figure 1.2. Recommendation techniques.

1.2.1. CONTENT-BASED FILTERING

Content-based technique is a domain-dependent algorithm and it emphasizes more on the analysis of the attributes of items in order to generate predictions. When documents such as web pages, publications and news are to be recommended, content-based filtering technique is the most successful. In content-based filtering technique, recommendation is made based on the user profiles using features extracted from the content of the items the user has evaluated in the past. Items that are mostly related to the positively rated items are recommended to the user. CBF uses different types of models to find similarity between documents in order to generate meaningful recommendations. It could use

Vector Space Model such as Term Frequency Inverse Document Frequency (TF/IDF) or Probabilistic models such as Naïve Bayes Classifier, Decision Trees or Neural Networks to model the relationship between different documents within a corpus. These techniques make recommendations by learning the underlying model with either statistical analysis or machine learning techniques. Content-based filtering technique does not need the profile of other users since they do not influence recommendation. Also, if the user profile changes, CBF technique still has the potential to adjust its recommendations within a very short period of time.

1.2.2. PROS AND CONS OF CONTENT-BASED FILTERING

CB filtering techniques overcome the challenges of CF. They have the ability to recommend new items even if there are no ratings provided by users. So even if the database does not contain user preferences, recommendation accuracy is not affected. Also, if the user preferences change, it has the capacity to adjust its recommendations in a short span of time. They can manage situations where different users do not share the same items, but only identical items according to their intrinsic features. Users can get recommendations without sharing their profile, and this ensures privacy. CBF technique can also provide explanations on how recommendations are generated to users. However, the techniques suffer from various problems as discussed in the literature. Content based filtering techniques are dependent on items' metadata. That is, they require rich description of items and very well organized user profile before recommendation can be made to users. This is called limited content analysis. So, the effectiveness of CBF depends on the availability of descriptive data. Content overspecialization is another serious problem of CBF technique. Users are restricted to getting recommendations similar to items already defined in their profiles.

1.2.3. COLLABORATIVE FILTERING

Collaborative filtering is a domain-independent prediction technique for content that cannot easily and adequately be described by metadata such as movies and music. Collaborative filtering technique works by building a database (user-item matrix) of preferences for items by users. It then matches users with relevant interest and preferences by calculating similarities between their profiles to make recommendations.

Such users build a group called neighborhood. An user gets recommendations to those items that he has not rated before but that were already positively rated by users in his neighborhood. Recommendations that are produced by CF can be of either prediction or recommendation. Prediction is a numerical value, Rij, expressing the predicted score of item j for the user i, while Recommendation is a list of top N items that the user will like the most as shown in Fig. 3. The technique of collaborative filtering can be divided into two categories: memory-based and model-based.

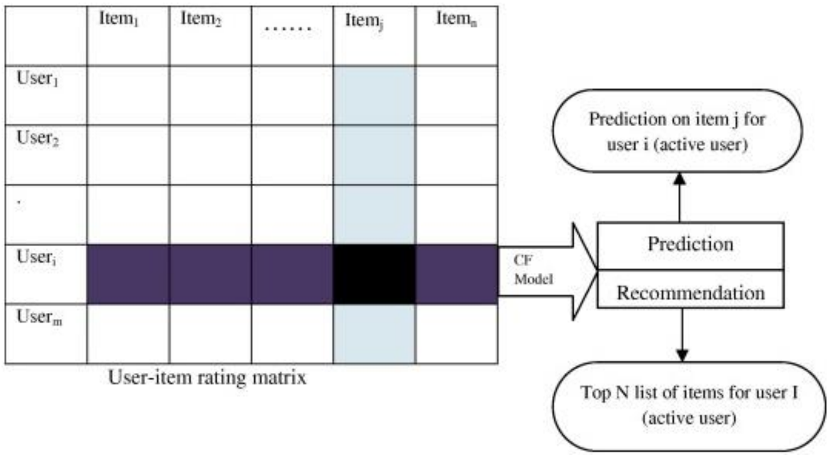


Figure 1.3. Collaborative filtering process.

1.2.3.1. MEMORY-BASED TECHNIQUES

The items that were already rated by the user before play a relevant role in searching for a neighbour that shares appreciation with him. Once a neighbour of a user is found, different algorithms can be used to combine the preferences of neighbours to generate recommendations. Due to the effectiveness of these techniques, they have achieved widespread success in real life applications. Memory-based CF can be achieved in two ways through user-based and item-based techniques. User based collaborative filtering technique calculates similarity between users by comparing their ratings on the same item, and it then computes the predicted rating for an item by the active user as a weighted average of the ratings of the item by users similar to the active user where weights are the similarities of these users with the target item. Item-based filtering techniques compute predictions using the similarity between items and not the similarity between users. It builds a model of item similarities by retrieving all items rated by an active user from the user-item matrix, it determines how similar the retrieved items are to the target item, then it selects the k most similar items and their corresponding similarities are also determined. Prediction is made by taking a weighted average of the active users rating on the similar items k.

1.2.3.2. MODEL-BASED TECHNIQUES

This technique employs the previous ratings to learn a model in order to improve the performance of Collaborative filtering Technique. The model building process can be done using machine learning or data mining techniques. These techniques can quickly recommend a set of items for the fact that they use pre-computed model and they have proved to produce recommendation results that are similar to neighborhood-based recommender techniques. Examples of these techniques include Dimensionality Reduction technique such as Singular Value Decomposition (SVD), Matrix Completion Technique, Latent Semantic methods, and Regression and Clustering. Model-based techniques analyze the user-item matrix to identify relations between items; they use these relations to compare the list of top-N recommendations. Model based techniques resolve the sparsity problems associated with recommendation systems.

1.2.4. PROS AND CONS OF COLLABORATIVE FILTERING

Collaborative Filtering has some major advantages over CBF in that it can perform in domains where there is not much content associated with items and where content is difficult for a computer system to analyze (such as opinions and ideal). Also, CF technique has the ability to provide serendipitous recommendations, which means that it can recommend items that are relevant to the user even without the content being in the user’s profile. Despite the success of CF techniques, their widespread use has revealed some potential problems such as follows.

1.2.4.1. COLD-START PROBLEM

This refers to a situation where a recommender does not have adequate information about a user or an item in order to make relevant predictions. This is one of the major problems that reduce the performance of recommendation system. The profile of such new user or item will be empty since he has not rated any item; hence, his taste is not known to the system.

1.2.4.2. DATA SPARSITY PROBLEM

This is the problem that occurs as a result of lack of enough information, that is, when only a few of the total number of items available in a database are rated by users. This always leads to a sparse user-item matrix, inability to locate successful neighbors and finally, the generation of weak recommendations. Also, data sparsity always leads to coverage problems, which is the percentage of items in the system that recommendations can be made for.

1.2.4.3. SCALABILITY

This is another problem associated with recommendation algorithms because computation normally grows linearly with the number of users and items. A recommendation technique that is efficient when the number of dataset is limited may be unable to generate satisfactory number of recommendations when the volume of dataset is increased. Thus, it is crucial to apply recommendation techniques which are capable of scaling up in a successful manner as the number of dataset in a database increases. Methods used for solving scalability problem and speeding up recommendation generation are based on Dimensionality reduction techniques, such as Singular Value Decomposition (SVD) method, which has the ability to produce reliable and efficient recommendations.

1.2.5. HYBRID FILTERING

Hybrid filtering technique combines different recommendation techniques in order to gain better system optimization to avoid some limitations and problems of pure recommendation systems. The idea behind hybrid techniques is that a combination of algorithms will provide more accurate and effective recommendations than a single algorithm as the disadvantages of one algorithm can be overcome by another algorithm. Using multiple recommendation techniques can suppress the weaknesses of an individual technique in a combined model. The combination of approaches can be done in any of the following ways: separate implementation of algorithms and combining the result, utilizing some content-based filtering in collaborative approach, utilizing some collaborative filtering in content-based approach, creating a unified recommendation system that brings together both approaches.

1.2.5.1 WEIGHTED HYBRIDIZATION

Weighted hybridization combines the results of different recommenders to generate a recommendation list or prediction by integrating the scores from each of the techniques in use by a linear formula. An example of a weighted hybridized recommendation system is P-tango [76]. The system consists of a content-based and collaborative recommender. They are given equal weights at first, but weights are adjusted as predictions are confirmed or otherwise. The benefit of a weighted hybrid is that all the recommender system's strengths are utilized during the recommendation process in a straightforward way.

1.2.5.2. SWITCHING HYBRIDIZATION

The system swaps to one of the recommendation techniques according to a heuristic reflecting the recommender ability to produce a good rating. The switching hybrid has the ability to avoid problems specific to one method e.g. the new user problem of content-based recommender, by switching to a collaborative recommendation system. The benefit of this strategy is that the system is sensitive to the strengths and weaknesses of its constituent recommenders. The main disadvantage of switching hybrids is that it usually introduces more complexity to recommendation process because the switching criterion, which normally increases the number of parameters to the recommendation system has to be determined [34]. Example of a switching hybrid recommender is the DailyLearner [77] that uses both content-based and collaborative hybrid where a content-based recommendation is employed first before collaborative recommendation in a situation where the content-based system cannot make recommendations with enough evidence.

1.2.5.3 CASCADE HYBRIDIZATION

The cascade hybridization technique applies an iterative refinement process in constructing an order of preference among different items. The recommendations of one technique are refined by another recommendation technique. The first recommendation technique outputs a coarse list of recommendations which is in turn refined by the next recommendation technique. The hybridization technique is very efficient and tolerant to noise due to the coarse-to-finer nature of the iteration. EntreeC [34] is an example of cascade hybridization method that used a cascade knowledge-based and collaborative recommender.

1.2.5.4. MIXED HYBRIDIZATION

Mixed hybrids combine recommendation results of different recommendation techniques at the same time instead of having just one recommendation per item. Each item has multiple recommendations associated with it from different recommendation techniques. In mixed hybridization, the individual performances do not always affect the general performance of a local region. Example of recommender system in this category that uses the mixed hybridization is the PTV system [78] which recommends a TV viewing schedule for a user by combining recommendations from content-based and collaborative systems to form a schedule. Profinder [79] and PickAFlick [80] are also examples of mixed hybrid systems.

1.2.5.5. FEATURE-COMBINATION

The features produced by a specific recommendation technique are fed into another recommendation technique. For example, the rating of similar users which is a feature of collaborative filtering is used in a case-based reasoning recommendation technique as one of the features to determine the similarity between items. Pipper is an example of feature combination technique that used the collaborative filter’s ratings in a content-based system as a feature for recommending movies [81]. The benefit of this technique is that, it does not always exclusively rely on the collaborative data.

1.2.5.6 FEATURE-AUGMENTATION

The technique makes use of the ratings and other information produced by the previous recommender and it also requires additional functionality from the recommender systems. For example, the Libra system [42] makes content-based recommendation of books on data found in Amazon.com by employing a naïve Bayes text classifier. Feature-augmentation hybrids are superior to feature-combination methods in that they add a small number of features to the primary recommender.

1.2.5.7 META-LEVEL

The internal model generated by one recommendation technique is used as input for another. The model generated is always richer in information when compared to a single rating. Meta-level [17] hybrids are able to solve the sparsity problem of collaborative filtering techniques by using the entire model learned by the first technique as input for the second technique. Example of meta-level technique is LaboUr [82] which uses instant-based learning to create content-based user profile that is then compared in a collaborative manner.

CHAPTER 2: LITERATURE REVIEW

2.1 INTRODUCTION

Recommender system is defined as a decision-making strategy for users under complex information environments. Also, recommender system was defined from the perspective of E-commerce as a tool that helps users search through records of knowledge which is related to users' interest and preference. Recommender system was defined as a means of assisting and augmenting the social process of using recommendations of others to make choices when there is no sufficient personal knowledge or experience of the alternatives.

2.2 EXISTING COLLABORATIVE FILTERING TECHNIQUES

Recently, various approaches for building recommendation systems have been developed, which can utilize either collaborative filtering, content-based filtering or hybrid filtering. Collaborative filtering technique is the most mature and the most commonly implemented. Collaborative filtering recommends items by identifying other users with similar taste; it uses their opinion to recommend items to the active user.

GroupLens is a news-based architecture which employed collaborative methods in assisting users to locate articles from massive news database. Ringo is an online social information filtering system that uses collaborative filtering to build users profile based on their ratings on music albums. Amazon uses topic diversification algorithms to improve its recommendation. The system uses collaborative filtering method to overcome scalability issue by generating a table of similar items offline through the use of item-to-item matrix. The system then recommends other products which are similar online according to the users' purchase history.

2.3 EXISTING CONTENT-BASED TECHNIQUES

Content-based techniques match content resources to user characteristics. Content-based filtering techniques normally base their predictions on user's information, and they ignore contributions from other users as with the case of collaborative techniques. Fab relies heavily on the ratings of different users in order to create a training set and it is an example of content-based recommender system. Some other systems that use content-based filtering to help users find information on the Internet include Letizia. The system makes use of a user interface that assists users in browsing the Internet; it is able to track the browsing pattern of a user to predict the pages that they may be interested in. Pazzani et al. designed an intelligent agent that attempts to predict which web pages will interest a user by using naive Bayesian classifier. The agent allows a user to provide training instances by rating different pages as either hot or cold.

2.4 HYBRID APPROACHES

2.4.1 LIMITATIONS OF EXISTING METHODS

Despite the success of these two filtering techniques, several limitations have been identified. Some of the problems associated with content-based filtering techniques are limited content analysis, overspecialization and sparsity of data. Also, collaborative approaches exhibit cold-start, sparsity and scalability problems. These problems usually reduce the quality of recommendations.

In order to mitigate some of the problems identified, Hybrid filtering, which combines two or more filtering techniques in different ways in order to increase the accuracy and performance of recommender systems has been proposed.

A general unified model which incorporates both content-based and collaborative filtering properties could be developed. The problem of sparsity of data and cold-start was addressed by combining the ratings, features and demographic information about items in a cascade hybrid recommendation technique.

2.4.2 HYBRID METHODS

In Ziegler et al., a hybrid collaborative filtering approach was proposed to exploit bulk taxonomic information designed for exacting product classification to address the data sparsity problem of CF recommendations, based on the generation of profiles via inference of super-topic score and topic diversification.

A hybrid recommendation technique is also proposed in Ghazantar and Pragel-Benett, and this uses the content-based profile of individual user to find similar users which are used to make predictions.

A simple and straightforward method for combining content-based and collaborative filtering was proposed by Cunningham et al. A music recommendation system which combined tagging information, play counts and social relations was proposed in Konstas et al.

To address the issue of data sparsity and cold-start in recommender system, social information (e.g., user-user trust links) has been introduced in Zhang et al. to complement rating data for improving the performances of traditional model-based recommendation techniques such as matrix factorization (MF) and Bayesian personalized ranking (BPR).

Top-N recommender systems have been investigated widely both in industry and academia. However, the recommendation quality is far from satisfactory. Kang et al. proposes a simple yet promising algorithm. We fill the user-item matrix based on a low-rank assumption and simultaneously keep the original information. To do that, a nonconvex rank relaxation rather than the nuclear norm is adopted to provide a better rank approximation and an efficient optimization strategy is designed.

Cheng et al proposes a context aware music recommendation approach, which can recommend music appropriate for users' contextual preference for music. This approach does not require songs to be described by features beforehand, but it learns music pieces' embeddings (vectors in low-dimensional continuous space) from music playing records and corresponding metadata and infer users' general and contextual preference for music from their playing records with the learned embedding.

The effectiveness of existing top-N recommendation methods decreases as the sparsity of the datasets increases. To alleviate this problem, Kabbur et al. presents an item-based method for generating top-N recommendations that learns the item- item similarity matrix as the product of two low dimensional latent factor matrices. These matrices are learned using a structural equation modeling approach, wherein the value being estimated is not used for its own estimation.

CHAPTER 3: DATASETS

3.1 MILLION SONG DATASET

3.1.1 INTRODUCTION

The Million Song Dataset, a freely-available collection of audio features and metadata for a million contemporary popular music tracks that were legally available to The Echo Nest. The songs are representative of recent western commercial music.

The core of the dataset comes from The Echo Nest API. This online resource provides metadata and audio analysis for millions of tracks and powers many music applications on the web, smart phones, etc.

The MSD contains audio features and metadata for a million contemporary popular music tracks. It contains:

- 280 GB of data
- 1, 000, 000 songs/files
- 44, 745 unique artists
- 7, 643 unique terms (Echo Nest tags)
- 2, 321 unique musicbrainz tags
- 43, 943 artists with at least one term
- 2, 201, 916 asymmetric similarity relationships
- 515, 576 dated tracks starting from 1922

3.1.2 USAGE

Since its inception, the Million Song Dataset has been used in various applications. Some of them are as follows:

Artist Recognition:

Recognizing the artist from the audio is a straightforward task that provides a nice showcase of both audio features and machine learning. In the MSD, a reasonable target is the 18, 073 artists that have at least 20 songs in the dataset.

Automatic Music Tagging:

The Echo Nest provides tags (called "terms") at the artist level, and we also retrieved the few terms provided by musicbrainz. Although less studied, the correlation between tags and metadata could be of great interest in a commercial system. Certain "genre tags", such as "disco", usually apply to songs released in the 70s. There are also correlations between artist names and genres; you can probably guess the kind of music the band Disembowelment plays.

Cover Song Recognition:

Cover song recognition has generated many publications in the past few years. One motivation behind this task is the belief that finding covers relies on understanding something deeper about the structure of a piece. This is done in association with the SecondHandSong dataset.

Recommendation:

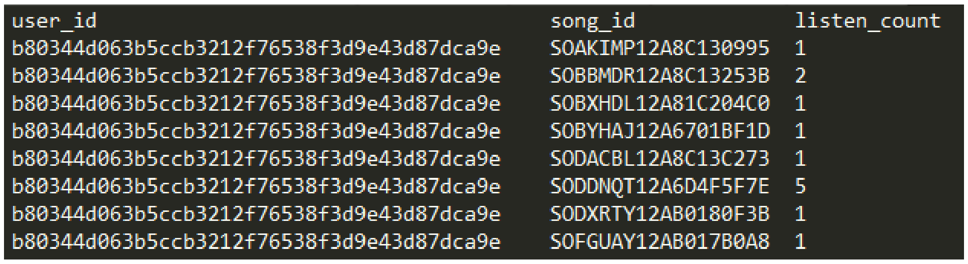
Music recommendation and music similarity are perhaps the best-studied areas in MIR. One reason is the potential commercial value of a working system. So far, content-based systems have fallen short at predicting user ratings when compared to collaborative filtering methods. One can argue that ratings are only one facet of recommendation (since listeners also value novelty and serendipity), but they are essential to a commercial system.

Ricky Martin	Weezer
Christina Aguilera	The Smashing Pumpkins
Shakira	Foo Fighters
Jennifer Lopez	Green Day

Figure 3.1: Some Similar Artists according to Echo Nest

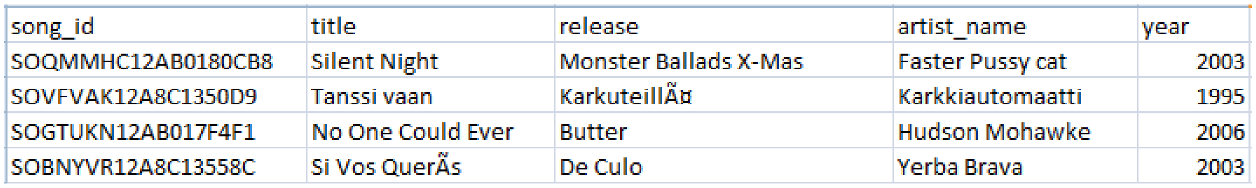
3.1.3 DATASET USED IN THE PROJECT:

The dataset that we use consists of 2 files: **triplet_file** and **metadat_file**. The triplet_file contains user_id, song_id and listen time. The metadat_file contains song_id, title, release_by and artist_name.



```
user_id                                    song_id             listen_count
b80344d063b5ccb3212f76538f3d9e43d87dca9e    SOAKIMP12A8C130995  1
b80344d063b5ccb3212f76538f3d9e43d87dca9e    SOBBMDR12A8C13253B  2
b80344d063b5ccb3212f76538f3d9e43d87dca9e    SOBXHDL12A81C204C0  1
b80344d063b5ccb3212f76538f3d9e43d87dca9e    SOBYHAJ12A6701BF1D  1
b80344d063b5ccb3212f76538f3d9e43d87dca9e    SODACBL12A8C13C273  1
b80344d063b5ccb3212f76538f3d9e43d87dca9e    SODDNQT12A6D4F5F7E  5
b80344d063b5ccb3212f76538f3d9e43d87dca9e    SODXRTY12AB0180F3B  1
b80344d063b5ccb3212f76538f3d9e43d87dca9e    SOFGUAY12AB017B0A8  1
```

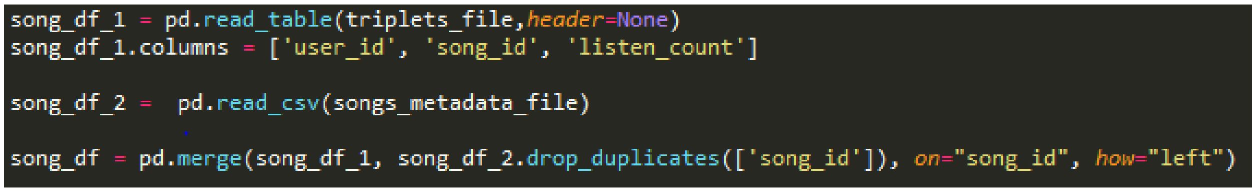
Figure 3.2: A snap of triplets_file



song_id	title	release	artist_name	year
SOQMMHC12AB0180CB8	Silent Night	Monster Ballads X-Mas	Faster Pussy cat	2003
SOVFVAK12A8C1350D9	Tanssi vaan	KarkuteillÃ¤	Karkkiautomaatti	1995
SOGTUKN12AB017F4F1	No One Could Ever	Butter	Hudson Mohawke	2006
SOBNYVR12A8C13558C	Si Vos QuerÃ s	De Culo	Yerba Brava	2003

Figure 3.3: A snap of metadat_file

The first thing that we did was to integrate our dataset, which is very important every time we want to build a data processing pipeline. To integrate both triplet_file and metadata_file, a popular Python library called **pandas** was used.



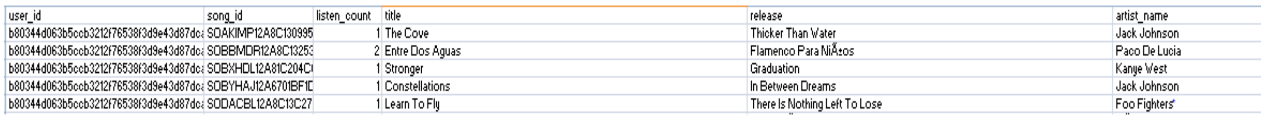
```
song_df_1 = pd.read_table(triplets_file,header=None)
song_df_1.columns = ['user_id', 'song_id', 'listen_count']

song_df_2 =  pd.read_csv(songs_metadata_file)

song_df = pd.merge(song_df_1, song_df_2.drop_duplicates(['song_id']), on="song_id", how="left")
```

Figure 3.4: Combining the datasets

We read the metadat_file and combine the metadata_file with triplets_file. Upon combining 2 or more datasets, there will be duplicate columns. Here we drop the duplicates between 2 datasets using song_id.



user_id	song_id	listen_count	title	release	artist_name
b80344d063b5ccb3212f76538f3d9e43d87dca	SOAKIMP12A8C130995	1	The Cove	Thicker Than Water	Jack Johnson
b80344d063b5ccb3212f76538f3d9e43d87dca	SOBBMDR12A8C13253	2	Entre Dos Aguas	Flamenco Para NiÃ±os	Paco De Lucia
b80344d063b5ccb3212f76538f3d9e43d87dca	SOBXHDL12A81C204C	1	Stronger	Graduation	Kanye West
b80344d063b5ccb3212f76538f3d9e43d87dca	SOBYHAJ12A6701BF1D	1	Constellations	In Between Dreams	Jack Johnson
b80344d063b5ccb3212f76538f3d9e43d87dca	SODACBL12A8C13C27	1	Learn To Fly	There Is Nothing Left To Lose	Foo Fighters

Figure 3.5: After merging the datasets

CHAPTER 4: METHODOLOGY

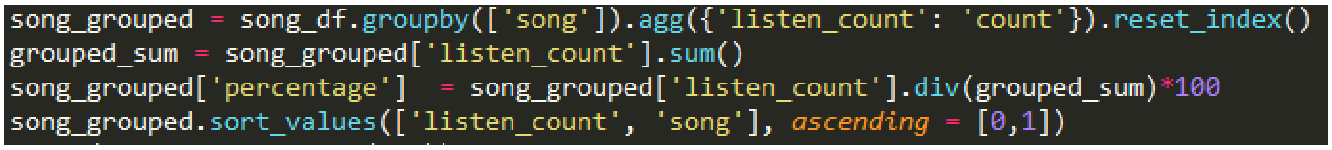
We implement 3 algorithms in our project. We then compare their results with each other and with some other popular algorithms in use in Music Recommendation field. The algorithms implemented are as follows:

- Popularity Based Recommender
- Item Similarity based Collaborative filtering model based Recommender
- Factored Item Similarity Model (FISM) based Recommender

4.1 POPULARITY BASED RECOMMENDER

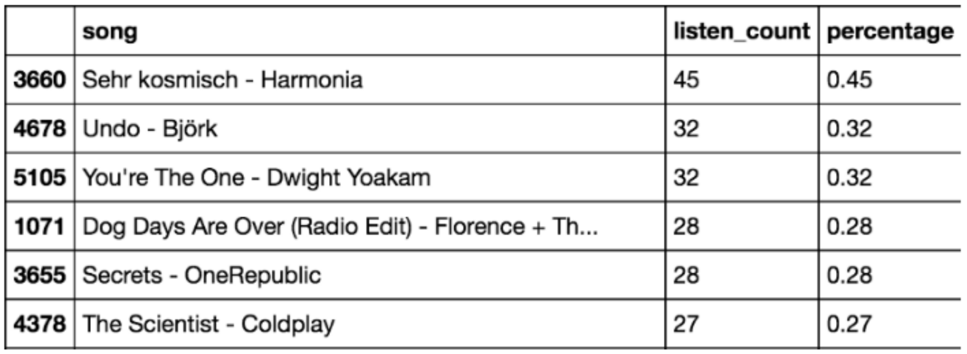
This is a naïve approach to build a recommender system. It doesn’t reflect a personalized system.

We first group the song_df by number of listen_count ascending. Then we calculate the group_sum by summing the listen_count of each song. Then add a new column called percentage and calculate this percentage by dividing the listen_count by the sum of listen_count of all songs and then multiply by 100. The last line lists the song in the ascending order of popularity for a given song.



```
song_grouped = song_df.groupby(['song']).agg({'listen_count': 'count'}).reset_index()
grouped_sum = song_grouped['listen_count'].sum()
song_grouped['percentage']  = song_grouped['listen_count'].div(grouped_sum)*100
song_grouped.sort_values(['listen_count', 'song'], ascending = [0,1])
```

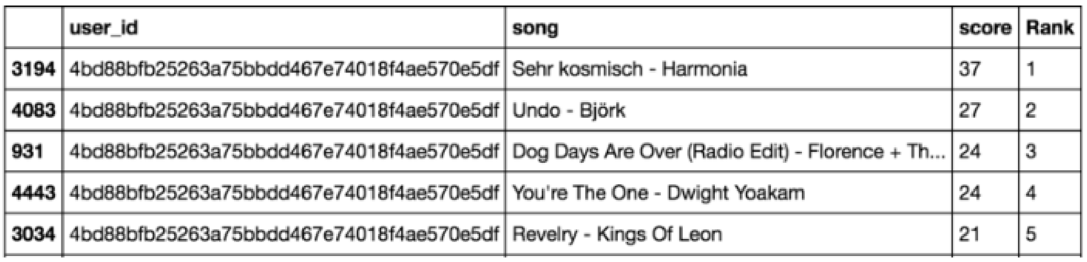
Figure 4.1: Code Snippet



	song	listen_count	percentage
3660	Sehr kosmisch - Harmonia	45	0.45
4678	Undo - Björk	32	0.32
5105	You're The One - Dwight Yoakam	32	0.32
1071	Dog Days Are Over (Radio Edit) - Florence + Th...	28	0.28
3655	Secrets - OneRepublic	28	0.28
4378	The Scientist - Coldplay	27	0.27

Figure 4.2: Dataset after Transformation

This system is a naive approach and not personalized. It first gets a unique count of user_id (ie the number of time that song was listened to in general by all user) for each song and tag it as a recommendation score. We then accept a user_id and output the top-N songs for that user.



	user_id	song	score	Rank
3194	4bd88bfb25263a75bbdd467e74018f4ae570e5df	Sehr kosmisch - Harmonia	37	1
4083	4bd88bfb25263a75bbdd467e74018f4ae570e5df	Undo - Björk	27	2
931	4bd88bfb25263a75bbdd467e74018f4ae570e5df	Dog Days Are Over (Radio Edit) - Florence + Th...	24	3
4443	4bd88bfb25263a75bbdd467e74018f4ae570e5df	You're The One - Dwight Yoakam	24	4
3034	4bd88bfb25263a75bbdd467e74018f4ae570e5df	Revelry - Kings Of Leon	21	5

Figure 4.3: Top 5 Recommendations for the given user id

Since this is the naive approach, the recommendation is not personalized and will be the same for all users.

This approach is really beneficial when we need to output recommendations for a new user who has just signed up on our platform (Eg: YouTube). Since the user has not used our platform yet, we cannot make any personalized recommendations. So, it is really convenient to show him/her the most popular items as recommendations.

4.2 Item Similarity based Collaborative filtering model-based Recommender

Recall that recommender system is divided into 2 types: *content based* and *collaborative based*. Content based system predicts what a user like based on what that user like in the past. Collaborative based systems predict what a particular user like based on what other similar users like. Most companies like Netflix and Hulu use the hybrid approach, which provide recommendation based on the combination of what content a user like in the past as well as what other similar user like.

Memory-based collaborative filtering can be divided into two main approaches: user-item filtering and item-item filtering.

Item-item filtering approach involves defining a co-occurrence matrix based on a song a user likes. We are seeking to answer a question, for each song, what a number of time a user, who have listened to that song, will also listen to another set of other songs. To further simplify this, based on what you like in the past, what other similar song that you will like based on what other similar user have liked.

We calculate a weighted average of the scores in cooccurence matrix for all user song. This cooccurence matrix will tend to be sparse matrix because it's not possible to predict if a user like a particular song, whether or not he/she will like a million other song. The possibility is so vast. Using our model, we will be able to predict the list of song that a user will like.

4.2.1 ALGORITHM:

1. For the given user, get the list of all songs that he has listened to. Let us call it user_songs.
2. Create a co-occurrence matrix of size len(user_songs) X len(all_songs)
3. Calculate similarity between user songs and all unique songs in the training data and fill the co-occurrence matrix.
4. For each song 'I' in all_songs:

   4.1 Calculate unique listeners of song I

   4.2 For each song 'j' in user_songs:

```
    4.2.1 Get unique listeners of song 'j'
    4.2.2 Calculate the intersection and union of listeners of song 'j' and song 'I'
    4.2.3 If intersection !=0
            co-occurrence matrix[j,i]= len(intersection)/len(union)
    4.2.4 else
            co-occurrence matrix[j,i]=0
```

5. Calculate a weighted average of the scores in the co-occurrence matrix for all user songs.
6. Sort it in descending order and select the first 'N' values.

Relevance:

CF based methods are very versatile and can easily be applied to any domain, be it music recommendation or artist recommendation etc. They work best when the user space is large and not sparse.

But the matrix formed is usually sparse since a user rates only few of the available items. Also, they suffer from "new item" problem much more than a content based system.

4.3 FACTORED ITEM SIMILARITY MODEL BASED RECOMMENDER:

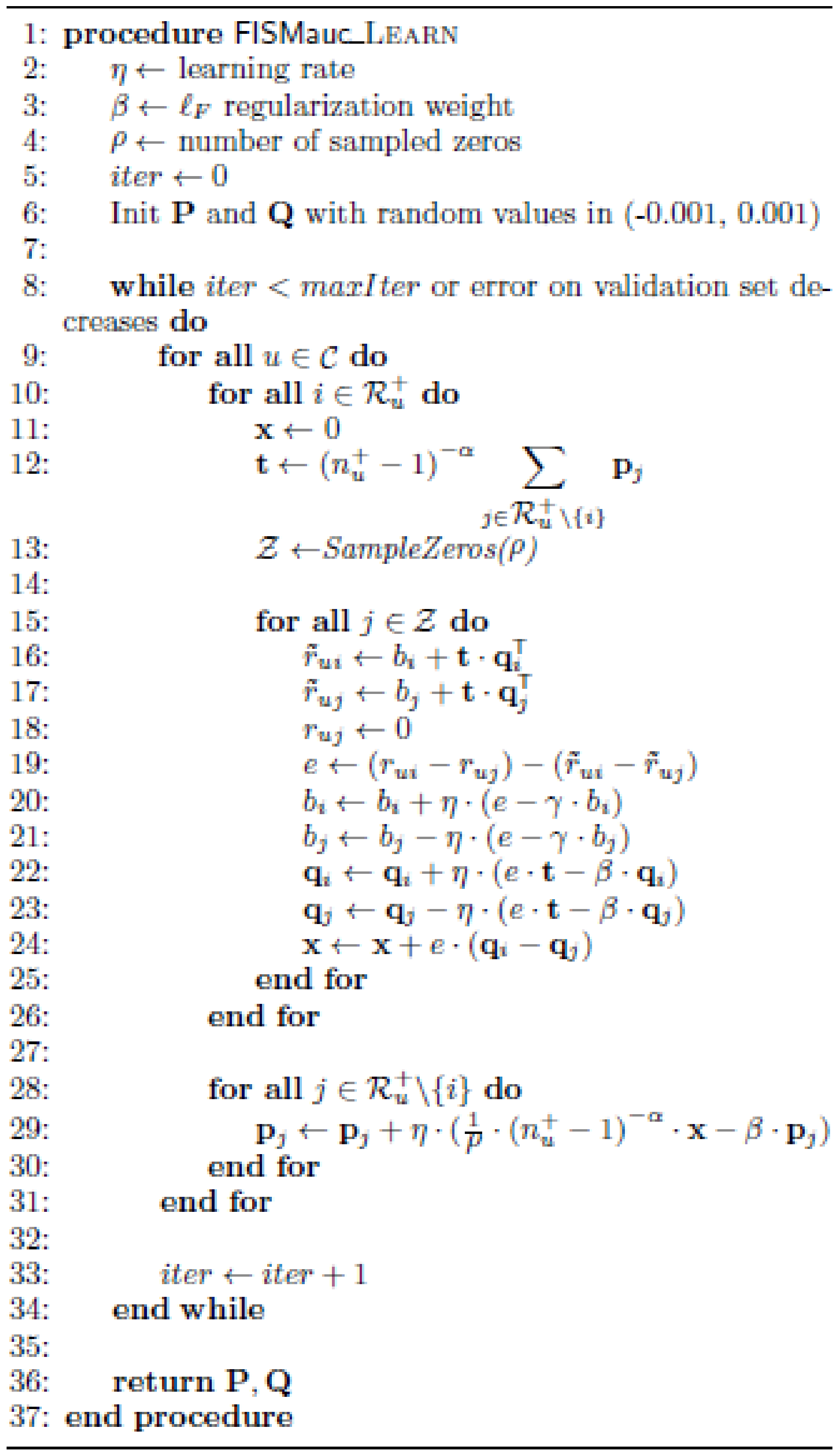
4.3.1 INTRODUCTION:

FISM is an item-based method for generating top-N recommendations that learns the item-item similarity matrix as the product of two low dimensional latent factor matrices. These matrices are learned using a structural equation modeling approach, wherein the value being estimated is not used for its own estimation. This factored representation of the item-item similarity matrix allows FISM to capture and model relations between items even on very sparse datasets.

4.3.2 MOTIVATION:

In real world scenarios, users typically provide feedback (purchase, rating or review) to only a handful of items out of possibly thousands or millions of items. This results in the user-item rating matrix becoming very sparse. Methods like ItemKNN which rely on learning similarities between items, fail to capture the dependencies between items that have not been co-rated by at least one user. Methods based on matrix factorization, alleviate this problem by projecting the data onto a low dimensional space, thereby implicitly learning better relationships between the users and items (including items which are not co-rated). It implicitly helps to learn transitive relations between items. Diagonal entries in the item similarities matrix correspond to including an item's own value while computing the prediction for that item. FISM explicitly excludes the diagonal entries while estimating.

4.3.3 FISM ALGORITHM:



```
procedure FISMauc_LEARN
    $\eta \leftarrow$ learning rate
    $\beta \leftarrow \ell_F$ regularization weight
    $\rho \leftarrow$ number of sampled zeros
    $iter \leftarrow 0$
    Init $\mathbf{P}$ and $\mathbf{Q}$ with random values in (-0.001, 0.001)

    while $iter < maxIter$ or error on validation set decreases do
        for all $u \in \mathcal{C}$ do
            for all $i \in \mathcal{R}_u^+$ do
                $\mathbf{x} \leftarrow 0$
                $\mathbf{t} \leftarrow (n_u^+ - 1)^{-\alpha} \sum_{j \in \mathcal{R}_u^+ \setminus \{i\}} \mathbf{p}_j$
                $\mathcal{Z} \leftarrow SampleZeros(\rho)$

                for all $j \in \mathcal{Z}$ do
                    $\tilde{r}_{ui} \leftarrow b_i + \mathbf{t} \cdot \mathbf{q}_i^\mathsf{T}$
                    $\tilde{r}_{uj} \leftarrow b_j + \mathbf{t} \cdot \mathbf{q}_j^\mathsf{T}$
                    $r_{uj} \leftarrow 0$
                    $e \leftarrow (r_{ui} - r_{uj}) - (\tilde{r}_{ui} - \tilde{r}_{uj})$
                    $b_i \leftarrow b_i + \eta \cdot (e - \gamma \cdot b_i)$
                    $b_j \leftarrow b_j - \eta \cdot (e - \gamma \cdot b_j)$
                    $\mathbf{q}_i \leftarrow \mathbf{q}_i + \eta \cdot (e \cdot \mathbf{t} - \beta \cdot \mathbf{q}_i)$
                    $\mathbf{q}_j \leftarrow \mathbf{q}_j - \eta \cdot (e \cdot \mathbf{t} - \beta \cdot \mathbf{q}_j)$
                    $\mathbf{x} \leftarrow \mathbf{x} + e \cdot (\mathbf{q}_i - \mathbf{q}_j)$
                end for
            end for

            for all $j \in \mathcal{R}_u^+ \setminus \{i\}$ do
                $\mathbf{p}_j \leftarrow \mathbf{p}_j + \eta \cdot (\frac{1}{\rho} \cdot (n_u^+ - 1)^{-\alpha} \cdot \mathbf{x} - \beta \cdot \mathbf{p}_j)$
            end for
        end for

        $iter \leftarrow iter + 1$
    end while

    return $\mathbf{P}, \mathbf{Q}$
end procedure
```

4.3.4 EXPLANATION:

Variables Used:

K: The reduced dimension of the Factor matrices.

MaxIter: The maximum number of times the algorithm runs.

η: The Learning Rate

RegU, RegI, RegB: Regularization weights used for user matrix, item matrix and bias matrix respectively.

Rho 'ρ': Number of zeroes to be sampled.

Alpha 'α': A user specifies parameter between 0 and 1.

P: NumPy matrix of dimension |user| X K. It is the user matrix. Initially assigned with random values.

Q: NumPy matrix of dimension |item| X K. It is the item matrix. Initially assigned with random values.

Bi: The Bias Matrix. Initially assigned with random values.

Name2id: A default dictionary. It provides a valid id to all the distinct users, songs and artists present in the dataset.

Id2name: A default dictionary. It provides a reverse mapping from distinct ids to the various names.

Listened: A default dictionary. It maps users and the songs that they have listened to.

UserRecord: A default dictionary. It maps user to user data. It is of the form: {user: record 1, record2]}

4.3.5 ALGORITHM:

Start procedure

```
1. While iteration < maxIter:
2. For each user in userRecord:
   2.1 nu=len(userRecord[user])
   2.2 coef = pow(nu - 1, -alpha)
       sum_Pj = A numPy array of dimension k assigned with 0 initially
   2.3 for each item in userRecord[user]:
       2.3.1 j= id of item retrieved from name2id
       2.3.2 sum_Pj+= P[j] //sum of the values corresponding to items rated by the user.
   End for
   2.4 for each item in userRecord[user];
       2.4.1: x=A numPy array of dimension k assigned with 0 initially
              I: item id
       2.4.2: Get 'ρ' items that the user has not rated
       2.4.3: r_ui=coef*(sum_Pj-P[i]).Q[i] +Bi[i]
              r_uj=coef*(sum_Pj-P[j]).Q[j] +Bi[j]
              error = 1-(r_ui-r_uj)
              loss += 0.5*error^2
           Update Bias Matrix
              Bi[i]+=lRate*(error-regB*Bi[i])
              Bi[j]+=lRate*(error-regB*Bi[j])
```

```
                Update Item Matrix
                  Q[i]+=.lRate*(error*coef*(sum_Pj-P[i])-regI*Q[i])
                  Q[j]+=.lRate*(error*coef*(sum_Pj-P[j])-regI*Q[j])
            End for
            Update User Matrix
            2.5 for each track in userRecord[user]:
                2.5.1 j= get id of track
                2.5.2 P[j]+=lRate*(1/float(rho)*coef*X[ind]-regI*P[j])
                End for
            End for
            2.6 loss += regU*(P^2).sum() + regI*(Q^2).sum() + regB*(Bi.dot(Bi))
            2.7 iteration += 1
            2.8 deltaLoss = (lastLoss-loss)
            2.9 if deltaLoss < 10^-3
                Break
            2.10 else update Learning Rate
  end While
Return P, Q
End procedure
```

CHAPTER 5: RESULTS & DISCUSSIONS

5.1 METRICS USED

Precision

Precision (P) is defined as the number of true positives (T_p) over the number of true positives plus the number of false positives (F_p).

$$P = \frac{T_p}{T_p+F_p}$$

Recall

Recall (R) is defined as the number of true positives (T_p) over the number of true positives plus the number of false negatives (F_n)

$$R = \frac{T_p}{T_p+F_n}$$

F_1 score

In statistical analysis of binary classification, the **F_1 score** (also **F-score** or **F-measure**) is a measure of a test's accuracy. It considers both the precision *p* and the recall *r* of the test to compute the score: *p* is the number of correct positive results divided by the number of all positive results returned by the classifier, and *r* is the number of correct positive results divided by the number of all relevant samples (all samples that should have been identified as positive). The F_1 score is the harmonic average of the precision and recall, where an F_1 score reaches its best value at 1 (perfect precision and recall) and worst at 0.

F_1 score is the harmonic mean of precision and recall:

$$F_1 = \frac{2}{\frac{1}{\text{recall}} + \frac{1}{\text{precision}}} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

5.2 GRAPHS

5.2.1 ON VARYING VARIOUS PARAMETERS USED IN FISM:

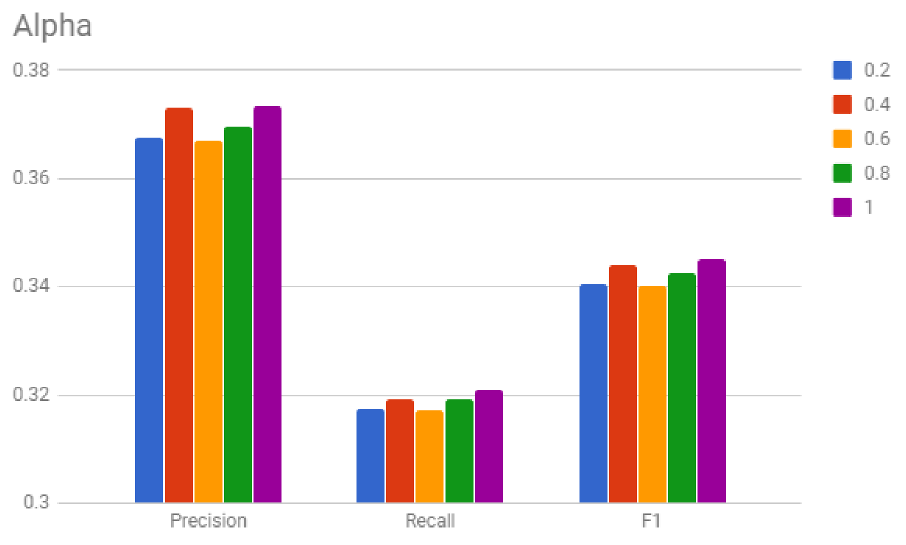


Figure 5.1: Effect of varying Alpha

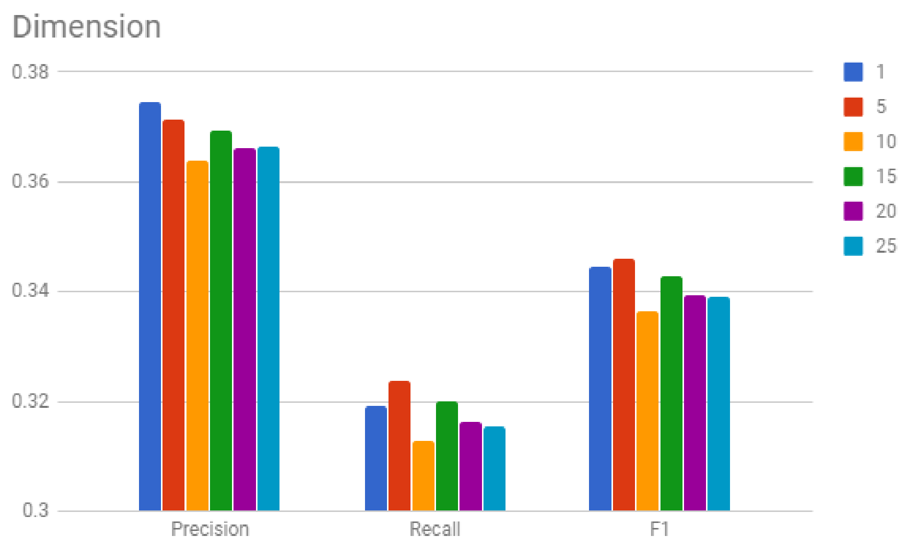


Figure 5.2: Effect of varying the dimensionality of factor matrices.

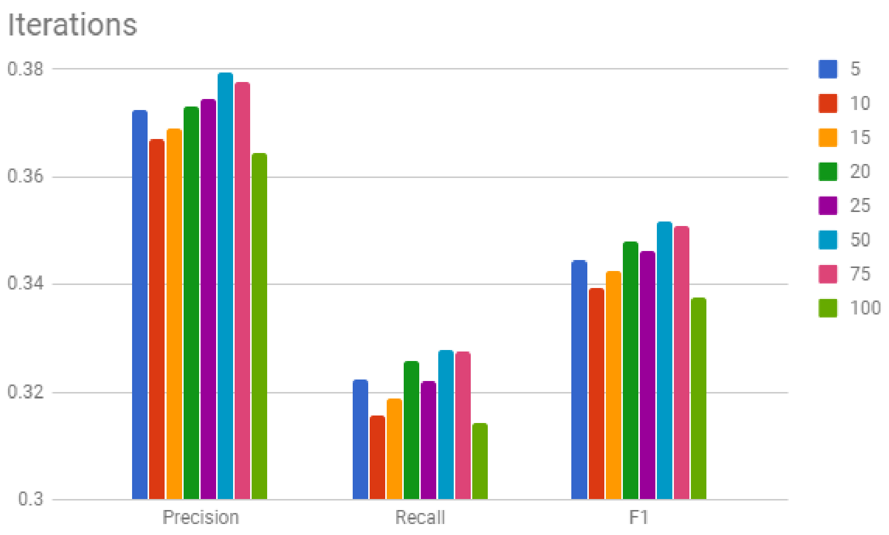


Figure 5.3: Varying number of iterations

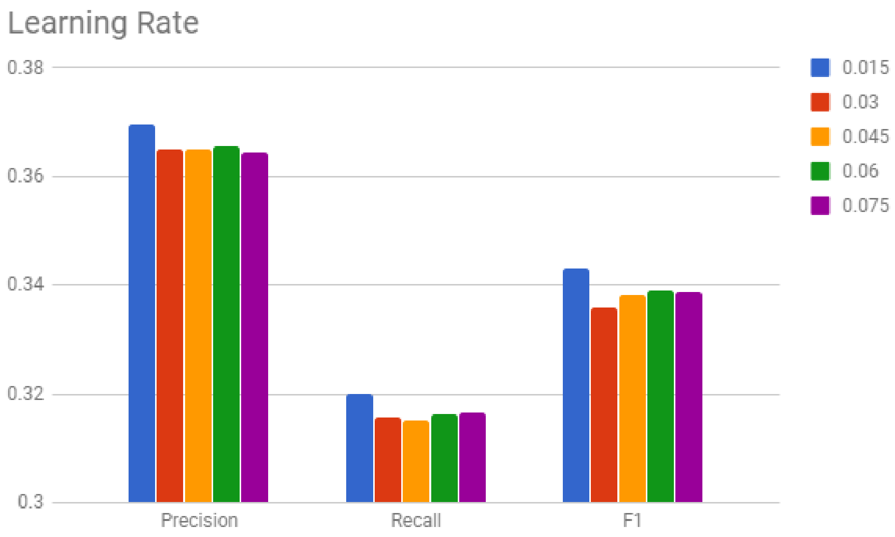


Figure 5.4: Varying the Learning Rate

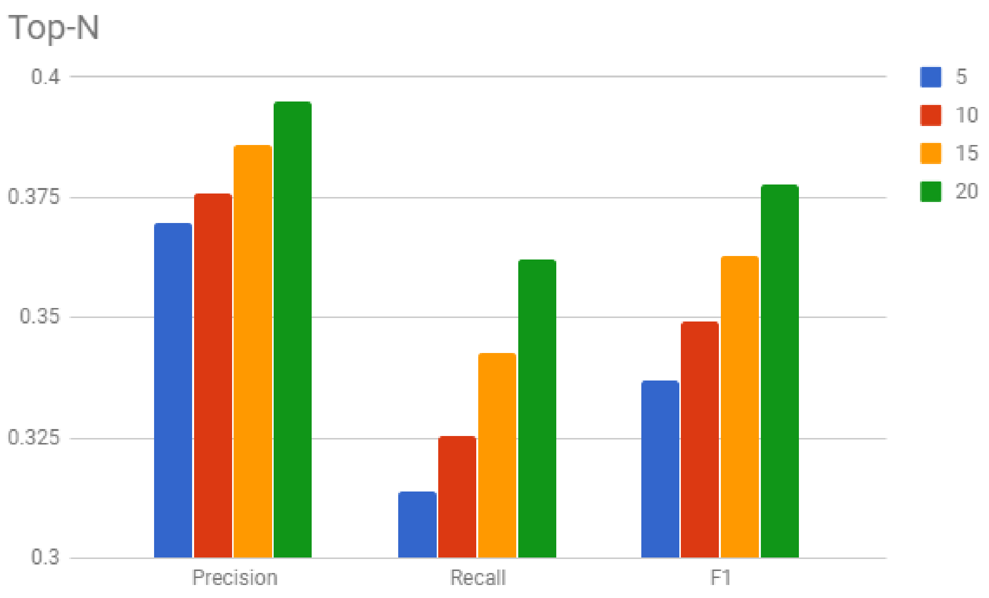


Figure 5.5: Varying N in Top-N Recommendations

5.2.2 COMPARING FISM WITH OTHER ALGORITHMS:

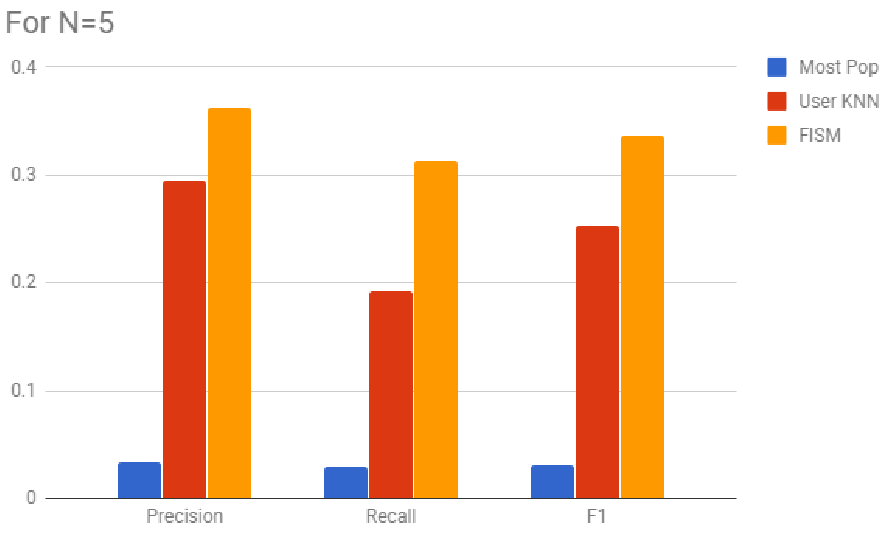


Figure 5.6: Comparing for Top 5 Recommendations

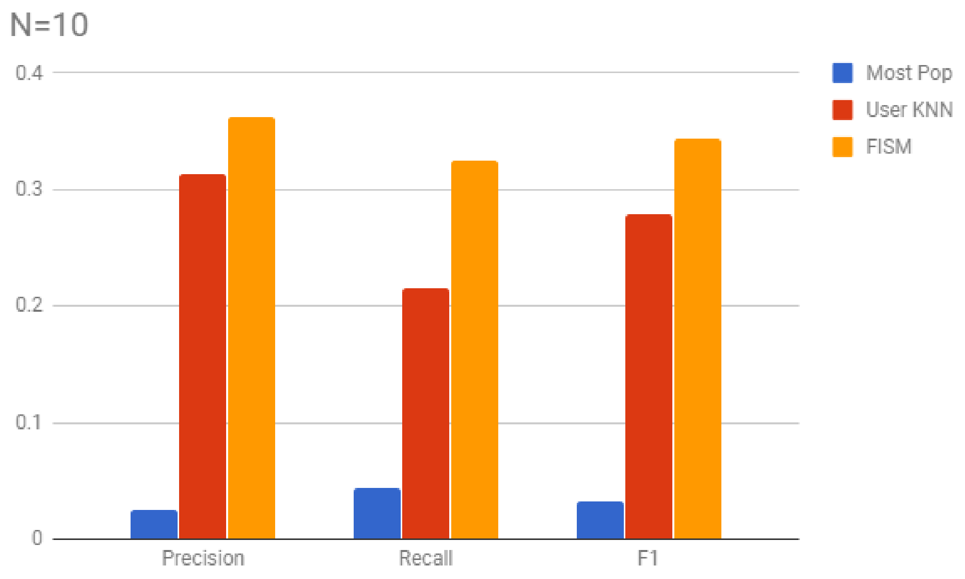


Figure 5.7: Comparing for Top 10 Recommendations

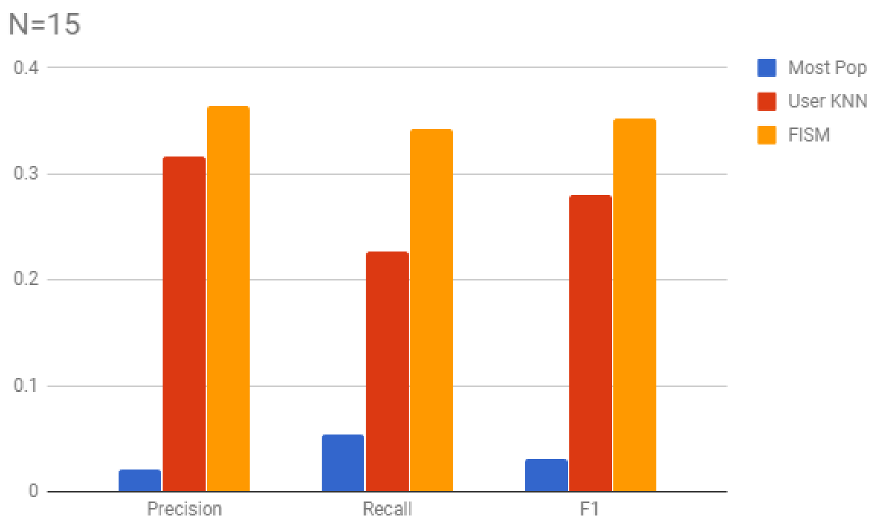


Figure 5.8: Comparing for Top 15 Recommendations

CHAPTER 6: CONCLUSION

FISM learns the item similarities as the product of two matrices, allowing it to generate high quality recommendations even on sparse datasets. The factored representation is estimated using a structural equation modeling approach

- FISM outperforms the common recommendation algorithms being used currently.
- The performance gaps increase as the datasets become sparser.
- It leads to better estimations as the number of factors increase.
- The performance varies a lot with the variance of N in Top N recommendations

REFERENCES

1. Kabbur et al., FISM: Factored Item Similarity Models for Top-N Recommender Systems, KDD 2013.
2. Ellis et al., The Million Song Dataset, ISMIR 2011
3. Kang et al., Top-N Recommender System via Matrix Completion, AAAI 16
4. Cheng et al., Exploiting Music Play Sequence for Music Recommendation, IJCAI-17
5. Zhang et al., Collaborative User Network Embedding for Social Recommender Systems, SIAM 15
6. L.S. Chen et al., Developing recommender systems with the consideration of product profitability for sellers Int J Inform Sci, 178 (4) (2008), pp. 1032 1048
7. G. Adomavicius et al., Toward the next generation of recommender system. A survey of the state-of-the-art and possible extensions IEEE Trans Knowl Data Eng, 17 (6) (2005), pp. 734-749
8. Ziegler CN et al., Improving recommendation lists through topic diversification. In: Proceedings of the 14th international conference on World Wide Web; 2005. p. 22–32
9. Ghazantar et al., A scalable accurate hybrid recommender system. In: the 3rd International conference on knowledge discovery and data mining (WKDD 2010), IEEE Computer Society, Washington, DC, USA.
10. Ziegler et al., Taxonomy-driven computation of product recommendations. In: Proceedings of the 13th international conference on information and knowledge management (CIKM ‘04), Washington, DC, USA; 2004. p. 406–15.
11. R. Burke Hybrid web recommender systems The adaptive web, LNCS 4321, Springer, Berlin Heidelberg, Germany (2007), pp. 377-408.