San Jose State University
SJSU ScholarWorks

Master's Projects | Master's Theses and Graduate Research

4-1-2012

Session Based Music Recommendation using Singular Value Decomposition (SVD)

Ruchit V. Mehta

Follow this and additional works at: http://scholarworks.sjsu.edu/etd_projects

Recommended Citation
Mehta, Ruchit V., "Session Based Music Recommendation using Singular Value Decomposition (SVD)" (2012). *Master's Projects.* Paper 237.

This Master's Project is brought to you for free and open access by the Master's Theses and Graduate Research at SJSU ScholarWorks. It has been accepted for inclusion in Master's Projects by an authorized administrator of SJSU ScholarWorks. For more information, please contact scholarworks@sjsu.edu.

Session Based Music Recommendation using Singular Value Decomposition (SVD)

A Writing Project

Presented to

The Faculty of the Department of Computer Science

San Jose State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

by

Ruchit V. Mehta
Spring 2012

Copyright © 2012

Ruchit V. Mehta

All Rights Reserved

SAN JOSE STATE UNIVERSITY

The Undersigned Project Committee Approves the Project Titled

Session Based Music Recommendation using Singular Value Decomposition (SVD)

by

Ruchit V. Mehta

Approved for the Department of Computer Science

Dr. Robert Chun	Department of Computer Science	Date
Dr. Soon Tee Teoh	Department of Computer Science	Date
Mr. Pradip Gajjar	Netflix	Date

Approved for the University

Associate Dean Office of Graduate Studies and Research Date

ABSTRACT

Technology in the modern world has over-simplified the access to information. At a click of a button we have volumes of music accessible on the Internet. Paradoxically, the abundance of available options has only made music discovery and recommendations a complex problem to solve. With huge collections of songs in the online digital libraries, finding a song or an artist is not a problem. However, an actual problem is what to look for that will intuitively satisfy a user's need. There exists multitude of recommendation algorithms, but many of them do not consider the contextual information in which a user listens to a song. This information is not quantifiable, but it needs to be extracted by some methods so as to provide an additional facet to music recommendations. There is active research in music recommendation to identify various factors that can influence the choice of a song. Songs that are often played together have some inherent correlations between them which at first, does not seem obvious. Thus, an approach is proposed that can extract information using a linear algebraic approach and generate context-aware music recommendations.

ACKNOWLEDGEMENTS

I am thankful to my project advisor, Dr. Robert Chun, for giving me an opportunity to work under his guidance. Dr. Chun's encouragement, input and support was very valuable to me. He is an educationalist in true sense. I am also grateful to my committee members Dr. Soon Tee Teoh and Mr. Pradip Gajjar for their timely input, advice and motivation. My committee members have guided me well to complete this project and helped me polish all the aspects of it, which I have presented in this report. It would have been difficult process without support from all of you, and without your persistent guidance this project would not have been possible.

Thank you.

Ruchit Mehta

TABLE OF CONTENTS

- INTROODUCTION ……. 1

1.1 Problem Statement …..…… 1

1.2 The Approach ……. 1

1.3 The Long Tail ……. 2

1.4 Literature Survey ……. 3

- BACKGROUND …….. 5

2.1 Music Recommendation …….. 5

2.1.1 Collaborative Filtering …….. 6

2.1.2 Content Based Filtering …….. 9

- DESIGNAND IMPLEMENTATION …… 10

3.1 Challenges in collaborative Filtering ……. 10

3.1.1 Data Sparsity ....……. 11

3.1.2 Scalability ……. 12

3.1.3 Synonymy ……. 13

3.2 Singular Value Decomposition ……. 13

3.2.1 Statement of Theorem ……..... 14

3.3 Last.fm Dataset ……..... 17

3.4 Realization of Session Based Algorithm using SVD ……..... 18

- EXPERIMENT ……. 23
- ANALYSIS AND EVALUATION ……. 28

5.1 Offline Analysis ……. 28

5.2 Coverage of the Algorithm ……. 32

5.3 Cluster Formation and k-value Evaluation ……. 33

- CONCLUSION ……. 35
- FUTURE WORK ……. 36
- REFERENCES ……. 37

LIST OF FIGURES

Figure 1 Simple user-based recommendation system showing the flow of information 7
Figure 2 A pseudo code to implement user based collaborative filtering approach 8
Figure 3 Pseudo code to implement Item based collaborative filtering approach 8
Figure 4 Plotting of regression line which is an approximation of all the data points 15
Figure 5 Alternative Regression line for approximating all the data points 16
Figure 6 User Record tuple in the dataset 17
Figure 7 Algorithmic Flow 19
Figure 8 SVD decomposition of Matrix X into sub matrices U, S and V 20
Figure 9 Eigen values in Matrix S are arranged in decreasing order of their importance 21
Figure 10 An example of session generated with the threshold value of 90 minutes 23
Figure 11 User-Item matrix after applying SVD decomposition 25
Figure 12 Applying clustering to Matrix V 26
Figure 13 True Positive Values for session based and unified session approach 29
Figure 14 False Positive Values for session based and unified session approach 30
Figure 15 False Negative Values for session based and unified session approach 30
Figure 16 Precision Values for session based and unified session approach 31
Figure 17 Recall Values for session based and unified session approach 31
Figure 18 Coverage of algorithm for different number of songs 32
Figure 19 Song feature in Matrix V 34

LIST OF TABLES

Table 1 A User-Item Matrix 11

Table 2 User –Item matrix for 4 users and 5 Songs 24

Table 3 Table to distinguish each of the recommended songs 28

LIST OF EQUATIONS

Equation 1 SVD Decomposition 11

Equation 2 SVD factorization equation after deciding threshold value k 21

Equation 3 Assigning weight to each song duals 27

Equation 4 Precision and Recall formulae 31

1. INTRODUCTION

1.1 PROBLEM STATMENT

The current scenario of music recommendation leaves something to be desired. Research to implement an algorithm that would address all the facets of music has been tried for quite some time; but it is unreasonable to expect a system to outperform human intuition, which is difficult to quantify. Choices are influenced by human behavior, thus we try to find important information in the multiple sessions of a user's listening history, and use it to make active recommendations. Sometimes, user data in its unprocessed form does not reveal important relationships between two users/items. Also, considering the field of music, where the size of music library is ever increasing, we need a viable linear algebraic approach that can address the computationally intensive approach of recommendation. Gathering data does not require mush effort; however processing Big Data is problematic. An approach is required which can infer intelligently but using less amount of bits. Thus, the goal is to implement and analyze an algorithm which uses the matrix factorization approach that can surface latent features in the user's listening history and could enhance the listening experience.

1.2 THE APPROACH

The approach here is based on certain hypotheses about the factors that can affect the choices of users when they listen to songs. These factors are apparently not given much importance in contemporary music recommendations. The algorithm attempts to analyze user's history by breaking it down into small sessions and try to deduce the user’s listening context from the kind of songs they are listening to; or by deducing a listening template that the users follow when they listen to a certain genre of music during a different time of the day. Considering these factors, the

algorithm tries to create a more intelligent recommendation than simply suggesting a song or a band to listen to [12]. Generally, music recommenders suggest a song or an artist by plainly noting the user's behavior and the kind of songs they scan through; however, it is important to note that a user may not like all the albums of a particular artist or all the songs on a particular album. The recommendation system should work so that it will give an enriching experience to its listener. Music portal Last.fm is leveraged to collect information and meta-data for approximately thousand users. The meta-data contains all the statistics such as: day, date and time of play for each song, artist and album of the song, etc. Other deductions are induced as per the requirement of the modules used in the algorithm [18].

The idea is to combine different behavioral patterns and try to explore ideas that might affect the choices of the songs that users make while listening songs.

1.3 THE LONG TAIL

Digital music libraries are increasing by thousands of songs every day and many songs, if not most, are getting lost in this huge pile. Thus, there are large collections of songs that people hardly bump into and they tend to ignore them for the most part of their listening cycle. This anomaly of never listening to large amounts of songs is called The Long Tail [8]. The long tail describes the statistical dispersion of a low-frequency population that is followed up by a high-frequency population [21]. For the above use-case, a high-frequency population refers to the songs that a user most often listens to. The low-frequency songs are the ones that are rarely or never played. Hence, it is an important aspect of music recommenders to explore this long tail, and retrieve lost music.

1.4 LITERATURE SURVEY

The literature studied for the project can be broadly categorized into two categories: Knowledge discovery through Data Mining and Music Recommendation. The latter category represents the concept of studying different attributes of music, identifying habits of users, and other related domain knowledge in order to make acute and effectual music recommendations.

Maes and Sharadanand used “word of mouth” to make recommendations [3]. The authors demonstrate the use of Pearson correlation and artist-artist correlation to analyze similarities between the user ratings and user profiles. The system they designed gave each song a rating from 1 through 7, one being the lowest and 7 being the highest. To test the system and get acceptable outputs, they had to perform several recommendation iterations on the input data.

In another study, CAI et al. designed a system that would recommend music based on the blogs or text that a user is reading [4]. Some extra data (e.g., textual information) is required to match the lyrics of the songs with the user content and check the type of emotions that prevail in those blogs. The approach was mainly influenced by classical IR (Information retrieval) techniques which, the author tries to find the frequency of all the terms in the documents and then use Bayes classifier to describe the psychology of the words that can point to a particular emotion.

Kaji et al. used the user’s environment as an influential factor in making music recommendations [5]. The authors relied on the lyrics of the songs and the tags given by the users to theorize favorite songs to generate a playlist. The initial method used was the content-based approach, but gradually the system takes the user’s ratings to the previously suggested songs into consideration and thus modifies the recommendations.

Song recommendations can also be influenced by the way in which a user interacts with the music player. It is important to note what users like to listen to, and yet it is equally important to

understand what a user does not like. Since humans sometimes tend to know what they don't like more than compared to what they like. Pampalk et al. have used this information to make active recommendations in their study [7]. Skipping a song is therefore an important option to enable when designing a system as it can provide some user insight. Later on, the heuristic technique used to pick the next song depends on: the songs that match closest to the last song, using k-nearest neighbor.

Finally, just like the network topologies, the existing music network topologies are examined in order to reveal some interesting patterns in which humans perceive the correlation between songs [6]. All the songs, artists and bands that they listen to form the nodes of the graphs. Further details are then embedded into those graphs like degrees of nodes, directed or undirected graph etc., which then becomes the data-points on which clustering is performed.

2. BACKGROUND

2.1 MUSIC RECOMMENDATION

We are offered options among different things that we come across throughout a day. We hear song on a radio, see a movie, read about some books, or see different clothes/accessories. We form an opinion: we like them, don't like them or sometimes we don't even care. This happens unconsciously [15]. Although these all seem random, we inherently follow a pattern and we call it personal taste. We tend to like similar things. For example, if someone likes bacon-lettuce-tomato sandwiches, then there are good chances that that person will also like a club sandwich because they are very similar– only with turkey replacing the bacon. We follow these kinds of patterns inherently [23]. In the crux, recommendations are all about pattern recognition and finding similarities.

Music is omnipresent. It is no surprise that there are millions of songs at everyone's finger tips. In fact, given the number of songs, bands, and artists coming up, music listeners are overwhelmed by choices. They are always looking for ways to discover new music so that it will match their taste. This has given birth to the field of music recommendations. In the past few years, there have been many services like Pandora, Spotify, and Last.fm that have come up in order to find a perfect solution, but haven't been completely successful. Choices are influenced by interests, trust, and liking towards any particular object and these emotions are very difficult to quantify– especially for a machine or software [19]. Hence, it has been a very difficult experience for these service providers to give a fulfilling experience. Every music recommendations system works on a given set of hypotheses, which they believe will result in the effective recommendations.

There are two fundamental styles of music recommendations: Collaborative filtering and Content-based Filtering. The next section describes the two methodologies used by existing music recommenders. Although my approach is primarily based on Collaborative Filtering, I have discussed content based filtering too.

2.1.1 COLLABORATIVE FILTERING

It is an approach in which information is gathered about the users' preferences for any particular item (books, videos, news articles, songs, etc.). The knowledge captured is then structured and used against all the unknown items and make intelligent predictions that a user might enjoy. In collaborative filtering, the interaction between users and items is important. The system relies on the past history to derive a suitable model for an entity [16]. The historical data acts as an input to the system. The preference or user history can be derived in two ways:

a. **Explicit Ratings:**

   Here the user should be willing to express his/her preferences for an item. The preference can be a simple true (like) - false (dislike) method or it can be a rating system (e.g., Rate a book on the scale of 1-5). This method comes with an assumption that a user will be actively participating to provide the feedback. The data they provide is to the best of their knowledge as an induction of false data/ noise can hamper the performance of the algorithms [22].

b. **Implicit Ratings**

   These are the inferred ratings which are interpreted as a result of user interactions with the items. These are the subtle algorithms which are used by many web-portals that are

working behind the curtains. Data collected by this method needs to be pre-processed as there is a greater probability of noise in the data.

Collaborative filtering is very popular and is being used widely by companies like Amazon, Google, Yahoo, etc. Collaborative filtering methodology tries to find similarity between two users or items. It is independent of the attributes of those entities. Thus collaborative filtering is a content agnostic approach.

Collaborative filtering can be further be categorized into following two groups:

2.1.1.1 USER BASED COLLABORATIVE FILTERING

In the user based collaborative filtering approach, we use user ratings for each item in his profile to infer interests and make recommendations.

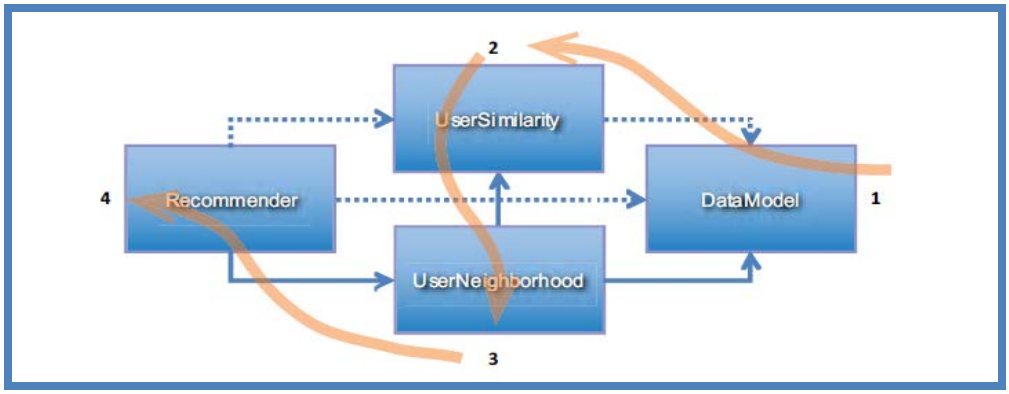


Figure 1: A simple user-based recommendation system showing the flow of information in such a system [25]

The crux of this approach is to find all the neighboring users for the current target user and try to fill in the missing pieces in order to guess the items that the target user would like. A simple user based collaborative filtering is shown in Figure 1 above.

```
for every other user w
  compute a similarity s between u and w
  retain the top users, ranked by similarity, as a neighborhood n
for every item i that some user in n has a preference for,
      but that u has no preference for yet
  for every other user v in n that has a preference for i
    compute a similarity s between u and v
    incorporate v's preference for i, weighted by s, into a running average
```

Figure 2: A pseudo code to implement user based collaborative filtering approach [25]

As shown in Fig. 2, the idea is to find similar users using various similarity measures such as cosine similarity, Pearson correlation coefficient similarity, and match their profiles for item discovery [24]. This kind of approach is useful when the system consists of huge collections of items as compared to the number of users, since it would be too costly to find similarities between items rather than users. A point in case is amazon.com where number of items exceeds number of users by a big margin.

2.1.1.2 ITEM BASED COLLABORATIVE FILTERING

In the item based collaborative approach, we construct item-profiles instead of user profiles and find similarities between any two given items using various measures like Euclidean distance similarity, Tanimoto coefficient similarity, and Log likelihood similarity. A simple approach to this kind of filtering is shown in Figure 3 below:

```
for every item i that u has no preference for yet
  for every item j that u has a preference for
    compute a similarity s between i and j
    add u's preference for j, weighted by s, to a running average
return the top items, ranked by weighted average
```

Figure 3: Pseudo code to implement Item based collaborative filtering approach

For any given item i, we compute its similarity with the one which is already present in the user profile to predict if the target item is worth recommending to the user or not. This type of approach is useful when new items are being added to the system too often.

2.1.2 CONTENT BASED FILTERING

In content-based filtering, we analyze the attributes or the content of a song in order to make recommendations. In the case of a song, we analyze the kind of instruments used, tempo, pitch of the song, and store all those information in a structured format. Now, when a user listens to a particular song, the system analyzes that song and finds the neighboring similar songs to make active recommendations. This approach is a content dependent approach because the methodology that is used to analyze or recommend songs would not work for analyzing books or videos since those items has different sets of attributes. Therefore, they should be approached differently. Pandora [2] is one of the music services that uses content based filtering for their music recommendations.

3. DESIGN AND IMPLEMENTATION

The section describes the challenges in designing the algorithm, the idea behind the use of singular value decomposition and the actual algorithmic approach.

3.1 CHALLENGES IN COLLABORATIVE FILTERING

A Recommendation system should be intuitive and consumer driven. It should be able to provide good results to gain the user's trust. The soul of the collaborative recommender system is in the users past history, i.e., user preferences and the history of like-minded users. The latter helps us to predict the unknown preferences of the new users. Using the data-points gathered from the above information tapestry, the system then plots a User - Item Matrix to find any correlation such that we can cluster with the nearest neighbor and return to the top N recommendations. Measurements like Cosine similarity, Euclidean distance, and Pearson correlation are used.

A typical Collaborative filtering system would have a data bank of the user's preferences for all the songs they have browsed or purchased. Essentially, we have a list of m users {u1, u2, u3 … um} along with a list of n songs {s1, s2, s3… sn}. As described previously, with the help of implicit or explicit ratings, user preferences are noted for all items i. The preference vectors of all the users are then converted to user-item matrix. Below is an example of such user-item matrix where we have captured the user preferences for an item with like-dislike ratings.

Here, the value in each cell represents the number of times a song has been listened by a user. The matrix shown here is moderately sparse, however, in a real world scenarios the matrices are much sparser than the one shown below in Table 1:

	Song 1	Song 2	Song 3	Song 4	Song 5	Song 6	Song 7	Song 8	Song 9
User 1	2	0	0	0	0	3	1	0	0
User 2	0	4	0	1	0	0	0	0	3
User 3	1	3	0	0	0	3	0	0	1
User 4	0	0	0	1	1	0	1	0	0

Table 1: A User – Item matrix

Processing and finding the important correlation in such sparse matrices is one of the challenges in the recommendation system; as generally, the number of songs greatly outnumbers the number of users. Some of the important challenges that need to be addressed and how SVD tries to overcome those problems are as discussed below:

3.1.1 DATA SPARSITY

While evaluating the Last.fm dataset, Data Sparsity was one of the foremost challenges that needed to be overcome. Since the dataset contained around 960,000 songs against only 1000 users, it was practically impossible for each user to listen to even half of the unique songs. Thus, each user vector was very light weight. One of the important factors that causes data sparsity problem is a cold start problem or a new-user / new-item problem. It is difficult to make any recommendations for a new user, because there is not a song listening history, nor user preferences in order to perform information retrieval algorithms and make recommendations. Also, the item preference vector is hugely sparse and unless that user starts providing input about their preferences, the system will have to make random recommendations. In the same way, new songs, bands or albums entered into the library cannot be suggested unless some users actively start using it and give explicit ratings. The Neighbor Transitivity is a problem that occurs with a sparse data set when the system cannot identify two similar users because they have not specified preferences for any of the same items. This can hamper the performance of the algorithm which banks on finding a similarity between two users.

In the classical information retrieval methodology we have seen the usage of Latent Semantic Indexing (LSI), which is based on Singular Value Decomposition (SVD). The dimensionality reduction technique which is the base of SVD is useful for recommendation systems. Here, for any user-item matrix, we remove the indifferent and insignificant data, which alone are not very important and only increase the vector calculations. Thus, with the help of SVD, we determine the similarity between users by mapping them in reduced space matrices.

3.1.2 SCALABILITY

Music recommendation domain contains an ever-growing library of songs. Many classical algorithms suffer the scalability issues, since there are thousands of users with millions of songs with ratings to map to. In such a case, techniques like SVD can be used to reduce dimension of the data and make computation easier to scale up along with the ever-increasing database [10]. An incremental system can be modeled without calculating the lower dimension data from scratch when new user data such as preference history or ratings are added. That makes the algorithm very scalable [9].

Pearson correlation CF algorithm (item-based algorithm) calculates similarities between co-rated items for a particular user instead of finding it between all the pairs of items [11]. In a similar fashion, Model-based CF algorithms rely on clustering all the highly similar data in the database. If any new data is introduced the iterative approach is cluster based on coercion parameters, although there are certain trade-offs between prediction performance and scalability. Thus, SVD provides an overall computation intensive, yet adaptive approach to compute good quality recommendations.

3.1.3 SYNONYMY

When there are songs, which are contextually very similar, but have different names and tags then that can lead to Synonymy. Modern recommender systems are indifferent in discovering this latent association and therefore the final product-rating indexes are clustered inappropriately. For example, a song is tagged with a ‘metallic’ tag while another song is tagged with a ’rock’ tag. Both the songs would sound very similar, but they would be perceived differently by the recommender system just because the tags given to them are different. Certainly, the degree of descriptiveness would vary from song to song, but Polysemy and Synonymy decreases the performance of the algorithms. There have been attempts to build a dictionary of related tags and to create intellectual and automatic expansion by inducing lyrics and instruments in the methodology, but it was limited to a very small increase in the performance and had potential trade-offs.

The Latent Semantic Indexing (LSI) in SVD is capable of addressing the synonymy problems. The resulting matrices after the factorization of the user-item association results in constituting a semantic space that places all the associated items close to each other and extracts associative patterns in data in order to ignore the smaller, and less important ones [12, 13].

3.2 SINGULAR VALUE DEOMPOSITION (SVD)

Singular Value Decomposition (SVD) is an approach where we factorize a matrix into a series of linear approximations. These approximations will expose the underlying structure of that matrix [14]. SVD can be expressed from three consistent viewpoints. First, SVD transforms a matrix of seemingly correlated variables into an uncorrelated one that provides a better understanding of the relationship between all the data points which might not be obvious in their original

formations. This is helpful because in some cases the relationships might be confusing; or it may suggest another relationship between 2 songs rather than what is apparent. Secondly, SVD is used to identify the relationship between various songs (mapped as data points in the matrix) and aligns the data in the product matrix so that the data points show the maximum variation. Thirdly, once we have figured out the vectors having most variations, it is possible to find the best approximations in the original data by using fewer dimensions. (Dimensions are extracted once we perform SVD factorization on the User-Item Matrix.). Thus we note that SVD is very useful in the dimensionality reduction technique [15].

SVD has a wide range of applications including signal processing, Latent Symantec Analysis, Pattern recognition, low range matrix approximation and weather prediction. Below is the definition of SVD theorem:

3.2.1 STATEMENT OF THEOREM

Consider a Matrix M with m rows and n columns. The SVD theorem in linear algebra states that such a matrix can be decomposed into a product of three matrices, which can be represented by following equation [15]:

$$\boxed{\mathbf{X_{mxn} = U_{m\,x\,r}\ S_{r\,x\,r}\ V_{r\,x\,n}}}$$

Equation 1: SVD Decomposition

Where,

- U – orthogonal matrix of dimension m x r
- S – diagonal matrix of dimension r x r
- V – transpose of an orthogonal matrix V with dimension r x n

The entire matrix is decomposed such that $U^t.U = I$, $V^t.V = I$, where I = Identity Matrix. The columns of Matrix U are orthonormal eigenvectors of XXt, the columns of Matrix V are orthonormal eigenvectors of Xt.X and Matrix S is a diagonal matrix that contains the square root of the Eigen value from Matrix U or V in descending order. Matrix S contains exactly r singular values where r is rank of Matrix X.

To understand the idea behind SVD further, we will consider a very simple example of data points being plotted in a 2-dimensional graph XY. Note: We take the results of Matrix S to determine the number of dimensions in which we will plot the data-point from matrix U or V.

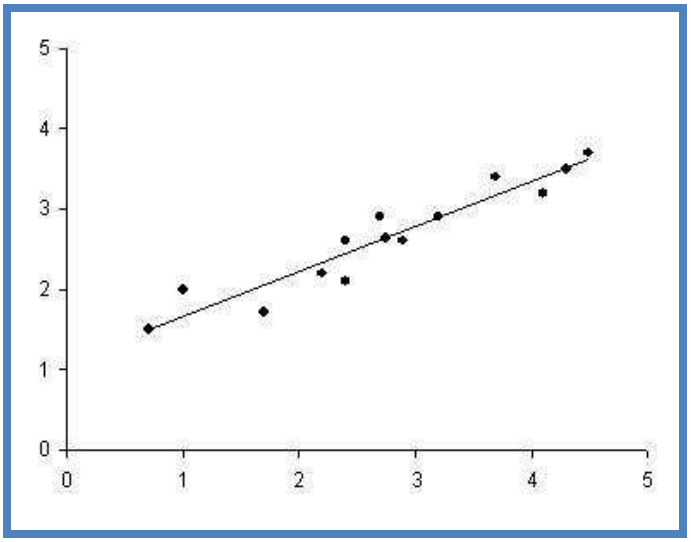


Figure 4: Plotting of regression line which is an approximation of all the data points

For simplicity, we take a 2-dimensional matrix, however for practical applications; the number can go in tens or even hundreds. In Fig. 4, we see a line running through an approximation of all the data-points. It is the best approximation as it averages all the data points perfectly by minimizing the distance between each point and the line. Now, if we draw perpendiculars from all the points to the line, and take the intersection of those lines as a new approximate representation of the actual points, then we would have the reduced representation with as

minimum variation as possible. Also, there is second approximation line possible as shown in the Figure 5 below.

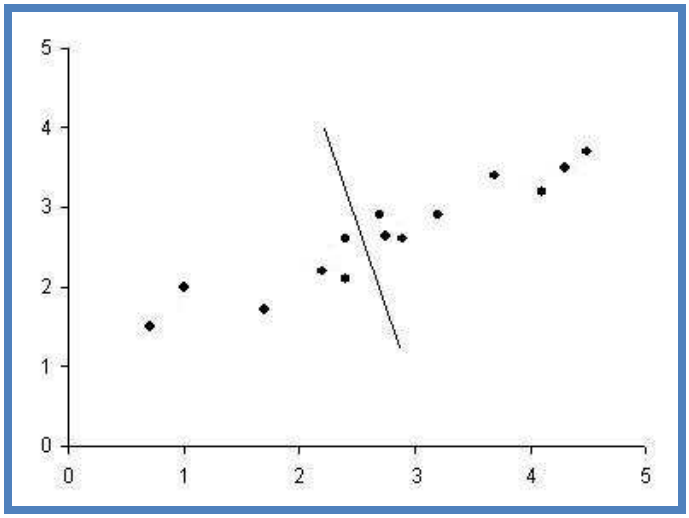


Figure 5: Alternative Regression line for approximating all the data points

In the above figure, the line once again tries to capture the best possible approximation of all the original data points. However, the above approximation seems poor because it corresponds to the dimension that exhibits the least variation (i.e., the dimension which is represented along Y-Axis whereas, in the previous figure we drew the line that corresponded to dimension along the X-Axis).

To describe SVD in crux, we take the user data, such as their song listening history, and plot it against their other data to form user-item matrices. This matrix represents a high dimensional and high variable data which will be reduced to a lower dimension. It will reveal a substructure of the actual data in a more defined way and will order it with the most consistent to the least. The most important part of SVD is that it makes it suitable for a recommendation system to ignore the values after a particular threshold and it still can be sure that all the important relationships between data will be conserved.

3.3 LAST.FM DATASET

The performance of the recommendation algorithm is determined by the quality of data that will be used to train those algorithms. The machine learning process is very data sensitive by nature; the computation accounts huge data sets. Runtime performance of the system in terms of memory and CPU usage also gets affected depending on the way data is prepared, the attributes selected and the order of magnitude to which the system can be scaled up. The gathering of data is not an issue as much as integrity of that data.

There are various kinds of errors that can be induced: data entry errors, measurement errors, distillation errors and data integration errors, etc. They can be introduced while gathering and compiling the databases and appropriate measures should be taken as it can turn into a significant roadblock. Datasets that are available contains lots of noise, so it becomes of prime importance to filter out the data based on the requirements.

For the purpose of this project, the Last.fm dataset has been used. Last.fm is a music web-portal that allows its user base, which has more than 30 million active users, to listen to millions of songs from its music library. All the users' activity is recorded in the Last.fm database, which in turn is used by the portal to make music recommendations [1].

The dataset for this project contains activities of 1000 users whose listening history has been captured anonymously for the period of 2 years. For every song that a user listens to, its activity is recorded in the following format:

User_000004 2009-04-09T12:49:50Z 078a9376-3c04-4280-b7d7-b20e158f345d A Perfect Circle 5ca13249-26da-47bd-bba7-80c2efebe9cd People Are People

Figure 6: User Record tuple in the dataset

The above record contains the following fields:

a. **User id (User_000004)** – Since the data is captured anonymously, we assigned each user, a user-id of the format user_000004.
b. **Date–Time (2009-04-09T12:49:50Z) –** Time of activity is recorded which will be used in our algorithm to determine the session in which it will belong.
c. **Album Id (078a9376-3c04-4280-b7d7-b20e158f345d) –** A unique identifier is attributed to each Album.
d. **Album name (A Perfect Circle) –** An album to which that song belongs to.
e. **Track id (5ca13249-26da-47bd-bba7-80c2efebe9cd) –** A unique identifier is attributed to each track / song.
f. **Track name (People are People) –** The songs which the user listened to.

3.4 REALIZATION OF SESSION BASED ALGORITHM USING SVD

This section describes all the stages of algorithmic implementation. The algorithm is implemented in JAVA along with the usage of some external libraries (JAMA and WEKA libraries).

a. Session Generation stage:

Once pre-processing of data is done, for each user i, where i is such that $0 < i < 1000$, we use the timestamps to perform an analysis to get a suitable threshold value of a session length. We can define a session such that the difference between the timestamps of any two consecutive songs is not greater than the threshold session value decided above. Here, we are working on a hypothesis that the users' choices of songs is influenced by external factors and that there exists a degree of correlation between any two songs that are listened to in the same user-sessions. Multiple such sessions are formed for each user i in the database as each of them has a listening history that spans over two years.

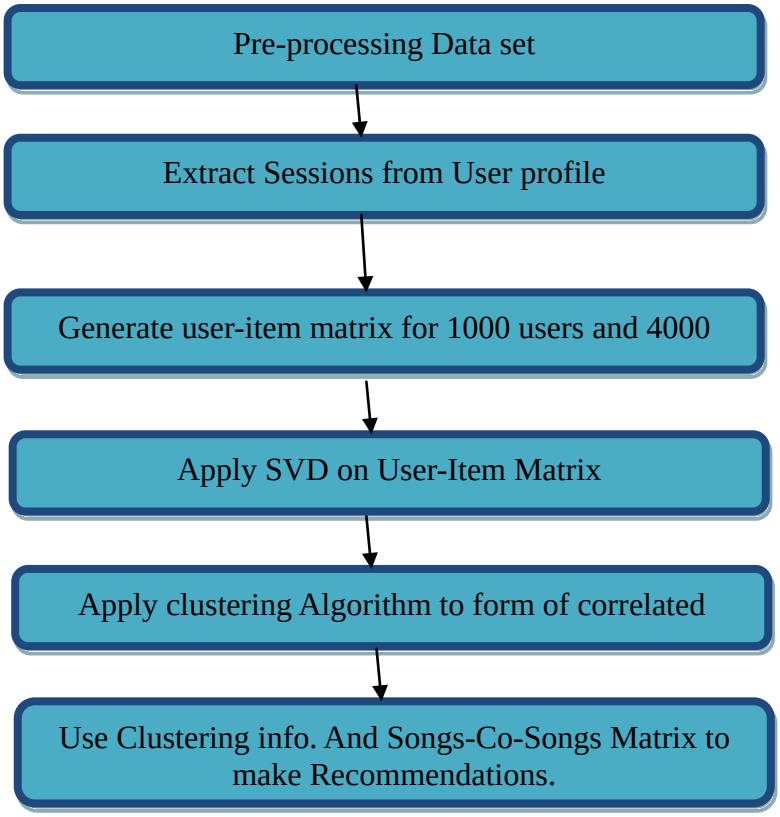


Figure 7: Algorithmic flow

We store all the user sessions in a flat file that falls into one of the three broad session's blocks and process them to extract the pair of all the songs that are played together in that particular session.

b. Constructing User-Item matrix stage:

Once we get the pairs of songs for each user i, we compose a user vector which consists of all the songs that are played in the user's history. Then we cross match it with the top 4000 songs. We construct a user-item matrix for 1000 users x 4000 songs; so that the value in each cell aij in the matrix is directly proportional to the number of times a user i has listened to song j. We call this matrix as Matrix M, which is a sparse matrix.

c. Applying SVD stage:

We use the JAMA (Java Matrix Package) library for applying SVD on the above user –item matrix which will decompose to three sub matrices U, S and V as shown in the Fig 8 below.

JAMA is used for the linear algebraic operation in JAVA. The package provides classes to manipulate and construct dense and sparse matrices [23].

$$\underbrace{\begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & \\ \vdots & \vdots & \ddots & \\ x_{m1} & & & x_{mn} \end{pmatrix}}_{m \times n}^{X} = \underbrace{\begin{pmatrix} u_{11} & \cdots & u_{1r} \\ \vdots & \ddots & \\ u_{m1} & & u_{mr} \end{pmatrix}}_{m \times r}^{U} \underbrace{\begin{pmatrix} s_{11} & 0 & \cdots \\ 0 & \ddots & \\ \vdots & & s_{rr} \end{pmatrix}}_{r \times r}^{S} \underbrace{\begin{pmatrix} v_{11} & \cdots & v_{1n} \\ \vdots & \ddots & \\ v_{r1} & & v_{rn} \end{pmatrix}}_{r \times n}^{V^{\mathsf{T}}}$$

Figure 8: SVD decomposition of Matrix X into sub matrices U, S and V

Matrix U consists of the user's information. It is comprised of the dimension m rows and r columns. Each cell in Matrix U represents a feature of each user. Similarly, Matrix V consists of Item (Song) information. It has the dimension of r rows and n columns, where each song is represented in n dimensions. One of the most important properties of Machine learning/ Information retrieval is its close association with Data Compression techniques. In Data Compression, if we can abstract out the most important concepts (clusters of songs, for example), then we can represent large data-points with very few bits. And the interesting part – its reverse also holds true! If we can manage to represent our data set in a very few bits then we have identified the most 'significant' concepts. This is where SVD plays an important role. We use the Matrix S that contains all the Eigen values represented in descending order, with topmost value being the most significant one as shown in Figure 8 below.

The diagram in Figure 9 shows only the top 8 Eigen vectors, however in practice there are r Eigen vectors where r is rank of the matrix. For experimental purposes, if we have the original user-item matrix of size 1000 x 4000, then we would have as many as 1000 Eigen values represented in Matrix S.

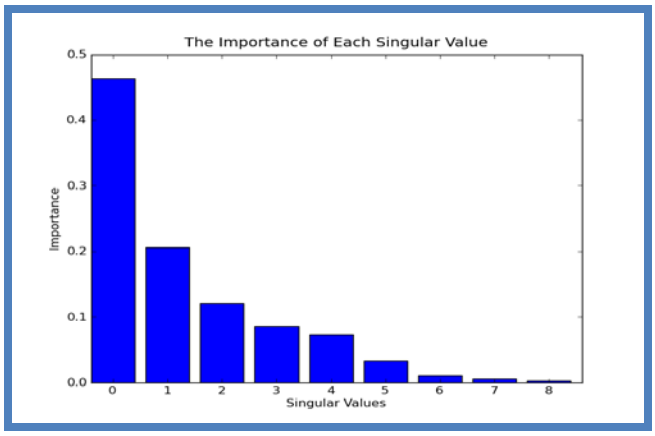


Figure 9: Eight Eigen values in Matrix S are arranged in decreasing order of their importance [26]

We can decide the threshold value k such that only top k-significant Eigen values are preserved and rest of them are discarded so that it doesn't contribute much to the precision of the algorithm. Thus, it has an analogy with data compression techniques as we require smaller number of bits to encode the first k values.

The value k plays is important as it not only reduces the size of Matrix S in computation and it also affects the dimension of Matrix U and V. The new dimensions are as follows:

$$\mathbf{X_{mxn} = U_{m\ x\ k}\ S_{k\ x\ k}\ V_{k\ x\ n}}$$

Equation 2: SVD factorization equation after deciding threshold value k

d. Clustering stage:

Now we apply the clustering algorithm on all the data points of Matrix Vk x n. We just use Matrix V, as our algorithmic approach is to find the similarities between all of the required songs and use that data as the corner stone of our recommendation system. After clustering, we get the group of all the similar songs. We store all the clusters into a flat file for the future decision making process (recommendations). For the purpose of clustering, we use WEKA (The University of Waikato) library. WEKA is a collection of many data processing and

clustering algorithms. It is mostly used for data mining and machine learning algorithms [22].

e. Constructing Song – Co-song Matrix stage:

From Making Session's stage, for each user, we have combinations of all the songs that are played in each session. The entire user history consists of hundreds of such sessions. We take this information and plot them in the Song – Co-Song matrix. Each cell in the matrix is assigned a weight that is a function of number of times those songs are played together. We started with an approach that the choices of songs played by a user is influenced by the time of day, kind of work he is doing, their mood, surrounding environment (for example: it's raining outside, it's Christmas holiday, working out in the gym, or if they are on their way to work, etc.), and due to this there exists some correlation between the songs played in a single session. The resultant Songs-Co-songs matrix is a sparse matrix. In the next stage we will recommend the songs using information assembled in the previous steps.

f. Recommending songs stage:

Once we have constructed the model from Clusters and Songs – Co-Songs matrix, we can give recommendations based on the patterns observed in the user's history and session information that can be extracted from the above process.

4. EXPERIMENT

To analyze the behavior of the algorithm, we take a use case similar to the real world scenario and trace all the algorithmic stages in this section.

Let's take a user scenario where we try to make recommendations based on some past preferences and session information gathered from all of our users. We build the model beforehand using the algorithm described in the previous section. To keeping things simple, we take a use case of just 4 users and 6 songs. The Matrix generated will not be as sparse as it is in the real world scenario because the number of items (songs) that are taken into consideration are very less. However, to understand the flow of the algorithm this information is sufficient.

The first stage is the preparatory stage where we take the users' listening history and try to eliminate all the data inconsistencies and rearrange it according to the timestamps in descending order.

User_000004 **2009-03-28T10:11:10Z** 0d360231-f492-46ac-baf0-4f84ac6e8b17 Air France 85e33d95-3523-4d80-bddc-a050d5a16e70 Collapsing At Your Doorstep
User_000004 **2009-03-28T10:02:26Z** 37116914-db39-443c-981d-75d6326450f1 The Phenomenal Handclap Band 04efa75f-276b-4acc-84a2-555fef9099ac 15 To 20 (Radio Version)
User_000004 **2009-03-28T09:55:23Z** dc21d171-7204-4759-9fd0-77d031aeb40c Frightened Rabbit 8b3c111a-b8a4-4823-9e0c-58043ed1af24 Old Old Fashioned
User_000004 **2009-03-28T09:50:02Z** 2aca01cc-256e-4a0d-b337-2be263ef9302 All Girl Summer Fun Band 722bd5fd-1b27-4ec1-ba21-3e7dc3c514b0 Cut Your Hair
User_000004 **2009-03-28T09:44:37Z** 2aca01cc-256e-4a0d-b337-2be263ef9302 All Girl Summer Fun Band 185b7f90-2624-49fa-bba9-910de231ab73 Video Game Heart
User_000004 **2009-03-28T09:36:39Z** da785f6e-86e2-4efa-aca6-6c3cbd15d91c Glasvegas bdf3729c-6720-46f0-b42d-6164bb76b1b1 Daddy's Gone
User_000004 **2009-03-28T09:13:30Z** 2fb4db11-8349-47ab-b1a6-f04f011699ff The Go-Betweens 70b543af-954c-4baa-97ad-a349214c3eeb The House That Jack Kerouac Built
User_000004 **2009-03-28T08:54:58Z** 2fb4db11-8349-47ab-b1a6-f04f011699ff The Go-Betweens c0fac49a-ab81-4457-bc02-2aa254c647c3 Was There Anything I Could Do?

Figure 10: An example of a session generated with the threshold value of 90 minutes

Now, in the **Session Generation stage** we select a threshold limit for the session length (in minutes) and generate multiple single sessions for each user that will be processed in the next few steps. One such session for a threshold value of 90 minutes is shown above in Fig. 10. Furthermore, we have to make songs – co-songs pair from each session by permuting each song with every other song in the same session.

Next, in **Constructing the User-Item matrix stage**, we plot the data for all 4 users and 5 songs into a matrix which is shown in Table 2, below. Each cell Cij, represents the number of times that a song is heard by the corresponding user. The value of each cell is directly proportional to the likeability of that song and the users' inclination for any songs is derived from these values.

	Ben	Tom	John	Fred
Song 1	5	5	0	5
Song 2	5	0	3	4
Song 3	3	4	0	3
Song 4	0	0	5	3
Song 5	5	4	4	5
Song 6	5	4	5	5

Table 2: User –Item matrix for 4 users and 5 Songs

Next, in the Applying SVD stage, we use JAMA library to compute SVD of the User-Item matrix. The SVD operation factorizes the matrix into three sub-matrices: U, S, and V. This reduced dimensional representation of the original matrix emphasizes a stronger relationship amongst users and items. It is also possible to reconstruct the original matrix with less information, however, the idea behind SVD is to analyze how many features or concepts are required to reconstruct it back.

Matrix S is a diagonal matrix where all of the features or concepts are arranged in an order of the most significant concept to the least significant one. It has r features, where r is the rank of user-item matrix. In linear algebraic notion, if two vectors cannot be expressed as scalar multiple or sum of any other vector in the same space then those two vectors are linearly independent. For example, if a user who likes a song *More Than You Know* also likes the song *All I Need to Hear*, then the two song vectors would be linearly dependent and thus will effectively contribute only one to the matrix rank. To perform a similar operation we take first k singular values from Matrix S, where k << r.

For this example, we take k = 2 and the matrix shown in the figure above is decomposed as shown below in Fig. 11:

U	
-0.4472	0.5373
-0.3586	-0.2461
-0.2925	0.4033
-0.2078	-0.6700
-0.5099	-0.0597
-0.5316	-0.1887

S	
17.7139	0.0000
0.0000	6.3917

V.transpose	
-0.5710	0.2228
-0.4275	0.5172
-0.3846	-0.8246
-0.5859	-0.0532

Figure 11: User-Item matrix after applying SVD decomposition

Next, in the **clustering stage**, we use clustering algorithms from WEKA library to cluster all the data from Matrix V. However, before clustering we need to do some pre-processing steps to convert it into the format that is acceptable by clustering algorithms. The features of the songs are rows-dominant and thus we take transpose of Matrix S and make it column dominant. Also, we convert the matrix into csv format before processing it further. The cluster can be visualized as shown below in Fig.12.

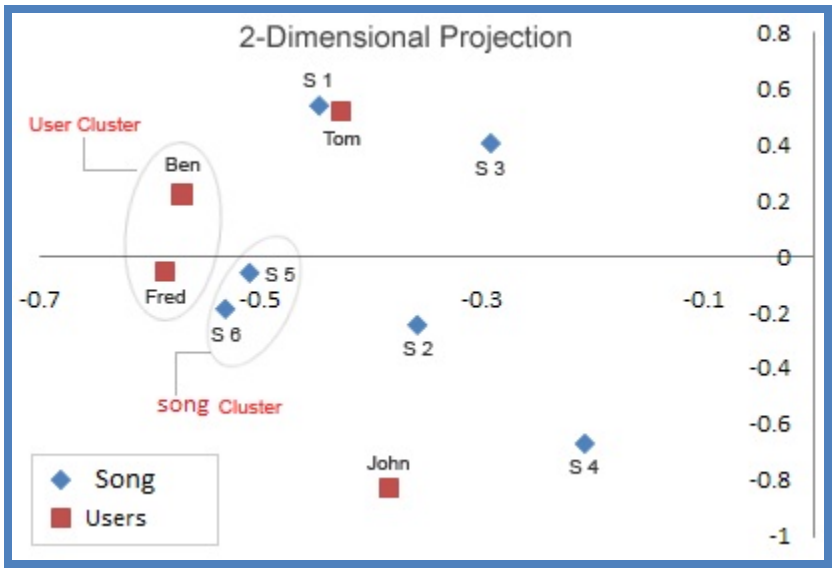


Figure 12: Applying Clustering to Matrix V

We can see in the above figure that Fred and Ben form a single cluster. It suggests that they have very similar listening pattern. Now, if we observe the user-matrix in Table 2, we see that Ben and Fred have user vectors as (5, 5, 3, 0, 5, 5) and (5, 4, 3, 3, 5, 5) respectively. It can be clearly seen that Ben and Fred have listened to Song 1, 2, 3, 5 and 6 for a similar number of times and thus they are rightfully clustered together. Using the above knowledge, we can recommend song 4 to Fred as he has not listened to that song yet; however there is a good chance that he will add that song to his listening list.

In the same way, Song 5 and 6 are clustered together. We can see the user-item matrix in figure x, song 5 and 6 have a song vector of (5, 4, 4, 5) and (5, 4, 5, 5), respectively. It indicates that those 2 songs are the most similar ones to each other as compared to other songs like song 1, 2, 3 and 4.

We can already make good recommendations using the SVD alone, as seen from the example above. However, we use the Songs-Co-songs Matrix generated in **Constructing Song – Co-song Matrix stage** of the algorithm to filter out the results further and use the session knowledge to

recommend songs that have strong correlation amongst each other. To construct Song- Co-Songs Matrix, we take all the pair of songs and construct a 6 x 6 matrix. Each cell in the matrix is given a value which is a function of how often they are listened together. The value is computed using the formula shown below:

Weight (Song Dual) = ∑ (frequency of two songs played together) / ∑ (frequency of all

Equation 3: Assigning weight to each song duals

Now, once we have both the clustering and Songs – Co-songs matrix information, we proceed to the **Recommending songs stage.** We take the listening history of a test user to whom we will be making recommendations. We try to find a pattern by taking every song in that user's history and use the above clustering model to find the cluster that each song will belong to. Once we find a dominant cluster, we take all the neighboring songs in that cluster (using the clustering model formed above) and put them into an Array-list. Now using the Songs – Co-songs matrix, which gives the probability of two songs being played together, we arrange all the songs in the Array list in decreasing order of their weight. Thus, the song which is at top of the list is most likely to be liked by the user and hence it is recommended. This is how we recommend the top N songs.

5. ANALYSIS AND EVALUATION

Recommendation algorithms are popular in research communities. Many researchers are working on various sub-problems as it is difficult to encompass all the candidate approaches into a single system. Thus, it is important to focus on a variety of properties that enhance user experience and can possibly add another dimension to the already existing approaches. We have discussed one such approach to recommend music and in this section we will try to evaluate the algorithm with some parameters as discussed below:

5.1 OFFLINE ANALYSIS

For the analysis purpose, we withheld some of the data (users' listening history) of 15 users to perform the evaluation of song recommendations and check if it matched the benchmarks. Rather than caring about the degree to which a user would like to a recommended song, we sometimes are more interested if a user will add that song to his listening queue. This is an important aspect of music discovery. Therefore, we hold back partial data of 15 users and try to complete that list using the recommendations made by our algorithm. For every test user, depending upon the criteria, each song that is recommended can be categorized into one of four the groups. They are as shown below in Table 3.

	Not Recommended	**Recommended**
Used	False-Negative (fn)	True Positive (tp)
Not	True Negative (tn)	False Positive (fp)

Table 3: Table to distinguish each of the recommended songs

Here, we can describe each of the term as:

- **True Negative (TN):** A song which is uninteresting to the user is not recommended to them.
- **False Positive (FP):** A song is recommended by the algorithm which a user is not interested in.

- **True Positive (TP):** An algorithm recommends a song to the user which they are interested in.
- **False Negative (FN):** The algorithm does not recommend a song to the user which they are interested in.

In this kind of offline evaluation, all the above recommendations are not given to an actual user to listen to, but are just compared with the data we already have, thus, we have to assume that unused recommendations would not make to user's listening list – i.e., they are unappealing to the users. However, this assumption can be false as some of the recommendations might be an interesting song with the user had not heard. However, once they are exposed to this recommendation they might get influenced to select it. This is one of the cases where false-positives are escalated.

In our algorithmic approach, we make song duals based on the sessions they are in and categorize them in any of the three categories depending upon the timestamp for respective sessions.

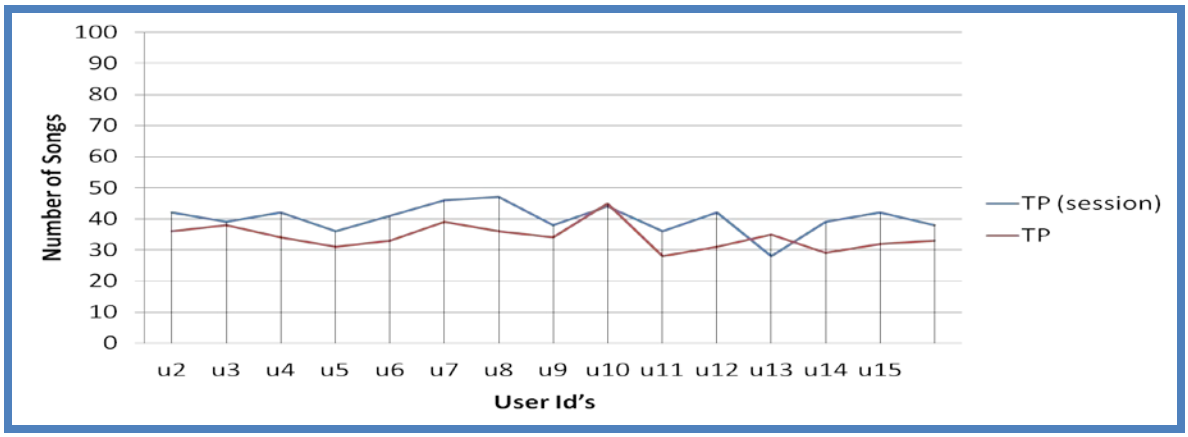


Figure 13: True Positive Values for session based and unified session approach

Another approach is to keep a unified list of all the possible song pairs without continuing to categorize them. We try to analyze both the approaches here with the help of recommendations made by the same algorithms, yet by considering different song-duals list. The graph for the number of True Positive (TP), False Positive (FP) and False Negative (FN) are as shown above.

In Fig, 13, the number of true positives, otherwise known as the songs that should actually be present in the user list, are recommended more accurately when we use the session based song-duals list as compared to unified songs list. The blue line represents the recommendations where session based list was used. In most of the cases, session based list outperforms the one without it, which is a positive result. There were some anomalies in the case of user 10 and user 13 but for maximum use cases– the session based song list gives better results.

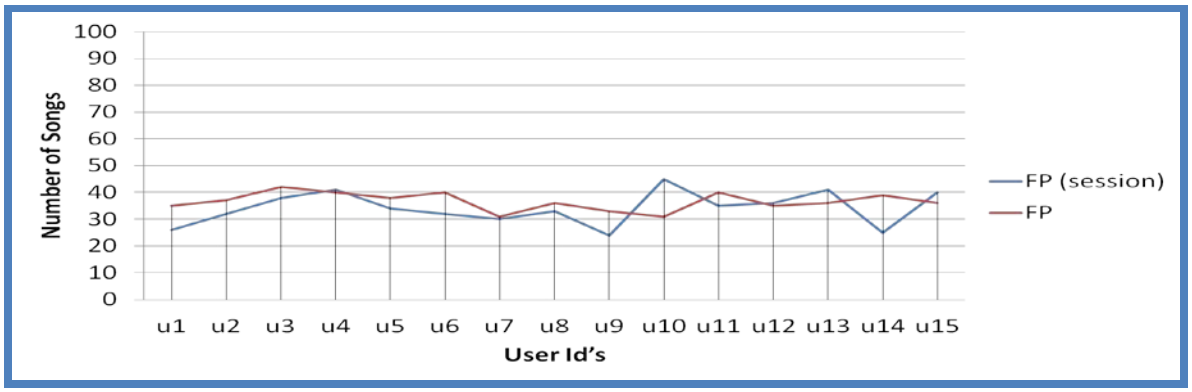


Figure 14: False Positive Values for session based and unified session approaches

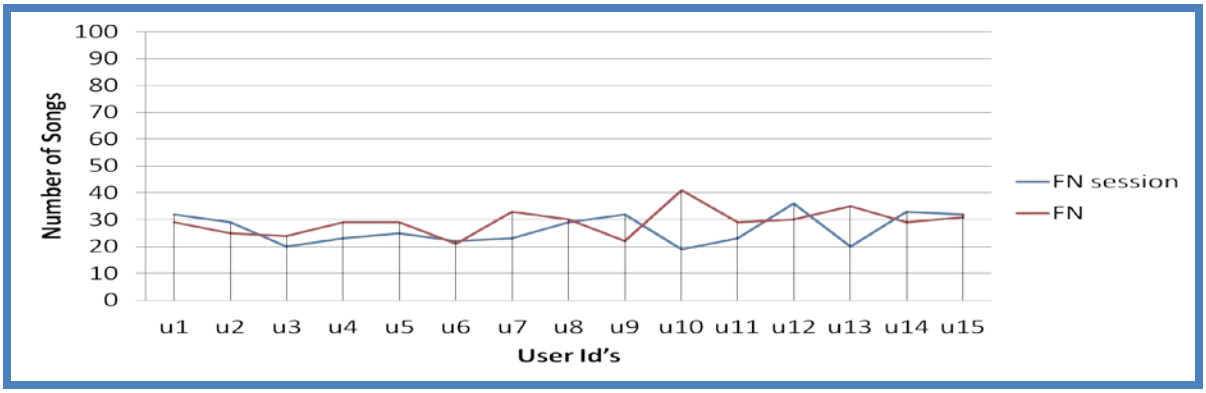


Figure 15: False Negative Values for session based and unified session approaches

The above Figs. 14 and 15 show a number of false positives (FP) and false negatives (FN) for all the 15 test users. The FP's are the songs that are recommended to the users but they actually are not interested in and the FN's are the songs in which a user is not interested and also not recommended to them. In both figures, number of songs recommended using session-based approach (blue lines) should be less than the ones using without it (red lines). Thus, the session-

based approach generates better results. Based on the above data, we calculate Precision and Recall using the formula shown in Equation 4:

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

$$Recall = \frac{TP}{TP + FN}$$

Equation 4: Precision and Recall formula

We can compute Precision and Recall values for each user profile and plot them in the graph shown in Fig.16 and 17:

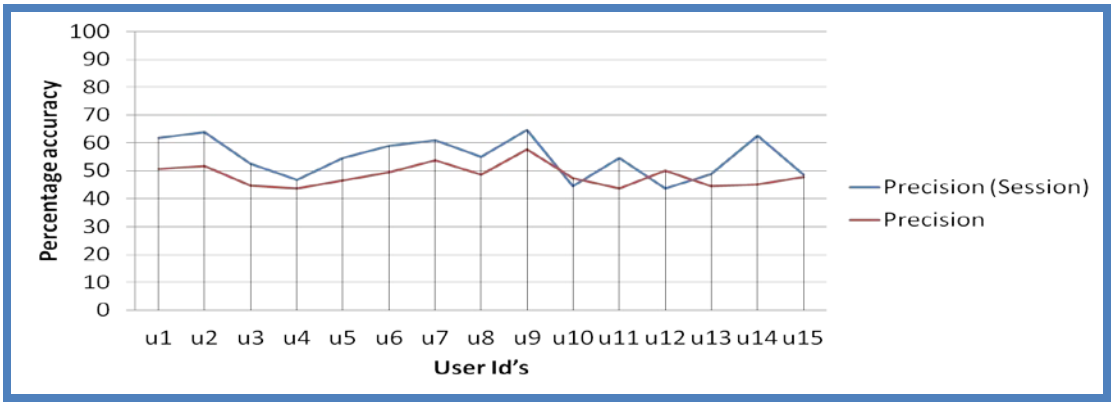


Figure 16: Precision Values for session based and unified session approaches

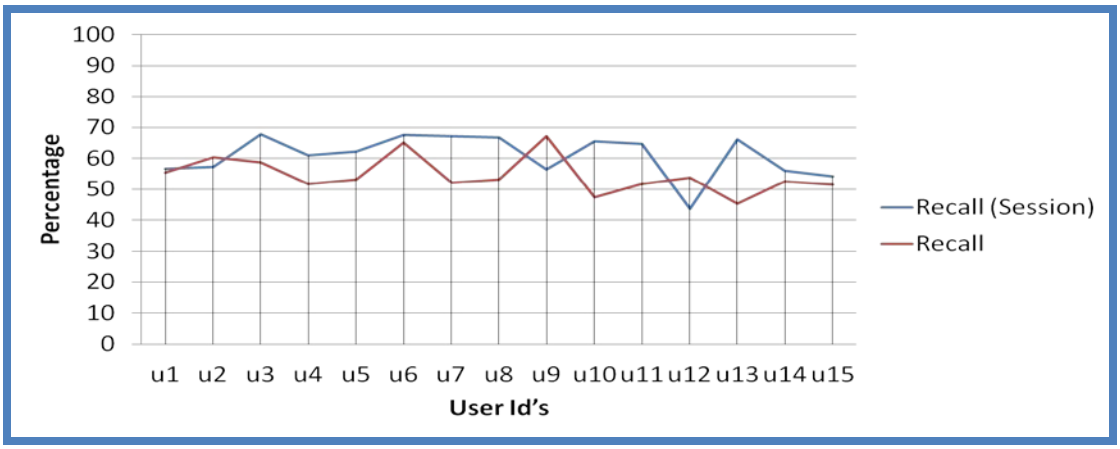


Figure 17: Recall Values for session based and unified session approaches

The graph shows that the session based songs list approach (showed in a blue line) has better precision and recall values. In addition, it outperforms the unified session based approach by approx. 12-18 % of most of the user profiles.

5.2 COVERAGE OF THE ALGORITHM

To measure the accuracy of recommendation algorithms, we measure the ratio of all the items that are finally shown up as recommendations to avoid any biasing in the results. We find the ratio of all the unique songs that appeared in the recommendation list compared to total number of songs in the dataset. That coverage corresponded to 58.29% of all the songs. However, every song is not equally popular. Thus, to calculate coverage in a more useful manner, it may be desirable to attach some importance contingent upon the popularity or utility of that item (song). Using this method, we won't suggest any songs that may be less appealing to a user. Yet, at the same time, it is costly to miss any high profile song. The weight assigned to each song is given by the formula:

Weight (song) = log (frequency of the song under consideration / summation of frequency of all songs)

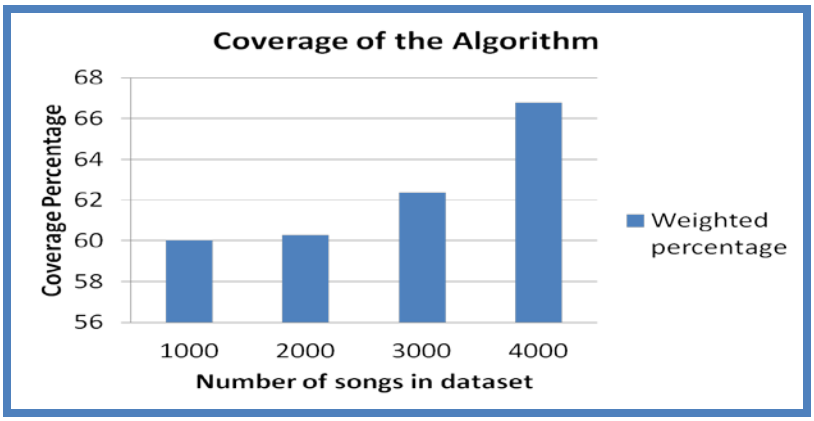


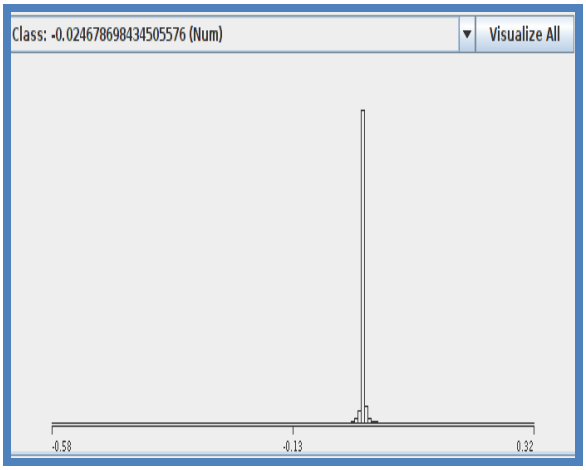
Figure 18: Coverage of algorithm for a different number of songs

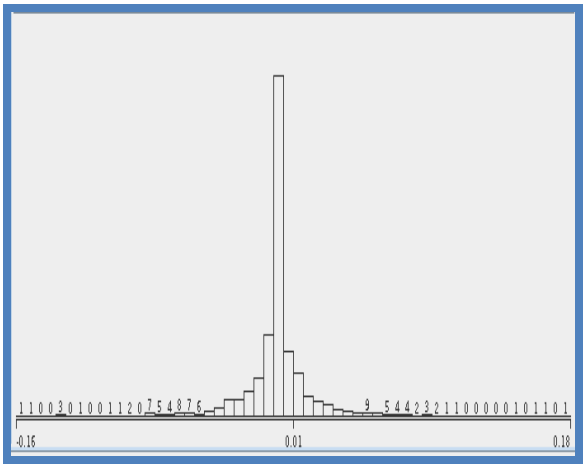
Once all of the songs are arranged in a decreasing order of their weights, we performed the above evaluation again and found out that the catalog coverage increased to 61.02%. The above use case is for the scenario when the user-item matrix had 1000 songs. The following graph presents the evaluation of catalog coverage for a different size of datasets. As we can see, the coverage increases proportionally with the number of songs in the dataset. When the algorithm can train the model with more songs, the weight of the songs in the dataset gets distributed and thus the coverage increases.

5.3 CLUSTER FORMATION AND k - VALUE EVALUATION

Cluster formation plays an important role as it will directly affect the kind of recommendations made by the algorithm. For our purpose, we analyzed the cluster distribution over three clustering algorithms: XMeans, Simple K means and DB Scan. Out of those, DB Scan gave a good cluster distribution. Simple K-Means and X means had many outlier points and thus it resulted in distant cluster formation.

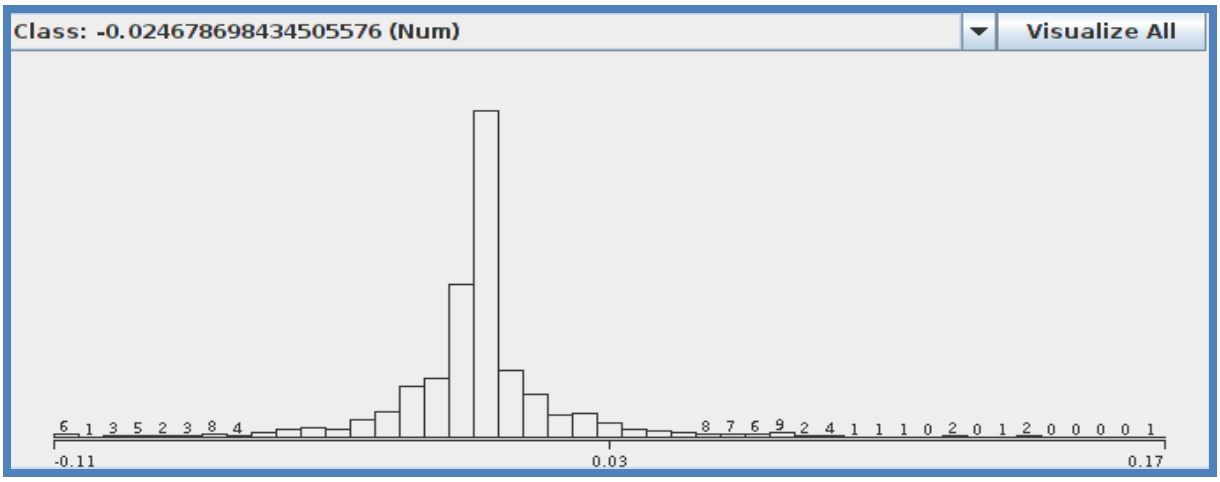
Also, we use the clustering algorithm to group all the songs together that have significant features aligned together. The feature distribution becomes less concentrated as we go down to the Matrix S. We use clustering to visualize as shown in the diagram below:





(a) **(b)**

Figures (a), (b) and (c) represent the Eigen values in descending order as it is present in Matrix S. We can see from Figures (a) to (c) that feature distribution becomes more variable. For example: Fig. (a) has a single dominant feature, while Fig. (c) has side features which are comparatively more dominant than earlier cases.



(c)

Figure 19: Song features in Matrix V

Thus, we experiment with different values of k i.e. for k = 200, 500, 600 and 700. This analysis points to important patterns in deciding the threshold value of k.

6. CONCLUSION

The algorithm present here is an attempt to design, implement and analyze an approach for a music recommendation system that takes user session into consideration while recommending songs. The goal was to provide an approach where we can leverage some of the Information retrieval techniques and the linear algebraic approach to solve big and sparse matrices in order to extract correct and valuable information. Many algorithmic approaches result in an exponential increase in mathematical calculations. An approach like SVD can benefit by reducing high dimensionality space into low dimensions. It also helps surface any relationship between items and users– which at first might not seem apparent or obvious. User tends to listen to the songs, which are inherently influenced by the environment or certain contexts. A mathematical approach is used to surface those correlations between the songs played in same session and use those facts to make music recommendations. The investigation outcome has shown that this technique can filter out noise and provide good results.

7. FUTURE WORK

Music is a complex item to analyze as it contains many layers of dependent and independent factors that can influence a user into making some random looking choices. The approach presented here to solve the problem was limited by the computational resources available. It would be interesting to see the use of some of the open source libraries, like Apache Mahout and Apache Hadoop, to facilitate the solving of bigger matrices than what is encountered here. This will certainly lead to better results, since more relationships can be formulated and eventually help in decision making.

The clustering algorithms can be manipulated to the specific needs of the problem so that the clustering of songs would result in appropriate groupings. Regression analysis can be performed on the dataset to extract the subject's mood and that can point to some valuable information. Study in the field of music psychology has revealed some very good results. However, that is dependent on the fact that we need to gather some good datasets (a very detailed user profile). The detailed user profile can then be processed to extract all the variables that will act as an input for this kind of research algorithm.

8. REFERENCES

[1] Last.Fm – A popular music web portal

http://www.last.fm

[2] Pandora – A free internet radio.

http://www.pandora.com/

[3] Maes, Pattie and Sharadanand, Upendra. (1995). Social Information filtering: Algorithm for automating word of mouth. SIGCHI'95 conference on Human Factors in computing systems. New York, NY, USA, (pp. 210-217).

[4] Chao Zhang, Rui CAI, Chong Wang. Contextual music recommendation using emotional allocation modeling. Proceedings of the 15th international conference on Multimedia, pages 553-556, New York, NY, USA, 2007

[5] Kaji Katsuhiko, Keiji Hirata, and Nagao Katashi. A music recommendation system based on annotations about listener's preferences and situations. Axmedis, 0:231-234, 2005.

[6] Pedro Cano and Javier M.Buldu. Topology of music recommendation networks. Chaos, 2006.

[7] Elias Pampalk and Gerhard Widmer. Dynamic playlist generation based on skipping behavior. 6th International Conference on Music Informational Retrieval, pages 634-637. University of London, 2005.

[8] C. Anderson. The long Tail. Wired Magazine, 12(10): 170-177, 2004.

[9] S. Deer ester, S. T. Dumais, T. K. Landauer, and R. Harshman, "Indexing by latent semantic analysis," Journal of the American Society for Information Science, vol. 41, no. 6, pp. 391–407, 1990.

[10] M. W. Berry, S. T. Dumais, and G. W. O'Brien, "Using linear algebra for intelligent information retrieval," *SIAM Review* pp. 573–595, 1995.

[11] K. Miyahara and M. J. Pazzani, “Improvement of collaborative filtering with the simple Bayesian classifier,” Information Processing Society of Japan, 2002.

[12] S. K. Jones, “A statistical interpretation of term specificity and its applications in retrieval, “Journal of Documentation, pp. 11–21, 1972.

[13] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, “Indexing by latent semantic analysis,” Journal of the American Society for Information Science, pp. 391–407, 1990.

[14] Singular Value Decomposition
http://en.wikipedia.org/wiki/Singular_value_decomposition

[15] Kirk Baker, Singular Value Decomposition Tutorial, March’29 2005

[16] Zhimin Chen, Yao Zhao. A Collaborative Filtering Recommendation Algorithm Based on User Interest Change and Trust Evaluation**,** Institute of Information Engineering, Yangzhou University, China

[17] Billsus D., and Pazzani M. J., “Learning Collaborative Information Filters”, ICML ’98, pp. 46-53, 1998.

[18] Wang Lan, Zhai Zheng Jun, “Collaborative filtering algorithm based on time weight”, Journal of Computer Applications, vol.27, no.9, pp. 2302-2305, 2007.

[19] G.Karypis, “Evaluation of item-based top-N recommendation algorithms,” CIKM 2001, pp. 247–254, 2001.

[20] E. W. De Luca and S. Albayrak. Multilingual ontology-based user profile enrichment. Workshop on the Multilingual Semantic Web, 2010.

[21] A. Hotho, R. Jaschke. Social tag prediction. Proceedings of the 31st annual international ACM SIGIR conference.

[22] WEKA – Data Mining and Open Source Machine Learning Library (The university of Waikato)

http://www.cs.waikato.ac.nz/ml/weka/

[23] JAMA – Java Matrix Package.

http://math.nist.gov/javanumerics/jama/

[24] Sibel Adali, Malik Ismail, and William Wallace. Measuring behavioral trust in social networks.

[25] Sean Owen, Robin Anil, Ted Dunning. Mahout in Action. Manning publications.

[26] SVD and LSI Tutorial
http://www.puffinwarellc.com/index.php/news-and-articles/articles/33-latent-semantic-analysis-tutorial.html?start=4

[27] Zhijun Yin and Jiawei Han on LINKREC: A Unified Framework for Link Recommendation with User Attributes and Graph Structure. April 26-30, Raleigh, NC, USA .

[28] R. Y. Nakamoto, S. H. Kato. Investigation of the effectiveness of tag-based contextual collaborative filtering in website recommendation. In Advances in Communication Systems and Electrical. Engineering, pages 309–318.