PROJECT REPORT

ON

“MUSIC TASTE PREDICTION”

Bachelor of Technology in
Electronics and Telecommunication

By

Utsav Shah	D036
Arnav Saxena	D059
Raunaq Singh Sahni	D073

Under the Guidance of
Prof. Nieves Crasto

Department of Electronics and Telecommunication
Mukesh Patel School of Technology Management and Engineering
Mumbai - 400056

2015-2016

NMIMS University

Mukesh Patel School of Technology Management and Engineering

Narsee Monjee Institute of Management Studies

Department of Electronics and Telecommunication

CERTIFICATE

This is certify that the project entitled

“Music Taste Prediction“

submitted by

UTSAV SHAH	**D036**
ARNAV SAXENA	**D059**
RAUNAQ SINGH SAHNI	**D073**

is a record of bonafide work carried out by them, in the partial fulfilment of the requirement for the award of Degree of Bachelor of Technology (Electronics and Telecommunication) at Mukesh Patel School of Technology Management and Engineering, Narsee Monjee Institute of Management Studies. This work is done during year 2015-2016, under our guidance.

Date:

(Prof. Nieves Crasto)
Project Guide

(Dr. Sharad Y. Mhaiskar)
Dean

(Dr. Vaishali Kulkarni)
HOD, EXTC Department

External Examiner

Acknowledgements

We take this opportunity to express our sincere deference to **Dr. Vaishali Kulkarni**, Head of Department, EXTC for including this project in the curriculum.

We would like to thank our project co-ordinators **Prof. Vipul Gohil, Prof. Vidya Sawant, Prof. Amey Raut and Prof. Jish Joy** for guiding us throughout the project.

We are very grateful to our project mentor **Prof. Nieves Crasto**, whose profound encouragement, co-operation, guidance and keen supervision at every stage of the project, inspired us in pursuing and completing the project within schedule. Without her guidance and support, this project would not have been possible.

We convey our immense gratitude to **Dr. S.Y.Mahiskar**, Dean, MPSTME for the continuous help and encouragement and the friendly atmosphere of education provided by him.

UTSAV SHAH
ARNAV SAXENA
RAUNAQ SINGH SAHNI

ABSTRACT

Music recommender system is a system which learns from the user's past listening history and recommends them songs which they would probably like to hear in future. Along with the rapid expansion of digital music formats, managing and searching for songs has become significant. High quality recommender systems are being produced, performing many recommendations per second for millions of users and items, achieving high coverage in the face of data sparsity.

To address these issues we have explored item-item collaborative filtering techniques. Item-based techniques first analyse the user-item matrix to identify relationships between different items, and then use these relationships to indirectly compute recommendation for users. K nearest neighbours algorithm and similarity functions have been implemented on the user-item matrix to predict ratings of songs.

Contents

List of Figures

Chapter 1

Introduction

1.1 Overview

1.1.1 Background

Music recommender systems have changed the way people listen to music. Patterns of behaviour are studied to track what someone will prefer from among a collection of songs one has never heard. The technology behind music recommender systems has evolved over the past 20 years into a rich collection of tools that enable the practitioner or researcher to develop effective recommenders. Using this project, the recommender systems can be used to predict music for different users. It is imperative that tools which are used for recommender systems are studied and implemented, including how they work, how to use them, how to evaluate them, and their strengths and weaknesses in practice [11].

1.1.2 Motivation and Scope of Thesis

Over the past two decades, immense research in this field has led to many different ways to provide an accurate recommendation to a user. Various algorithms are implemented in different situations to understand and predict the user preference. Hence, the advancement in automated recommendation systems have become an interesting topic among machine-learning and human-computer interaction domains.

1.1.3 Salient Contribution

This recommendation system builds up a user's profile based on his/her past records, and compares it with some reference characteristics, then seeks to predict ratings of a song to a user would not have listened before.

One of the most promising technologies is collaborative filtering. Collaborative filtering works by building a database of preferences for items by users.

1.2 Million Song Dataset (MSDS)

The Million Song Dataset is a freely-available collection of audio features and meta-data for a million contemporary popular music tracks.

The core of the dataset is the feature analysis and metadata for one million songs, provided by The Echo Nest. The dataset does not include any audio, only the derived features. However, that sample audio can be fetched from services like 7digital, using the code that they have provided[1].

For this project, the database for the songs and the model created to recommend songs, has been implemented by using the subset of the MSDS. The matrix extracted and is then worked upon by making it concise and readable.

1.3 Collaborative Filtering

Collaborative filtering (CF) is a popular recommendation algorithm that bases its predictions and recommendations on the ratings or behaviour of other users in the system. The fundamental assumption behind this method is that other user's opinions can be selected and aggregated in such a way as to provide a reasonable prediction of the active user's preference. Intuitively, they assume that, if users agree about the quality or relevance of some items, then they will likely agree about other items if a group of users likes the same things as one user, then that user is likely to like the things they like which the user hasn't yet seen.

Tapestry was a manual collaborative filtering system: it allowed the user to query for items in an information domain, such as corporate e-mail, based on other user's opinions or actions (give me all the messages forwarded by user X). It required effort on the part of its users, but allowed them to harness the reactions of previous readers of a piece of correspondence to determine its relevance to them[2].

Automated collaborative filtering systems soon followed, automatically locating relevant opinions and aggregating them to provide recommendations. In the late 1990's, commercial deployments of recommender technology began to emerge. Perhaps the most widely-known application of recommender system technologies is

Amazon.com. Based on purchase history, browsing history, and the item a user is currently viewing, they recommend items for the user to consider purchasing[3].

Collaborative filtering techniques depend on several concepts to describe the problem domain and the particular requirements placed on the system. Many of these concepts are also shared by other recommendation methods. The information domain for a collaborative filtering system consists of users which have expressed preferences for various items. A preference expressed by a user for an item is called a rating and is frequently represented as a (User, Item, Rating) triplet[2].

Methods used for collaborative filtering :

1.3.1 User-user Collaborative Filtering:

User - user Collaborative Filtering is a straightforward algorithmic interpretation of the core premise of collaborative filtering: find other users whose past rating behaviour is similar to that of the current user and use their ratings on other items to predict what the current user will like. To predict a user's preference for an item the user has not rated, user user CF looks for other users who have high agreement with the former user on the items they have both rated. These users' ratings for the item in question are then weighted by their level of agreement with the former user's ratings to predict user preference.

Besides the rating matrix R, a user - user CF system requires a similarity function $S : U \times U \to R$ computing the similarity between two users and a method for using similarities and ratings to generate predictions[3].

1.3.2 Item-item Collaborative Filtering:

Itemitem CF generates predictions by using the users own ratings for other items combined with those items similarities to the target item, rather than other users ratings and user similarities as in user user CF. Similar to useruser CF, the recommender system needs a similarity function, this time $S : I \times I \to R$, and a method to generate predictions from ratings and similarities[3].

Chapter 2

Literature Survey

One of the earliest implementation of collaborative filtering-based recommender systems is Tapestry. This system relied on the explicit opinions of people from a close-knit community, such as an office workgroup. However, recommender systems for large communities cannot depend on each person knowing the other. Later, several ratings-based automated recommender systems were developed.

Music is subjective and universal. It not only can convey emotion, but also can it modulate a listener's mood. The tastes in music are varied from person to person, therefore, the previous approaches cannot always meet the users' needs. An emotion-based model and a context-based model have been proposed. The former one recommends music based on mood which allows the user to locate their expected perceived emotion on a 2D valence-arousal interface. The latter one collects other contextual information such as comments, music review, or social tags to generate the playlist. Though hybrid music recommender systems would outperform the conventional models, the development is still at very early stage. Due to recent studies in psychology, signal processing, machine learning and musicology, there is much room for future extension[6].

The GroupLens research system provides collaborative filtering solution for Usenet news and movies. Ringo and Video Recommender are email and web based systems that generate recommendations on music and movies respectively. A special issue of Communications of the ACM presents a number of different recommender systems. Perhaps the most widely known application of recommender system technologies is Amazon.com. Based on purchase history, browsing history, and the item a user is currently viewing, they recommend items for the user to consider purchasing similar items according to the user's history.[4].

Other technologies have also been applied to recommender systems, including Bayesian networks, clustering and Horting. Bayesian networks create a model based on training set with a decision tree at each node and edges representing user information. The model can be built off-line over a matter of hours. The resulting model is very small, very fast and essentially accurate as nearest neighbour methods. Bayesian networks may prove practical for environments in which knowledge of user's preference change slowly with respect to the time needed to build the model but not suitable for environments in which user preference models must be updated rapidly or frequently[5].

Chapter 3

Problem Statement

The central issue is the recommendation of songs to a user. For a given user, song history and play count is given for each song. Generally, this is done by looking at the songs that are most similar to the user's songs, as well as the users who are most similar to the user according to their listening history.

However, given that there is a lot of data sparsity for each rating given to the songs by a particular user, a significant issue was to normalize the these rating values due to the empty values of ratings. Also, the issue of dealing with large quantities of data exists. The subset of the MSDS and accompanying datasets offer over 1.8 gigabytes of information about songs and ratings, i.e., user play counts[2].

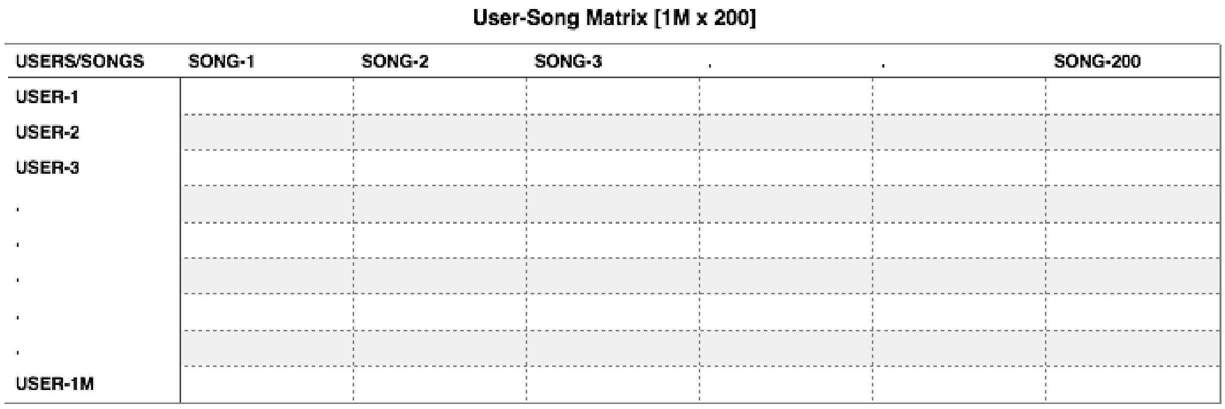
Algorithms such as K Nearest Neighbours Algorithm (KNN) [7] and different Similarity Functions are implemented in order to give an optimum rating prediction. The results obtained from cosine - based similarity function are compared on the basis of Mean Absolute Error (MAE) to give the optimum output.

Chapter 4

Project Description

4.1 User - Song Matrix

The matrix is extracted from the Million Song Dataset [1] which contains the ratings of 1 million users given to 200 distinct songs, the rows represent the users which are 1 Million in total and the columns represent the 200 songs. Each column row intersection signifies a particular play count of a song heard by the particular user.



User-Song Matrix [1M x 200]

USERS/SONGS	SONG-1	SONG-2	SONG-3	.	.	SONG-200
USER-1						
USER-2						
USER-3						
.						
.						
.						
.						
.						
USER-1M						

Figure 4.1: The User-Song Matrix

One of the drawbacks and problems with this matrix is the sparsity of data which is inevitably present, sparsity of data means the absence of substantial data, in the above matrix most of the fields are zero, as all users may not have heard each and every song [2].

To overcome this, normalization of the values between 0 and 1 using normalization technique has been done.

4.2 Normalisation

There are two methods of normalisation that have been implemented on the R matrix. They are:

4.2.1 L1 Normalisation

In this attempt, the values of the play counts are normalized for each user before splitting the matrix and applying the rest of the cosine similarity and weighted average procedure. Normalizing the matrix improves results because play counts of a song do not have rigid boundaries and different users may have different average play count numbers. In order to address the above issue, we decided to normalize the dataset between 0 and 1. On using this formula, each play count is between 0 and 1 such that the cumulative play count will be 1.

In this attempt the normalization formula, each play count is normalized between 0 and 1:

$$\boxed{\textbf{L1 Norm} = \frac{\textbf{User's Play Count for Particular Song}}{\textbf{User's Cumulative Play Count}}}$$

4.2.2 Min-Max Normalisation

On using this formula, every play count lies between 0 and 1 but the sum of the row is not necessarily 1. For the sake of convenience, this formula is referred to as min-max normalization.

$$\boxed{\textbf{Min-Max Norm} = \frac{\textbf{Play Count - Minimum Play Count}}{\textbf{Maximum Play Count - Minimum Play Count}}}$$

4.3 K Nearest Neighbour (KNN) Algorithm

K Nearest Neighbour is an non parametric lazy learning algorithm. Non parametric means that it does not make any assumptions on the underlying data distribution. This is useful as in the real world, most of the practical data does not obey the typical theoretical assumptions made(e.g. Gaussian mixtures, linearly separable, etc.) Non parametric algorithms like KNN are used here.

KNN algorithm, also called the lazy algorithm, means that it does not use the training data points to do any generalization. In other words, there is no explicit training phase or it is very minimal. This means the training phase is quite fast. Lack of generalization means that KNN keeps all the training data. More exactly, all the training data is needed during the testing phase. This is in contrast to other techniques like Support Vector Machine (SVM) where you can discard all non support vectors without any problem. Most of the lazy algorithms especially KNN make decision based on the entire training data set.

Given a query instance x_q to be classified, let x_1, x_2, ... x_k denote the k distances from training examples that are nearest to x_q.

For each training example $< x, f(x) >$, add the example to the list of training examples.

On applying the K Nearest Neighbour algorithm, the class is returned that represents the maximum of k instances[7].

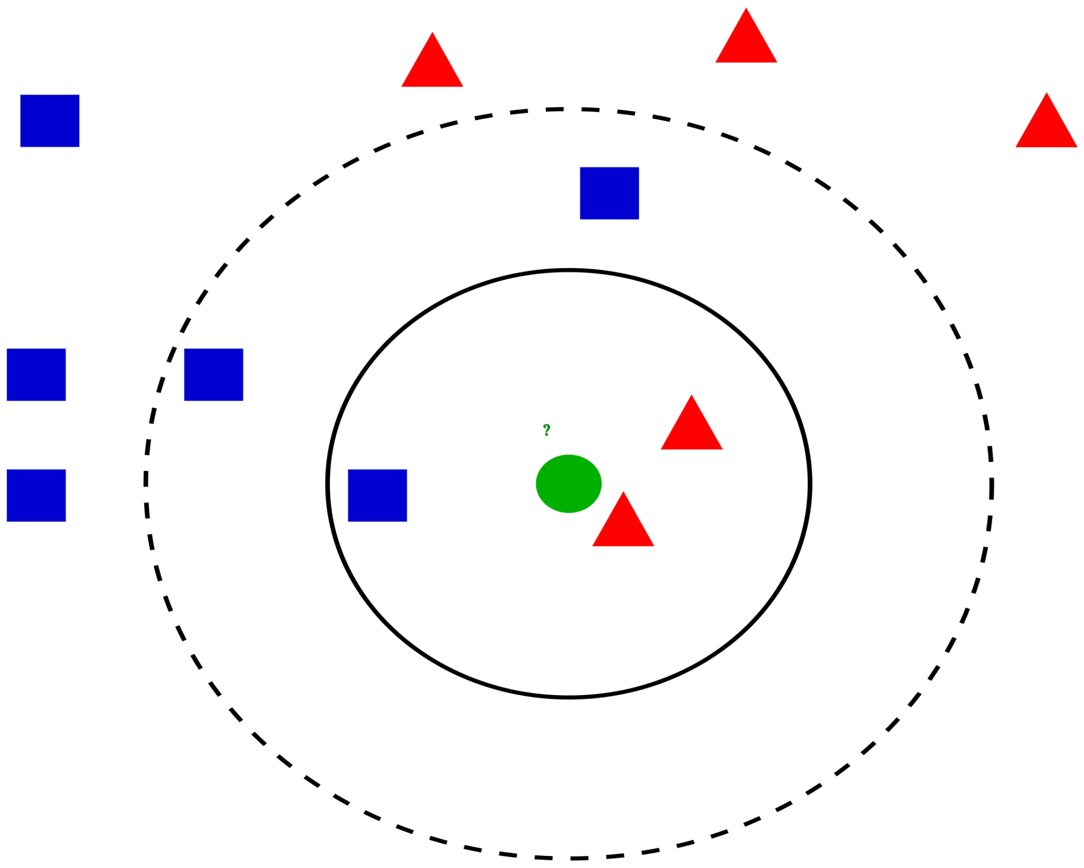


Figure 4.2: KNN Algorithm

where

🟦 is User 1

🔺 is User 2

🟢 is X_q

4.4 Similarity Functions

4.4.1 Weighted Average

This approach computes the prediction on an item *i* for a user *u* by computing the sum of the play counts given by the user on similar items to *i*. Each play count is weighted by the corresponding similarity $S_{i,j}$ between items i and j [8].

$$\boxed{P_{u,i} = \frac{\sum_{\text{all similar items, N}}(S_{i,N} * R_{u,N})}{\sum_{\text{all similar items, N}}(|S_{i,N}|)}}$$

where

$P_{u,i} \rightarrow$ Prediction on Item *i* for User *u*

$S_{i,N} \rightarrow$ Sum of ratings given by items *i* and *N*

$R_{u,N} \rightarrow$ Rating of User *u* on Item *N*

4.4.2 Cosine-based Similarity

The cosine similarity between two vectors is a measure that calculates the cosine of the angle between them. This is a measurement of orientation and not magnitude, it can be seen as a comparison between objects on a normalized space, because not only the magnitude, but also the angle between the object is also taken into consideration. For this it is necessary to build the cosine similarity equation is to solve the equation of the dot product for cos θ.

Two items are considered as two vectors in a dimensional space. The similarity between them is measured by computing the cosine of the angle between these two vectors [8].

$$\boxed{sim(i,j) = \frac{\sum_{u \in U}((R_{u,i})(R_{u,j}))}{\sqrt[2]{\sum_{u \in U}(R_{u,i})^2}\sqrt[2]{\sum_{u \in U}(R_{u,j})^2}}}$$

where

$sim(i,j) \rightarrow$ Similarity between item *i* and *j*

$R_{u,i} \rightarrow$ Rating of User *u* on Item *i*

$R_{u,j} \rightarrow$ Rating of User *u* on Item *j*

4.4.3 Adjusted Cosine Similarity

Computing similarity using basic cosine measure in item-based case has one important drawback-the difference in rating scale between different users are not taken into account. The adjusted cosine similarity offsets this drawback by subtracting the corresponding user average from each co-rated pair [3]. Formally, the similarity between items i and j using this scheme is given by

Here, corresponding user average is subtracted from each co-rated pair.

$$\boxed{sim(i,j) = \frac{\sum_{u\in U}(R_{u,i}-\vec{R_u})(R_{u,j}-\vec{R_u})}{\sqrt[2]{\sum_{u\in U}(R_{u,i}-\vec{R_u})^2}\sqrt[2]{\sum_{u\in U}(R_{u,j}-\vec{R_u})^2}}}$$

where

$sim(i,j) \rightarrow$ Similarity between item i and j

$R_{u,i} \rightarrow$ Rating of User u on Item i

$R_{u,j} \rightarrow$ Rating of User u on Item j

$\vec{R_u} \rightarrow$ Average of $U^{\textbf{th}}$ User Rating

4.5 Mean Absolute Error (MAE)

Measure of deviation of recommendations from their true user specified values.

Given formula:

$$\boxed{\textbf{MAE} = \frac{\sum_{i-1}^{N}|p_i - q_i|}{N}}$$

- p_i : Obtained Value
- q_i : Predicted Value
- N : Total Number of Songs

Chapter 5

Implementation

The EchoNest Taste Profile Subset [2] is used, which contains music history of 1 million users. It is in the form of triplets namely: $(user\_id, song\_id, play\_count)$.

5.1 Python Implementation

Step 1:
Use the EchoNest Taste Profile Subset and map it to a vectorized matrix in the same format using Python [9].

Step 2:
Derived the user-item matrix R for this subset using the above matrix, R contains one row for every user and the columns represent the play count for every unique song. For the songs that the user has not heard, the play count is zero. For performance reasons, the dimensions were limited to $\mathbf{R}_{1M\times200}$ [**1M users x 200 songs**] of the R matrix. Thus, the R matrix is the user-item matrix for 1 million users and 200 songs.

Step 3:
Divided $\mathbf{R}_{1M\times200}$ to a train data of 180 songs $\mathbf{A}_{1M\times180}$ and test data of 20 songs $\mathbf{E}_{1M\times20}$.

Step 4:
For each song (E_i) in the test data, found K similar songs from the training dataset. Calculated cosine similarity E_i and all songs in the training data and extracted K songs with the highest similarity. This is put into another matrix $\mathbf{K}_{1M\times k}$. This matrix K gives the k most similar songs from training data to the test song E_i.

Step 5:
Take the weighted average of the K similar song vectors (columns of matrix K) for each user. After this, the predicted vector for each song E_i in the test data is obtained. Then, stacked the predicted vector for each song horizontally to make a predicted data matrix $\mathbf{P}_{1M\times 20}$.

Step 6:
Apply the cosine similarity algorithm to find the 10 most similar songs for each of the 20 test songs and take a weighted average of the 10 most similar songs for each of the 20 test songs and compute a predicted version of the test dataset matrix.
The MAE is the error between the original test data and the predicted data.

Step 7:
Normalize the values of the play counts for each user before splitting the matrix and applying the cosine similarity and weighted average algorithms. Normalization improves the results as play counts of a song do not have rigid boundaries and different users may have different average play count numbers, also normalized the dataset between 0 and 1.

Step 8:
Use a different formula to normalize the matrix and use the same algorithms to calculate the MAE.

Step 9:
Split the matrix before it is normalized and then normalize the training matrix $\mathbf{A}_{1M\times 180}$ and the test matrix $\mathbf{E}_{1M\times 20}$ separately. Use the formula in step 8 for normalizing them. This gives the best MAE value so far.

Chapter 6

Results and Discussions

After splitting the matrix into two parts,

- Training dataset
- Test dataset

Item-item based collaborative filtering technique [10] such as Cosine Similarity Algorithm is used and the weighted average is taken. After this, MAE is calculated. Based upon the calculated MAE after each step, the matrix which is split into,

- Training dataset $\mathbf{A}_{1M\times380}$
- Test dataset $\mathbf{E}_{1M\times20}$

yields the lowest MAE.

Sr.No.	Approach	Calculated MAE
1	Initial Approach	0.097
2	L1 Norm	0.045
3	L1 Norm (Min-max)	0.111
4	Normalization Train and Test	0.029
5	1Mx20 (400)	0.010
6	1Mx80 (400)	0.0137

Chapter 7

Conclusion and Future Scope

7.1 Conclusion

Have been implemented the KNN Algorithm and Similarity Functions, i.e, Weighted Average and Cosine-Similarity Function to be precise, on the R Matrix, the basis of comparison is the Mean Absolute Error (MAE).

The test and train data is changed from training matrix $\mathbf{A}_{1M\times180}$ and the test matrix $\mathbf{E}_{1M\times20}$ to training matrix $\mathbf{A}_{1M\times380}$ and the test matrix $\mathbf{E}_{1M\times20}$. This is done to consider more songs to give out a better predicted value keeping in mind the data sparsity.

7.2 Future Scope

- The test and train data should be chosen at a random. This will help in having a better predicted ratings.
- As the basis of comparison of the result of different algorithms is very subjective, the one in this project is MAE. The basis of comparison could be changed and the different now could be noted [12].
- The predicted data ratings could be implemented on a dynamic database of music.

References

[1] Million Song Dataset — scaling MIR research. *Labrosa.ee.columbia.edu,* 2011. `http://labrosa.ee.columbia.edu/millionsong`

[2] Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere. The Million Song Dataset. *In Proceedings of the 12th International Society for Music Information Retrieval Conference (ISMIR),* 2011.

[3] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl,Item-Based Collaborative Filtering Recommendation Algorithms, *GroupLens Research Group/Army HPC Research Center Department of Computer Science and Engineering,* University of Minnesota, Minneapolis, MN 55455, April 2001.

[4] Sarwar, B. M., Karypis, G., Konstan, J. A., and Riedl, J. (2000). Application of Dimensionality Reduction in Recommender System - A Case Study. *In ACM WebKDD Workshop,* 2000.

[5] Sarwar, B. M., Karypis, G., Konstan, J. A., and Riedl, J. (2000). Analysis of Recommendation Algorithms for E-Commerce. *Proceedings of the ACM EC'00 Conference.* Minneapolis, MN. pp. 158-167.

[6] Yading Song, Simon Dixon, and Marcus Pearce, ”A Survey of Music Recommendation Systems and Future Perspectives”, *Centre for Digital Music Queen Mary University of London,* London, 2012.

[7] OpenCV-Python Tutorialss documentation, OpenCV-Python Tutorials 1 documentation. *Opencv-python-tutroals.readthedocs.org,* 2013. `https://opencv-python-tutroals.readthedocs.org/en/latest`

[8] C. Perone, ”Machine Learning :: Cosine Similarity for Vector Space Models (Part III) — Terra Incognita. *Blog.christianperone.com,* 2013. `http://blog.christianperone.com/2013/09/machine-learning-cosine-similarity-for-vector-space-models-part-iii`

[9] HDF5 for Python. *H5py.org,* 2014. `http://www.h5py.org/index.html`

[10] J. A. Konstan, B. N. Miller, D. Maltz, J. L. Herlocker, L. R. Gordon, and J. Riedl, ”GroupLens: Applying collaborative filtering to Usenet news,” *Communications of the ACM,* vol. 40, no. 3, pp. 7787, Mar. 1997.

[11] R. Burke, Hybrid recommender systems: Survey and experiments, *User Modeling and User-Adapted Interaction,* vol. 12, no. 4, pp. 331370, November 2002.

[12] G. Adomavicius and A. Tuzhilin, Towards the next generation of recommender systems: a survey of the state-of-the-art and possible extensions, *IEEE Trans. on Data and Knowledge Engineering,* 17:6, pp. 734 749, 2005.