Analysis of Time Aware and Semantic Feature Based Music Recommender System

7th Semester Mini Project

Submitted by

Shubham Mehrotra (IIT2012156)
Dhruv Kumar (IIT2012171)
Ronish Kalia (IIT2012139)

Supervisor:

Prof. OP Vyas
IIIT-Allahabad

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY, ALLAHABAD

CERTIFICATE FROM SUPERVISOR

I do hereby recommend that the mini project report prepared under my supervision, titled "**Analysis of Time Aware and Semantic Feature Based Music Recommender System**" be accepted in the fulfillment of the requirements of the completion of end semester of seventh semester of Bachelor of Technology in Information Technology**.**

Date: 02.01.2015 **Guide's name**

Place: Allahabad

_____

Prof OP Vyas

Table of Contents

1. Introduction

Recommender systems or recommendation systems (sometimes replacing "system" with a synonym such as platform or engine) are a subclass of information filtering system that seek to predict the 'rating' or 'preference' that a user would give to an item.

Recommender systems typically produce a list of recommendations in one of two ways - through collaborative or content-based filtering. Collaborative filtering approaches building a model from a user's past behavior (items previously purchased or selected and/or numerical ratings given to those items) as well as similar decisions made by other users. This model is then used to predict items (or ratings for items) that the user may have an interest in. Content-based filtering approaches utilize a series of discrete characteristics of an item in order to recommend additional items with similar properties. These approaches are often combined (see Hybrid Recommender Systems).

The goal of recommender systems is to make personalized product recommendations based on users' taste. As the Netflix challenge demonstrated, one of the most effective way to build such systems is through matrix factorization. Matrix factorization algorithms utilize prior product feedback given by users to automatically build user and product profiles. A product can then be recommended to a user if the user's profile closely matches that of the product.

Unfortunately, most of the research in matrix factorization focuses on explicit feedback datasets, where users make their preferences known by directly rating subsets of available products on a fixed scale. In many real-worlds applications, however, such direct ratings are unavailable. Instead, implicit feedback must be used, such as browsing history and view counts. In this work we present a recommendation algorithm which uses implicit play frequency information to recommend artists to the users of the Last.fm1 music website. We demonstrate how an existing algorithm for explicit feedback datasets can be adapted to work with implicit data. We further evaluate the performance of our recommendation algorithm on a Last.fm dataset, investigating the effects of regularization, normalization and the number of features on the quality of recommendations. We also discuss the viability of our approach in a real-world setting.

2. Literature survey

In this section we first explain the basic Recommender Systems then Semantic Web and at the end we explained some approaches already presents for combining the information of rating matrix and semantic web for the task of rate prediction.

a) Recommender System

Research on Recommender systems are not new, it has been in progress since 90's. Tapestry (UserBased CF, 1992) [1], GroupLens (Collaborative Filtering, 1994) [2], NewsWeeder (Content Based, 1995) [3], Bellcore video recommender (Collaborative Filtering, 1995) [4], InfoFinder (Content Based, 1996) [5], PHOAKS (Collaborative Filtering, 1997) [6], Fab (Hybrid Based, 1997) [7], PTV (Hybrid Based, 2001) [8], Amazon (Item based Collaborative Filtering, 2003) **[9],** Pandora (Content Based, 2008) and plethora of applications are designed in this journey of research. RSs act as the agent (software) to recommend a person before s (he) explicitly show his/her likings towards a particular item. It also helps a user to overcome the problem of information overflow. Web 2.0 based recommendation most commonly known as social recommendation is another growing branch of Recommender System, since 2004. In recent past tag based recommendations **[10]** are used for recommending semantically related tags for recommendation purpose. Tags help users for searching and quantify the objects according to their thinking. Mining opinion or reviews and comments also add extra information to the features and produce better meaningful prediction. Many other Recommender Systems **[10-15]** are proposed in last few years not only to improve the prediction but also its application for new area and with new dimensions, examples are blog recommendation and context aware movie recommendation for mobile app. RS had given a boom in traditional marketing environment (E- Commerce) **[16].** Elearning is also an emerging area, where the scope of RSs has been increases in recent years. These systems improve the quality of presenting E-learning material by associate it various interactive or visual examples. These gathering of information have been possible by different Knowledge Base and open source learning material ex: MOOC. Agrawal et. al (2013) **[17]** presents an idea for study navigator application which improves the experience of studying from electronic textbooks. RSs use techniques from different well establish area of research.

b) Semantic Web

In 2006, Berners-Lee et. al. **[18]** introduces a new technology (SW) for providing structured data in place of unstructured web pages, with sufficient background knowledge, less ambiguous and ease of access. Besides of these advancements it gives partial benefit in RS because most of web pages are un-annotated, thus cannot utilize by SW. However, the joint efforts of the project team of Linked Open Data (LOD)crafted a big picture of SW technology and ready to reduce the above disadvantages (Lorna 2010, Heiner et al. 2012) **[19] [20].** Web 3.0 (Internet of things), has structured knowledge in RDF (Resource Description Format). It has power of linkage, which can be explained as follows: the concept node (subject) has links to the related information, here the related information is called as 'object' node and links are called as 'properties'. These objects can be further explored by different properties. Thus, object node of the first triplet will represent as a concept node (subject) for the second triplet i.e. (<subject1>-<property1>-<object1>---<subject2>-<property2>-<object2>). Here, 'object1' and 'subject2' denotes same node. This chain continues until the object does not have any further information. The ending of the object node will always be represented by a literal or constant value, however in former (chain) a URI denotes the corresponding object and continues further. The links or properties have well defined meaning provided by Web Ontology Language (OWL). Many researchers and organizations contribute to donate RDF datasets (after conversion) For ex. IMDB-->LinkedMDB, MusicBrainz->LinkedBrainz, Wikipedia-->DBpedia. These datasets are not independent from each other. The concept node of one dataset can be linked with various other domain/crossdomain datasets residing in Linked Open Data. For ex. DBpedia and DrugBank have 4220 same as links. These links have been assigned by special vocabulary term, "owl-sameAs". Linked Open Data provides knowledge enhancement quality from fetch able and openly available data source. Linked Open Data provide various SPARQL endpoints to access this data directly from the online server using SPARQL querying [21]. However, in implementation point of view these global servers don't allow more than 7000 queries from one IP address. There are various local databases like Sesame, Alleograph, Virtuoso, 4Store, Star Dog **[22] [23] [24] [25]** dedicated to RDF storage in memory efficient and for quick retrieval.

c) Recommender System with Side information

Approaches deal with Recommender Systems are broadly classified in memory based and model based techniques. In memory based we need to store whole matrix in the system and process it every time when we need to recommend a target item to the user. Thus, unlike model based it suffered from scalability, however it provide more accurate result in comparison of using less information in the case of model based approaches. Many authors tried to use memory based approach for combining the side information (user and item contextual information) [26][27][28]. Other authors proposed to use social trust network information in 2005 [29] and 2009 [30]. In 1998, Basu et. al. [31] solved recommendation for each user as a classification problem in model based. The main drawback of this approach was the limited ability to generalize thus suffered from over fitting. In 2006 & 2009 Wang et. al & Wetzer et. al. [32][33] proposed LSA and probabilistic LSA to integrate tag relation with the User * Item rating matrix. Rattenbury et. al (2009) [34] combined geographical topics from social media based on the flicker tags. Beside the application of Data Mining approaches Deep learning was also applied to integrate side information in 2006 by Hinton [35]. To improve the performance on high dimensional matrix and sparse dataset, Matrix factorization machines was also applied for this using CMF (Collective Matrix Factorization) or JMF (Joint Matrix Factorization) by Singh & Gordan et al. (2008) [36], Ma. et al. (2009,2010) [30], Zhen (2009) [37], Shi et. al. (2010,2011)[38][39]. These authors incorporate different type of contextual information for the Rating prediction for ex: Explicit & Implicit social tags, tag based information, location correlation, Wikipedia build category for landmark, mood specific information).

3. Proposed Approach

Our main aim in this project is to incorporate side information in basic rating matrix on explicit data. Here, we considered semantic item * feature matrix and tag-time as our side information.

We have completed the analysis of the last fm dataset and used the implicit technique (where we have taken into account the number of clicks a user has made on a particular artist.

3.1 Matrix Factorization:

Just as its name suggests, matrix factorization is to, obviously, factorize a matrix, i.e. to find out two (or more) matrices such that when you multiply them you will get back the original matrix.

In a recommendation system such as Netflix or MovieLens, there is a group of users and a set of items (movies for the above two systems). Given that each users have rated some items in the system, we would like to predict how the users would rate the items that they have not yet rated, such that we can make recommendations to the users. In this case, all the information we have about the existing ratings can be represented in a matrix

The intuition behind using matrix factorization to solve this problem is that there should be some latent features that determine how a user rates an item. For example, two users would give high ratings to a certain movie if they both like the actors/actresses of the movie, or if the movie is an action movie, which is a genre preferred by both users. Hence, if we can discover these latent features, we should be able to predict a rating with respect to a certain user and a certain item, because the features associated with the user should match with the features associated with the item.

In trying to discover the different features, we also make the assumption that the number of features would be smaller than the number of users and the number of items. It should not be difficult to understand this assumption because clearly it would not be reasonable to assume that each user is associated with a unique feature (although this is not impossible). And anyway if this is the case there would be no point in making recommendations, because each of these users would not be interested in the items rated by other users. Similarly, the same argument applies to the items.

Firstly, we have a set U of users, and a set D of items. Let $\mathbf{R}$ of size $|U| \times |D|$ be the matrix that contains all the ratings that the users have assigned to the items. Also, we assume that we would like to discover K latent features. Our task, then, is to find two matrics matrices $\mathbf{P}$ (a $|U| \times K$ matrix) and $\mathbf{Q}$ (a $|D| \times K$ matrix) such that their product approximates $\mathbf{R}$:

$$\mathbf{R} \approx \mathbf{P} \times \mathbf{Q}^T = \hat{\mathbf{R}}$$

In this way, each row of $\mathbf{P}$ would represent the strength of the associations between a user and the features. Similarly, each row of $\mathbf{Q}$ would represent the strength of the associations between an item and the features. To get the prediction of a rating of an item d_j by u_i, we can calculate the dot product of the two vectors corresponding to u_i and d_j:

$$\hat{r}_{ij} = p_i^T q_j = \sum_{k=1}^{k} p_{ik} q_{kj}$$

Now, we have to find a way to obtain $\mathbf{P}$ and $\mathbf{Q}$. One way to approach this problem is the first intialize the two matrices with some values, calculate how `different' their product is to $\mathbf{M}$, and then try to minimize this difference iteratively. Such a method is called gradient descent, aiming at finding a local minimum of the difference.

The difference here, usually called the error between the estimated rating and the real rating, can be calculated by the following equation for each user-item pair:

$$e_{ij}^2 = (r_{ij} - \hat{r}_{ij})^2 = (r_{ij} - \sum_{k=1}^{K} p_{ik} q_{kj})^2$$

Here we consider the squared error because the estimated rating can be either higher or lower than the real rating.

To minimize the error, we have to know in which direction we have to modify the values of p_{ik} and q_{kj}. In other words, we need to know the gradient at the current values, and therefore we differentiate the above equation with respect to these two variables separately:

$$\frac{\partial}{\partial p_{ik}} e_{ij}^2 = -2(r_{ij} - \hat{r}_{ij})(q_{kj}) = -2e_{ij}q_{kj}$$
$$\frac{\partial}{\partial q_{ik}} e_{ij}^2 = -2(r_{ij} - \hat{r}_{ij})(p_{ik}) = -2e_{ij}p_{ik}$$

Having obtained the gradient, we can now formulate the update rules for both p_{ik} and q_{kj}:

$$p'_{ik} = p_{ik} + \alpha \frac{\partial}{\partial p_{ik}} e_{ij}^2 = p_{ik} + 2\alpha e_{ij}q_{kj}$$
$$q'_{kj} = q_{kj} + \alpha \frac{\partial}{\partial q_{kj}} e_{ij}^2 = q_{kj} + 2\alpha e_{ij}p_{ik}$$

Here, α is a constant whose value determines the rate of approaching the minimum. Usually we will choose a small value for , say 0.0002. This is because if we make too large a step towards the minimum we may run into the risk of missing the minimum and end up oscillating around the minimum.

A question might have come to your mind by now: if we find two matrices and such that approximates $\mathbf{R}$, isn't that our predictions of all the unseen ratings will all be zeros? In fact, we are not really trying to come up with and such that we can reproduce $\mathbf{R}$ exactly. Instead, we will only try to minimise the errors of the observed user-item pairs. In other words, if we let T be a set of tuples, each of which is in the form of (u_i, d_j, r_{ij}), such that T contains all the observed useritem pairs together with the associated ratings, we are only trying to minimise every e_{ij} for $(u_i, d_j, r_{ij}) \in T$. (In other words, T is our set of training data.) As for the rest of the unknowns, we will be able to determine their values once the associations between the users, items and features have been learnt.

Using the above update rules, we can then iteratively perform the operation until the error converges to its minimum. We can check the overall error as calculated using the following equation and determine when we should stop the process.

$$E = \sum_{(u_i,d_j,r_{ij})\in T} e_{ij} = \sum_{(u_i,d_j,r_{ij})\in T} (r_{ij} - \sum_{k=1}^{K} p_{ik}q_{kj})^2$$

3.2 Implicit Method

Our work is based on implicit user feedback for a music dataset. For each user we know the number of times the user played songs by a given artist, but we do not have direct ratings for that artist. As a result, in order to use the matrix factorization approaches described earlier, ratings must be estimated from play count information. We define play frequency freq for a given user i and artist j to be the user's play count for that artist normalized by user's total plays

$$freq_{i,j} = \frac{count(i,j)}{\sum_{j'} count(i,j')}$$

We also adopt the notation freqk(i) to denote the frequency of the k-th most listened to artist for user i. As Figure 1 shows, play frequencies have a clear power law distribution. A rating for an artist with rank k is computed as a linear function of the frequency percentile:

$$r_{i,j} = 4 \cdot \left(1 - \sum_{k'=1}^{k-1} freq_{k'}(i)\right)$$

It follows that we assign a rating in the 3-4 range to the artists in the top 25% frequency percentile, a rating of 2-3 to the artists in the subsequent 25% percentile, and so on. The least frequently listened to artists have a rating close to 0. The formulation in Equation 9 appears to transform the problem into the domain of explicit feedback, and makes it tempting to directly use unmodified

algorithms from that domain. In fact, as we show in the Experiments section, such an approach does indeed quickly converge to a reasonable global error, but the result is of no practical use due to the peculiarities of implicit music datasets. We describe these peculiarities, and how we deal with them, below.

The Implicit Problem

One of the most significant differences between explicit and implicit feedback datasets is the distribution of ratings. In an explicit setting, the average user only rates a reasonably small subset of products, and the ratings are not heavily skewed towards one end or the other. This is in contrast to implicit music datasets and ratings derived from play frequencies, where most artists have a rating close to 0 - a consequence of a power law distribution (see Figure 1). Such a skewed distribution has an impact on parameter estimation. A naive approach, i.e. directly optimizing Equation 1, is susceptible to getting stuck in a local minimum where, for each user, all available products have a low rating. While this is a reasonable approximation of the distribution, it is of little practical use, since in recommender systems we are interested in identifying products a user might actually like. We demonstrate this problem in the Experiments section, where we apply a Stochastic Gradient Descent algorithm, designed for explicit datasets, to implicit ratings derived from a Last.fm dataset.

Percentile Normalization

In order to deal with the skewed rating distribution, we propose a normalization scheme which amplifies the error (Equation 2) for highly rated artists. We accomplish this, for each user i, by assigning all artists to four non-overlapping bins based on the artists' rating (which in turn is based on the frequency percentile,

$$B_i^1 = \{j : 4 \geq r_{i,j} > 3\}$$
$$B_i^2 = \{j : 3 \geq r_{i,j} > 2\}$$
$$B_i^3 = \{j : 2 \geq r_{i,j} > 1\}$$
$$B_i^4 = \{j : 1 \geq r_{i,j} > 0\}$$

We then normalize errors within each bin by the square root of the bin's size. Formally, we define the

Percentile-normalized error as:

$$e_{i,j}^{norm} = \frac{r_{i,j} - u_i^T p_j}{\sqrt{\|B_i^t\|}}$$

where Bt i is the bin to which artist j belongs for user i. The normalization ensures that each individual bin's contribution to the total squared error has approximately the same weight, regardless of the bin's size. To make this clearer, suppose we always make a constant error _ on all samples. The total squared error for all samples in a bin t is then:

$$\sum_{r_{i,j} \in B_i^t} \left(\frac{r_{i,j} - u_i^T p_j}{\sqrt{\|B_i^t\|}} \right)^2 = \sum_{r_{i,j} \in B_i^t} \left(\frac{\epsilon}{\sqrt{\|B_i^t\|}} \right)^2 =$$

$$\|B_i^t\| \cdot \frac{\epsilon^2}{\|B_i^t\|} = \epsilon^2$$

Parameter Estimation

We estimate the user and product matrices U and P using a variant of the Stochastic Gradient Descent algorithm modified to work with the normalized error. For each rating ri,j and its associated user and artist vectors ui and pj , we compute the normalized error enorm i,j and update parameters according to:

$$u_i \leftarrow u_i + \gamma \cdot (e_{i,j}^{norm} \cdot p_j - \lambda \cdot u_i)$$
$$p_j \leftarrow p_j + \gamma \cdot (e_{i,j}^{norm} \cdot u_i - \lambda \cdot p_j)$$

We repeat the process for all ratings until convergence, or until a predefined maximum number of iterations.

3.3 Semantic Web

Semantic Web methodologies take advantages of the advancements in the semantic web technologies and features such as ontologies, taxonomies, social networks, tagging. We will

combine semantic web information into our data and try to decide how importance it is in producing the overall ratings.

Dbpedia is the source from where we have fetched data regarding each Artist or we can say items. In our case R = User X Artist and S = Artist X Categories are the Rating (R) and Semantic Information (S) matrix respectively. Because we want to add the information into Artist or item feature vector, so the updating equation will be changed as below

$$u_i \leftarrow u_i + \gamma \cdot \left(e_{i,j}^{norm} \cdot p_j - \lambda \cdot u_i\right)$$
$$p_j \leftarrow p_j + \gamma \cdot \left(e_{i,j}^{norm} \cdot u_i + 2\alpha\left(S - p_j c_n\right) - \lambda \cdot p_j\right)$$

3.4 Temporal Factor

Much of the temporal variability is included within the baseline predictors, through two major temporal effects. The first addresses the fact that an item's popularity may change over time. The second major temporal effect allows users to change their baseline ratings over time. Therefore we will also try to decide how important these temporal effects could be to the overall ratings by combining them in our method to calculate the overall ratings.

$$postScore_i = \lambda^{\triangle Time_i}$$

where

is the time decaying function which is smaller than '1', _=0.9.

other formula is related to Tag specificity, or we can say how often the tag has been rated using same tag by other users

$$TagSpeceficity = log(50 + tagCount_i)$$

To measure the overall tag score they have used following formula:

$$TagScore_i orc_{i,j} = \frac{\sum_i^n (PostScore)}{TagSpeceficity_i}$$

We can say Tag Score as ci;j , or the confidence coefficient for the Matrix T (Tag X Latent Factor), so to include this coefficient we will modify the Equation as follows:

$$\mathrm{u}_i \leftarrow u_i + \gamma \cdot \left(e_{i,j}^{norm} \cdot p_j - \lambda \cdot u_i \right)$$
$$p_j \leftarrow p_j + \gamma \cdot \left(e_{i,j}^{norm} \cdot u_i + 2\alpha \left(S - p_j c_n \right) + 2\beta \times c_{i,j} \left(T - p_j g_k \right) - \lambda \cdot p_j \right)$$

Comparison

Final aim of the project will be to compare existing results with the one calculated after taking the effects of semantic web and temporal factors into consideration. Our system would show the final recommendations based on this.

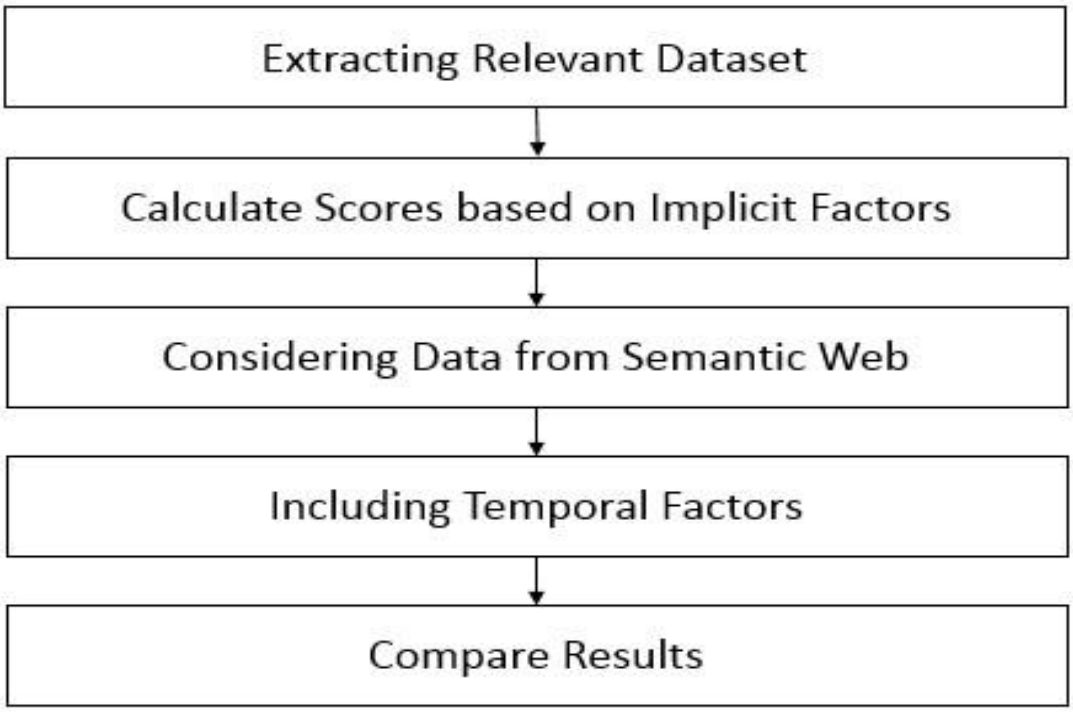
4. Software Used

Language: Python

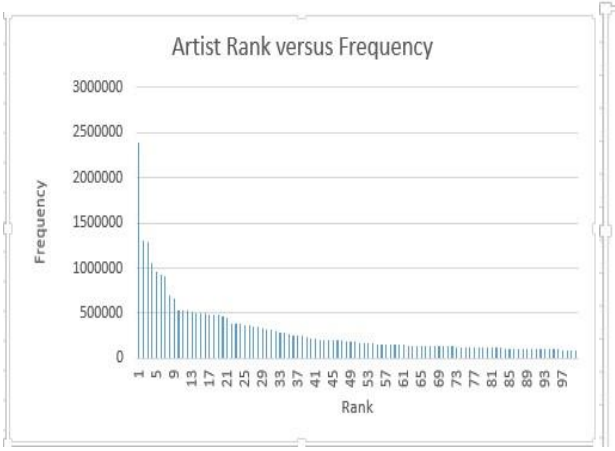
Libraries: NumPy, DateTime, Re, CSV

Software: MS Excel, IDLE

5. Activity Time Chart



6. Results



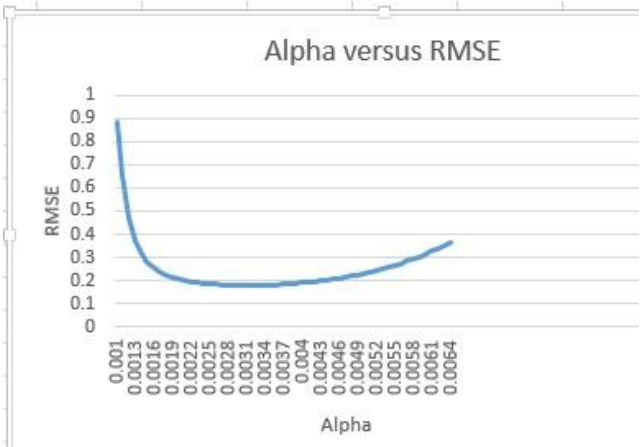
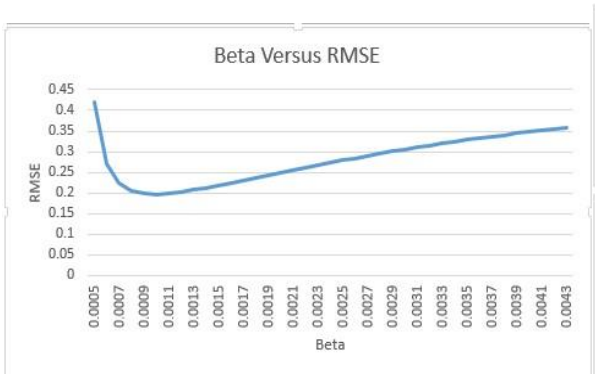


Fig 1. Artist Rank and Frequency of being heard Fig 2. Alpha versus Root Mean Square Error



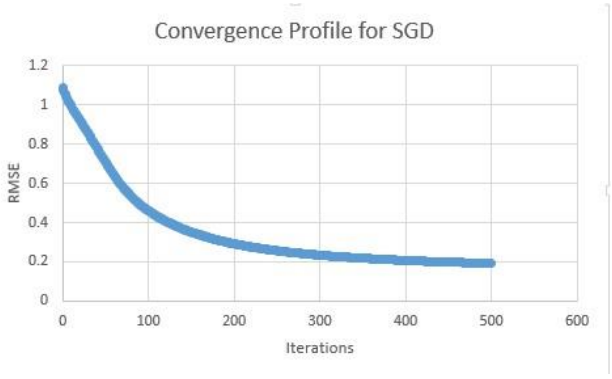
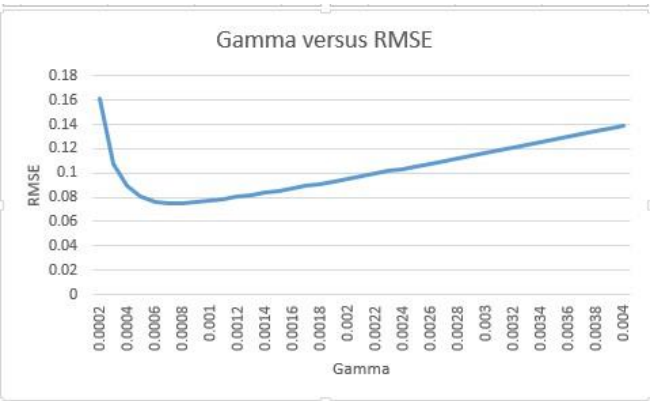


Fig 3. Beta Versus Root Mean Square Error Fig 4. Convergence profile MatrixFactorization



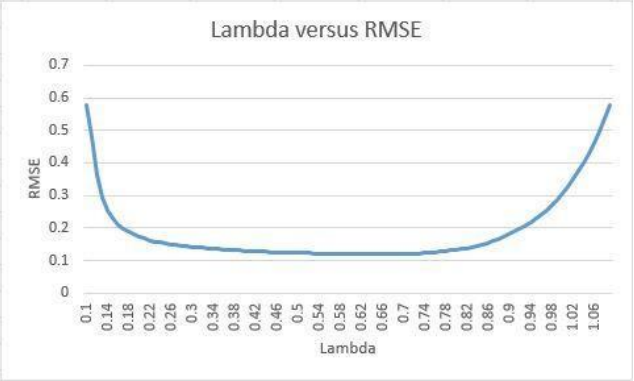


Fig 5. Gamma (Time Variant versus Root Mean Square Error)

Tag_time

Training Size	Test Size	Train Error	Test Error
8156 (20%)	33132	0.384579322934	0.486217337033
16611 (40%)	24677	0.34319899727	0.394204147192
24854 (60%)	16434	0.336208159975	0.384122427242
33005 (80%)	8283	0.323183635267	0.378329296078

Semantic

Training Size	Test Size	Train Error	Test Error
8198 (20%)	33090	0.291794410351	0.455146682153
16683(40%)	24605	0.270813517029	0.383589007124
24721 (60%)	16567	0.252987018573	0.335888042357
33061 (80%)	8227	0.240566742213	0.294084896122

Semantic_tag_time

Training Size	Test Size	Train Error	Test Error
8301 (20%)	32987	0.299473772339	0.369949341672
16611 (40%)	24677	0.274483475109	0.292670808659
24838 (60%)	16450	0.259931881909	0.244602200161
33100 (80%)	8188	0.250994254996	0.234908290027

7. Conclusion and Future Scope

We wish to incorporate time as a factor which would help us in predicting music to the user in a more personalized manner. Time factor would include the when a user prefers to listen to a particular type of music. We also plan to add information from semantic web technologies and features such as ontologies, taxonomies, social networks, tagging.

An addition of these techniques to the already existent methods would help us in giving a more personalized recommendation and hence improve the accuracy of our result.

We wish to implement this algorithm in a real time software to give more personalized recommendations. We would also like our system to be an active learner.

8. References

1. Goldberg D, Nichols D, Oki BM, Terry D (1992) Using Collaborative Filtering to weave an information TAPESTRY. Communications of the ACM. Vol. 35, No. 12.
2. Resnick P, Iacovou N, Suchak M, Bergstorm P, Riedl J (1994) GroupLens: An Open Architecture for Collaborative Filtering of Netnews. CSCW, ACM
3. Lang K (1995) NewsWeeder: Learning to Filter Netnews. ML 95
4. *Hill W, Stead L, Rosenstein M, Furnas G (1995) Recommending And Evaluating Choices In A Virtual Community Of Use. CHI'95*
5. Krulwich B, Burke C (1996) Learning user information interests through the extraction of semantically significant phrases. AAAI Technical Report
6. *Terveen L, Hill W, Amento B, McDonald D, Creter J (1997) PHOAKS: a system for sharing recommendations. Communications of the ACM. 40(3): p59(4)*
7. *Balabanovic M, Shoham Y (1997) Fab: content-based, collaborative recommendation. Special Section: Recommender Systems. v40; 66-73*
8. *Cotter P, Smyth B (2001)* PTV: Intelligent Personalised TV Guides. Proceedings of the 17th National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence. 957-964
9. Linden G, Smith B, York J (2003) Amazon.com recommendations: Item-to-item collaborative filtering. Internet Computing, IEEE. 7(1):76–80
10. Montanes E, Quevedo JR, Dıaz I, Ranilla J (2009) Collaborative Tag Recommendation System based on Logistic Regression. ECML PKDD Discovery Challenge.
11. Eyjolfsdottir EA, Tilak G, Li N (2007) MovieGEN: A Movie Recommendation System.
12. Symeonidis P, Nanopoulos A, Manolopoulos Y (2007) Feature-weighted User Model for Recommender Systems. User Modelling 2007, LNCS, Springer-Verlag Berlin Heidelberg. 4511: 97-106
13. Sarwar B, Karypis G, Konstan J, Riedl J (2010) Item-Based Collaborative Filtering Recommendation Algorithms. ACM, WWW10
14. Pazzani MJ, Billsus D (2007) Content-Based Recommendation Systems. The Adaptive Web, LNCS, Springer-Verlag Berlin Heidelberg. pp: 325-341

15. Uluyagmur M, Cataltepe Z, Tayfur E (2012) Content Based Movie Recommendation Using Different Feature Sets. Proceedings of the World Congress on Engineering and Computer Science. vol. 1
16. Vakratsas D, Ambler T (1999) How Advertising Works: What Do We Really Know?. Journal of Marketing. 63(1): 26-43
17. Agrawal R, Gollapudi S, Kannan A, Kenthapadi K (2013) Study Navigator: An Algorithmically Generated Aid for Learning from Electronic Textbooks. Microsoft Research Technical Report, MSR-TR-2013-68.
18. Bizer C, Heath T, Berners-Lee T (2009) Linked data – the story so far. Int. J. Semantic Web Inf. Syst.
19. Stuckenschmidt H (2012) Data semantics on the web. Journal on Data Semantics. 1:1-9
20. Campbell LM, MacNeill S (2010) The semantic web, linked and open data. JISC CETIS.
21. Prud'hommeaux E, Seaborne A (2008) Sparql query language for RDF. W3C recommendation. W3C.
22. Broekstra J, Kampman A, van Harmelen F (2012) Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema. ISWC.
23. Franz Inc (2010) AllegroGraph RDFStore Web 3.0's Database.
24. Erling O, Mikhailov I (2009) RDF support in the Virtuoso DBMS. Studies in Computational Intelligence. 221: 7-24
25. Harris S, Lamb N, Shadbolt N (2009) 4store: The Design and Implementation of a Clustered RDF Store.
26. Prem Melville, Raymod J. Mooney, and Ramadass Nagarajan. 2002. Content-boosted collaborative filtering for improved recommendations. In Proceedings of the 18th National Conference on Artificial Intelligence. American Association for Artificial Intelligence, Menlo Park, CA, 187–192. http://dl.acm.org/citation.cfm?id=777092.777124
27. Shilad Sen, Jesse Vig, and John Riedl. 2009. Tagommenders: Connecting users to items through tags. In Proceedings of the 18th International Conference on World Wide Web (WWW'09). ACM, New York, NY, 671–680. DOI:http://dx.doi.org/10.1145/1526709.1526800.

28. Shilad Sen, Jesse Vig, and John Riedl. 2009. Tagommenders: Connecting users to items through tags. In Proceedings of the 18th International Conference on World Wide Web (WWW'09). ACM, New York, NY,671–680.

DOI:http://dx.doi.org/10.1145/1526709.1526800

29. Jennifer Ann Golbeck. 2005. Computing and Applying Trust in Web-Based Social Networks. Ph.D. Dissertation. College Park, MD.

30. Mohsen Jamali and Martin Ester. 2009a. TrustWalker: A random walk model for combining trust-based and item-based recommendation. In Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'09). ACM, New York, NY, 397–406. DOI:http://dx.doi.org/10.1145/1557019.1557067

31. Chumki Basu, Haym Hirsh, and William Cohen. 1998. Recommendation as classification: Using social and content-based information in recommendation. In Proceedings of the 15th National/10th Conference on Artificial Intelligence/Innovative Applications of Artificial Intelligence. 714–720.

32. Xuanhui Wang, Jian-Tao Sun, Zheng Chen, and ChengXiang Zhai. 2006. Latent semantic analysis for multiple-type interrelated data objects. In Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'06). ACM, New York, NY, 236–243. DOI:http://dx.doi.org/10.1145/1148170.1148214

33. Robert Wetzker, Winfried Umbrath, and Alan Said. 2009. A hybrid approach to item recommendation in folksonomies. In Proceedings of the WSDM Workshop on Exploiting Semantic Annotations in Information Retrieval (ESAIR'09). ACM, New York, NY, DOI:http://dx.doi.org/10.1145/1506250.1506255

34. Tye Rattenbury and Mor Naaman. 2009. Methods for extracting place semantics from Flickr tags. ACM Transactions on the Web 3, 1 (January 2009), Article 1. DOI:http://dx.doi.org/10.1145/1462148.1462149

35. Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. 2006. A fast learning algorithm for deep belief nets.Neural Computation 18, 7, 1527–1554.

36. Ajit P. Singh and Geoffrey J. Gordon. 2008. Relational learning via collective matrix factorization. In Proceeding of the 14th ACM SIGKDD International Conference on

Knowledge Discovery and Data Mining (KDD'08). ACM, New York, NY, 650–658. DOI:http://dx.doi.org/10.1145/1401890.1401969

37. Yi Zhen, Wu-Jun Li, and Dit-Yan Yeung. 2009. TagiCoFi: Tag informed collaborative filtering. In Proceedings of the 3rd ACM Conference on Recommender Systems (RecSys'09). ACM, New York, NY, 69–76.DOI:http://dx.doi.org/10.1145/1639714.1639727
38. Yue Shi, Pavel Serdyukov, Alan Hanjalic, and Martha Larson. 2011c. Personalized landmark recommendation based on geotags from photo sharing sites. In Proceedings of the 5th International Conference on Weblogs and Social Media (ICWSM'11). 622–625.
39. Yue Shi, Pavel Serdyukov, Alan Hanjalic, and Martha Larson. 2013. Nontrivial landmark recommendation using geotagged photos. ACM Transactions on Intelligent Systems and Technology 4, 3 (July 2013), Article 47. DOI:http://dx.doi.org/10.1145/2483669.2483680

9. Suggestion by Board Members