Building A Recommendation System: Million Song Dataset Challenge

Sujay Jagtap[1] , Raja Parmar[2] , Prof. Sanjeev Dwivedi[3]
[1, 2, 3] Dept of Computer
[1, 2, 3] Vidyalankar Institute of Technology , Mumbai , India.

Abstract- *The Million Song Dataset Challenge is an open, offline music recommendation evaluation challenge. The task of this challenge is to predict what people might want to listen to based on their pre-recorded listening history. This is an offline evaluation challenge meaning the actual listening history is already available, so the recommendations made by the system will be directly compared with the original history. This was a Kaggle competition introduced in 2008 and the aim of it was to design a recommendation model and the participant with the best results would be awarded.*

We took upon this challenge and built a recommendation system for it using with combination of Collaborative filtering and content Based recommendations and have presented our results in this paper.

Keywords- recommendation systems, collaborative filtering, machine learning

I. INTRODUCTION

With increase in number of services like Spotify, Netflix, Amazon Prime etc. there is strong need for big companies to build a reliable recommendation system which not only would be enticing to the users but also be good financial incentive to the businesses. A good recommendation system provides a user with valuable options(songs in this case) based on their preferences and also improves its performance overtime as more users join in the service and new items are introduced in the system.

There are many issues which arise when building a robust recommendation system. One of the issues is that there is no direct similarity measurement tool to measure the similarity between two songs. Of course we can check if two songs are similar based on their metadata, but even that would not be a great way to do it as an artist may have two completely different sounding songs which could be very similar but a user may not like both of them equally. Another issue is in identifying useful features of the songs; determining which features would be best for a learning algorithm is important, since good feature selection could improve recommendations.

Another major issue which frequently comes up is something which called as “Cold Start”. Which is essentially a situation deals with the fact that how does a recommendation system handles a new user or a new item. If there is enough data already available associated with the users preferences it is particularly easy to understand a users likes and dislikes and then model the recommendations accordingly. But when a new user comes in the system there is hardly any relevant information available. In such a situation it is imperative of a good recommendation to accommodate new user as well. So while developing a recommendation model both the situations “Warm Start”and “Cold Start”should be managed with proficiency. This problem is one of the active areas of research in this field. Majority of the research focuses on creating a dense representation of a user as well a song which would take care of the computing resources as well the cold start situation. Our approach also relies on creating a dense representation of a user and an item.

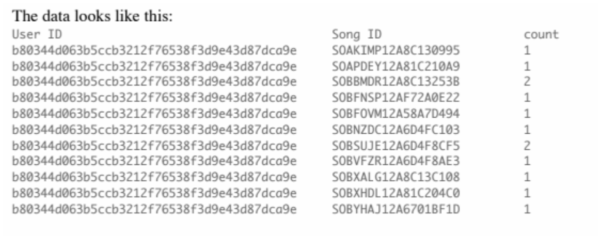
While developing the system we also needed to decide what all metadata and song features should we use. The MSD dataset was of huge size approx. 280 GB which was not possible for us to process. So we decided to do pure collaborative filtering which was essentially a metadata less approach to this problem. The specifics of the problem have been discussed in detail in the next section.

II. PROBLEM DEFINITION

The task of this challenge is to provide recommendations for 110K users. The datasets available for this are:

1. The Taste Profile Subset

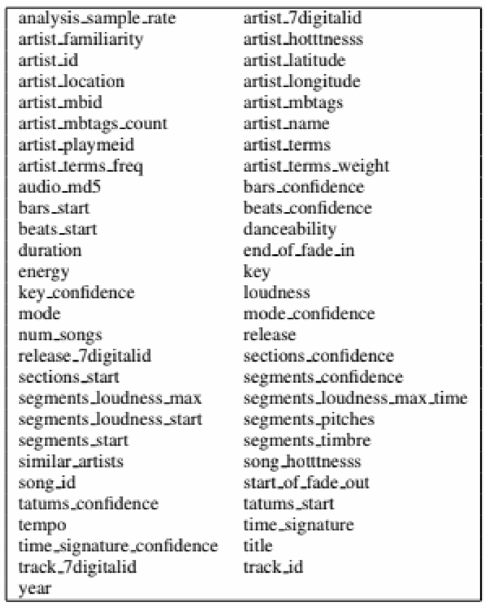
This dataset contains information about 1M users and their listening history. It is structured in form of triplets (userid,songid,playcount)


The data looks like this:

User ID	Song ID	count
b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOAKIMP12A8C130995	1
b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOAPDEY12A81C210A9	1
b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOBBMDR12A8C13253B	2
b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOBFNSP12AF72A0E22	1
b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOBFOVM12A58A7D494	1
b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOBNZDC12A6D4FC103	1
b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOBSUJE12A6D4F8CF5	2
b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOBVFZR12A6D4F8AE3	1
b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOBXALG12A8C13C108	1
b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOBXHDL12A81C204C0	1
b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOBYHAJ12A6701BF1D	1

2. The Million Song Dataset

This dataset contains the metadata of 1M songs. Each song contains of 55 unique features.



analysis_sample_rate	artist_7digitalid
artist_familiarity	artist_hotttnesss
artist_id	artist_latitude
artist_location	artist_longitude
artist_mbid	artist_mbtags
artist_mbtags_count	artist_name
artist_playmeid	artist_terms
artist_terms_freq	artist_terms_weight
audio_md5	bars_confidence
bars_start	beats_confidence
beats_start	danceability
duration	end_of_fade_in
energy	key
key_confidence	loudness
mode	mode_confidence
num_songs	release
release_7digitalid	sections_confidence
sections_start	segments_confidence
segments_loudness_max	segments_loudness_max_time
segments_loudness_start	segments_pitches
segments_start	segments_timbre
similar_artists	song_hotttnesss
song_id	start_of_fade_out
tatums_confidence	tatums_start
tempo	time_signature
time_signature_confidence	title
track_7digitalid	track_id
year	

Table 2. List of the 55 fields provided in each per-song HDF5 file in the MSD.

3. The test and validation dataset

This dataset is structured similarly to the taste profile subset. But it contains information about 110k users. This information is then divided into two datasets visible and hidden. Visible is the one which is available to us and hidden is the one on which the actual recommendations will be tested.

So the task is if given only half of the listening history of 110k users ,we need to predict the remaining half of the actual listening history. For that we can use all the above datasets to model our system. Once we are done with the recommendations we then need to check how accurate is our system. For testing the accuracy we have used Precision, Recall and F-score as our three evaluation parameters.

III. SCOPE

Even though we built a recommender system for Kaggle’s challenge, the general approach which we used to do it can be scaled to multitude of different domains. Our system takes into account only the history of users and item , and that information is readily available with most of websites. The fact that we are using a latent representation of a user and an item, that representation is as concise as possible and it also takes into account the entirety of user history available.

Also, the vector is of much smaller dimension than the data available ,which makes it storage friendly. The smaller size also accounts to faster computations of recommendations which would always be preferred by an organisation.

With simple modifications our proposed system can be tuned to any recommendation domain such as E-commerce, Video Streaming etc.

IV. PROPOSED SYSTEM

Symbol	Description
n_u	Number of users
n_m	Number of songs
$x^{(i)}$	Feature vector of song *i*
n	Number of features
$\Theta^{(i)}$	User preference for song *i*
R_{ij}	Rating provided by user *j* to song *i*
r_{ij}	Binary value depending upon the presence of R_{ij}
α	Learning rate
□	Regularisation constant

Prior Information:

Dataset provided by Kaggle structures data into tuples governed by following schema.

User_ID	Song_ID	Playcount

For convenience the data is structured again into a matrix form which has the following representation

$$\begin{bmatrix} R11 & \cdots & R1N \\ \vdots & \ddots & \vdots \\ RN1 & \cdots & RNN \end{bmatrix}$$

The problem is to predict the missing entries of the above matrix.

There are two common approaches generally used to solve recommender system problems.

- Content Based Recommendation.
- Collaborative Filtering.

Both the topics can be summarised as follows:

Content Based Recommendation:Content based algorithms are algorithms that attempt to recommend items that are similar to items the user liked in the past. They treat the recommendation's problem as a search for related items. Information about each item is stored and used for the recommendations. Items selected for recommendation are items that content correlates the most with the user's preferences. Content based algorithms analyze item descriptions to identify items that are of particular interest to the user.

Collaborative Filtering: Collaborative filtering algorithms (CF) are algorithms that require the recommendation seekers to express their preferences by rating items. In this algorithm, the roles of recommendation seeker (a user) and preference provider are merged; the more users rate items (or categories), the more accurate the recommendation becomes. In most CF approaches, there is a list of users U = u_1, u_2, etc and a list of items I = i_1, i_2, etc. Each user u_i has a list of itemI_{ui} on which he has expressed his opinion.

Mathematical representations of the above ideas involve using concepts like linear regression and gradient descent (optimization purpose).

Before showing how above ideas can be proposed mathematically it is implicitly assumed that the following approaches have been unraveled in the context of the problem of solving the Million Song Recommendation challenge, however with some efforts one can map the technique to most of the recommendation problems existing in the real world
The same ideas can be mathematically stated as follows:

Content Based Recommendation:

For data representation we need two vectors which are named as x and Θ. Every element j of the vector $x^{(i)}$ is score of the song j for the feature'i'. Every element i of the vector $\Theta^{(i)}$ represents choice of the user for the feature 'i'.

Now if we assume that one already has both the vectors readily available one can easily predict a rating for a song which has not been rated by the user previously as $\Theta^{(i)T} * x^{(i)}$

Now the problem for solving the recommendation problem using Content we just need the Preference vectors for every user which can be mathematically stated as follows
Cost function:

To learn $\theta^{(j)}$ (parameter for user j):

$$\min_{\theta^{(j)}} \frac{1}{2} \sum_{i:r(i,j)=1} \left((\theta^{(j)})^T x^{(i)} - y^{(i,j)} \right)^2 + \frac{\lambda}{2} \sum_{k=1}^{n} (\theta_k^{(j)})^2$$

To learn $\theta^{(1)}, \theta^{(2)}, \ldots, \theta^{(n_u)}$:

$$\min_{\theta^{(1)},\ldots,\theta^{(n_u)}} \frac{1}{2} \sum_{j=1}^{n_u} \sum_{i:r(i,j)=1} \left((\theta^{(j)})^T x^{(i)} - y^{(i,j)} \right)^2 + \frac{\lambda}{2} \sum_{j=1}^{n_u} \sum_{k=1}^{n} (\theta_k^{(j)})^2$$

Optimisation function

Optimization algorithm:

$$\min_{\theta^{(1)},\ldots,\theta^{(n_u)}} \frac{1}{2} \sum_{j=1}^{n_u} \sum_{i:r(i,j)=1} \left((\theta^{(j)})^T x^{(i)} - y^{(i,j)} \right)^2 + \frac{\lambda}{2} \sum_{j=1}^{n_u} \sum_{k=1}^{n} (\theta_k^{(j)})^2$$

Gradient descent update:

$$\theta_k^{(j)} := \theta_k^{(j)} - \alpha \sum_{i:r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)}) x_k^{(i)} \quad (\text{for } k = 0)$$

$$\theta_k^{(j)} := \theta_k^{(j)} - \alpha \left(\sum_{i:r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)}) x_k^{(i)} + \lambda \theta_k^{(j)} \right) \quad (\text{for } k \neq 0)$$

Collaborative Filtering:

In the above method we saw how one can predict the missing values in the matrix if one has feature vectors and preference vectors available for computation.

But Collaborative filtering solves the problem of recommendation even if feature vectors are unavailable. It computes both the vectors $\Theta^{(i)}$ and $x^{(i)}$ in dovetailing fashion and uses optimisation function in turn to optimize both the vectors.

Optimization algorithm

Given $\theta^{(1)},\ldots,\theta^{(n_u)}$, to learn $x^{(i)}$:

$$\min_{x^{(i)}} \frac{1}{2} \sum_{j:r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)})^2 + \frac{\lambda}{2} \sum_{k=1}^{n} (x_k^{(i)})^2$$

Given $\theta^{(1)},\ldots,\theta^{(n_u)}$, to learn $x^{(1)},\ldots,x^{(n_m)}$:

$$\min_{x^{(1)},\ldots,x^{(n_m)}} \frac{1}{2} \sum_{i=1}^{n_m} \sum_{j:r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)})^2 + \frac{\lambda}{2} \sum_{i=1}^{n_m} \sum_{k=1}^{n} (x_k^{(i)})^2$$

Using the above approach one has to follow a specific sequence of computation to get minimal errors, however one can solve the problem by combining both the equations and integrating it as a single program efficiently in the following manner.

Collaborative filtering optimization objective

Given $x^{(1)},\ldots,x^{(n_m)}$ estimate $\theta^{(1)},\ldots,\theta^{(n_u)}$:

$$\min_{\theta^{(1)},\ldots,\theta^{(n_u)}} \frac{1}{2} \sum_{j=1}^{n_u} \sum_{i:r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)})^2 + \frac{\lambda}{2} \sum_{j=1}^{n_u} \sum_{k=1}^{n} (\theta_k^{(j)})^2$$

Given $\theta^{(1)},\ldots,\theta^{(n_u)}$ estimate $x^{(1)},\ldots,x^{(n_m)}$:

$$\min_{x^{(1)},\ldots,x^{(n_m)}} \frac{1}{2} \sum_{i=1}^{n_m} \sum_{i:r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)})^2 + \frac{\lambda}{2} \sum_{i=1}^{n_m} \sum_{k=1}^{n} (x_k^{(i)})^2$$

Minimizing $x^{(1)},\ldots,x^{(n_m)}$ and $\theta^{(1)},\ldots,\theta^{(n_u)}$ simultaneously:

$$J(x^{(1)},\ldots,x^{(n_m)},\theta^{(1)},\ldots,\theta^{(n_u)}) = \frac{1}{2} \sum_{(i,j):r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)})^2 + \frac{\lambda}{2} \sum_{i=1}^{n_m} \sum_{k=1}^{n} (x_k^{(i)})^2 + \frac{\lambda}{2} \sum_{j=1}^{n_u} \sum_{k=1}^{n} (\theta_k^{(j)})^2$$

$$\min_{\substack{x^{(1)},\ldots,x^{(n_m)} \\ \theta^{(1)},\ldots,\theta^{(n_u)}}} J(x^{(1)},\ldots,x^{(n_m)},\theta^{(1)},\ldots,\theta^{(n_u)})$$

Hence the entire collaborative filtering process can be summarised as follows.

Collaborative filtering algorithm

1. Initialize $x^{(1)},\ldots,x^{(n_m)},\theta^{(1)},\ldots,\theta^{(n_u)}$ to small random values.
2. Minimize $J(x^{(1)},\ldots,x^{(n_m)},\theta^{(1)},\ldots,\theta^{(n_u)})$ using gradient descent (or an advanced optimization algorithm). E.g. for every $j = 1,\ldots,n_u, i = 1,\ldots,n_m$:

$$x_k^{(i)} := x_k^{(i)} - \alpha \left(\sum_{j:r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)})\theta_k^{(j)} + \lambda x_k^{(i)} \right)$$

$$\theta_k^{(j)} := \theta_k^{(j)} - \alpha \left(\sum_{i:r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)})x_k^{(i)} + \lambda \theta_k^{(j)} \right)$$

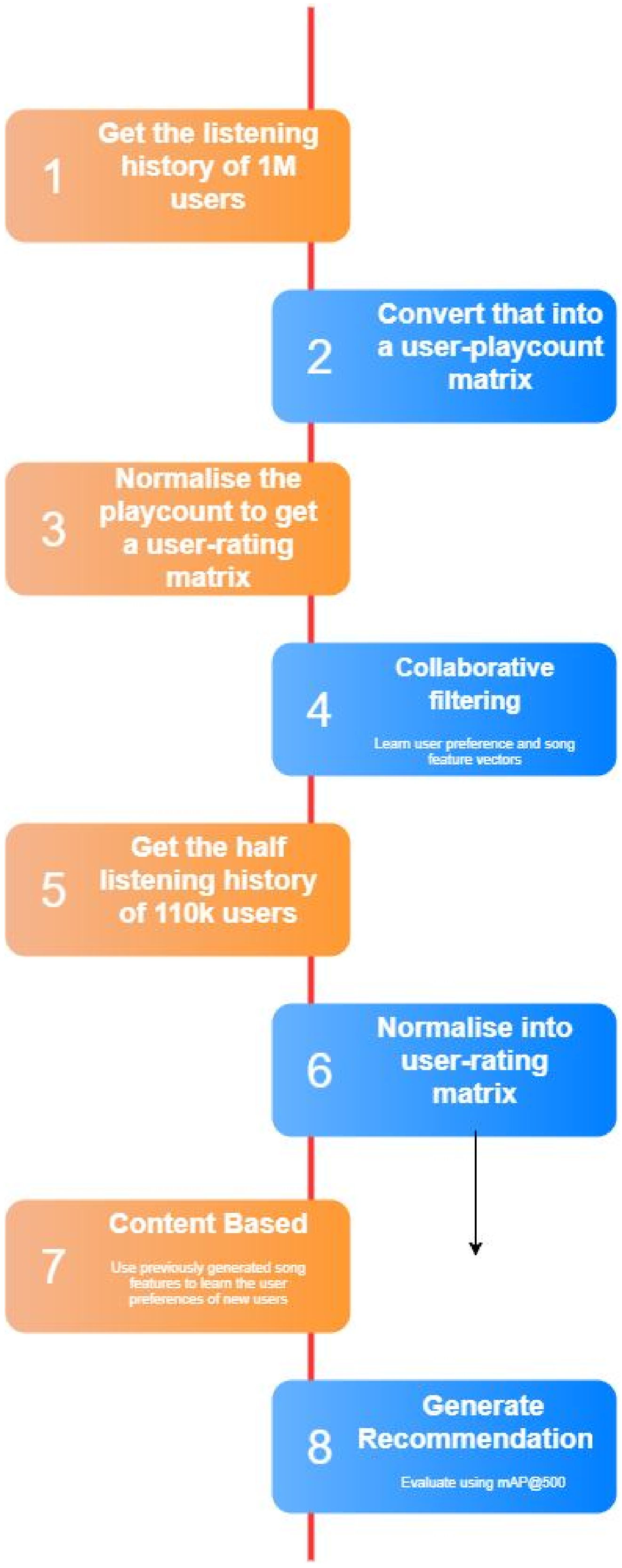
3. For a user with parameters θ and a movie with (learned) features x, predict a star rating of $\theta^T x$.

Above functions give a detailed idea of how one can use any of the approaches to solve the recommendations problem by using basic machine learning principles.

Our approach is to basically combine both the techniques to solve the MSDC problem.

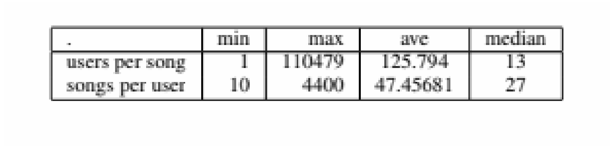
We first learn the feature vectors and preference vectors using collaborative filtering using the million song dataset. We later ignore the preference vectors learned from this dataset as this dataset does not comply with the users that are featured in the 110k dataset. We then run the collaborative filtering algorithm on the 110k dataset and get user preference vectors.

Now as both the vectors are available to us i.e. user preference vectors (CF on 110K) and feature vectors (CF on Million song dataset) we use the Content based recommendation approach to predict the final ratings of the users by merely multiplying them.


1
Get the listening history of 1M users
2
Convert that into a user-playcount matrix
3
Normalise the playcount to get a user-rating matrix
4
Collaborative filtering
Learn user preference and song feature vectors
5
Get the half listening history of 110k users
6
Normalise into user-rating matrix
7
Content Based
Use previously generated song features to learn the user preferences of new users
8
Generate Recommendation
Evaluate using mAP@500

V. IMPLEMENTATION

The way we implement our system is to first use the listening history of 1M users to obtain feature vectors for all the songs which are available in the dataset.



.	min	max	ave	median
users per song	1	110479	125.794	13
songs per user	10	4400	47.45681	27

Based on the statistics above we can make out that there are many people who have listened to far fever songs. We think that people with a small listening history may affect our algorithm in a negative way. Using all the users would unnecessarily increase the complexity of our model and also increase computation time exponentially. So to tackle this we filtered out those users who've listened to less than 250 songs, i.e. All the users now have at least listened to 250 songs individually. This process of filtering out not only makes the dataset a little smaller but also reduces the amount of sparsity in the user song matrix considerably.

The next step is now to normalize the play-count of each user into a fixed range. Doing this will improve the speed of the algorithm and in the end will enable the algorithm to learn more precise features.

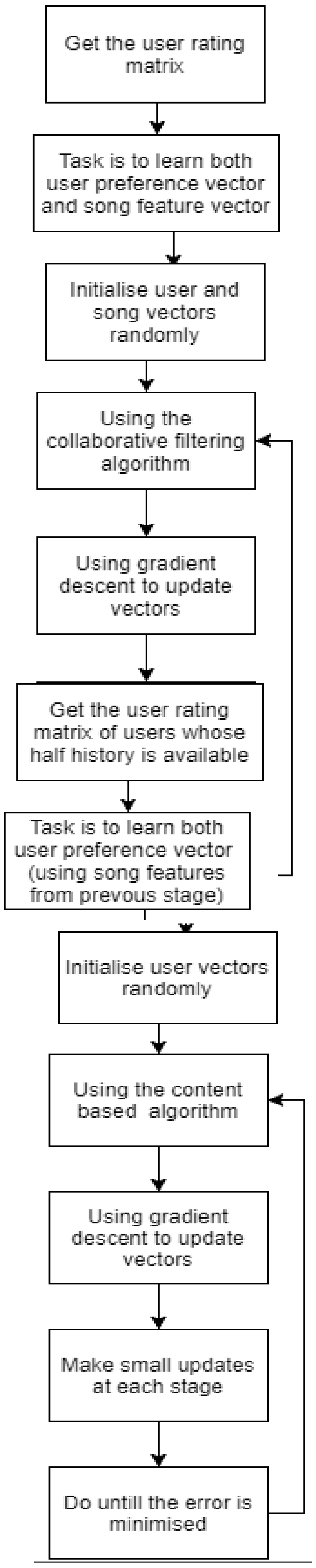
```
newvalue= (max'-min')/(max-min)*(value-max)+max'
or
newvalue= (max'-min')/(max-min)*(value-min)+min'.
```

Using the above formulae, we normalised the ratings within a range of 0-5.

Collaborative Filtering:

This collaborative algorithm learns both user preferences and song feature vectors. When we run on our dataset, we can obtain feature vectors of all the songs which are available.
We ran this algorithm twice once on listening history of 1M users and once on visible listening of 110k users. We combined the song vectors obtained from training on both the datasets. And we discarded user preference vectors from both.
These song vectors are now carried forward to the content based recommendation stage.

Content based Recommendations:


Get the user rating matrix
Task is to learn both user preference vector and song feature vector
Initialise user and song vectors randomly
Using the collaborative filtering algorithm
Using gradient descent to update vectors
Get the user rating matrix of users whose half history is available
Task is to learn both user preference vector (using song features from prevous stage)
Initialise user vectors randomly
Using the content based algorithm
Using gradient descent to update vectors
Make small updates at each stage
Do untill the error is minimised

Content based algorithm only gives us user preference vectors based on already obtained song feature vectors. We use this on 110k dataset to obtain user preference vectors of the 110k users.

Finally we have all the required user preferences and song vectors. To obtain a rating by a user for a particular song, we take a dot product of the vector of the user and the song.

In this way we obtain rating by each user(among 110k) to each unlistened song by them. Then sort all the ratings in descending order. After sorting we are able to recommend then top N songs to each user and evaluate our system based on those predictions.

VI. EXPERIMENTS AND RESULTS

To test our system we used the hidden history of 110k users against the songs predicted by our system.

We used 3 parameters for evaluation:

1. Precision

$$Precision = \frac{tp}{tp + fp}$$

Precision is number of songs correctly predicted by our system divided by the total number of songs recommended by our system.

2. Recall

$$Recall = \frac{tp}{tp + fn}$$

Recall is number of songs correctly predicted by our system divided by the total number of songs actually listened by the user.

3. F – measure

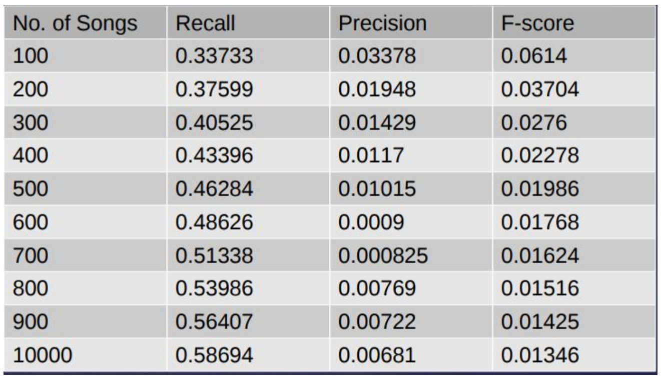
$$F = 2\frac{Precision * Recall}{Precision + Recall}$$

F-measure is nothing but the harmonic mean of precision and recall.

We calculated recall,precision and f-measure for each user and then averaged the results over 110k users.

We experimented by recommending varying the amount of songs our system recommended, we started by recommending 100 songs per user and went on till 1000 songs per user.

Below are results:



No. of Songs	Recall	Precision	F-score
100	0.33733	0.03378	0.0614
200	0.37599	0.01948	0.03704
300	0.40525	0.01429	0.0276
400	0.43396	0.0117	0.02278
500	0.46284	0.01015	0.01986
600	0.48626	0.0009	0.01768
700	0.51338	0.000825	0.01624
800	0.53986	0.00769	0.01516
900	0.56407	0.00722	0.01425
10000	0.58694	0.00681	0.01346

We have obtained the highest F-score of 0.0614 for 100 songs along with recall of 0.33 and precision of 0.033.

VII. CONCLUSION

We have obtained the highest F-score of 0.0614@100 songs. Its apparent that as the number of songs increase the recall increases as expected but the precision takes a hit ,which in turn affects the F-score as well.

Precision is so severely affected as the number of songs listened by each user is not same but the number of system recommended songs is same.

The recall of our system is fairly decent. 0.33 percent of recall means that 33% of the recommended songs are correct. The reason we feel that 33% match is good enough because a user only listens to some of the recommended items and not all(Ideally he would listen to all the songs).But in reality even if a system can recommend some relevant items ,the system is worthy of deploying in a real world scenario.

VIII. FUTURE WORK

Every Collaborative filtering solution suffers with the problem of cold start i.e. one cannot recommend to a user by knowing nothing about him/her. Most of the solutions offered to the cold start problem are context specific, however pertaining to this topic one could try clustering such users and recommending them subsets of each other's song if present or try recommending them the most popular songs. This could engage them to provide further information about their taste. Once sufficient history has been collected from that, one can use our approach for normal recommendations.

One could entirely change the approach by using similarity metrics to solve the same problem by either of the approaches. For content based recommendations one would have to collect information of the songs using the metadata of the songs and form user and item profiles. As item and user profiles are generated one could use any similarity metrics to find how much a single user leans towards that particular song profile. Collaborative filtering requires the same idea however using similarity metrics one could find similar users and then perform recommendation to each other according to the user's history.

One could even try clustering various users with similar taste and then applying CF algorithm, thus narrowing down the problem.

REFERENCES

[1] Gediminas Adomavicius and Alexander Tuzhilin, "Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and PossibleExtensions", IEEETKDE: IEEE Transactions on Knowledge and Data Engineering, 17, 2005.

[2] Prodan Andrei-Cristian, "Implementation of a Recommender System Using Collaborative Filtering", Studiauniv.Babes_bolyai, Informatica, volume lv,number 4, 2010.

[3] B. Sarwar, G. Karypis, J. Konstan, and J. Reidl, "Item-based collaborative filtering recommendation algorithms", In WWW '01: Proceedings of the 10thinternational conference on World Wide Web, New York, NY, USA, 2001.

[4] Michael J. Pazzani and Daniel Billsus, "Content Based Recommendation System."

[5] Zhou and T.Luo, "A Novel Approach to Solve the Sparsity Problem in Collaborative Filtering."

[6] Fabio Aiolli.: Preliminary Study on a recommender system for theMillion Songs Dataset challenge. (2011)

[7] http://www.kaggle.com/c/msdchallenge

[8] http://labrosa.ee.columbia.edu/millionsong/

[9] http://en.wikipedia.org/wiki/Collaborative_filtering