Music Recommendation System

Stony Brook University

Piyush Khemka	Sharang Bhat	Prachi Poddar	Snigdha Kamal
110828688	110920852	110815897	110937472

Abstract

With the tremendous increase in music consumption and a plethora of options available for music streaming such as Spotify, Pandora and others, music recommendation systems have been fast gaining importance. A music recommendation system recommends music to a user customized on their taste. The recommendations are based on the user's past listening history which creates a positive customer experience and at the same time increases a company's profit. In this project, we have developed a music recommendation system based on the Million Song Dataset using Content Based Recommendation and Item - user collaborative filtering and drawn a comparative analysis between them. In this paper, we present our methodology along with the results and analysis of our findings.

1 Introduction

There has been a huge rise in the number of online music streaming companies recently. Companies such as Spotify, Pandora, Gaana.com, 8tracks,iTunes and even the rest of the music industry have migrated from distributing music in the form of physical records to digital music for users on the internet. In this new model, value for the company is derived when the customer buys the song he/she is interested in, either by purchasing the song online or paying for a subscription to that company's music service. Both these methods bring revenue to the company by recommending customers new songs they might like, which improves user experience and in turn creates a higher likelihood of the customer buying a premium service with that music provider company. Therefore, there is a strong financial incentive to implement a robust song recommendation system.

Amongst the many issues that make song recommendation a difficult problem, the main one is the lack of similarity measure between songs- two similar sounding songs could turn out to be completely different with an equally distinct user base. Hence, we need a method to extract useful features that could be utilized to recommend other songs. This poses the problem of deciding which features would be good for learning, as the accuracy of the recommendation system can vary greatly depending on the features that are selected. The dataset used for our problem - the Million Song Dataset contains a million unique songs of 280 GB of data, posing a major challenge to process and utilize efficiently.

2 Related Works

The million song dataset challenge was a competition hosted on Kaggle. The specific details of the implementations and algorithms used by the competitors weren't revealed, hence we used the scoreboard rankings of the competition as a measure of comparison for our algorithm against others. The highest average precision achieved in the Kaggle competition was 17% and the average precision of the authors of the paper[1] was 14%. We were able to achieve a precision of 6.1 % with content based recommendation. It is to be noted that the Kaggle competition was run on a dataset of 80MB while our content based recommendation was tested on

[1] http://www-personal.umich.edu/ yjli/content/projectreport.pdf

the official subset of the Million song dataset which was roughly 2.8 GB in size. There have also been other related works showcasing different algorithms to generate song recommendations, as detailed below.

There are a number of different ways to measure the similarities between songs, namely semantic embedding model(Law et al., 2010) and acoustic similarity model(Bertin-Mahieux et al., 2010). However, music similarity measures are subjective and vary from person to person, which makes it tough to rely on ground truth. This issue is addressed in (Berenzweig et al., 2004) and (Ellis et al., 2002). In our project work, we have used the song history of the user to generate better results.

There have been various algorithms that have been used to implement recommendation systems. One method called Latent factor mode Bayesian personalized ranking (BPR) is very similar to the SVD approach as the matrix factorization is the same.

$$M = \overline{M} = U^T V \tag{1}$$

However, this algorithm involves elements of machine learning. The algorithm learns the U and V values such that for every user, the positive(songs listened) items are ranked higher than the negative ones(no-feedback items). This learning is accomplished by stochastic gradient descent.This is a unique approach, transforming a recommendation problem into an optimization problem.

3 Dataset

The data for this project is the Million Song Data Set (MSDS), released by Columbia University's Laboratory for the Recognition and Organization of Speech and Audio. It is a freely available collection of audio features and metadata for a million popular music tracks(280 GB) gathered from 110k users, and consists of audio features for over 1 million popular and contemporary songs, varying in genre, time period, and country of origin. We used a subset of the Million Song dataset which consists of 10,000 songs totaling 2.8 GB of data in zipped format. It contained individual files in h5 format, which when converted to a csv format totaled to 10 GB of data (hdf,). Supplementary data provided by other groups and matched to songs in the MSDS was also used. This includes the User Taste Profile Dataset containing over 48 million triplets(User Id, Song ID, count) gathered from over 1 million users and the user-defined tags extracted from the Last.fm dataset.

4 Methodology

The workflow of the algorithm is shown in Figure 1

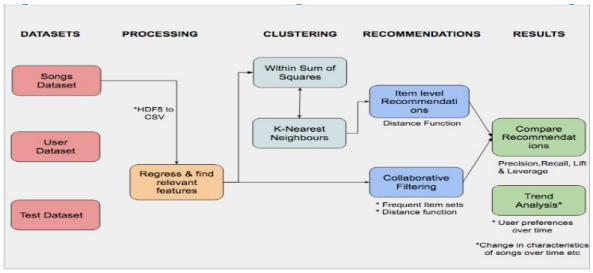

Figure 1: Algorithm Pipeline

4.1 Collaborative Filtering

Collaborative filtering is a common measure used in recommendation systems. This technique is used to fill the holes or missing entries in a user-item association matrix. Usually, a matrix of this form is used to fill missing ratings. However, in our dataset we had count i.e. number of times the user had heard a particular song instead of ratings. We have used count as a proxy for ratings and built our association matrix on user-song counts. We employed Spark's MLLib library to perform collaborative filtering where users and songs are characterized by a set of latent factors. These factors can then be used to predict missing values in the matrix. We have implemented the Alternating Least Squares(ALS) algorithm to build the matrix and figure out the latent factors.(lat,)

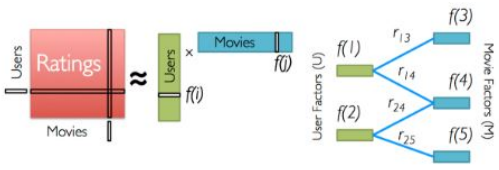

Figure 2: Low Rank Matrix Factorization

$$f[i] = argmin_{w \epsilon R^d} \sum_{j \epsilon Nbrs(i)} (r_{ij} - w^T f[j])^2 + \lambda \|w\|_2^2 \tag{2}$$

4.1.1 Training

Generally every recommendation system, while using collaborative filtering uses a distance function as a similarity measure. However, we used a different approach wherein latent factor methods were used to train on some known data, extracting K User features, or their latent features, and by multiplying K user features with respective item or latent features.

The matrix factorization is a latent factor model method representing items and users as vectors derived from a rating pattern. The rating pattern in this case was the counts of the songs. In other words, the latent factor model maps K users and items to a set of K features such that user-item ratings are modeled as inner vector products. We assigned different values of K, in each iteration of training. We assigned the value of K at sets of intervals, such as $\{4, 8, 12\}$. We simulated the training and validation in each iteration and computed the Root Mean Squared Error or the RMSE of the ratings produced. In the first case, we divided the training data into three parts - 60% used for training, 20% for testing and 20% for validation. In the training iterations, the model was trained over different latent factors, and at each stage, the RMSE was computed and compared to the next model's RMSE. The rank, or the value or K for the least RMSE was selected as the best rank, and retained for the testing phase. The baseline RMSE was 9.38294018346.

4.1.2 Testing

For the testing phase, we used four files, tested the model and generated their RMSE values, finally comparing them with the baseline RMSE as documented in Table1. The average RMSE obtained was 8.5954855 and the difference from baseline was 0.787454.

File Name	Entries	RMSE	Diff. from Baseline
Test1(80MB)	200,000	8.954	0.429
Test_Validation(80 MB)	200,000	9.536	0.153
Test2(60MB)	500,000	7.436	1.947
Test2_Validation(60MB)	500,000	8.456	0.927
Average		**8.595**	**0.787**

Table 1: Collaborative Filtering Results

4.2 Content Based Filtering

The content based recommendation required a different approach, as the this is not natively supported in spark or in python. We extracted certain features from the dataset, which describes features of a song. The features were namely - $\{title, song\_id, releaseartist\_id, artist\_name, duration, artist\_familiarity, artist\_hotness, year, end\_of\_fade\_in, energykey, key\_confidence, loudnessmode, mode\_confidence, song\_hotness, start\_of\_fade\_out, tempo, time\_signature, time\_signature\_confidence, artist\_latitude, artist\_location, artist\_longitude\}$. We normalized certain features by taking its product with its confidence to get a final value, such as mode and mode confidence to get a final mode estimate. We then further clustered similar songs based on these features, and attached each song to its respective cluster in a higher dimensional space. Further, we used Euclidean distance metric to find the most similar song.

4.2.1 Training

For the training part, the user-triples dataset was split in a ratio of 80:20. 80% of the dataset was used to train a clustering model based on the features of the songs. Each feature of the song corresponded to a point in a higher dimensional space and they were clustered together using k-means clustering.

4.2.2 Testing

For our testing dataset, we first created a profile for each user by merging the user-song dataset with the songs dataset which consisted of all the song features. Then we took the mean of all song features found for each user. Therefore each user profile consisted of a list of song features which was the mean of all the songs heard by the user in his lifetime and represents the kind of songs that the user like to listen to. Using this feature vector of each user, we found 10 nearest neighbors in the cluster generated by the training data and found 10 nearest neighbors as recommendations for the user. To evaluate the results we compared our recommendations with the actual values present in the testing dataset and calculated precision.

5 Results

5.1 Recommendation results

5.1.1 Collaborative Filtering

The sample recommendations for a given user-ID using Collaborative Filtering are shown in Figure 3 Table 1 shows the RMSE values derived for four files, along with their respective differences from the baseline RMSE values. The average RMSE obtained was 8.595 and its difference from baseline was 0.787.

	Artists	Songs
0	Genesis	Invisible Touch
1	Gemma Hayes	Back Of My Hand
2	Dropkick Murphys	The Wild Rover
3	Bryan Adams / Sting	All For Love
4	Marc Almond & Gene Pitney	Something's Gotten Hold of my Heart

Figure 3: Top 5 recommendations for User ID: adb7806be460224c4fbd0ee5c8409e365f0d6723 using Collaborative Filtering

5.1.2 Content Based Filtering

The average precision value achieved for content-based filtering was 6.1% We compared our precision with the precision of the winner of the Kaggle Challenge for Million Song Dataset. The highest precision achieved in the competition was 17% and the average precision of the owners of the dataset was was 14%. Given that the Kaggle Challenge was run on 80MB of data while our model was tested on a subset of 2.8GB of data, we find a result of 6.1% fairly satisfactory. Figure 4 shows a sample recommendations for a specific user-ID using Content-based filtering.

5.2 Visualizations and Trend Analysis

We used the subset of the Million Song dataset to perform visualizations and trend analysis as shown in the accompanying figures.

- Location of various artists visualized on the World Map as shown in Figure 5
- Top 10 songs in the Year 1990 as shown in Figure 6
- Top 10 Metal Songs in the Year 2000 shown in Figure 7

	000ebc858861aca26bac9b49f650ed424cf882fc
0	Genio Atrapado
1	Did We Not Choose Each Other
2	So So So
3	Life Deprived
4	Warhead (Live in Croatia_ 1993)
5	Baltech's Lament
6	Saturday
7	Take Your Leave Of Me Baby
8	Man I Used To Be
9	The west's awake

Figure 4: Sample Recommendations from Content-based filtering

- Number of songs released over the decades in Figure 8
- Trend of song tempo over the decades shown in Figure 9
- Trend of song confidence over the decades in Figure 10
- Top genre distributions of the dataset shown in Figure 11

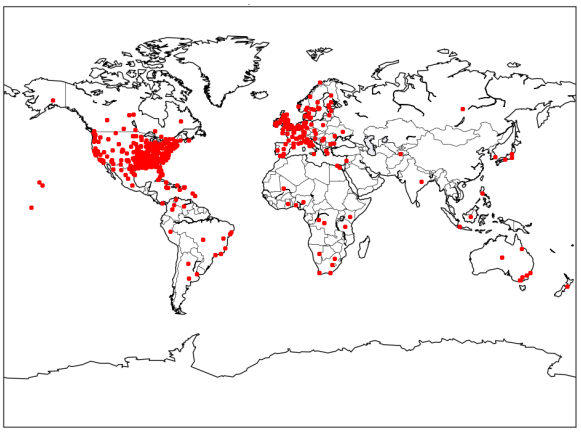

Figure 5: Location of artists on world map

6 Discussion

In this project, we have used several methods to extract similarities between users and songs. Our system yielded a precision of 6.1%, for Content-based

	Top 10 Songs in the Year 1990
0	Entre Caníbales
1	Indian Love Call
2	Commercial Reign
3	Biding Her Time
4	Entre Caníbales
5	Indian Love Call
6	Commercial Reign
7	Biding Her Time
8	I Love You More Than You'll Ever Know (LP Vers...
9	Love Takes Time

Figure 6: Top 10 songs in the Year 1990

	Top 10 Metal Songs in the Year 2000
0	Burning In The Aftermath
1	Wicker Chair
2	Inolvidable
3	Amber Changing
4	English Summer Rain
5	Burning In The Aftermath
6	Wicker Chair
7	Inolvidable
8	Amber Changing
9	English Summer Rain

Figure 7: Top 10 Metal songs in the Year 2000

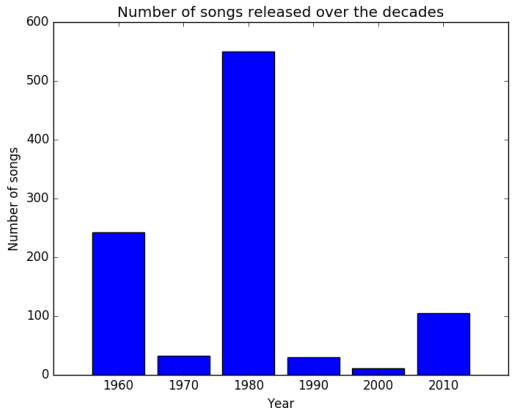

Figure 8: Number of songs released over the decades

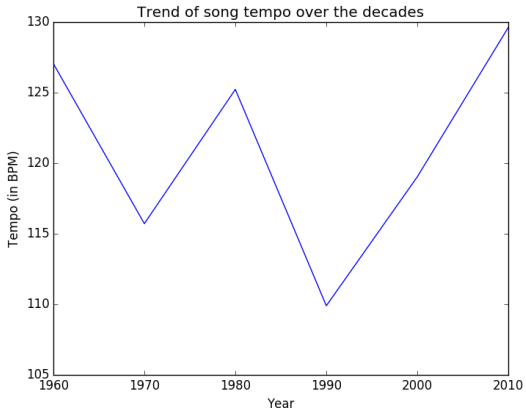

Figure 9: Trend of song tempo over the decades

filtering and a RMSE of 8.595 with a 0.787 difference from baseline for Collaborative filtering, emphasizing the fact that song recommendation problem is tough and can be attributed to the lack of similarity or correlation between a user's listening history and their song preferences in the dataset. The highest average precision on the award winning Kaggle algorithm was 17%, however their dataset was much smaller than our dataset.

7 Conclusion

This project work has provided us with a chance to understand how recommendation systems work in large companies by getting a hands-on experience implementing our own system. We realized that song recommendation is a tough problem which yields low precision values both in our work and all the other related works we studied. We implemented and evaluated song recommendation systems using Content Based Filtering and Collaborative Filtering. The collaborative filtering method used the latent factor model as opposed to the conventional usage of a distance function. From what we understood, the collaborative filtering method proves to be easier to implement and evaluate, as it runs primarily on a user's previous choices, as opposed to the inherent features within a song,which may or may not represent the users prediction towards the song or the genre. We also analyzed trends across the years, varying from popularity to loudness.

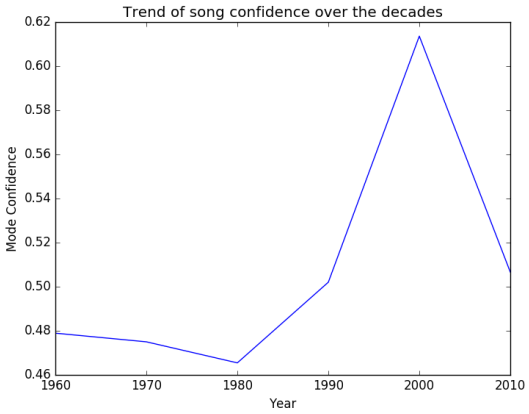

Figure 10: Trend of song confidence over the decades

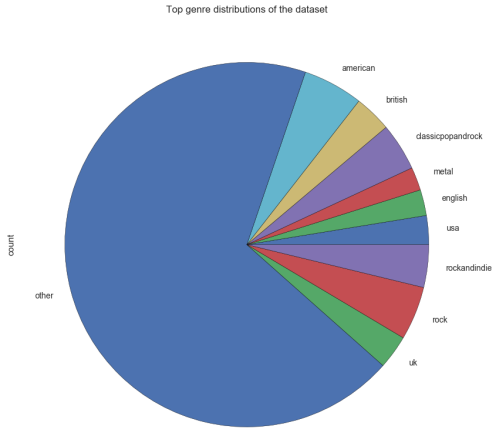

Figure 11: Top genre distributions of the dataset

References

[Berenzweig et al.2004] Adam Berenzweig, Beth Logan, Daniel PW Ellis, and Brian Whitman. 2004. A large-scale evaluation of acoustic and subjective music-similarity measures. *Computer Music Journal*, 28(2):63–76.

[Bertin-Mahieux et al.2010] Thierry Bertin-Mahieux, Douglas Eck, and Michael Mandel. 2010. Automatic tagging of audio: The state-of-the-art.

[Ellis et al.2002] Daniel PW Ellis, Brian Whitman, Adam Berenzweig, and Steve Lawrence. 2002. The quest for ground truth in musical artist similarity.

[hdf] Data conversion. `https://github.com/tbertinmahieux/MSongsDB/blob/master/PythonSrc/hdf5_getters.py`.

[lat] Latent factor model. `https://databricks-training.s3.amazonaws.com/img/matrix_factorization.png`.

[Law et al.2010] Edith Law, Burr Settles, and Tom Mitchell. 2010. Learning to tag from open vocabulary labels. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 211–226. Springer.