Song Recommender System

Ashwini Devendra Prabhu

Introduction

The amount of data content available in today's world exceeds the capacity of single individual to explore it. This data content could be related to anything such as movies, songs, items that we purchase on websites such as amazon etc. Hence, service providers need an efficient way to manage these contents and help their costumers to discover items by giving quality recommendations. Recommender systems help in filtering millions of contents and make personalized recommendations to the users.

Recommender systems rely on different types of input. High quality explicit feedback is the most convenient, which includes explicit input by users regarding their interest in items such as ratings. However, explicit feedback is not always available. Thus, recommenders can infer user preferences from the more abundant implicit feedback, which indirectly reflect opinion through observing user behavior. This includes purchase history based on certain factors like favorite author.

In this project, I have implemented and analyzed a song recommendation system. I have used Million Song Dataset provided by Kaggle to find correlations between users and songs and to learn from the previous listening history of users to provide song recommendations which users would prefer to listen. I have explored the collaborative filtering and KNN algorithm approach in this project.

Dataset

The dataset used for this project is the Million Song subset. The dataset is open; meta-data, audio content analysis, etc. available for all the songs. It is also very large and contains histories of over one million users and metadata (280 GB) of millions of songs. I have used the validation dataset as my main dataset [2]. It consists of 10,000,000 triplets of 10000 users. The data is stored using HDF5 format to handle the heterogeneous types of information such as audio features in variable array lengths, names as strings, similar artists, etc. Each song detail is described by a single file.

I used metadata of only 10,000 songs which is around 3GB. One of the main files of the dataset is the kaggle_visible_evaluation_triplets.txt [1] which consists of the ['user-id', 'song-id', 'play-count']. The feedback is implicit as the song play-count is given instead of explicit ratings.

Methods and Algorithms

Recommender systems can be broadly divided into two categories: content based, and collaborative filtering based.

Collaborative Filtering

Collaborative filtering systems work by collecting user feedback in the form of ratings for items in a given domain and exploiting similarities in rating behavior among several users. The ratings could be explicit (score such as a rating e.g.: movie ratings on a scale of 5 or 10) or implicit (such as number of views or purchases, number of times a song is heard). There are two types of collaborative filtering: Memory-based algorithm and Model based algorithm.

Memory-based algorithms recommend items based on similarity between users or items (songs). User-based model make prediction for a certain user based on its similar users. The intuition is that if user A has similar taste to another user B then A will be likely to enjoy the songs that B listened. Alternatively, item-based model make recommendation by finding similar songs to the songs in certain user's listening history. Here I have implemented memory-based algorithm. The method I have used is alternating least square error (ALS) which is a part of Spark library for collaborative filtering.

Alternating Least Square (ALS)

ALS is an iterative optimization method where we arrive closer to a factorized representation of our data in each iteration. In this method we can identify the relation between the user and the items by applying matrix factorization. The ALS method accepts a large user-item matrix and figures out the hidden features that relates them to each other in a much smaller user-item feature matrix.

If our original matrix ***R*** is of size *u x i* (u as users, i as items and x as some type of feedback data). We should then find a way to turn this into a matrix with users and hidden features (size *u x f)* and a matrix with items and hidden features (size *f x i).* In ***U*** and ***V,*** we have weights for how each user - item relates to each feature.

We can then calculate ***U*** and ***V*** such that: ***R ≈ U x V***.

We can find out what weights yield the best approximation of **R** by randomly assigning the **U** and **V** values and using least square. With the alternating least squares approach we use the same idea but iteratively alternate between optimizing **U** and fixing **V** and vice versa. We do this for each iteration to arrive closer value to **R = U x V.**

For implementing collaborative filtering only Kaggle triplets file was sufficient. This file consists of ['user_id','song_ids','play_count']. The ALS algorithm accepts inputs only in the form of numbers. Hence the user_id and song_id which were in the string format was converted to integer format. The data was then split into training and testing data. Once this was fed to the ALS algorithm, the top 10 recommendations for all the users was displayed. We could also query the recommendations based on the user_id for a particular person.

KNN Algorithm

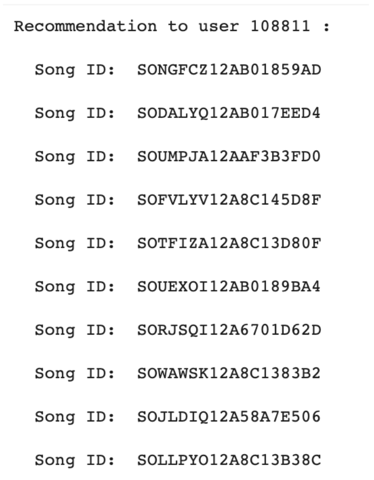
K-nearest neighbor finds the k most similar items to a particular instance based on a given distance metric. KNN does not make any assumptions on underlying data distributions. It relies on item feature similarity. When KNN makes a song prediction, it will calculate the distance between the target song and every other song in the dataset. It then ranks its distances and returns the top k nearest neighbor songs as the most similar song recommendations.

I have implemented both user based KNN and item based KNN. The distance is calculated using the cosine metric. Initially the data is loaded from both the files. The Kaggle triplet file consists of the [‘user_id’, ‘song_id’, ‘play_count’]. The metadata of the song consists of many columns out of which I chose [‘song_ids’, ‘song_title’, ‘artist_name’]. These columns were combined and divided into training and testing data. I then checked if there were any missing values and filled them with zeroes. This data was then made fit to be fed to the KNN model by converting into a sparse matrix. For User-based KNN model, the sparse matrix consists of song_ids as the columns and the user_id as the rows. For Item-based collaborative, the sparse matrix consists of user_id as columns and song_ids as rows. Then the recommendations were provided based on either the song_ids or the user_id provided.

Conclusions and Results

The million-song dataset is very sparse. From the implementation, I felt the recommendations were better for a subset of data rather than the whole dataset. The song play_count varies highly. The play_count for most of the songs were 1 whereas some songs had the play_count as high as 900. Even in the subset of data, the play_count value was high as 300. I got a high RMSE value of around 7 which I think is due to the imbalance of song play count values.

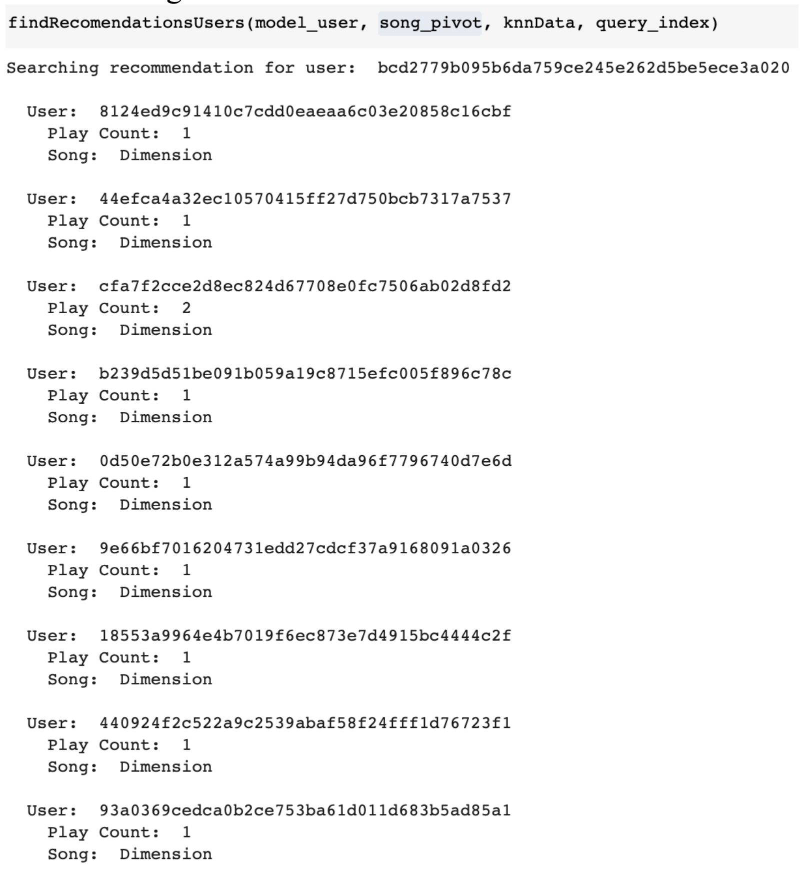
Collaborative filtering using ALS: The below image shows the song recommendations for a user based on user_id.



```
Recommendation to user 108811 :

  Song ID:  SONGFCZ12AB01859AD

  Song ID:  SODALYQ12AB017EED4

  Song ID:  SOUMPJA12AAF3B3FD0

  Song ID:  SOFVLYV12A8C145D8F

  Song ID:  SOTFIZA12A8C13D80F

  Song ID:  SOUEXOI12AB0189BA4

  Song ID:  SORJSQI12A6701D62D

  Song ID:  SOWAWSK12A8C1383B2

  Song ID:  SOJLDIQ12A58A7E506

  Song ID:  SOLLPYO12A8C13B38C
```

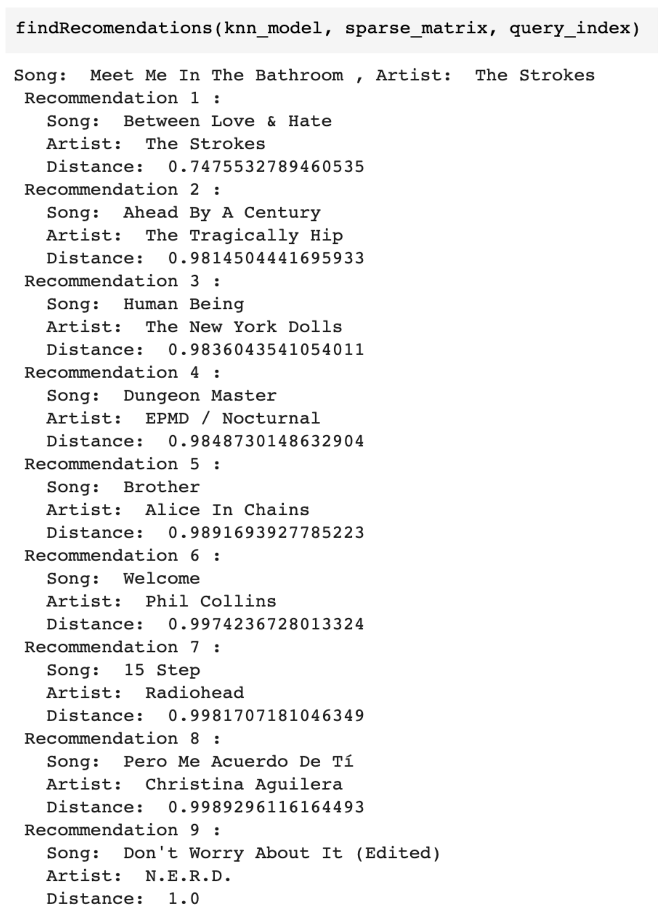
Fig:1

KNN: The fig: 2 image shows the recommendations for different users and fig.3 shows the recommendation based on song.



```
findRecomendationsUsers(model_user, song_pivot, knnData, query_index)

Searching recommendation for user:  bcd2779b095b6da759ce245e262d5be5ece3a020

  User:  8124ed9c91410c7cdd0eaeaa6c03e20858c16cbf
    Play Count:  1
    Song:  Dimension

  User:  44efca4a32ec10570415ff27d750bcb7317a7537
    Play Count:  1
    Song:  Dimension

  User:  cfa7f2cce2d8ec824d67708e0fc7506ab02d8fd2
    Play Count:  2
    Song:  Dimension

  User:  b239d5d51be091b059a19c8715efc005f896c78c
    Play Count:  1
    Song:  Dimension

  User:  0d50e72b0e312a574a99b94da96f7796740d7e6d
    Play Count:  1
    Song:  Dimension

  User:  9e66bf7016204731edd27cdcf37a9168091a0326
    Play Count:  1
    Song:  Dimension

  User:  18553a9964e4b7019f6ec873e7d4915bc4444c2f
    Play Count:  1
    Song:  Dimension

  User:  440924f2c522a9c2539abaf58f24fff1d76723f1
    Play Count:  1
    Song:  Dimension

  User:  93a0369cedca0b2ce753ba61d011d683b5ad85a1
    Play Count:  1
    Song:  Dimension
```

Fig 2:



```
findRecomendations(knn_model, sparse_matrix, query_index)
```

```
Song:  Meet Me In The Bathroom , Artist:  The Strokes
 Recommendation 1 :
   Song:  Between Love & Hate
   Artist:  The Strokes
   Distance:  0.7475532789460535
 Recommendation 2 :
   Song:  Ahead By A Century
   Artist:  The Tragically Hip
   Distance:  0.9814504441695933
 Recommendation 3 :
   Song:  Human Being
   Artist:  The New York Dolls
   Distance:  0.9836043541054011
 Recommendation 4 :
   Song:  Dungeon Master
   Artist:  EPMD / Nocturnal
   Distance:  0.9848730148632904
 Recommendation 5 :
   Song:  Brother
   Artist:  Alice In Chains
   Distance:  0.9891693927785223
 Recommendation 6 :
   Song:  Welcome
   Artist:  Phil Collins
   Distance:  0.9974236728013324
 Recommendation 7 :
   Song:  15 Step
   Artist:  Radiohead
   Distance:  0.9981707181046349
 Recommendation 8 :
   Song:  Pero Me Acuerdo De Tí
   Artist:  Christina Aguilera
   Distance:  0.9989296116164493
 Recommendation 9 :
   Song:  Don't Worry About It (Edited)
   Artist:  N.E.R.D.
   Distance:  1.0
```

Fig 3:

Reference

[1] http://www.kaggle.com/c/msdchallenge
[2] http://labrosa.ee.columbia.edu/millionsong/
[3] T. Bertin-Mahieux, D.P.W. Ellis, B. Whitman, and P. Lamere. The million song dataset. In ISMIR 2011: Proceedings of the 12th International Society for Music Information Retrieval Conference, October 24-28, 2011, Miami, Florida, pages 591–596. University of Miami, 2011.