Song recommender system based on collaborative filtering using Neo4j knowledge graphs

Gattu Ramanadham
School of Informatics and Computing
Indiana University
Bloomington, United States
ramgattu@iu.edu

Jainendra Kumar
School of Informatics and Computing
Indiana University
Bloomington, United States
jaikumar@iu.edu

Ishneet Singh Arora
School of Informatics and Computing
Indiana University
Bloomington, United States
iarora@iu.edu

Abstract—**With the growing pervasive computing age, exponential increase in the textual and multimedia data presents new challenges to data analysis involving challenges in storing, cleaning and querying the data. Past research has shown that graph database is an efficient alternative to relational database management systems for processing complex data. Graph databases also play an important part in the recommender systems which are very important to the information and e-commerce ecosystem. They play a very important part in the development of an efficient and very powerful system which enable users to filter through huge datasets and give useful insights. Recommender systems can be built in different ways based on different choices of algorithms, design methods, and their ability to cater to a specific business problem. In this paper, we show how a song recommender system can be built using nearest-neighbor collaborative filtering techniques on top of Python programming language and Neo4j, a graph database platform. Additionally, we give our opinion on how current offerings of Graph databases have improved the performance of the recommender systems.**

Keywords—Graph Database, Recommender System, graph data storage model, collaborative filtering, k nearest-neighbors, Python, Exploratory Data Analysis.

I. Introduction

A recommender system or a recommendation system is a system that performs information filtering with an aim to predict the preference of a user towards a certain item. These are primarily very popular in e-commerce industry and are widely used by services such as Netflix, YouTube and Spotify. Similarly, product recommender systems are used in most of the e-commerce websites such as Amazon, E-Bay and content recommendation systems in Facebook, Instagram and Twitter [1]. They have also been used where they predict the research articles to a researcher based on his requirements and experiments [2] and are also been used in financial services industry such as insurance [3].

Like many machine learning techniques, a recommender system works based on how a user performs in the past. In other words, a recommender system would look, analyze and query the past actions of the user with an assumption that the users who have agreed in the past will also agree in the future. They usually make use of either or both Content based filtering or Collaborative Filtering. Collaborative filtering dwells more on user's past behavior as well as similar actions taken by other users [4]. Content based approach on the other hand requires sufficient amount of pre-tagged information or metadata about the user which can be extracted or created by other techniques such as Natural Language Processing [5]. For example, a metadata about a song would be genre, year, artist, record label, language, etc. which can be extracted to make predictions.

II. Related Work

In the work of Barrington, Luke, et.al "Smarter than Genius? Human Evolution of Music Recommender Systems compared Genius, a popular commercial music recommendation engine to two other canonically similar music recommender systems: first based on artist similarity and other based on acoustic content. They benchmarked the performance with a user study of 185 subjects and found that Genius overall produces the best recommendations.

In the work of Yang, Xiwang, et.al "A survey of collaborative filtering based social recommender systems" gave a brief overview of recommender systems and other traditional approaches that do not use social network information. They also demonstrated how accuracy of recommender system can be increased by using social network information.

Lee, Haesung and Joonhee, Kwon in "Efficient Recommender System based on Graph Data for Multimedia Application" proposed a new graph data storage model for the collaborative filtering-based recommendation system.

III. Dataset

For our project we the Million Song Dataset which is a "freely-available collection of audio features and metadata for a million contemporary popular music tracks" [6]. The aim of this data as stated on their official website is to encourage and promote research on the scalability of algorithms. To provide a reference dataset for evaluating experimental research results and as a way to create large dataset with APIs.
The dataset does not include any audio but includes the derived features and metadata for one million songs. The entire dataset is of 280 GB but for our recommendation system we have taken a subset of the data with two files:

- The first file contains user_id, song_id and listen time. This file measures the number of listens for a particular song.
- The second file is the metadata file that contains song_id, title, release_by, artist_name and year. (1 million observations)

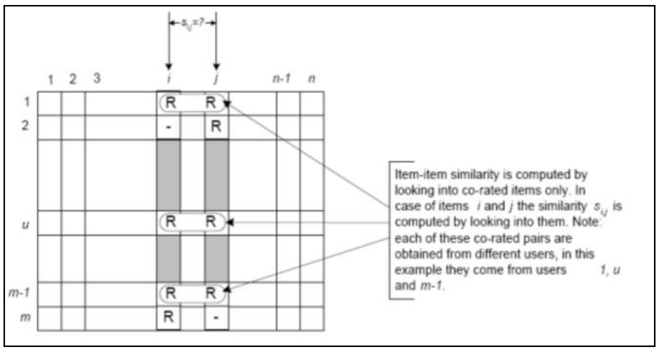
The dataset contains data about millions of songs which are maintained by multiple websites such Last.fm, thisismyjam, musixmatch, etc.

IV. METHODOLOGY

A. *Collaborative Filtering*

In our project we have used nearest neighbor Collaborative Filtering technique. In a Collaborative Filtering based recommendation system taste or preference of a user is predicted based on a collaboration of users which also have a similar taste or preference to the current user. In other words, collaborative technique means if a person X has the same preference towards a product as person Y then X is more likely to have the same preference as Y on some other product as well. Therefore, the rating/taste/preference of multiple users are collaborated and correlated together in order to make a prediction of a certain user on the taste and preferences of other users.

As shown in the figure 1 below similarity measure between two items i and j is calculated by observing the ratings of all the users who have rated an item i and item j.



Item-item similarity is computed by looking into co-rated items only. In case of items *i* and *j* the similarity s_{ij} is computed by looking into them. Note: each of these co-rated pairs are obtained from different users, in this example they come from users *1*, *u* and *m-1*.

Figure 1.
http://www.cs.carleton.edu/cs_comps/0607/recommend/recommender/itembased.html

To calculate the similarity between two items there are number of mathematical formulae such as:
Cosine-based similarity or vector-based similarity is calculated by viewing two items and their ratings as vectors and similarity is the angle between those vectors [7].

$$sim(i,j) = \cos(\vec{i},\vec{j}) = \frac{\vec{i}\cdot\vec{j}}{\|\vec{i}\|_2 * \|\vec{j}\|_2}$$

Pearson (correlation) based similarity on the other hand is based on the ratings by common users for two items i and j [7].

$$sim(i,j) = \frac{\sum_{u\in U}(R_{u,i}-\bar{R}_i)(R_{u,j}-\bar{R}_j)}{\sqrt{\sum_{u\in U}(R_{u,i}-\bar{R}_i)^2}\sqrt{\sum_{u\in U}(R_{u,j}-\bar{R}_j)^2}}$$

Adjusted cosine similarity is developed to remove the limitations of vector based similarity where we subract the average ratings of each user from each user's rating for item i and j [7].

$$sim(i,j) = \frac{\sum_{u\in U}(R_{u,i}-\bar{R}_u)(R_{u,j}-\bar{R}_u)}{\sqrt{\sum_{u\in U}(R_{u,i}-\bar{R}_u)^2}\sqrt{\sum_{u\in U}(R_{u,j}-\bar{R}_u)^2}}$$

Once the model has been constructed by any of the similarity measures the prediction formula is given by:

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

The prediction formula is based on the idea of weighted sum [7]. We take items which the user has rated and are similar to our target item and then we find the similarity index between them. Then we sum the similarities to find the actual prediction. In our project we have used cosine similarity between user ratings as the similarity.

B. *Graph Database and k-nearest neighbors*

K-nearest neighbors is perhaps the most basic algorithm in the data-mining field. Similarity is calculated by one of the similarity metric that we have defined before. In this project we have used cosine similarity to measure the similarity index.

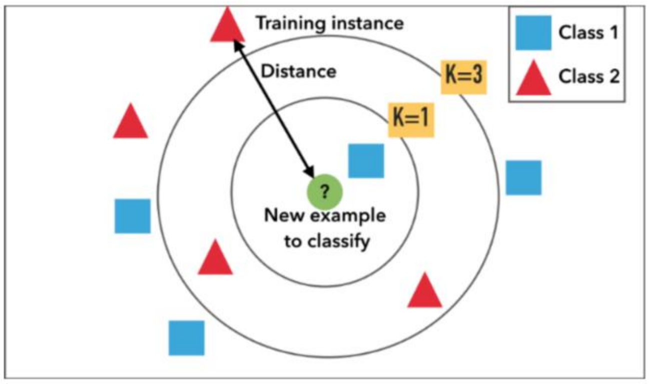


Figure 2.
https://blog.usejournal.com/a-quick-introduction-to-k-nearest-neighbors-algorithm-62214cea29c7

We have used Neo4j graph database for collaborative filtering using k-nearest neighbors.

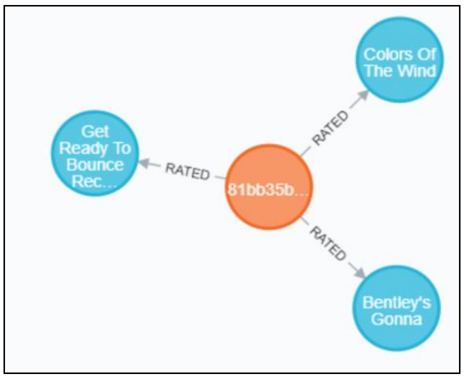


Figure 3.

Graph is a very effective tool in modeling the relationships between two different entities. As we can see from the figure 3 taken from our actual project graph user with user_id “81bb35” has rated three songs represented by the nodes in blue color.

V. APPROACH

A. *Read the data*

Our goal is to create a song recommender system based on nearest neighbors collaborative filtering. In the first step we read the two data files in our Python program.

B. *Understand the features*

In the first step we read the data through our Python program and then try to analyze each and every attribute of the data.

C. *Exploratory Data Analysis*

Our data set is huge so to understand the data more clearly we performed some data analysis using Python's matplotlib library. For the data analysis we first stored the data into python data frames and then perform calculations to calculate the total listen count of each song. In other words, we calculate how much times a song with a song ID has been listened.

Figure 4 below shows the song distribution with their listen count. As we can see from the graph some of the songs are so popular that they have been listened more than 50K times while the average listen count is in the range of 8 to 10K.

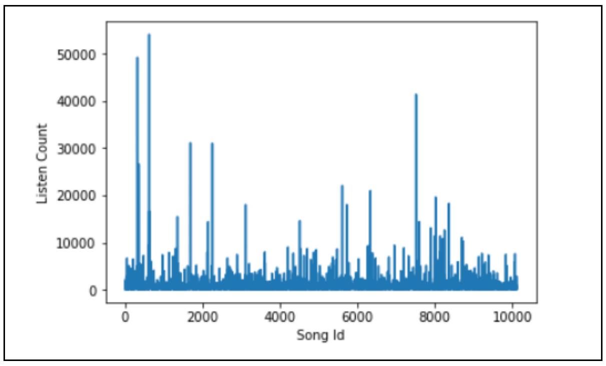


Figure 4.

In the figure 5 we can see the distribution between total number of unique songs people listen to and the frequency of overall total listens. There we see that listeners listen to more unique songs than they listen to old songs or the songs they have already listened.

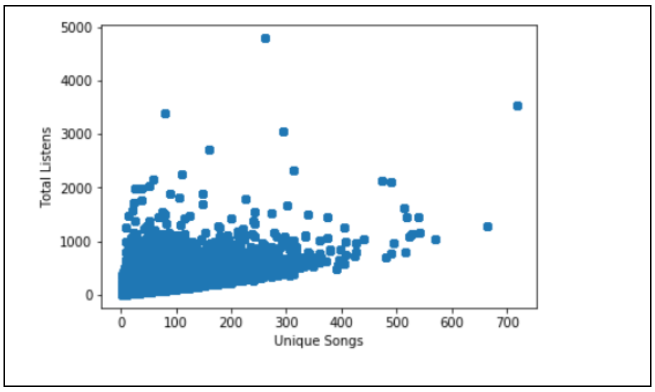


Figure 5.

D. *Clean the data*

Here in this step we clean the data by removing duplicate song ids. We further store the data in the Python Data frames. We then join the data frame by user ID and then summarize the data by song.

E. *Model the data*

In this step we filter the data frame, calculate the total listen count of each user, filter out users based on song ratings and display users with different sets of ratings.

F. *Data Sampling*

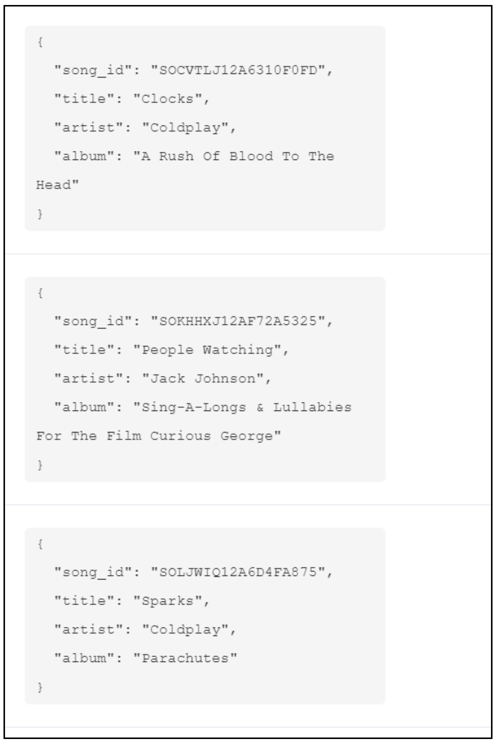
We prepare the data for modeling by taking a random sample from the data, once the data is cleaned and stored in structured way.

G. *Setting up Neo4j Server*

Once our data has been cleaned, structured and modeled we load the data into Neo4j Server after connecting the neo4j server with our Python program.

H. *Generate the graph, map nodes and relationships*

In this step we create nodes for different users, nodes for unique songs and define the relationship between the users and songs through the rating relationship. i.e. an edge connecting the two nodes.



```
{
  "song_id": "SOCVTLJ12A6310F0FD",
  "title": "Clocks",
  "artist": "Coldplay",
  "album": "A Rush Of Blood To The
Head"
}
```

```
{
  "song_id": "SOKHHXJ12AF72A5325",
  "title": "People Watching",
  "artist": "Jack Johnson",
  "album": "Sing-A-Longs & Lullabies
For The Film Curious George"
}
```

```
{
  "song_id": "SOLJWIQ12A6D4FA875",
  "title": "Sparks",
  "artist": "Coldplay",
  "album": "Parachutes"
}
```

Figure 6.

Figure 6 shows the sample data loaded into the Neo4j server from the Python program. Next step would be to perform collaborative filtering using cosine similarity.

I. Perform collaborative filtering using cosine distance

We run the Neo4j query to calculate the cosine distance between user ratings which will denote the similarity between multiple users.

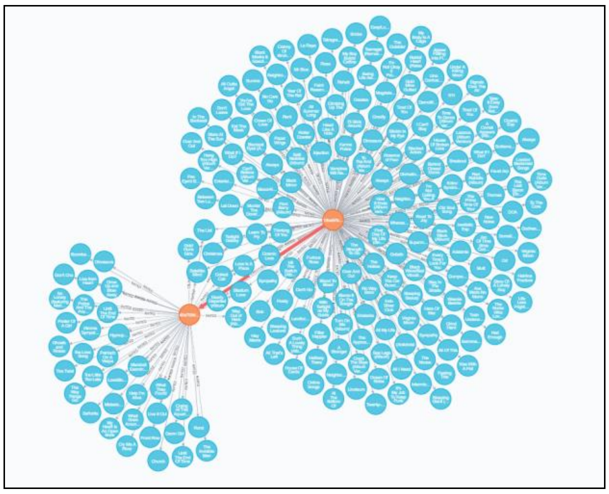
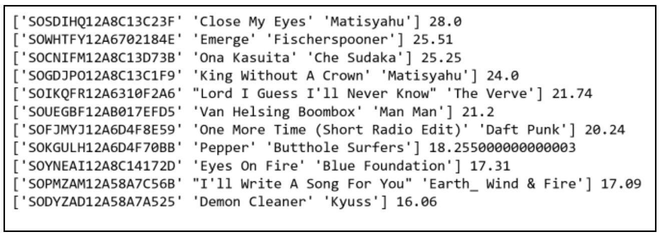


Figure 7.

As we can examine the sample of graph database generated between two users the orange nodes are the users while the blue nodes are the songs they have rated. The red line shows the similarity between the two users.

J. Perform testing on recommendation system

To test our recommendation, we perform multiple queries on the system. For instance, we test the similarity of the two users. We test to find user's actual rating and find the recommended songs that a particular user has not listened to based on preference and taste of other users.



```
['SOSDIHQ12A8C13C23F' 'Close My Eyes' 'Matisyahu'] 28.0
['SOWHTFY12A6702184E' 'Emerge' 'Fischerspooner'] 25.51
['SOCNIFM12A8C13D73B' 'Ona Kasuita' 'Che Sudaka'] 25.25
['SOGDJPO12A8C13C1F9' 'King Without A Crown' 'Matisyahu'] 24.0
['SOIKQFR12A6310F2A6' "Lord I Guess I'll Never Know" 'The Verve'] 21.74
['SOUEGBF12AB017EFD5' 'Van Helsing Boombox' 'Man Man'] 21.2
['SOFJMYJ12A6D4F8E59' 'One More Time (Short Radio Edit)' 'Daft Punk'] 20.24
['SOKGULH12A6D4F70BB' 'Pepper' 'Butthole Surfers'] 18.255000000000003
['SOYNEAI12A8C14172D' 'Eyes On Fire' 'Blue Foundation'] 17.31
['SOPMZAM12A58A7C56B' "I'll Write A Song For You" 'Earth_ Wind & Fire'] 17.09
['SODYZAD12A58A7A525' 'Demon Cleaner' 'Kyuss'] 16.06
```

Figure 8.

For example, a sample result of song recommendations to a specific user has been displayed in figure 8.

VI. CONCLUSION

In this project we demonstrated the capabilities of graph based databases in recommendation systems. We also demonstrated how modern algorithmic techniques can be applied to these databases to perform complex data analysis and prediction. Neo4j makes it very easy to define relationships among the data as compared to other relational database management systems. Therefore, graph databases can prove to be very efficient in biological, multimedia, semantic, network and recommendation systems. If there is a need of a database for data that is highly interconnected, then graph database would be the answer. For majority of other problems, relational database management systems are still a suitable solution which are more stable, reliant and secure**.**

VII. ACKNOWLEDGEMENT

The authors would like to thank professor Ding for her constant guidance and excellent teaching on NoSQL and graph databases throughout the INFO-I590 course. The authors would also like to thank associate instructor Bu for his help, support, and guidance throughout this course.

REFERENCES

[1] Pankaj Gupta, Ashish Goel, Jimmy Lin, Aneesh Sharma, Dong Wang, and Reza Bosagh Zadeh WTF:The who-to-follow system at Twitter, Proceedings of the 22nd international conference on World Wide Web

[2] H. Chen, A. G. Ororbia II, C. L. Giles ExpertSeer: a Keyphrase Based Expert Recommender for Digital Libraries, in arXiv preprint 2015

[3] Alexander Felfernig, Klaus Isak, Kalman Szabo, Peter Zachar, The VITA Financial Services Sales Support Environment, in AAAI/IAAI 2007, pp. 1692-1699, Vancouver, Canada, 2007.

[4] Prem Melville and Vikas Sindhwani, Recommender Systems, Encyclopedia of Machine Learning, 2010

[5] https://towardsdatascience.com/intro-to-recommender-system-collaborative-filtering-64a238194a26

[6] https://labrosa.ee.columbia.edu/millionsong/

[7]http://www.cs.carleton.edu/cs_comps/0607/recommend/recommender/itembased.html

[8] https://rpubs.com/ksb357/388973

[9] Gupta, Mehak & Rani, Rinkle. (2014). Transforming Relational Database to Graph Database Using Neo4j.

[10] Constantinov, Calin & Mocanu, Mihai & Poteras, Cosmin & Popa, Bogdan. (2018). Using a graph database for evaluating and enhancing a social reputation engine. 518-523. 10.1109/CarpathianCC.2018.8399685.