Amanreet Bajwa

KDD Cup

Introduction

The KDD Cup is a data mining challenge hosted by Yahoo! Labs. In this project we were given the task of creating a working solution for Track 1 of the KDD Cup 2011 Challenge. The dataset for this challenge consists of Yahoo! Music data. It is made up of tracks, albums, artists, and genres which all tie together. The different items (albums, users, tracks, etc.) are given meaningless anonymous numbers so that no identifying information is released.

Preprocessing

The first step in any Data Mining project is to take the time to explore and familiarize yourself with the data. This particular data set is made up of tracks, albums, artists and genres. Each track has a unique id, associated album id, associated artist id, and optional genres. Each album has a unique id, an associated artist id, and optional genres. The genre and artist files each hold a list of unique ids. Looking at this non-standardized text format the first step was to put it into a database. The database includes four main tables: albums, artists, genres, tracks, and two join tables: albums\_genres and genres\_tracks. First, I created a SQLITE database that met these specifications. Then I took the time to write a program using Active Record that handles reading in the data files and loading them into the correct places in the database. The loading process ended up taking a lot longer than I was expecting. It took around 2 hours to load all of the data in to my SQLITE database. At this point I realized I forgot to load the users and their ratings into the database so I went back and added a users and ratings table. The loading of the ratings file didn’t take nearly as long since there are only 11696 lines in the files.

Statistics

After all the data was loaded into the database my first thought was to see what exactly the item id rated in the ratings file was referring to. Is it just tracks, albums, and artists? Or does it also include ratings for genres? First I checked the number of ratings there were for albums which turned out to be only 1391. Then I checked the number of ratings for the tracks table tracks which was 7295.The tracks ended up being the most rated item which seems natural. Artists had 2487 and surprisingly there were 479 ratings for genres. I did not expect there to be any ratings for genres because it doesn’t seem like a common thing to rate to me. I decided to see what the average overall rating was and it turned out to be approximately 50.024. Figure 1 shows the average rating for each user which is the next statistic I decided to calculate. The average ratings for each user tend to be pretty high, only a hand full of them fall below 50%. After some thought, I tried to find the number of unique items rated. This seemed like a useful statistic since it would show how the data is distributed in terms of how many times an item was rated. It turns out there are 9308 unique items rated out of 11652 total items (in the sample dataset). This means that each rated item has not always been rated more than once.

Clustering

After spending time on creating the database and generating some simple statistics I decided to think about different algorithms that could be used on this data and compare the options. The first approach that came to mind was clustering. In this dataset I think it would be possible to cluster similar users, tracks, or albums. The only problem with this approach is that it would be hard to judge what items in the dataset are similar. Determining if users are similar would be difficult. It should depend on the ratings each user gave a song but it would have to make sure each of the users actually rated the song. It could also take into account the genres a user likes and compare using that. A lot of different factors can go into finding the similarities between users. One possible way to cluster tracks is to base clusters on genres. This would involve altering the dataset a little though. The genre attributes for a track could be made binary by creating an attribute for each genre that is either true or false. This would allow binary similarity metrics to be used on the data. This type of idea can also be applied to other items in the dataset such as albums, artists, or even users.

Conclusion

Nearest Neighbor idea

* Make a list of all tracks, albums, artists and genres for each
* Make each a binary attribute
* Do KNN and find most similar items and find ratings