Books Recommendation System

Luis Aguilar, Sonali Sharma

1. Introduction

In order to help individuals identify other books they may be interested in, the goal of the project is to build a comprehensive recommendation system for books based on user ratings. We also take into account user's location and age to improve relevance of recommendation. We take dataset consisting of quantifiable book rating values given by users who have provided their age and where they reside. We pre-calculate pair wise books similarity for different ages groups and location. A user is asked to rate a fixed number of books from our dataset and based on the user's rating for these selected books and ratings given to them by other individuals, our recommendation system recommends other books from our dataset that matches user's interest based on ratings based similarity.

2. Problem

The motivation comes from our Open Data project where we are trying to increase accessibility to free public domain electronic books by combining disparate web applications of book sources. Even though many books are available, we believe many books are not selected because classic books may not be known and people want a little help with decision-making.

While the data set we chose does contain ratings of many books that are not in the public domain and will not help with us directly with our motivation, the methods that we are using can be easily replicated as long as we have similar profile and rating data to utilize.

The dataset from http://www.informatik.uni-freiburg.de/~cziegler/BX/ was collected in 2004 from the community at http://www.bookcrossing.com. Below is the schema of the three CSV files, which contain book ratings, books and users who have rated books:

BX-Book Ratings: UserId, ISBN, Book Ratings

	UserID	ISBN	rating
1	User-ID	ISBN	Book-Rating
2	276725	034545104X	0
3	276726	0155061224	5

BX-Books: ISBN, Book-Title, Book-Author

	ISBN	Title	Author	pubyear
1	0195153448	Classical Mythology	Mark P. O. Morford	2002
2	0002005018	Clara Callan	Richard Bruce Wright	2001
3	0060973129	Decision in Normandy	Carlo D'Este	1991

BX-Users: UserId, Location, Age

	UserID	Location	Age
1	1	nyc, new york, usa	37.8951466694
2	2	stockton, california, usa	18
3	3	moscow, yukon territory, russia	32.1151738546

Note: The ratings for the books are on a scale of 1 - 10. 0 rating is the default for un-rated books.

3. Solution/Details

We used item-based collaborative filtering technique. Adjusted Cosine Similarity was used as a similarity measure. First, we created a MapReduce application using mrjob, which calculates the similarities among every pair of books. If this were an application that received new ratings periodically, we would theoretically run the job in sync in order to update all the pairwise similarities.
Second, in order to provide recommendations, we ask the users to rate 5 application-selected books (based on user's age and location). We run another mrjob that then takes user provided ratings, reads the pre-computed pair wise similarity files and recommends top 5 books to the user.

4. Data Preprocessing and mining

The files themselves contain 278,858 users, 271,379 books and 1,149,780 book ratings. There are 716,109 records with missing ratings (defaulted to 0). Since there are 168,096 records that contain ages, many missing ages were defaulted to the mean for that specific data set since it was a symmetric distribution. Many of the ratings turned out to be 0 and many of the location names were not standardized, hence we decided to take rating>0 and divide locations into USA and Non-USA based groups. We also took only those books that were rated by at least 5 users in order to get useful similarity calculation. In order to account for user's demographic, the ratings had to be split up by location (USA vs Non-USA) and by age.
Since our normalized dataset was smaller, to remove outliers we used z-score method. We calculated z-score for all ages and decided to keep data within the z-score range of -3.0 to +3.0. For raters from the USA, the mean age was 37.89514, the standard deviation was 11.18987, which created an age range of 5 to 71. For Non-USA raters, the mean age was 32.11517, the standard deviation was 13.25913, which created an age range of 0 to 71.
Ipython notebook for data analysis: http://nbviewer.ipython.org/5593987

5. Collaborative Filtering

Collaborative filtering is the process of filtering information or patterns using techniques involving collaboration between agents, viewpoints or data sources. The two main methods are item based and user based. We chose item based collaborative filtering due to the sparse rating habits and general dissimilarity of the users in the dataset. Furthermore, in order to handle the differences in rating scale between different users, we chose **adjusted cosine similarity**, which subtracts the user average from each co-rated item pair. Similarity between books i and j is calculated as:

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

Here $\bar{R_u}$ is the average of the *u*-th user's ratings. R(u,i) is the rating given to ith book by user u.

An obvious shortcoming of adjusted cosine similarity for our dataset is that it needs ratings from multiple users in order to calculate pairwise item similarity. For those book pairs that were rated by only one user, the convention adjusted cosine similarity would always be 1, which was misleading. To overcome this we decided to take the difference of ratings between the two books and use a linear scale to translate them to a similarity value. Hence if a user rated a book 1 and another 10, then the difference of ratings which 9 would give it a similarity of -1 (dissimilar), using this we came up with a linear line equation to calculate similarity.

predicted similarity = -(2/9) difference of rating + 1

6. Recommendations

We calculated and stored the pair wise book similarity for different age groups and location. In order to provide recommendation for books, we asked the user to provide ratings for 5 application-selected books (based on user's age and location). We then use high rated books (rating of 5 and above) and use another mrjob to fetch top 5 recommendations. For each highly rated book we read the pre-computed similarity file for that specific age, location and all combinations that include the user rated book. We then arrange this list in descending order of similarity and pick the top 5 books for recommendations. Here we make an assumption that the recommendation will only be provided based on the highly rated books. If the user does not rate any of the books highly then the system does not give any recommendations.

Verification of the results was done using a subset of our data as test set. We took one user at time and chose a set of 5 books highly rated by the user. We then passed these books as an input to our recommendation mrjob. The returned recommendations consisted of books that were also highly rated by the same user. We repeated this step for multiple users. We got an accuracy of about 84% on our results.

7. Process/Reproducibility

The data was transferred from csv file to an sqlite3 database. Sqlite3 is a lightweight file based database, which was sufficient for our project. We preprocessed data using Python in order to set defaults, remove outliers, and partition by demographics. The data was analyzed using an iPython notebook by loading it into pandas data frames from the database. We calculated adjusted cosine similarity for all book rating combinations and cosine similarity for ratings given by user for five books in order to retrieve similar books using mrjob. Using mrjob we got a performance gain of at-least 90% as compared to serial processing that we were doing earlier.

The collaborative filtering, using adjusted cosine similarity can be easily reused or any rating system. We have seen similar application for movie recommendations. The code for this recommendation system is modular. In order to run the code, refer to the steps provided in README.md file

Github: https://github.com/fchasen/bookhunters/tree/master/data_mining/mrjob

Steps to run the code: https://github.com/fchasen/bookhunters/blob/master/data_mining/mrjob/README.md

8. Related Work

Collaborative filtering. (n.d.) In Wikipedia. Retrieved April 22, 2013 from http://en.wikipedia.org/wiki/Collaborative_filtering

Sarwar, B., Karypis, G., Konstan, J., Riedl, J. (2001). Item-Based Collaborative Filtering Recommendation Algorithms. Retrieved from http://www.grouplens.org/papers/pdf/www10_sarwar.pdf

Amatriain, X., Jaimes, A., Oliver, N., Pujol, J. (2011). Data Mining Methods for Recommender Systems. Retrieved from http://www.newbooks-services.de/MediaFiles/Texts/7/9780387858197_Excerpt_001.pdf

Introduction to Recommendations with MapReduce and mrjob. (2012, August 23). Artificial Intelligence in Motion. Retrieved April 24, 2013 from http://aimotion.blogspot.com/2012/08/introduction-to-recommendations-with.html

9. Further Work

In future we plan to extend this algorithm and incorporate content-based recommendation system. It would also be interesting to cluster books based on their rating, content and reviews by the user. we also plan to build a web application for this recommendation system, this will provide a better user interaction.