Book Recommendation System

IS 688 – Web Mining

Group Seven

Submitted To:
Christopher Markson

Submitted By:
Harshad Kolhe
Lakshmi Narayana Dhavala
Suhrud Reddy
Tejas Redkar
Vamsi Krishna Channa

Table of Contents

1 What is a recommendation system?

In our day to day life, we come across a large number of Recommendation engines like Facebook Recommendation Engine for Friends' suggestions, and suggestions of similar Like Pages, Youtube recommendation engine suggesting videos similar to our previous searches/preferences.

A recommendation system performs extensive data analysis in order to generate suggestions to its users about what might interest them. R has recently become one of the most popular programming languages for the data analysis. Its structure allows you to interactively explore the data and its modules contain the most cutting-edge techniques thanks to its wide international community. This distinctive feature of the R language makes it a preferred choice for developers who are looking to build recommendation systems.

These systems serve two important functions:

- They help users deal with the information overload by giving them recommendations of different products.
- They help businesses make more profits, i.e., selling more products.

1.1 What we built?

Using R we built a Recommendation System which interacts directly with the user database for suggesting books to an individual user based on the user's previous readings/ratings . Similar books are retrieved using the similarity score computed by our system. Books are then suggested to the user based on a predicted rating computed by our system for the particular user.

1.2 What does our system do exactly?

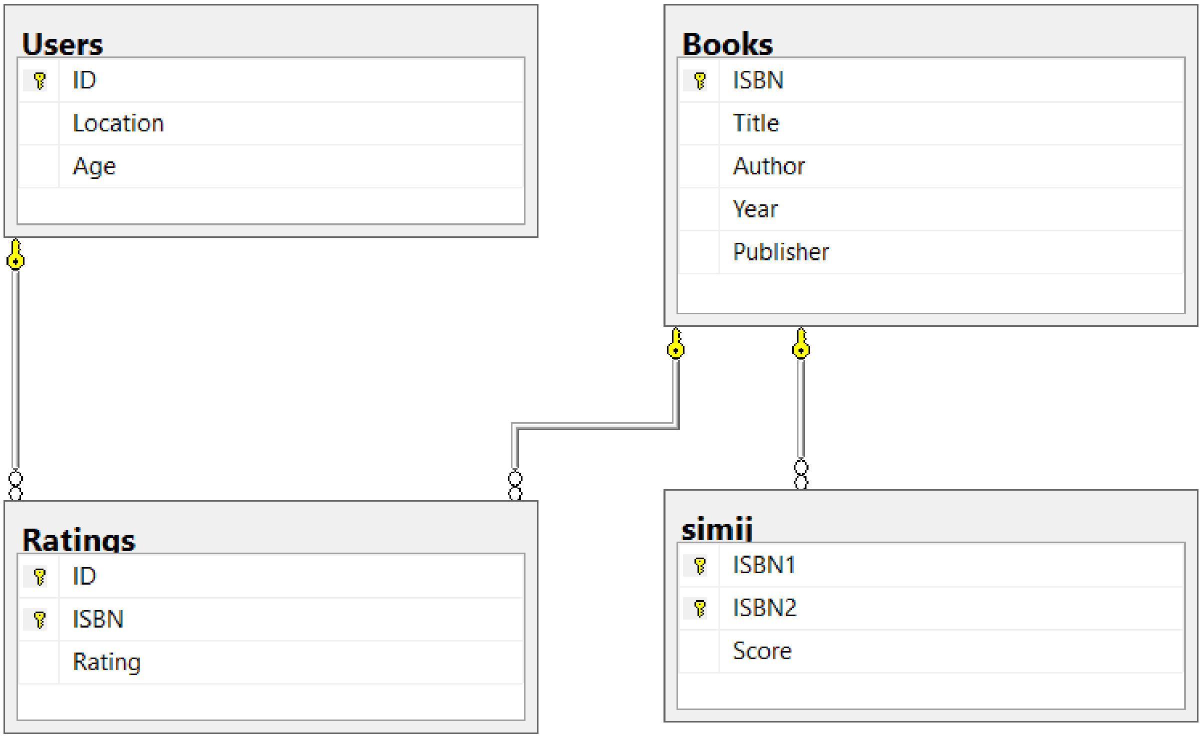
Item-based Collaborative filtering

Recommender systems and collaborative filtering became a topic of increasing interest among human–computer interaction, machine learning, and information retrieval researchers. One approach to the design of recommender systems that has wide use is Collaborative filtering methods are based on collecting and analyzing a large amount of information on users' behaviors, activities or preferences and predicting what users will like based on their similarity to other users.

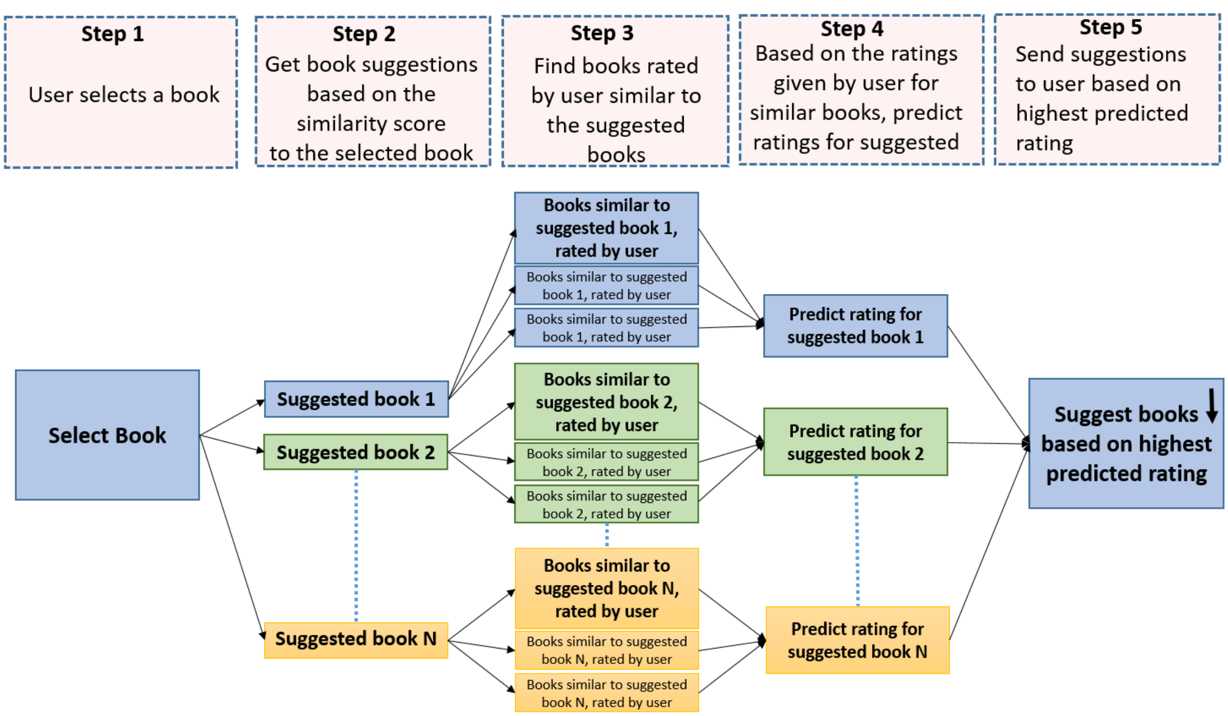
The fundamental assumption behind this method is that other users' opinions can be selected and aggregated in such a way as to provide a reasonable prediction of the active user's preference. Intuitively, they assume that, if users agree about the quality or relevance of some items, then they will likely agree about other items — if a group of users likes the same things as Mary, then Mary is likely to like the things they like which she hasn't yet seen. There are other methods for performing recommendation, such as finding items similar to the items liked by a user using textual similarity in metadata (content-based filtering or CBF). The focus of this survey is on collaborative filtering methods, although content-based filtering will enter our discussion at times when it is relevant to overcoming a particular recommender system difficulty. The majority of collaborative filtering algorithms in service today, including all algorithms detailed in this section, operate by first generating predictions of the user's preference and then produce their recommendations by ranking candidate items by predicted preferences. Often this prediction is in the same scale as the ratings provided by Baseline Predictors users, but occasionally the prediction is on a different scale and is meaningful only for candidate ranking. This strategy is analogous to the common information retrieval method of producing relevance scores for each document in a corpus with respect to a particular query and presenting the top-scored items. Indeed, the recommend task can be viewed as an information retrieval problem in which the domain of items (the corpus) is queried with the user's preference profile. Therefore, this section is primarily concerned with how various algorithms predict user preference. In later sections we will discuss recommendation strategies that diverge from this structure, but in actual implementation they frequently start with a preference-ranked list of items and adjust the final recommendation list based on additional criteria. A key advantage of the collaborative filtering approach is that it does not rely on machine analyzable content and therefore it is capable of accurately recommending complex items such as movies without requiring an "understanding" of the item itself.

2 Design

2.1 Database



2.2 Prediction Process



3 Detailed Process

3.1 Dataset

Books	271,379
Users	278,858
Ratings	1,149,780

http://www2.informatik.uni-freiburg.de/~cziegler/BX/

For the purpose of the project, we computed the similarity between 1000 books. OpenRefine was used to clean and filter the data before importing it into Microsoft SQL Server. The similarity score between the books has been stored in the 'simij' table, which contains 416,824 similarities.

3.2 Computing Similarity

The RODBC package has been used to connect R with the database, to compute and store the similarity score between each book.

The first step is to retrieve all the available books in the database

select isbn from books

Next, we run two loops on the retrieved books to compute the similarity score of one book with all the others. Inside these loops, users who have rated both the books running in the loops are retrieved.

*select id from ratings where isbn=**book in first loop***
intersect
*select id from ratings where isbn=**book in second loop***

After this, a loop is run on the users and the rating given by the users retrieved for each of the book is retrieved

*select rating from ratings where isbn=**book isbn***

We retrieve the average rating the user has given to the books rated by thyself using the SQL average function

*select avg(rating) as avgr from ratings where id=**user id***

Using all the information retrieved, the similarity score is then computed using the item based collaborative filtering formula, and then stored in the database.

3.3 Prediction Process

The user selects a book, based on this selection, books with a high similarity score to the selected book are retrieved from the database. The following SQL query is used to retrieve similar books with a similarity rating above 0.85, excludes books already read by the user

*select isbn2 from simij where isbn1=(**selected book isbn**)* *and isbn2 not in (select isbn from ratings where id=(**Current user's id**) and score>**0.85***

Next, the system then retrieves books read and rated by the user which are similar to the suggested books.

*select ISBN2 from simij where isbn1 =(**isbn of suggested books**)* *intersect* *select isbn from ratings where id=**current user id***

After this, the ratings given by the user for the suggested similar books are retrieved

*select rating from ratings where isbn=(**isbn of similar books read by user**)* *and id=(**current user's id**)*

Finally the similarity rating of the suggested books and the similar books(to the suggested) read by the user are retrieved.

*select score from simij where ISBN1=(**suggested book**)* *and ISBN2=(**similar books rated by the user, to the suggested**)*

After all this information is retrieved for each book, a predicted rating for the current user is computed using the prediction formula, and the books are then suggested using the highest predicted rating.

3.4 Formulae utilized

Item based collaborative filtering to compute the similarity between the books

$$sim(i,j) = \frac{\sum_{\mathbf{u}\in U}(r_{\mathbf{u},i} - \bar{r}_{\mathbf{u}})(r_{\mathbf{u},j} - \bar{r}_{\mathbf{u}})}{\sqrt{\sum_{\mathbf{u}\in U}(r_{\mathbf{u},i} - \bar{r}_{\mathbf{u}})^2}\sqrt{\sum_{\mathbf{u}\in U}(r_{\mathbf{u},j} - \bar{r}_{\mathbf{u}})^2}}$$

Rating prediction formula

$$p(\mathbf{u},i) = \frac{\sum_{j\in J} r_{\mathbf{u},j} \times sim(i,j)}{\sum_{j\in J} sim(i,j)}$$

4 Algorithms

4.1 Similarity computation

1. Retrieve all books from the database
2. First loop to select a book
3. Second loop to compare the selected book to all the other books in the database
4. Retrieve users who have rated both books
5. Third for loop to go through each user
6. Retrieve current user's rating for both book along with his average rating
7. Perform required calculations and append to variables
8. After user loop, compute and store similarity for the two books
9. Proceed to compare the first book, with the other books in the database
10. Perform the same for all the other books

4.2 User rating prediction

1. Unique user selects a book
2. The user's id and selected book's isbn is passed to the function
3. Books similar to the selected book are retrieved
4. First for loop on the similar books retrieved
5. Books similar to the suggested book, but have been rated by the user are retrieved
6. Second for loop on the retrieved books
7. Similarity score and user's rating on the first retrieved book are fetched
8. Perform required calculations and append to variables
9. After second for loop, compute predicted score for first suggested book
10. Perform the same for all the other similar/suggested books

5 Conclusion

Recommender systems have become ubiquitous. People use them to find books, music, news, smart phones, vacation trips, and romantic partners. Nearly every product, service, or type of information has recommenders to help people select from among the myriad alternatives the few they would most appreciate. Sustaining these commercial applications is a vibrant research community, with creative interaction ideas, powerful new algorithms, and careful experiments.

Recommender Systems in Books will help the customer select books similar to their interest, which in turn will lead to increase in business revenue.

There still are many challenges for the field, especially at the interaction between research and commercial practice.

5.1 Challenges

We have framed these challenges here in a four-part taxonomy.

5.1.1 Data

The data used by recommender systems are sometimes biased in unexpected ways that can have a dramatic effect on outcomes.

5.1.2 User experience

The goal of recommender systems is to improve user experience. In both research and practice, crafting the user experience to fit the application and the user lifecycle remains a substantial challenge.

5.1.3 Evaluation and metrics

Evaluation of recommender systems has advanced significantly over the past decade, but many challenges remain. Even with a better understanding of the relationships among different algorithm performance metrics, the field struggles to standardize evaluation.

5.1.4 Social impact

Because collaborative filtering recommender systems must by their nature collect substantial personalized data about their users, they face important privacy and security challenges. Researchers have been approaching these challenges head-on by attempting to develop ways to collect and store data in such a way that extracting personalized data is provably difficult.

5.2 Room for Improvement

1. If user does not have a profile with a sufficient amount of ratings, recommendations can be given based on similar user profiles like age, demographics, gender, etc.
2. Recommend items based on inverse similarity if data is insufficient for similar predictions

5.3 References

1. G. Adomavicius and A. Tuzhilin, “Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions,” IEEE Transactions on Knowledge and Data Engineering, vol. 17, no. 6, pp. 734–749, 2005.
2. E. Agichtein, E. Brill, S. Dumais, and R. Ragno, “Learning user interaction models for predicting web search result preferences,” in ACM SIGIR ’06, pp. 3–10, ACM, 2006.
3. D. Aha and L. Breslow, “Refining conversational case libraries,” in Case-Based Reasoning Research and Development, vol. 1266 of Lecture Notes in Computer Science, pp. 267–278, Springer, 1997