International Journal of Computer Applications (0975 – 8887)
Volume 159 – No 9, February 2017

SuggestABook: A Book Recommender Engine with Personality based Mapping

Shivani Bhosale
Dept. of Computer Engineering,
PCET’s Pimpri Chinchwad
College of Engineering, Nigdi

Pranjal Nimse
Dept. of Computer Engineering,
PCET’s Pimpri Chinchwad
College of Engineering, Nigdi

Siddhi Wadgaonkar
Dept. of Computer Engineering,
PCET’s Pimpri Chinchwad
College of Engineering, Nigdi

Aishwarya Yeole
Dept. of Computer Engineering,
PCET’s Pimpri Chinchwad
College of Engineering, Nigdi

ABSTRACT

Recommendation systems are used to filter information to seek or predict the preference of a user for a particular item. Online book recommender engines recommend books to the users based on their categories of interest. This paper proposes a book recommender system which accepts the user’s preferences as well as analyses Facebook profile to get user’s data and recommends books with a certain degree of support and confidence. It makes use of continuous learning algorithms for recommending books to the user each time the user is logged in. In case a user does not have a Facebook account, his preferences will be accepted explicitly.

Keywords

Recommender systems, content-based filtering, collaborative filtering, support, confidence, books

1. INTRODUCTION

Recommender systems are tools and techniques for suggesting items that could be of use to users. They provide suggestions related to various decision making processes. Recommender systems typically produce a list of recommendations either by Collaborative filtering or Content-based filtering. Recommender system is a useful alternative tosearch algorithms as it helps users discover items they might not have found by themselves. Recommender systems today are often implemented using search engines indexing non-traditional data.

2. COLLABORATIVE FILTERING

Collaborative filtering is the most commonly adopted technique used in commercial recommender systems. Its basic idea refers to making recommendations based upon ratings that users have given to a particular product. Ratings can either be based on user’s opinion about a given product or when the purchasing or mentioning of an item counts as an expression of appreciation. While implicit ratings are generally more facile to collect, their usage adds noise to the collected information.

2.1 User-based collaborative filtering

User-based collaborative filtering predicts a test interest of user in a test item based on rating information from similar user profiles. Cosine similarity and Pearson’s correlation are popular similarity measures in collaborative filtering.

2.2 Item based collaborative filtering

Item-based collaborative filtering is a form of collaborative filtering based on the similarity between items calculated using people's ratings of those items .Item-item models use rating distributions *per item*, instead of user. Each item tends to have more ratings than each user, so usually an item's average rating does not change quickly. This leads to more stable rating distributions in the model. When users consume and then rate an item, similar items are picked from the existing system and added to the user's recommendations.

3. CONTENT-BASED FILTERING

Content-based filtering (CBF) is one of the most widely used recommendation approach. Main component of CBF is the user modeling process, in which users’ interests are concluded from the items that users have used or searched for. Items are usually textual, for instance emails or web pages. "Interaction" actually means downloading, buying, authoring, or tagging an item. Some recommender systems also use features like writing style layout information and XML tags. Content-based recommendation systems try recommending items similar to those a given user has liked in the past, whereas collaborative recommendation systems identify users whose preferences are similar to those of the given user and recommend items they have liked.

4. EXISTING SYSTEM

Traditional book recommender systems suggest books that are mostly likely to be preferred by the users. The proposed system not only recommends books but also uses support and confidence measures to recommend books.

5. PROPOSED ARCHITECTURE

The proposed system has three major components and can be represented as a 3-tier architecture. The main Processing module i.e. the Recommender Engine collaborates with both the Android Application module and Database and data sources.

Android app is used for providing user interface to the system. When the user will log in for the first time to the app, he will be asked to sign up with Facebook. Sign-up with Facebook helps the system to connect with the user's Facebook account from which it can access his Profile details such as "Interests' , "Likes" , "Books Read" etc. This information will be used to predict the personality of the person which can in turn be used to recommend books for him.

The Recommender Engine will have appropriate intelligent algorithms which will recommend relevant books to the user. Support and Confidence measures are the major sub-modules which will calculate Support and confidence for the recommendations. After the final top-5 or top-10 books are obtained, it will be recommended to the user via Android app.

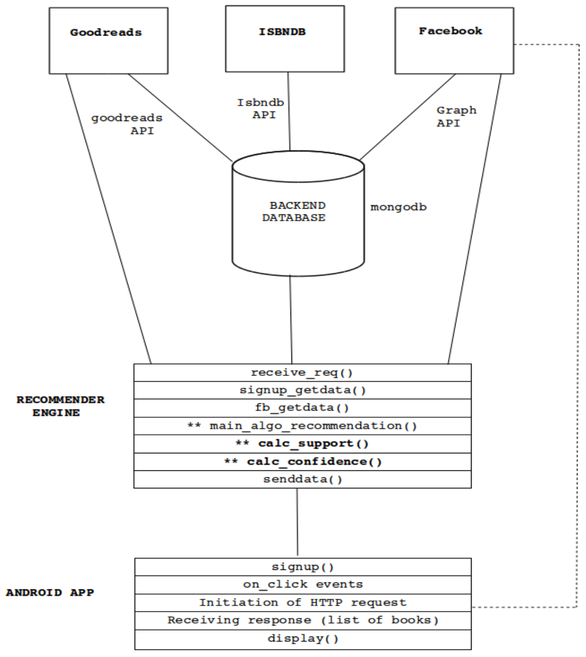
The backend database will contain a number of attributes. Majorly, there will be two categories: Users Database and Books Database. The Books database will have a huge number of books and its related information such as Title, Author, Genre, and Year etc. This information can be retrieved from Goodreads, ISBNDB and other such online datasets. To use Goodreads and ISBNDB, their respective APIs are available to connect to the online datasets. The Users database will store information containing User Name, Books Read by the user, Personality (predicted by the system) etc. Preferably MongoDB will suit best for such type of data.

6. MATHEMATICAL FORMULAE

6.1 SUPPORT and CONFIDENCE

Support (X , Y) = #tuples containing both X and Y / total number of tuples Confidence (X , Y) = #tuples containing both X and Y / #tuples containing X

Support and Confidence will be used to measure the relevancy of a certain trait in that person, as well as relevancy of the book suggested to him. For example, if a person is classified as a Spiritual person, so to what level does this classification is correct; is calculated using support and confidence. Also, when a book is recommended, what is the % prediction that the person will like this book? For example, there are 80% chances that the user will like this particular book. So it's a better option to recommend. For some other case, if the confidence comes out to be only 20% then it's not a better option to recommend this book to the person.



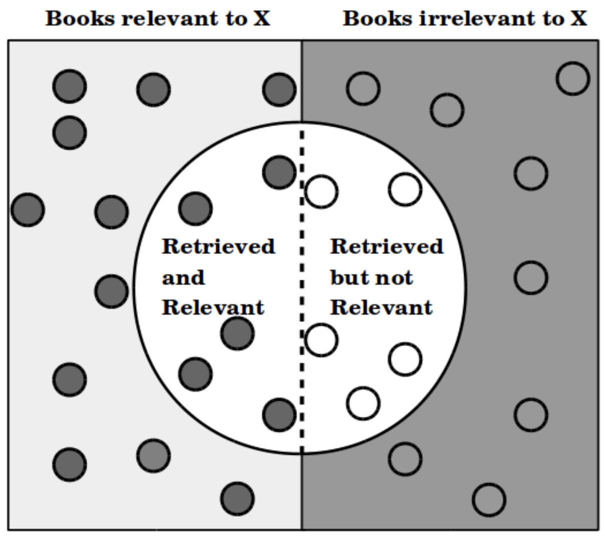
5.1 Proposed System Architecture

6.2 PRECISION and RECALL

Precision and Recall are another means of measuring the performance/accuracy of the recommendations. Considering that the system recommends/retrieves a set of books after performing its intelligent algorithms; it is important to check the relevancy of these predictions and that can be done using Precision and Recall.

Precision = Retrieved Relevant Books / Total Retrieved Books

Recall = Retrieved Relevant Books / Total Relevant Books



6.1 Precision and Recall

For a specific user X : say originally there are some relevant books and some irrelevant (less relevant) books. This is represented by the complete square. Circle represents the Books that are Retrieved or Recommended by the System.

7. FP-GROWTH ALGORITHM

FP-growth is used for finding frequent item set mining. The FP-Growth Algorithm is a way to find frequent item sets without using candidate generations, thus results into improving performance. It uses a divide-and-conquer strategy. The core of this method is the usage of a special data structure named frequent-pattern tree (FP-tree), which retains the item set association information.

The algorithm works as follows: first it compresses the input database creating an FP-tree instance to represent frequent items. It divides the compressed database into different set of conditional databases, where each and every database is associated with one frequent pattern. At the end, every such database is mined separately. Using this strategy, search costs is reduced here by looking for short patterns recursively and then concatenating all of them in the long frequent patterns, offering good selectivity.

7.1 ALGORITHM

Input: A transaction database DB and a minimum support threshold #. [1]

Output: Its frequent pattern tree, FP-tree

Method: The FP-tree is constructed in the following steps. [1]

1. Scan the transaction database DB once. Collect the set of frequent items F and their supports. Sort F in support descending order as L, the list of frequent items.

2. Create a root of an FP-tree, T, and label it as \null". For each transaction Trans in DB do the following. Select and sort the frequent items in Trans according to the order of L. Let the

sorted frequent item list in Trans be [p j P], where p is the first element and P is the remaining list. Call insert tree ([p j P]; T).

The function insert tree ([p j P]; T) is performed as follows. If T has a child N such that N.item-name = p.item-name, then increment N's count by 1; else create a new node N, and let its count be 1, its parent link be linked to T, and its node-link be linked to the nodes with the same item-name via the node-link structure. If P is nonempty, call insert tree (P; N) recursively.

Analysis: From the FP-tree construction process, we can see that one needs exactly two scans of the transaction database, DB: the first collects the set of frequent items, and the second constructs the FP-tree. The cost of inserting a transaction Trans into the FP-tree is O(j Trans j), where j Trans j is the number of frequent items in T rans. We will show that the FP-tree contains the complete information for frequent pattern mining.[1]

7.2 APPROACH

Top-k genres of books that user likes can be taken at the time of sign-up. Those pair of users who have liked books from same genre can be suggested the other books on the basis of user-user collaborative approach.

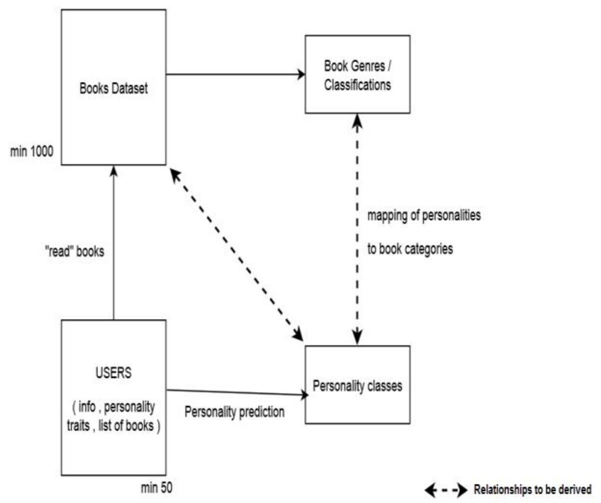
Clusters of such users with similarity measures can be formed and nearest neighbours to the cluster can be recommended.

There are few more approaches:

user-item similarity

item-item similarity

8. DATASET OVERVIEW



Required Dataset

1. Book Dataset classified into book Genres

2. Users database with personality classification

Flow of database:

User's database consist of User info, personality traits, list of books read by user

“Books read by user” should be present in book dataset.

Relationships to be derived:

1. Assigning books to personality classes (of user database)
2. Mapping of personalities to book genre

The main task of algorithm will be to establish these links and suggest the books accordingly.

9. EXPERIMENTAL SETUP

Our experimental setup consists of android application integrated with facebook sdk and graph API.

Cloud storage is the model used in our project for storing the data extracted from Facebook and the book dataset.

9.1 FACEBOOK SDK FOR ANDROID

The Facebook SDK for Android is the easiest way to integrate your Android app with Facebook. It enables:

9.1.1 Facebook Login: Authenticate users with their Facebook credentials.

9.1.2 Share, Send dialogs: Enable sharing content from your app to Facebook.

9.1.3 App Events: Log events in your application.

9.1.4 Graph API: Read and write to Graph API

9.2 GRAPH API

The Android SDK has support for integrating with Facebook Graph API. With the GraphRequest and GraphResponse classes, you can make requests and get responses in JSON asynchronously.

Android SDK sends any permissions your app has in the access_token and this controls data access. Ifyour app has no available access token, Graph API returns only publicly available information

A Request can be executed either anonymously or representing an authenticated user. If requests are executed in a batch, a Facebook application ID must be associated with the it, either by setting the application ID in the AndroidManifest.xml or via FacebookSdk or by calling the setDefaultBatchApplicationId.setDefaultBatchApplicationId(String) method.

9.3 Cloud Storage

GoDaddy Cloud storage is the model used in our project for storing the data extracted from facebook and

the book dataset. The cloud storage providers are responsible for keeping the data available and accessible. Cloud storage services may be accessed through a cloud computer service, a web service API or by applications that utilize the API, such as cloud desktop storage, a cloud storage gateway or Web-based content management systems.

10. RESULTS

Experiments to be held mainly focus on the calculation of support and confidence for various book recommendations . Various sources can be used to predict the personality of the user, based on which certain recommendations can be made. Then, the top 4-5 books under the same class label will be recommended to the user. At the initial stage of testing the algorithm, the results will be used to check the accuracy of the same. Also, these results will be used to optimize the system further. Regular feedback will also enhance the knowledge of the system and increase its ability to produce accurate recommendations.

11. CONCLUSION

In this paper we present a recommendation system that is based on collaborative filtering and content based filtering methods. This book recommendation has considered many parameters like attributes of the books and also personality based mapping of users. This recommender system also uses support and confidence measures to give stronger recommendations. They are very useful in recommending books to users according to their need and interests.

12. ACKNOWLEDGMENTS

We thank our project guide Mr. S. S. Deshmukh for contributing his thoughts and ideas. We also thank Persistent Sytems for the project sponsorship and Mr. Snehkumar Shahani, our mentor from Persistent Systems, for his guidance.

13. REFERENCES

[1] Mining Frequent Patterns without Candidate GenerationJiawei Han, Jian Pei and YiwenYinSchool of Computing ScienceSimon Fraser Universityfhan, peijian, yiweny g @cs.sfu.ca

[2] Hybrid book recommender system for an e-commerce application, Laxmi V, Dr M C Padma, Vol. 3, Issue 2, pp: (921-924), Month: April - June 2015

[3] Improving Recommendation Lists Through Topic Diversification, Cai-Nicolas Ziegler, Sean M. McNee, Joseph A. Konstan,Georg.Lau