Book Recommender System & A Study of Effectiveness of Different Visualization Techniques in Recommender Systems

Aditya Narula
anarula@cs.umass.edu

Ramya Sarma
rsarma@cs.umass.edu

ABSTRACT

Recommender systems are found in almost all walks of life helping their users to decide, compare, explore and discover on the basis of information available through online resources or gathered from relevant community. Book recommendation is a popular use case of such a system. Users rely heavily on these to choose their next read. The most popular book recommendation websites such as GoodReads and Amazon leverage this to generate recommendations based on the user's reading history and books liked by similar users. But users are not always content with the suggested books. They find no available ways to get new sets of recommendations unless they rate/like more books or just go and explore the popular collections (Top authors, Most read books in 2018) on these websites. This is a major limitation of these websites which, in addition to limiting user satisfaction, also has other implications such as the new titles remaining unexplored by users.

In this project we aim to study the effectiveness of visual recommendation system to produce high quality recommendations for the user. We study and implement visualizations to promote sense making of the user-specific retrieval results. The project aims at developing a web based application, that would provide book recommendations on the basis of user's profile, i.e. a list of their previous ratings. The application aims to base its decisions on reliable, authentic and filtered source of information, thus saving its users from the problem of unnecessary information overload.

1. MOTIVATION

Recommender system is a software agent that improves decision making process of its users, by making appropriate recommendations on the basis of their interests, preferences or peculiar requirements, elicited explicitly or implicitly [4]. Additionally, Recommender systems have also become valuable means for dealing with the information overload problem. Explosive growth of world wide web and countless sources of information for single query have not only overwhelmed users, but has also led to poor decision making. Thus over-provision of choices and unprecedented load of data has killed the very purpose of it, for which it was initially designed. Recommender systems relieve user from the problem of "too many choices" and present information to the user according to his context and needs [3]. These systems try to mimic the basic human psychology of relying on others' suggestions and advice, while making decisions in one's day to day life. The tendency becomes even more prevalent, when that source of recommendation is considered knowledgeable, resourceful, transparent and unambiguous in its approach. For example, it is a common human norm to select a movie on someone's suggestion or newspaper review, one trust, or even hiring people on the basis of recommendation letters from renowned institutions, without much hesitation [3]. Similarly, Recommender systems are directed towards individuals, not having enough knowledge or competence to decide among various alternatives. In doing so, they generate recommendations on the basis of information available about users, items and previous relationship history between the two, if any. These recommendations may be in the form of ranking list on the basis of user's preferences, capabilities and constraints etc. The user may select or decline the options with relevant feedback, that is stored in the database; and can be used for new recommendations in any of the next sessions [3]. Recommendation systems have found applications in a plethora of products and services. A Recommendation system (RS) essentially assists the customer to better select items of choice, while offering serendipitous suggestions. Netflix, uses these systems to recommend movies that viewers may enjoy. RS have a large impact on such businesses as users tend to rely heavily on these recommendations which, in turn, drive sales. For instance, Amazon generates 35% of the revenue by its own Recommender engine. One such area where RS are being extensively utilized is in online book suggestion websites like Goodreads and retail websites like Amazon Book Store. Current research in this domain is focused on improving recommendations based on specific applications. To accurately gauge users' interest in books, we integrated the two strategies in the field of recommendation systems to build a hybrid recommender model and evaluate different visualization techniques.

2. RELATED WORK

There are several recommendation websites already in existence for various domains. The methodology adopted for giving recommendations by these sites may vary but have still have a lot in common. Item-based collaborative filtering and User-based collaborative filtering are the two commonly adopted techniques. In Item-based recommendations, the similarity between items is taken into account and then predictions are made. Whereas in the latter, users with similar tastes are found and on the basis of their ratings, predictions are made. Different algorithms like Cosine Similarity Measure, Pearson Correlation Similarity Measure are used for the same. Book recommendation websites have flourished over the web over the past decade. Huddersfield Book Rec-

ommender system, BookPsychic, WhatshouldIreadnext.com, LibraryThing, Goodsreads.com and Bookexplorer.com are some of the popular ones. Content-based recommendations are also provided by some of the mentioned systems. But most current related work we found do not focus on using visualization features to generate recommendations for users. In [1], Bostandjiev et al explored an interactive hybrid music recommendation system that generates item predictions from multiple social and semantic web resources, such as Wikipedia, Facebook, and Twitter. It goes on to describe a supervised user study was performed using the system to explore research questions relating to visual interactive recommendation systems. We have adopted the same evaluation technique for our system as future scope of the project. Information Visualization for Interactive Information Retrieval [2] This paper focuses on how information visualization supports interactive information retrieval. It goes on to review and survey existing search interfaces which leverage this idea.

3. DATASETS

3.1 Dataset information

For this project, we used the GoodBooks 10K dataset: The dataset contains 5,976,479 ratings for ten thousand most popular books (with most ratings). There are also:
1. books marked to read by the users
2. book metadata (author, year, etc.)
3. tags/shelves/genres
Ratings.csv contains ratings sorted by time. It is 70MB: Ratings go from one to five. Both book IDs and user IDs are contiguous. For books, they are 1-10000, for users, 1-53424. toread.csv provides IDs of the books marked "to read" by each user, as userid, bookid pairs, sorted by time. There are close to a million pairs.
books.csv has metadata for each book (goodreads IDs, authors, title, average rating, etc.). The metadata have been extracted from goodreads XML files, available in books.xml. booktags.csv contains tags/shelves/genres assigned by users to books. Tags in this file are represented by their IDs. They are sorted by goodreadsbookid ascending and count descending.
Here, each tag/shelf is given an ID. tags.csv translates tag IDs to names.
Each book may have many editions. goodreadsbookid and bestbookid generally point to the most popular edition of a given book, while goodreads, workid refers to the book in the abstract sense.
Note that bookid in ratings.csv and toread.csv maps to workid, not to goodreadsbookid, meaning that ratings for different editions are aggregated.

3.2 Dataset cleaning

1. We only made use of books which had complete information in the books.csv file. We removed all the other data points.
2. We removed null values present in the ratings.csv files.
3. Further, we assigned the ratings a categorical value so that we can feed it into the TF-IDF vectorizer to extract feature vectors.

4. APPROACH

The two most popular approaches in recommendation systems are Content Based and Collaborative Filtering. We implement our recommendation system using a hybrid model, which uses a combination of these approaches to generate the book recommendations. We use the weighted hybridization approach.
Steps:
1. For a particular user, we initially display a set of top books read by the user. This is based on the user's history i.e. the list of books he has liked and rated higher.
2. Based on these, our system generates an initial set of recommended books for the user.
3. These recommendations have been generated using a certain set of features and corresponding weights.
4. The features we selected were Genre, Author and Ratings.(Approach described later)
5. Below the recommended books collection, is a visualization describing the relative weights of the primary features used to create the model.
6. If the user wants better recommendations, the user can choose to interact with this visualization and modify the weights of the features.
7. The system then recomputes the weights and provides these updated weights to the model, which in turn generates new recommendations.
8. The user can also choose to ask for recommendations completely based on one feature using the selection radios on the left side of the visualization.

Selecting features

We chose the ratings, author name and genre as the three main features to use. We chose these three because studies have shown that users prefer to read books from previously liked genre. For instance, if a user has liked books from the 'horror' genre in the past, the user will like to see books from the 'horror' genre in the future as well. Studies have also shown that users who who like a particular author, like to stick to the same author and read all books by that author.

Selecting Visualizations

We researched and did some basic user testing with a multiple visualizations such as bar graph, pie chart and histogram to get a sense of which one would be the most effective in helping users interact with the system and change weights of the features. It was found that pie-charts served the purpose the best.

5. SYSTEM DESIGN AND INTERACTION

We implemented the book recommender system using the Hybrid Model. We used visual features for analysis and displaying of the results. This way we built an interactive book recommender system where user has control over the recommendation process. For this purpose, we generate an initial set of recommendations for the user. And then allow the user to interact with the system with the help of visualizations we generate from the data.
As for the visualizations, we will explored different visualization techniques based on features available in the dataset and whether individual features work well or a combination of certain features help the user better.

5.1 Hybrid Model

For a given user id we initially present the user with an initial set of recommendations. For genre based recommen-

dations, we took the genre that the user has like the most in the past. By "most" we imply that we got the counts of the genre of the books in the user's history. And then, we used content and collaborative filtering to find the top rated books in that same genre which the user might like. When it comes to rating based, we have used a different kind of technique. Since the rating in the dataset is just numbers between 1 and 5, getting useful information from this field is a challenge. To make sense of this rating, we took the title of the books rated by the user and performed content based filtering on the same to derive better recommendation predictions. Furthermore, given a book id is liked or rated in the past by the user, we generate new recommendations based on these ratings. We used BM25 to rank other books in the dataset which are similar to this book id. Ranking is based on similar authors as well as their respective ratings in the given genre. Following this, the user can get revised recommendation based on his preference. For this purpose, we have assigned weights to each of the three features selected. Initially each feature has an equal weight. Then, the user can increase or decrease the weight of a feature so as to assign it an important value. For example, if the weight of "genre" is highest, the user expects to see books based on his his past likes in this genre. So we limit our search to books of the users favorite genre and limit our search within those books. Following this, we try to predict which books within these subsets best match the user's interest.

We used TfidfVectorizer function from scikit-learn, which transforms text to feature vectors that can be used as input to estimator. Cosine Similarity to calculate a numeric value that denotes the similarity between two books.

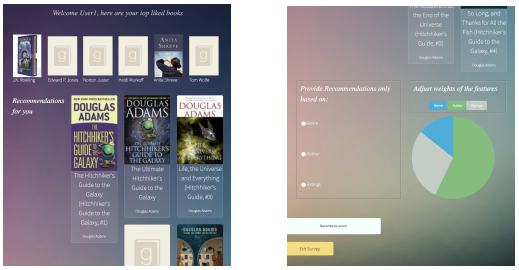
The hybrid algorithm uses two matrices: the book x user matrix, where $cell_{i,j}$ contains the rating that $user_j$ gave to $book_i$; and the author x user matrix, where $cell_{i,j}$ contains the average of ratings that $user_j$ gave to books of $author_i$. These matrices are used to calculate both book x book and author x author matrices where the $cell_{i,j}$ contains similarity between items i and j. Similarity is calculated using item co-occurrences and cosine similarities. Our system uses the active user preference vector and the book x book/author x author matrix to yield the book/author rank vector. Position i of the rank vector contains the rank predicted for $book_i/author_i$ according to the active user preferences. The author rank vector is then expanded into a book rank vector by assigning to each book its author predicted rank. Rank vectors are calculated by aggregating the columns of the similarity matrices corresponding to the items preferred by the active user.

In this way, depending on the order of the weights, our system will generate further recommendations based on what feature is important to the user.

5.2 User Interface and Visualizations

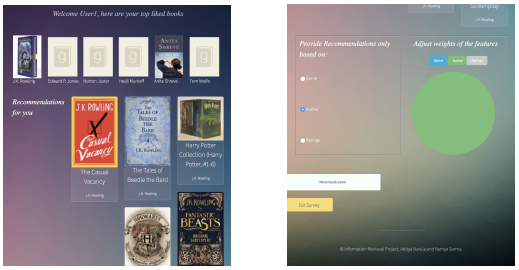
On the UI front, we used a lightweight web-app design. The webpage was created using HTML and CSS. For the visualizations, we found that JavaScript with D3 and it's associated libraries would provide an ideal framework to incorporate dynamic changes to the UI. Due to time and resource constraints, collecting and analysing implicit feedback data like transactional log analysis, eye tracking data was found to be infeasible.

6. EVALUATION



(a) Part 1 (b) Part 2

Figure 1: Screenshot showing Initial recommendations

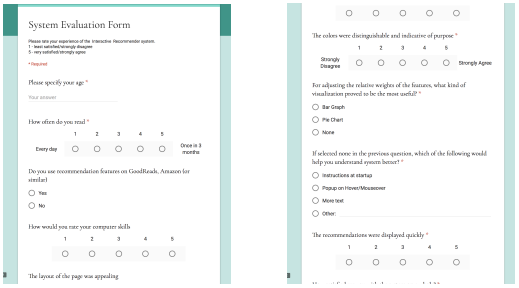


(a) Part 1 (b) Part 2

Figure 2: Screenshot showing revised recommendation based on user's preference

Evaluation of recommendation systems is difficult since we do not have a set collection of results to verify our system generated results with. The ideal metrics to evaluate such a system would be to see how it adds value to the end user. One of the ways is through user studies. We created a survey which the user can voluntarily participate in before leaving the application.

The survey rates the user experience on a linear scale. Some of the questions we think are pertinent and will help us design a better system involve asking users about the most visualization, the layout of results and how the speed of the retrieval was.



(a) Part 1 (b) Part 2

Figure 3: Screenshot of the evaluation survey form

7. FUTURE WORK AND DIRECTION

The role of visualization in Information retrieval has not been explored much, hence this remains an interesting area to explore. One of primary future work with respect is testing the efficiency of this system by carrying out some user

testing. We need the feedback to understand how comfortable users are using this type of a system, do they require instructions to use the system to its full capabilities and what visualizations do they prefer to interact with. far as our system was considered, the web-app design based design, though easy to use without installation or any other hassles, faced a drawback of lower retrieval speeds. Considering that a recommender system must churn out results fast, we would want to work on porting the system to a dedicated server to support efficient retrieval.

8. REFERENCES

[1] S. Bostandjiev, J. O'Donovan, and T. Höllerer. Tasteweights: A visual interactive hybrid recommender system. In *Proceedings of the Sixth ACM Conference on Recommender Systems*, RecSys '12, pages 35–42, New York, NY, USA, 2012. ACM.

[2] O. Hoeber. *Information Visualization for Interactive Information Retrieval*. March 2018.

[3] F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor. *Recommender Systems Handbook*. Springer-Verlag, Berlin, Heidelberg, 1st edition, 2010.

[4] B. Xiao and I. Benbasat. E-commerce product recommendation agents: Use, characteristics, and impact. *MIS Q.*, 31(1):137–209, Mar. 2007.

APPENDIX

Prerequisites

1. Python 3.x
2. Apache Server
3. PHP 5.0
4. Pandas
5. Numpy
6. Scikit Learn

Installing

1. Install the Apache Server: *sudo apt install apache2*
2. Install PHP 5.0: *sudo apt install php-pear php-fpm php-dev php-zip php-curl php-xmlrpc php-gd php-mysql php-mbstring php-xml libapache2-mod-php*
3. Install Pandas, Numpy, SciKit-Learn *python -m pip install –user numpy scipy pandas*
4. Unzip the uploaded code folder:
tar xvzf <path to file>

Running

To start the server *sudo service apache2 restart* To view the web app open the following link http://localhost/<folder where the repository was cloned>