Book Recommendation System

SRUSHTI S N
PES2UG20CS352
srushtisn703@gmail.com

VISMAYA R
PES2UG20CS391
vismayar2811@gmail.com

SHUBHANGI SADHWANI
PES2UG20CS406
shubhangi.sadhwani@gmail.com

***Abstract*— As recommendation systems become critical in many industries, we aim to build one for the bibliophiles. The primary goal of our project is to make it feasible for the users to find books on the basis of their interest and taste.**

As the user selects a book, we recommend other book titles he might be interested in, based on various similarity factors The dataset provides features such as book title, author, unique identification number, etc. As the user selects a book, we provide other book titles he might be interested in, based on various factors, using our recommendation model.

***Index Terms*— Recommender system, Content-based Book Recommendation, Cosine Similarity.**

I. INTRODUCTION AND BACKGROUND

In recent times, it has become very difficult for a user to find items that they are interested in as the amount of resources on the internet increases exponentially. In a world where search engines can supply the user with any information and resource they need, searching manually for the next thing a user is interested in can become laborious. Also, without in-depth information about the item, it is difficult to decide if it is an object that the user would react positively to.

They direct users towards those items by using various types of information such as either user profile, ratings of similar users, etc. which can meet their needs through cutting down a large database of Information thus also saving time.

A recommendation system helps save time for the user and helps the company cater to new customers while also retaining the existing ones.

Books are the summary of human knowledge and a world of magnificent escape. While there are multiple popular and effective recommendation models for domains such as Movies, Series, etc. the models used to recommend books do not have much diversity.

There are various models to select the appropriate recommendations: Content based, Collaborative based, Knowledge based and Hybrid approach etc. which use different kinds of data on the books. This makes it easier to choose the next book to read without going through the hassle of searching for it online or asking other people for recommendations.

II. REVIEW OF LITERATURE

A. A Content-Based Movie Recommendation Using Different Feature Sets

This paper introduces a content-based movie recommendation method with implicit rating mechanism. This means that features of a movie are given as much weight as the viewing engagement of the user with that movie. First, the viewing duration of the movies are converted to ratings, then the weights are assigned for the features and the final rating predicted. Lastly, the different combinations of feature sets are compared and evaluated based on the recommendations they produce. The different permutations and combinations of the feature set are validated by comparing their Precision, Recall and F-measure.

One of the major disadvantages of this approach is that it is ineffective for providing recommendations for new users

B. Building a Book Recommender system using time based content filtering

The model focuses on a content-based system for books which takes into account the choices of all users instead of only the similar users. This aspect introduces the popularity criteria to the recommendations by fetching the links which are the most visited for a period of time. The temporal factor effect on the content based filtering is analyzed by introducing a “temporal dimension”. The temporal dimension improves the recommendations by updating the item’s counter every time it is updated as time goes on.

While, the survey results show that more users preferred diverse recommendations, the complexity it brings is not a fair trade off in many domains where similarity is preferred in the recommendations such as in e-learning and Academic research content.

C. Recommendation Systems: a review

The article provides an overview of the following recommendation systems: Collaborative Filtering (CF), Content-Based Filtering (CBF), and Hybrid Recommendation approaches. These recommendation systems have various advantages and disadvantages that this literature paper has tried to look into. We use this article to select a suitable

recommendation system for our project: "Book recommendation system" which involves analysis of behavior of systems with works like Amazon, Netflix etc. Recommendation system is chosen based on which system works better with which type of data. Content-Based Filtering is most suitable due to its ability to work well with locally similar objects and smaller data and also the fact that it doesn't necessarily need domain knowledge.

D. College Library Personalized Recommendation System Based on Hybrid Recommendation Algorithm

This paper compares the performance of different recommendation models on a dataset from the library in "Inner Mongolia University of Technology(IMUT)". This involves calculation of precision of different recommender systems and sparsity of different matrices. Despite the fact that the paper concludes that the hybrid model is accurate, it is so only in case of very large models. However, the precision of collaborative filtering model and content-based filtering model was found to be similar. Hybrid recommendation system is a combination of content-based filtering and collaborative filtering methods. Collaborative filtering under hybrid recommender uses clustering. This clustering is performed using the "K-means algorithm". However, the Hybrid recommender system is not suitable for a comparatively smaller dataset like ours.

E. Design and Implementation of Movie Recommender System Based on Graph Database

This research paper describes preference information on users which consists of explicit user feedback and implicit user feedback. It also contains different data sources to find data correlation methods that can be divided into, context-based recommendation, demographic based recommendation, collaborative filter based recommendation. The paper also describes the two types of database: NoSQL and SQL.

While the query speed was found to depend only on the number of concrete relationships, the amount of data was found to have no effect on it. Results were obtained in real time which made the model difficult to scale. The model was designed as a one-tier architecture and there was no uniform query language.

F. Book Recommendation System

The approach used in this paper involves a system that combines the features of association rule mining with aspects from content based and collaborative recommender systems in order to provide the user with effective and efficient recommendations.

Collaborative filtering collects feedback from users in the form of ratings on the objects or items. In addition, collaborative filtering also considers many parameters like quality of the book as well as content of the book by performing filtering on ratings by the other users. This recommender model also uses an associative model to provide the user with stronger recommendations than before.

The automatic learning of the embeddings act as an advantage as it doesn't require domain knowledge but the heuristics required to generate embeddings of new items is a disadvantage of the approach used in this paper.

G. Summary

While few of the papers have used either collaborative or hybrid recommender systems, we are using content based recommender systems because hybrid models are only effective when used for a very large dataset and collaborative filtering suffers from 'cold-start problem'.

However, it is easier to scale-up to a large number of users in the case of a content based recommender system if one has appropriate amount of relevant domain knowledge.

The system personalizes the recommendations to each user based on the user profile generated, therefore capturing the preferences of that user with respect to various item attributes. This enables the system to suggest unpopular items which may interest the user. To facilitate the problem where this is not a lot of data available on a user, the system uses item ratings.

III. PROPOSED SOLUTION

The problem statement involves Recommending top 5 most similar books to the book the user has selected, based on similarity between various attributes of the book and the ratings provided.

While choosing our recommendation model, we consider the fact that our dataset contains a significant amount of attribute information as opposed to user rating data and also that our attributes in our dataset are text-rich, so a content-based system is particularly well suited.

If a user likes a particular book, there is a very high chance that he would like other books which are also under the same genre, and therefore books in the same category act as accurate recommendations. To consider this factor, in our model, as the user selects a book, we fetch the category of the book and then filter out the dataset to reduce it to the ones with the same category as the selected book.

In the Movie Recommendation system from paper [1] that was part of our literature survey, it was found that 'Director' is an important attribute contributing to the content-based filtering as users who like a movie were found to react positively to other movies by the same director. An equivalent attribute to 'Director' in books may be considered as 'Author' but it is known that it does not play a major role in selecting a book to read unless the book is part of a series. So, instead of a filtering based on 'Author' attribute, we merge the column with the 'Description' column, and then use the newly

combined column ‘keywords’ for measuring the similarity between the books and to compute recommendations.

A. Dataset

‘7k Books’ is a books dataset scraped from the web using the ‘Goodreads API’ and the ‘Google Books API’. It is a free public dataset hosted on Kaggle. It provides a comprehensive list of about 6810 books listed in goodreads. It contains a total of 12 columns including identifiers, title, subtitle, authors, categories, thumbnail url, description, number of pages, published year, average rating, and ratings count.

The International Standard Book Number stands for a number that uniquely identifies books and book-like products worldwide. The dataset contains both ‘isbn10’ and ‘isbn13’ attributes. ‘isbn13’ was introduced in 2007 to combat the increasing demand worldwide.

The ‘authors’ column contains 98% of unique values, 1% of null values and 1% values as ‘Agatha Christie’. This is evident to the fact that she is the best-selling author of all time. Multiple authors of the same book are separated by a ‘;’. In addition, since there are not many repeated authors, we have decided not to use ‘authors’ attribute as a filtering measure.

The ‘title’ column had 6398 unique values while the ‘subtitle’ column contained 65% of null values, 3% of ‘A Novel’ and the rest 32% was found to be other unique subtitles.

The ‘average_rating’ attribute offers valuable insight into how popular the book is, and how an average book reader would react to the book. The ‘ratings_count’ attribute is important as it ensures that there is no bias in the average rating of the book due to a few people having similar opinions rating it similarly.

B. Data Preprocessing

After the analysis of the missing data, it was found that the ‘subtitle’ attribute contained more than 65% of data missing, and therefore the attribute was dropped from the data.

The numerical columns such as ‘average_rating’ and ‘rating_count’ contained 0.63% of null values. Mean values were used to impute such instances.

The attributes ‘authors’, ‘categories’ and ‘description’ contained missing values of below 5%. These instances were entirely dropped from the dataset.

Also, as part of the Dimensionality reduction, attributes which were not contributing to the recommendation such as ‘thumbnail’, ‘published_year’, ‘num_pages’ are dropped.

C. Fitting the Models

The attributes we use for the recommendation model are: ‘title’ , ‘categories’ , ‘description’ and ‘authors’.

To prepare the data for the model fitting, we used multiple feature representation and cleaning methods. We used the libraries nltk(Natural Language Toolkit) and sklearn in python along with the string library for the same.

We start by removing the non-ASCII characters, punctuations and HTML codes from the 3 attributes which are used for the content-filtering. Words such as “a,” “an,” and “the” will not be specific to a particular book instance. as they are commonly part of the English vocabulary. In general, stopwords commonly include articles, prepositions, conjunctions, and pronouns. To retrieve the stopwords for the English language, we used the nltk.corpus function and then removed them from the ‘description’ and ‘categories’ attributes. The next step was to remove extra spaces from the ‘description’ and ‘categories’ columns.

The ‘authors’ column contains spaces between the first name and last name. This can be misleading because ‘John’ from ‘John Saul’ and from ‘John Huey’ are different people. Thus, we concatenate the authors’ name,removing the space in-between.

Weighted rating:

The ‘average_rating’ provides the ranking of the books and the comparative assessment of their quality by different users. The ‘ratings_count’ provides a validation of the accuracy of the ‘average_rating’ attribute. The simple logic being that if more people have rated a book, the more accurate its average rating is found to be. To combine these into an effective equivalent attribute, we use the weighted average.

$$W = \frac{Rv + Cm}{v + m}$$

- *W=Weighted rating*
- *R=Average ratings*
- *C=Mean of the average ratings*
- *m=Minimum number of ratings*
- *v=ratings count*

We created a new column ‘weighted_rating’ which uses the above mentioned formula to combine the 2 column values. We finished the feature representation by concatenating the ‘description’ and the ‘authors’ columns to a new column called ‘keywords’.

On executing the above steps, the keywords attribute is converted to a vector-space representation using the TfidfVectorizer method. TF-IDF ,a numerical statistic, is used to compare the importance of each word to a document in a corpus. Term Frequency-Inverse Document Frequency is often used as a weighting factor in text mining, searches of information retrieval and user modeling.

We compute the TFIDF matrix using the fit_transform function based on the ‘keyword’ column.

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

tf_{ij} = number of occurrences of i in j
df_i = number of documents containing i
N = total number of documents

Cosine similarity is a metric used in data science to help determine the similarity between the data objects irrespective of their size. This is also a reason why we decided not to use euclidean distance. We used the built-in cosine_similarity function from the 'sklearn' library and computed the similarity scores from the vectorized attribute values.

Before calling the recommender function, we change the numerical sequential indexing of the dataframe to 'title' based indexing.

The recommender function is called each time a user inputs a book title to receive recommendations. The data frame is filtered based on the 'category' of the selected book. The cosine similarity matrix is indexed through the selected book title and the top 10 most book instances with the highest similarity values are selected.

Now, the weighted average rating of the top 10 books are fetched and the book instances are sorted in decreasing order based on the value of the weighted rating. The 5 instances with the highest weighted rating are selected. Hence, these become the top 5 book recommendations by the system.

IV. RESULTS AND INFERENCES

There were certain assumptions made during the course of the model building. We assumed that despite the dataset being unbalanced with respect to the 'categories' attribute, it won't affect the recommendations as the 'categories' attribute is only used for filtering the books dataframe into the ones with the category value matching with the book selected by the user.

On the basis of the book selected by the user, the recommendation model finds books within the same category thereby catering to the user's interests and tastes. The books are then recommended based on the similar keywords of the book, which combines the keywords extracted from the 'description' and the concatenated author name. These filtered similar books are then sorted on the basis of the weighted ratings.

Thus, our model ensures that the recommendations given to the users are both popularly well rated and contains similar describing keywords as the selected book, thus ensuring that the user's interest is captured.

As the recommendations are sorted based on similarity, this means that out of the five recommended books, the first book is the most similar to the book read by the user. The second recommended book will be less similar to the book read by the user as compared to the first recommended book but still more than the third book and so on.. In addition,the first recommended book also has the highest weighted rating in comparison to the rest of the books making it the best match for the user.

So, as we go down, the similarity of recommended books with the book selected by the user will keep decreasing and so will the weighted average rating.

Model Result 1:

```
recommend("Murder on the Orient Express")

515     absent in the spring and other novels
2370                 the body in the library
2377                                 prophet
2369                murder at the vicarage
2374                     death on the nile
2373                 a murder is announced
Name: title, dtype: object
```

Picture above demonstrates the working of the recommendation system for the book: 'Murder on the Orient Express'.

While we expect our model to work accurately on all cases, we did discover a few drawbacks. For example, when recommending a book based on "Harry Potter", all the recommendations would be other "Harry Potter" books. This is because 'Harry Potter' is a series and thus it is actually good to recommend the next book or the other books in the series if they've selected one. But this becomes a problem if the user is done reading the series and wants to explore something which is new to him but contains an overall similarity to the series. But according to the system we've trained, the books most similar to any 'Harry Potter' book will be the other Harry Potter books. This is a problem because the system does not recommend books that satisfy the user's needs in this case.

Model Result 2:

```
recommend("harry potter")

243                          the harry potter collection
239      harry potter and the half-blood prince (book 6)
229    harry potter and the prisoner of azkaban (book 3)
389                  harry potter and the goblet of fire
202    harry potter and the order of the phoenix (boo...
219       harry potter and the sorcerer's stone (book 1)
Name: title, dtype: object
```

The image above demonstrates the working of the recommendation system for the book: 'Harry Potter'.

Evaluation:

Evaluation of recommender systems is usually done in 3 ways: Online Evaluation, User Studies and Offline Evaluation. As we do not have already existing users, we cannot do user studies. For offline evaluation, we would need historical data of the recommendations, which were not a part of our dataset and therefore we chose to go with Online evaluation.

The Online evaluation results for this model were obtained by performing a user survey on our peers and fellow book readers. The survey was conducted to evaluate the quality of the recommender model for effectiveness based on secondary goals of the recommender systems such as Novelty, Serendipity and Diversity.

Respondents were asked to provide their favorite book or a book that they would like to obtain recommendations on. Then the model was used to get the top 5 recommendations and the results were shown to the user along with a questionnaire. The questionnaire contains questions which measure the secondary goals of our system by the user feedback.

Observed Results:

We found that 76.2% of the users who participated in the survey enjoyed reading books on a habitual basis, while 23.8% of them did not have an avid habit.

Novelty of a recommendation system is one of the primary metrics of customer satisfaction. It measures if the user gets a recommendation which they have not come across before, thus ensuring that recommendations should not prove redundant to the user.

Among the surveyed group, it was found that 76.2% of the users received a recommendation they did not know of before.

Serendipity is a recommender system evaluation criteria used for making user appealing and relevant recommendations. Serendipity suggests the "lucky discovery" aspect of a recommender system. In the survey group, it was observed that 81% of the users took interest in the books that were recommended to them.This implied that the users were able to find something that piqued their interest without looking for it.

Diversity of a recommender system is measured using similarity between the recommended books. 85.7% of the users agreed that they found variety in the books recommended to them by the recommender system. This implies that the recommendations made by the recommender system appear to be very diverse.

V. CONCLUSIONS

In conclusion, the goal of our project was to build a content-based recommender system for books using the dataset extracted from 'Goodreads'. As modern applications like kindle and google play books revolutionise the landscape of books and bring it to the mainstream again, a recommendation system configured specifically to the domain becomes crucial. The project depends deeply on domain knowledge to be able to provide more relevant recommendations which is the case in other content-based systems as well.

The model shows positive results with respect to the feasibility, trust and confidence of the users which plays a crucial role in the evaluation of the model's performance. Despite the model satisfying secondary measures, its accuracy cannot be calculated since there is no availability of historical data.

Although this model provides recommendations with good secondary evaluation measures , there is still a lot of future scope to improve its working.

With a dataset that consisted of about 7 thousand book instances, we tested various methods and approaches to build the Content-based model. Through this process, we learnt a lot about how various recommendation systems work and got an insight into how popular applications make use of them to provide us with accurate recommendations almost instantly.

VI. REFERENCES

[1] Content-Based Movie Recommendation Using Different Feature Sets- Mahiye Uluyagmur, Zehra Cataltepe and Esengul Tayfur , Proceedings of the World Congress on Engineering and Computer Science 2012 Vol I WCECS 2012, October 24-26, 2012, San Francisco, USA

http://www.iaeng.org/publication/WCECS2012/WCECS2012_pp517-521.pdf

[2] Building a Book Recommender system using time based content filtering. Chhavi Rana, Sanjay Kumar Jain. WSEAS TRANSACTIONS on COMPUTERS

http://www.wseas.us/journal/pdf/computers/2012/54-571.pdf

[3] Nabizadeh, Amir Hossein & Rafsanjani, Nabizadeh & Salim, Naomie & Rezaei Aghdam, Atae & Fard, Karamollah. (2013). Recommendation Systems: a review.

https://www.researchgate.net/publication/325074466_Recommendation_Systems_a_review

[4] Yonghong Tian, Bing Zheng, Yanfang Wang, Yue Zhang, Qi Wu, College Library Personalized Recommendation System Based on Hybrid Recommendation Algorithm

https://www.sciencedirect.com/science/article/pii/S2212827119307401

[5] Design and Implementation of Movie Recommender System Based on Graph Database . Ningning Yi; Chunfang Li; Xin Feng; Minyong Shi

https://ieeexplore.ieee.org/abstract/document/8332600

[6] BOOK RECOMMENDATION SYSTEM

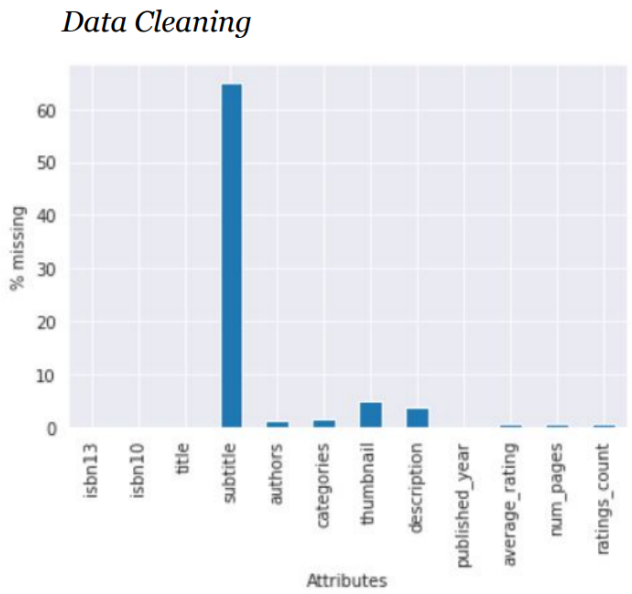
https://www.academia.edu/15972400/BOOK_RECOMMENDATION_SYSTEM

[7] Weighted-rating:

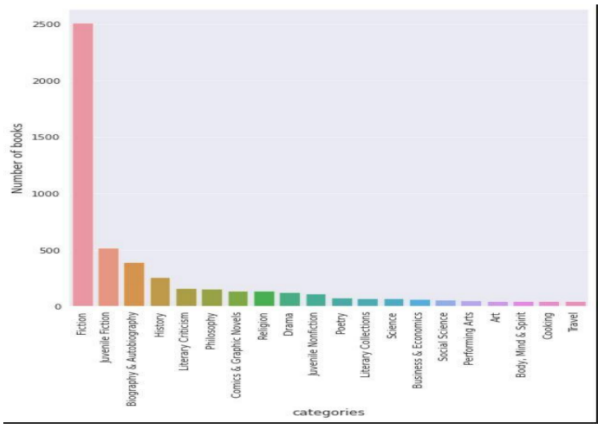
https://medium.com/@developeraritro/building-a-recommendation-system-using-weighted-hybrid-technique-75598b6be8ed

VII. APPENDIX

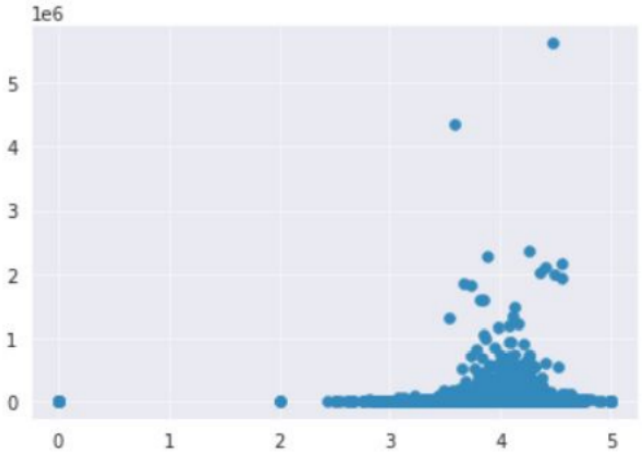
A. Data Visualizations



Bar graph between number of books and categories.



Correlation plot between number of ratings and rating count.



B. Contributions

Srushti S N : Literature survey, exploratory data analysis, calculating the similarity measure, analyzing model results.

Vismaya R: Literature survey, feature extraction and text processing, evaluation of the model.

Shubhangi Sadhwani: Literature survey, drafting the problem statement, data preprocessing and cleaning.