Book Recommender Systems

Anish Sunil
2nd year Computer Science Engineering
Student
PES University
Bengaluru, India
anishsunil01@gmail.com

Sunaina T
2nd year Computer Science Engineering
Student
PES Unversity
Bengaluru, India
tsunaina123@gmail.com

***Abstract*—Recommender systems are software applications that help users to find items of interest in situations of information overload. This paper tries to explain the basic two different types of recommender systems using books as the item of recommendation while also visiting the application of some Statistical concepts like Cosine Similarity and Pearson Correlation into the recommender models.**

Keywords—Content based Recommendation, Collaborative Filtering, Cosine Similarity, Cosine Distance, Pearson Correlation, K-Nearest Neighbors, Tf-Idf, Pivot, Sparse Matrix.

I. Introduction

Recommender Systems (RS) refer to computerized systems with the purpose of filtering out potentially items that are of little or no interest to the users. The output of the systems will be a smaller set of items that are of interest to the users. It is a form of information filtering, similar to how news editors aggregate news and present the most relevant news only. With the advent of the Internet, information overload is a challenge and recommender systems become crucial to ensure only relevant items are presented and other items are filtered out.

In this paper, we will understand the two main popular types of recommender systems: (1) Content or Item based recommendations and (2) Collaborative Filtering based recommendations. We will be executing 3 models to analyze the two types of recommendations using 'Books' as the item of recommendation.

The objective of this paper to obtain a beginner's perspective of Recommender Systems and to understand the concepts involved in some basic recommender systems and it's applications based on researches made by others regarding this.

II. Concepts Used

A. *Tf-Idf(Term frequency- Inverse document frequency)*

In information retrieval, Tf-Idf, is a numeric statistic that is intended to reflect how important a word is to a document in a collection or corpus. It is often used a weighting factor in searches of information retrieval, text mining and user modelling. The Tf-Idf value increases proportionally to the number of times a word appears in the document and is offset by the number of documents in the corpus that contain the word, which helps to adjust for the fact that some words appear more frequently in general.

The Tf-Idf is the product of two statistics, term frequency and inverse document frequency. There are various ways of determining the exact values of both statistics.

(i) $Tf(t,d) = f(t,d)$ (Its simply the frequency of a particular word of interest.)

$f(t,d) \rightarrow$ frequency of the term 't' in a document 'd'

(ii) $Idf(t,D) = \log (N/ |\{d \in D: t \in d\}|)$

$N \rightarrow$ total no. of documents in the corpus $N = |D|$

$|\{d \in D: t \in d\}| \rightarrow$ no. of documents where the term 't' appears.

Then the Tf-Idf value is calculated as:

$Tfidf (t, t, D) = tf (t, d) \bullet idf (t, D)$

B. *Cosine Simialrity*

Cosine similarity is a measure of similarity between two non-zero vectors of an inner product space. It is defined to equal the cosine of the angle between them, which is also the same as inner product of the same vectors normalized to both have length 1.

The cosine similarity of two non-zero vectors can be represented using dot product and magnitude as:

$\text{Similarity} = \cos(\theta) = (A \bullet B)/ ||A|| \, ||B||$

C. *Cosine Distance*

Cosine distance is used a type of metric and is related to cosine similarity through this:

$\text{Cosine Distance} = D_C(A,B) = 1 - S_C(A,B)$

Where $S_C(A,B) \rightarrow$ Cosine Similarity b/w the two vectors under consideration.

D. *Pearson Correlation Coefficient*

In statistics, the Pearson correlation coefficient (PCC), or the bivariate correlation, is a statistic that measures linear correlation between two variables X and Y.

Pearson's Correlation Coefficient (ρ), when applied to a population is commonly represented for a given pair of random variables (X, Y):

$\rho (X, Y) = cov(X, Y)/ (\sigma(X) \bullet \sigma(Y))$

where cov $\rightarrow$ covariance

$\sigma(X) \rightarrow$ std. deviation of X

$\sigma(Y) \rightarrow$ std. deviation of Y

E. *K-Nearest Neighbors*

In statistics, the K-Nearest Neighbors algorithm(k-NN) is a non-parametric method used for classification and regression. In both cases, we assume a feature space consisting of k closest training examples from a testing vector/sample/point. A useful technique often used is assigning weights to the contributions of the neighbors. A common weighting scheme consists in giving each neighbor a weight of 1/d, where 'd' is the distance to the neighbor.

III. MODELS' WORKING METHODOLOGY

Recommender Systems (RS) can be divided as item based and ratings based. Here in this paper, we will implement three models in total, one in item-based category and two in ratings category. All the models are executed using Python programming language in Jupyter Notebooks.

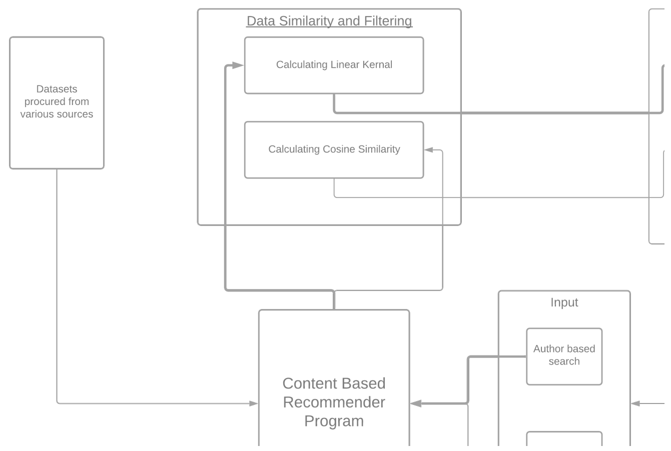
Item based (or Content based) RSs recommend items (here books) based on some similarities between the input book (or the book read or the book of interest) and 'to-be' recommended books. Ratings based (or Collaborative Filtering based) RSs recommend items (here books) based on the ratings given to these items by various users.

A. Content Based RS (Using Cosine Simialrity)

The dataset used for this model is called 'goodbooks - 10k'. Also, this model consists of two types of recommendations: (1) Author Based and (2) Content Based. The link to obtain this dataset will be mentioned in the 'References and Links' section of this paper. The model is executed as follows:

- All the necessary python libraries and functions are imported and all the csv files are taken as input. A dataframe is produced consisting of all the necessary columns by merging data from various csv files.
- Some basic data cleaning is done for proper utilization of data in the model. This process includes removing unwanted columns from the data, removing rows with NA values and removing duplicate data.
- Next the data is visualized (in the from of stacked bar and Wordclouds) to represent Top Rated Books, Top Popular Books, Top Popular Authors etc.
- Now a dataframe 'X' is produced with columns as 'original_title', 'authors', 'average_rating', 'content'. The rows under 'content are formed through string addition of data under other columns mentioned above.
- Now the data in the 'authors' column is fed into a Tf-Idf Vectorizer which first removes stop words, converts all the data into tf-idf vector matrix.
- This matrix is fed into a linear kernel function and the output of this function is fed into a cosine_sim function inside a 'get_recommendations_books' function that takes book title as an input to produce top 10 recommendations by comparing the cosine similarity values of all book-authors taken with respect to the input book-author. These top 10 recommendations are displayed using another function called 'author_book_shows. This is an author-based recommendation using cosine similarity.
- Now the data under 'contents' column of dataframe 'X' is fed into a CountVectorizer function to produce a vector matrix. (**Note**: CountVectorizer counts the number of times a word appears in the document which results in biasing in favour of most frequent words, but this ignores rare words which maybe needed for further processing. So, to overcome this issue, we can use TfidfVectorizer where we consider the overall weightage of the word and assign it a value.)
- This matrix is fed into a cosine_sim function (which is inside a 'get_recommendatons' function which takes a certain input book) to produce recommendations based on comparison of cosine similarity values of all the books-contents taken with respect to the book-content of the input. This is the content-based recommendation using cosine similarity.

Here's an architecture diagram to represent our model and summarize the methodology in a pictorial format:


Data Similarity and Filtering
Calculating Linear Kernal
Calculating Cosine Similarity
Datasets procured from various sources
Input
Author based search
Content Based Recommender Program

For the Jupyter Notebook with the complete working code, check the 'References and Links' section for the GitHub link.

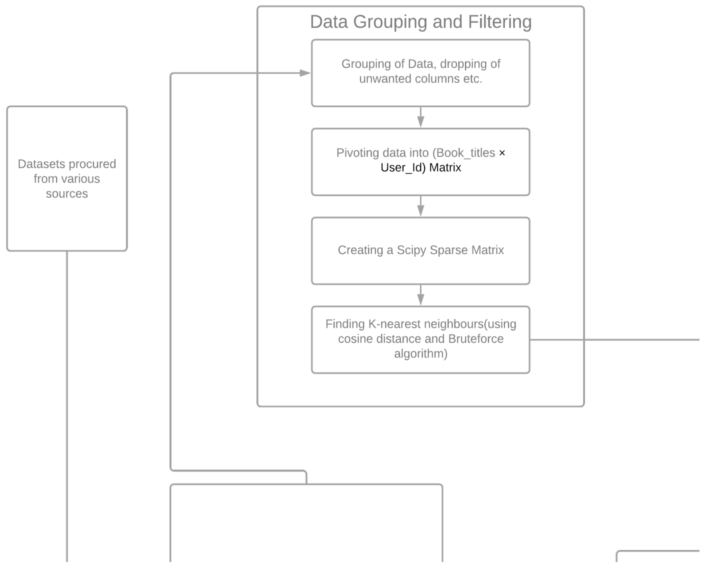
B. K-Nearest Neighbours based Collaborative Filtering RS

The dataset for this model is procured from 'Books-Crossings'. This is a ratings-based recommender model. The link to obtain this dataset will be mentioned in the 'References and Links' section of this paper. The model is executed as follows:

- All the necessary python libraries and functions are imported and all the csv files are taken as input. A dataframe is produced consisting of all the necessary columns by merging data from various csv files.

- The top 5 rated books were visualized in the form of a table. Some basic data cleaning is done for proper utilization of data in the model. This process includes removing unnecessary columns and dropping duplicates.
- Now a dataframe is created consisting of combined book ratings with each book having a total ratings count. By statistically analyzing the 'total_ratings_count' dataframe, we find that the book count reaches nearly 2.5 lakh with a max rating count of 2.5k and mean ratings count of 4.3. Also, by observing the distribution of ratings in some top 'X' percentage of books, we find that the top 1% of the books have a total_ratings_count of 50 or more. This means that only the top 1% of the data regarding books and their ratings is statistically significant. So now a popularity threshold of value 50 is chosen to filter out the top 1% of the books for further processing.
- Now the dataset procured from Books-Crossings also contains the locations of the users who have rated the books. In order to reduce the runtime of the filtering process, we restrict the ratings to only those given by users in USA, Canada, UK and Australia because these are the countries from where most of the ratings have occurred. This is done to also ensure the statistical significance of the filtering.
- Now we create a dataframe which consists of user_id, isbn, rating, book_title, img_l (book cover image URL), totalRatingCount and location as features or columns for each of the filetered books. Let's call this dataframe as 'X'.
- Now we pivot this dataframe to a 2-D matrix consisting of "book title' as rows and 'user_id' as columns. Also, all the NA values are replaced with '0'.
- We run this matrix through a csr_matrix function which returns a Scipy sparse matrix which doesn't consist of any '0' valued cell.
- This matrix is fed to NearestNeighbors function which calculates the k-nearest neighbors of a specified book based on cosine distance. The Brute force algorithm is used to filter out the top k nearest neighbors to the specified book.

Here's an architecture diagram to represent our model and summarize the methodology in a pictorial format:


Data Grouping and Filtering
Grouping of Data, dropping of unwanted columns etc.
Pivoting data into (Book_titles × User_Id) Matrix
Creating a Scipy Sparse Matrix
Finding K-nearest neighbours(using cosine distance and Bruteforce algorithm)
Datasets procured from various sources

For the Jupyter Notebook with the complete working code, check the 'References and Links' section for the GitHub link.

C. Pearson Correlation Coefficient based Collaborative Filtering RS

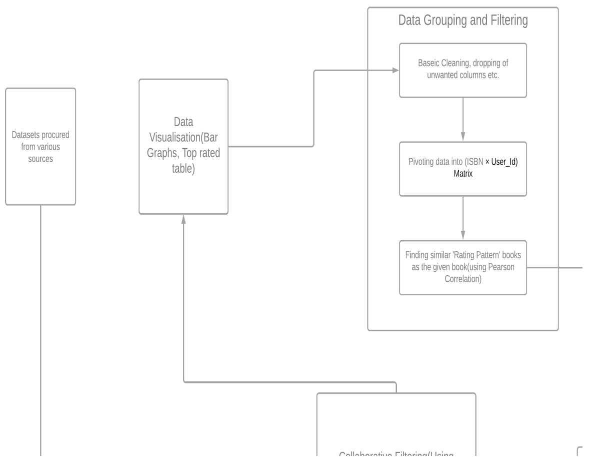
The dataset for this model is procured from 'Books-Crossings'. This is a ratings-based recommender model. The link to obtain this dataset will be mentioned in the 'References and Links' section of this paper. The model is executed as follows:

- All the necessary python libraries and functions are imported and all the csv files are taken as input. A dataframe is produced consisting of all the necessary columns by merging data from various csv files.
- The data procured from various csv files are visualized to represent ratings distribution and most rated books. We learn from the Ratings distribution bar graph that most of the ratings in the dataset are 'null'.
- We have considered and filtered out only books with more than 100 ratings and users who have given ratings for more than 200 books, for statistical significance. We have come up with the above mentioned numbers after some statistical analysis. We now create a dataframe consisting of necessary columns.
- Now we create a pivot 2-D matrix with 'userID' as the rows and 'ISBN' as the columns. The matrix will be sparse because not every user has rated every book.
- We then use the function 'corrwith ()' with the input book(here we used the ISBN of the book as the input) and the pivot matrix(from the previous step) to calculate the pearson coefficients for each and every book. We then

filtered out the books which have a non-real pearson coefficient.

- We merge the above produced dataframe with the rating count (number of users who have rated the book) for the corresponding books. We filter out the books that have a count of less than 300 and arrange the dataframe in the descending order of their coefficients.
- The top 10 books (other than itself) are given as the output for recommendation.

Here's an architecture diagram to represent our model and summarize the methodology in pictorial format:


Datasets procured from various sources
Data Visualisation(Bar Graphs, Top rated table)
Data Grouping and Filtering
Baseic Cleaning, dropping of unwanted columns etc.
Pivoting data into (ISBN × User_Id) Matrix
Finding similar 'Rating Pattern' books as the given book(using Pearson Correlation)

For the Jupyter Notebook with the complete working code, check the 'References and Links' section for the GitHub link.

IV. Observation and Conclusion

Disclaimer: The below observations and conclusions are made with respect to researches from different people.

Collaborative algorithm uses "User Behavior" for recommending items. They exploit behavior of other users and items in terms of transaction history, ratings, selection and purchase information. Other user's behavior and preferences over the items are used to recommend items to the new users. It has the limitations of Cold-start problem where a recommender does not have the adequate information about a user or an item to make relevant predictions. Data Sparsity is the problem that occurs as a result of lack of enough information, that is, when only a few of the total number of items available in a database are rated by users. This leads to a sparse user-item matrix, inability to locate successful neighbors resulting in the generation of weak recommendations [3][6].

In content-based filtering we have to know the content of both user and item. Usually we construct user-profile and item-profile using the content of shared attribute space. Here, we have text description about the product. It has the limitations of advocating the same types of items because of which it suffers from an overspecialization problem. It is harder to acquire feedback from users in CBF because users do not typically rank the items (as in CF) and therefore, it is not possible to determine whether the recommendation is correct[4][5].

We conclude by saying that we have an understanding of the nature of the different models with its advantages and drawbacks. In this paper we discuss about 3 models: one model on content based and two models on collaborative filtering.

V. Acknowledgment

We would like to thank our guide Prof. Sandesh.B.J for guiding us and helping us walk into uncharted parts of Machine Learning and Data Analysis and apply them into working models for better understanding of the learned concepts. We would also like to offer our gratitude to Ms. Vandana and Ms. Pooja for judging our works in this project and offering their valuable insight and helpful criticism (where necessary) into bettering our project and experience of learning.

VI. References and Links

[1] Jia-Ming Low, Ian K.T.Tan and Choo-Yee Ting "Recent Developments in Recommender Systems" .

[2] Costas Vassilakis and Dionisis Margaris. Editorial on "Moden Recommender Systems: Approaches, Challenges and Applications"

[3] P. Aggarwal, V. Tomar, A. Kathuria, "Comparing Content Based and Collaborative Filtering in Recommender Systems", International Journal of New Technology and Research (2017).

[4] . P. B. Thorat, R. M. Goudar, S. Barve, " Survey on Collaborative Filtering, Content-based Filtering and Hybrid Recommendation System", International Journal of Computer Applications (2015).

[5] Michael J. Pazzani1and Daniel Billsus2, "Content-based Recommendation Systems"

[6] Atisha Sachan and Vineet Richariya, "A Survey on Recommender Systems based on Collaborative Filtering Technique"

[7] Prem Melville and Vikas Sindhwani, "Recommender Systems"

[8] Swaleha Zubair , MuaadhAbdo Al Sabri, Afreen Khan, "Correlation among similarity measurements for collaborative filtering techniques: an improved similarity metric"

[9] Haifeng Liu , Zheng Hu , Ahmad Mian, Hui Tian, XuzhenZhu , "A new user similarity model to improve the accuracy of collaborative filtering"

Links:

(1) Link for good-reads-10k dataset: https://github.com/Sage101201/Book-Recommender-Systems-/tree/Good-reads-10k-dataset

(2) Link for the three recommender models: https://github.com/Sage101201/Book-Recommender-Systems-/tree/main

(3) Link for Books-Crossings dataset: https://github.com/Sage101201/Book-Recommender-Systems-/tree/Books-Crossings-dataset