**Book Recommendation System**

Naga Abhilash Reddy Julakanti

Computer Engineering Department, San Jose State University, CA

1. **Introduction:**

From the past decade we can see the sharp decline in number of users who reads books and books are major source of knowledge. So, in this system, we are building a recommendation model that recommends books to the users to infuse interest in the users and help them choose the book that they love to read. The data for this system is collected by web scraping from Goodreads website. This system recommends books for the user based on his previous ratings and certain measure of similarity between the books he rated and the books that he hasn’t read but rated by others. To get data from the website I have used a scraping technique which scrape through HTML script of the page and get the required data from it. I used both content-based and Item – based collaborative filtering techniques to get the recommendations for the given book and recommendations for the user based on his previous ratings.

1. **Model workflow:**

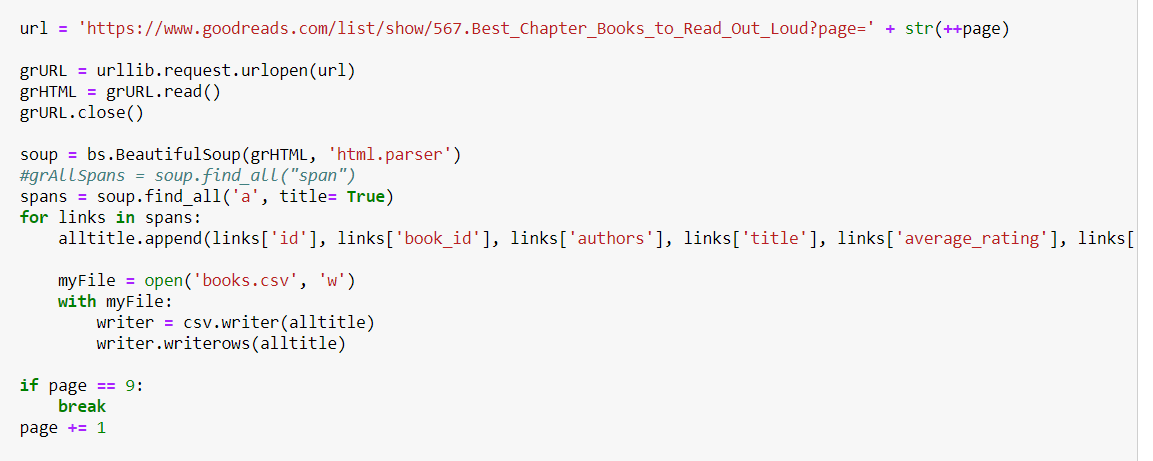
For this system to recommend books properly, we need lots of data and “goodreads.com” is one such kind of website that provides lot of information about the books, users, ratings, tags and many more. So, first I used scrapping technique to scrape data from goodreads.com website. We scrape data in three different ways. One to get data about the books, second to get all types of genres that are available in the website and last to get user ratings for a given book. The collected data from scraping techniques has unnecessary data and need to be cleaned before passing it to recommendation systems.

Once, the data is ready to be processed, I have created a user and added random data to get recommendations for this user. This data is then sent to content-based filtering technique and also to Item – based collaborative filtering to find the recommendations for the user that we created.

1. **Data Acquisition:**

Data Acquisition is an important part of recommendation system. The larger the data, the better the model predictions would be and the so does the recommendations. To acquire data, we are using Beautiful Soup 4(bs4) and Urllib packages in python. First, we provide link to first page of books and the scrapper will go through the HTML script of the page and get the necessary details.

We are scraping details about books, genres and ratings given by users. In each and every scraping technique, we provide the links to the code and the rest is taken care by the scrapper. If we look at Fig 1,we can see that the code runs through each page till it reaches the end of all pages and scrapes titles, ID’s, ISBN and other required attributes of books and this data is stored in books.csv file. Similarly, we scrape data for Genre’s, tags, ratings and book - tags.

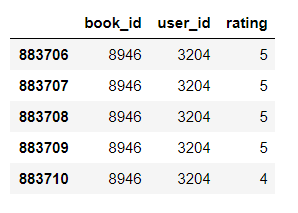


**Fig :** Code to scrape books data

Using the scrapping technique, we have collected data for about 10,000 books which have 981,756 ratings given by 53,424 users. Also, these books have more than 34,000 unique tags. Now let us go to data cleaning and data preprocessing steps before proceeding to recommender models.

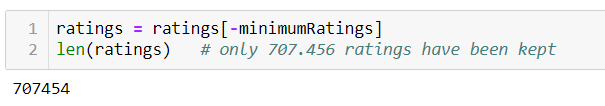
1. **Data Cleaning and Preprocessing:**

After web scraping, the data is not clean and consists of lot of unnecessary values like user comments, Reviews which will come with the required data and needs to be removed. I have added my personal ratings to the account to see how the recommender system works for a user. Also, I have seen that most of the users rated same book multiple times and we have to remove the duplicate data in order to predict the recommendations correctly. To handle them, we are averaging all the ratings for the book by particular user and replace one rating with mean and remove the other duplicates.

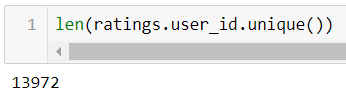


**Fig :** Duplicate ratings by user3204 for book id 8946

Apart from that we also have users who have rated as few as one book. This creates data sparsity problem and to solve this, we are deleting the users who have rated less than 20 books. This will sole the data sparsity problem which in turn helps us to improve the efficiency of the model.



**Fig :** No of ratings after removing users with < 20 ratings

****

**Fig :** Users after deleting users with < 20 ratings.

After handling users with few ratings, now we have to handle books with few ratings. Few books have as few as 1 rating and we need to remove such kind of books or they can’t be recommended to others. After removing books that have few ratings, we are let with around 8,800 books,13,971 users and around 685,000 ratings and we are ready to enter recommender stage.

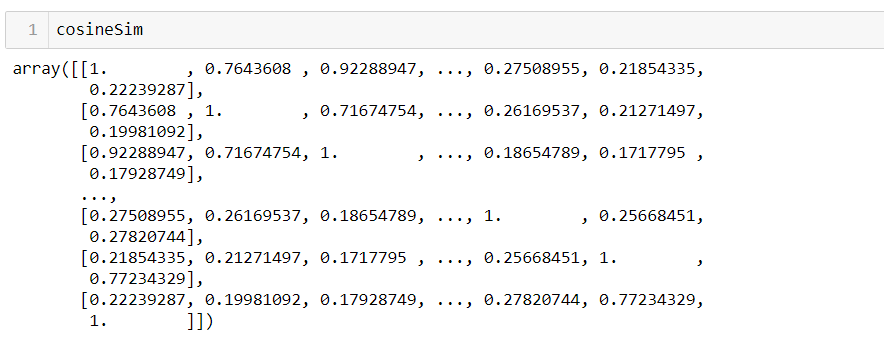
1. **Content Based Recommendation Systems:**

I have used the genres.csv files, which was generated by scraping Goodreads website, and it is shown in the other notebook. Then I have merged the bookTags dataset with the genreTags, so that now I have the information of what books in the dataset have been tagged with at least one of the 832 genre-tags used by Goodreads in their genres section.

The idea is then to measure the similarity between each book pair, so that the system will be able to recommend Top-N similar books for any selected title from the ones included in the analysis. In order to be able to properly perform content - based filtering, first I have to put all the tags related to one single book in a single string: each tag must be separated from each other with a space, to be accounted as a single word. Then I will perform pairwise similarity scores between books.

Other than the genre-tags, a very important information that we can add to the stringed tag of each book is the author. Including the author will produce a higher similarity score between 2 books written by the same author, which seems reasonable a as someone who likes a book from an author is most likely going to like also other books from the same person. Now I will calculate the cosine similarity between the books.

The next step is retrieving the titles of the books (from the 'stringedTags' dataset, column 'title') and use them as index, so that I can create a function, called 'topRecommendations' that takes in a book title and scores all cosine similarity of that title with each book in the dataset; then - after sorting by the highest similarity and after dropping the first observation (that will show the similarity of the book with itself, which is the maximum value) it grabs only the first 10 most similar books to that title and return them in the form of a top-10 recommendation.

****

**Fig :** Cosine similarity of books

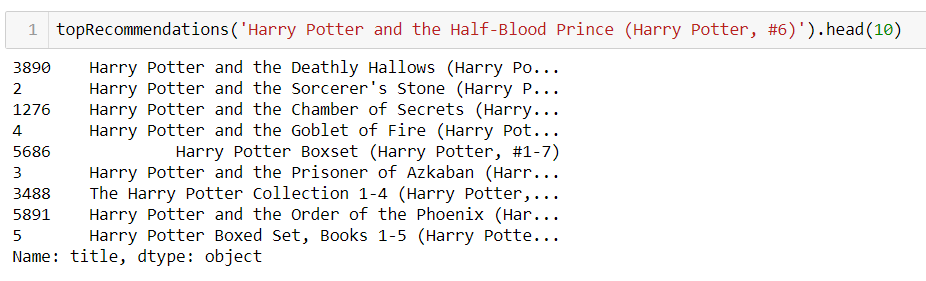
1. **Item based Collaborative Filtering:**

Another type of recommender that I have decided to implement is based on 'Collaborative Filtering' technique. In this case, I have considered an Item-Based Collaborative Filtering. The starting point of this type of system is a user by book matrix, in which each row indicates a single user, and each column represents a single book. With the intent of making the table easier to read and displaying the book titles rather than the book ids, I have used the 'ratingsWithTitles' table.

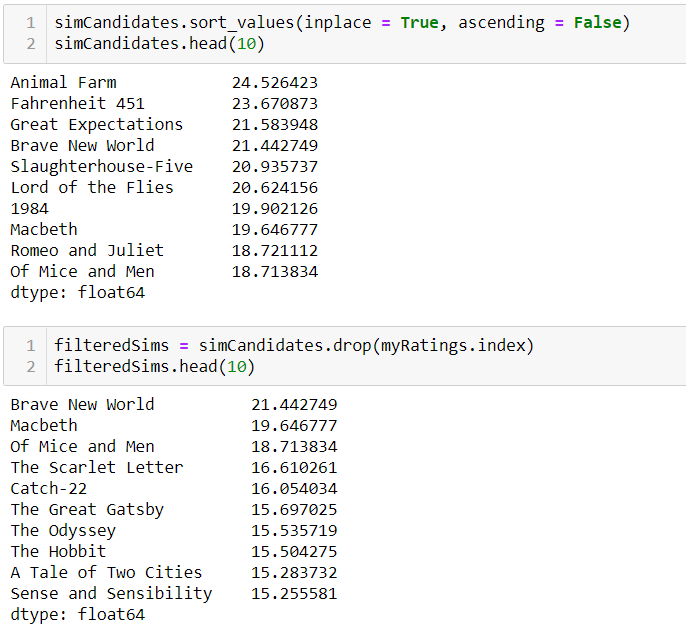
The resulting userByBook table contains an extremely high number of NAs, due to the fact that - as seen before - even the users who have read the highest number of books in our dataset, they have just read roughly 1% of all the books in the analysis. The goal in item-based collaborative filtering is to score the similarity between book pairs, based on the ratings they have received by the pool of users. To obtain a matrix in which each rows and columns will represent a single book and the values in the matrix represent the similarity between them, I have used the corr() function, which computes pairwise correlation of columns, excluding NA values, using Pearson as default method.

1. **Results:**

Now let us see the results of content - based and item - based collaborative filtering.



**Fig :** Content based recommendations.



**Fig :** Item-based collaborative filtering recommendations

1. **Conclusion:**

Finally, we have built a recommender system that recommends books to user based on his previous ratings. We have scraped data from goodreads.com and used cleaned it by removing duplicates and then preprocessed it with different parameters. The content based filtering is different from collaborative as we need to provide book title in content based and it recommends based only on what user likes. Whereas, collaborative filtering is more intelligent and recommends books based on what user likes and also combines it with other data which user have not rated yet but his peers rated. The results for both the models were printed above and can look at the observations clearly.

1. **References:**
2. <https://www.blackhatworld.com/seo/how-to-scrape-goodreads.1084349/> - To Scrape data from good reads
3. <https://www.goodreads.com/list/show/567.Best_Chapter_Books_to_Read_Out_Loud?page=1>
4. <https://www.goodreads.com/>
5. <https://towardsdatascience.com/my-journey-to-building-book-recommendation-system-5ec959c41847>