# AI Cap. Final Project

**Team 2, Member** 黃芷柔，李宜蓁，劉詠

# Goal

In this project, we implement a book recommendation system using two methods: user-based recommendation and content-based recommendation. We will evaluate and compare the effectiveness of each method to determine which one performs better.

# Motivation

Nowadays, many social media and shopping platforms use recommendation systems. We are undertaking this project due to our curiosity about recommendation algorithms. Through this project, we aim to deepen our understanding and become more familiar with the workings of recommendation systems.

# Dataset

Dataset: goodbooks_10k_rating_and_description

The dataset is about books with user ratings and book descriptions and is sourced from the Kaggle. Following is the introduction of the columns we will use in this project:

**goodbooks_10k_rating_and_description.csv**

- book_id: the index of each book
- book_title: the title of the book
- book_desc: the description of the book

**ratings.csv**

- user_id: the index of each user
- book_id: the book the user has rated
- ratings: the ratings user gives to the books

# Methods and Techniques

## User-based

In this method, we will implement user based collaborative filtering with cosine similarity.

First, we will need to create a user-item matrix, where each row represents the users, and each column represents the books. However, we

can’t directly create such a matrix from the dataset, since the matrix will be too great and our cpu can’t afford it. Therefore, we will need to preprocess our data at the beginning.

Since in the dataset “rating”, each user has the rating of the books that he/she has read, and in the dataset “goodbooks_10k_rating_and_description”, we have more important features that we need.

Combine the features in datasets above to get the data frame with user id, book id, users’ ratings of the books, books’ titles. And discard the books with fewer comments, later create another data frame,”user_book_df”, with index of user id and columns of books title , and the values in each column are the ratings that users gave that book.

Next we select a random user who is going to be recommended books for, first find the user’s read books. Split those read books into two parts in if we need do evaluation later. Use the titles of the read books to extract the ratings of the books of the data frame “user_book_df”, and save the result in “books_read_df”. Count the numbers without null value for each user in “books_read_df”.

The numbers count in the above represents the number of books read by each user that are the same as the random user. We define a particular ratio and percentage, if the number of a user is higher than that percentage, the user will be considered the one that has similar behavior as the random selected user.

Since we have known similar behavior users, withdraw all the rating data of those users, use the Cosine Similarity to compute similarity between those users based on their rating values of “books_read_df”. Then we will obtain the users that are correlated with the selected random user based on the Cosine Similarity score, note that the higher the better.

For those filtered users, we further use the score obtained by the Cosine Similarity to multiply the ratings of all the books that were read by those filtered users. Therefore, each book will have many values that come from different users. We calculate the mean of those values as the weighted rating of that particular book.

Finally we choose a threshold and use the weighted rating obtained to select the books with value that surpass that threshold as recommended books for the selected random user.

# Content-based

We analyzed whether the content of the books are similar by checking the column "book_desc_tags_FE", which is an introduction of every book in nearly 300 words. Then we use TF-IDF(term frequency - inverse document frequency) to vectorize the description of all books. TD-IDF is a measure of the importance

of a word, which is dependent on TD and IDF. TF is the frequency of a term in the document. If the term appears in the document more frequently, TD becomes larger, that means the word is more important. IDF is the frequency of this term in all documents**.** If the term appears in many documents, that means this term may be a commonly used word (such as “I”, ”the”), so IDF becomes smaller. We can get TD-IDF from this equation.

$$\mathrm{tf}(t,d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}},$$

$f_{t,d}$ : the raw count of a term in a document

$$\mathrm{idf}(t,D) = \log \frac{N}{|\{d : d \in D \text{ and } t \in d\}|}$$

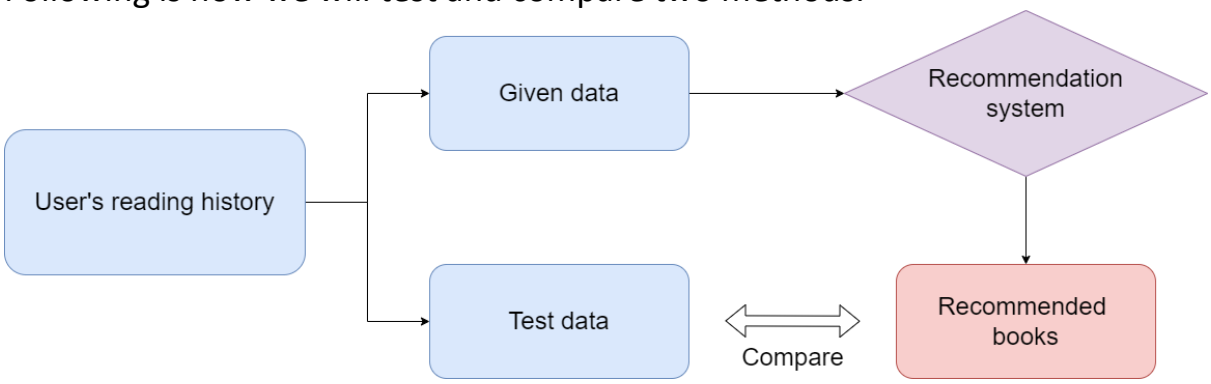
$N$: total number of documents in the corpus $N = |D|$
$|\{d: d \in D \ and \ t \in d\}|$: number of documents where the term $t$ appear (i.e. $tf(t,d) \neq 0$)

$$\mathrm{tfidf}(t,d,D) = \mathrm{tf}(t,d) \cdot \mathrm{idf}(t,D)$$

Then, we apply cosine similarity to produce a similarity matrix with the vectorized data. This matrix allows us to look up the relationships between books. To recommend books, we choose a book, find the data related to this book in the similarity matrix and extract them. Next, we sort the data by similarity, make the books with higher similarity in front of the list. We can get the five books closest to the selected book.

# Testing & Result

Following is how we will test and compare two methods.




We wanted to know if the books we recommend match the readers' preferences, so we analyzed our recommendation system by examining the users' read history. We chose some random users, divided the read history of each user into 2 parts. We put one part into the recommendation system to produce recommended books. Then we checked whether these books are in the

other part of the user's read history. If there are books matching, that means the user may be interested in what we recommended.

In a user-based system, we split the users' read history into two parts to put one part in the recommendation system directly. In a content-based system, since the unit of analysis is a single book, we analyze the viewing records book by book for recommendations. If there are duplicates, they are removed, and the number of recommended books is adjusted to match the quantity in the user-based system.

We let: $Matching\ rate = \frac{The\ number\ of\ matching\ books}{Total\ test\ data\ of\ an\ user}$

while the scale of recommended books is same in two methods (almost 500 books while the number of total books is 10000)

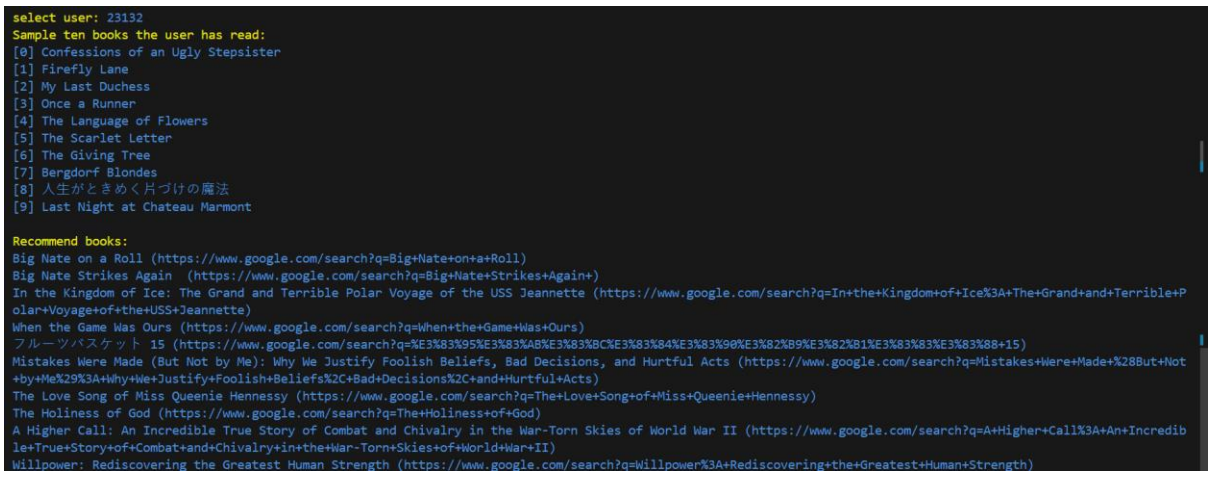
Result of 30 random users:

| | User-based | Content-based |
|---|---|---|
| Average of matching rate | 33.28% | 59.88% |

We can find that the content-based model overperforms the user-based model. We speculate that this is because the preferences between users who read the same books are not that close.

## Demo

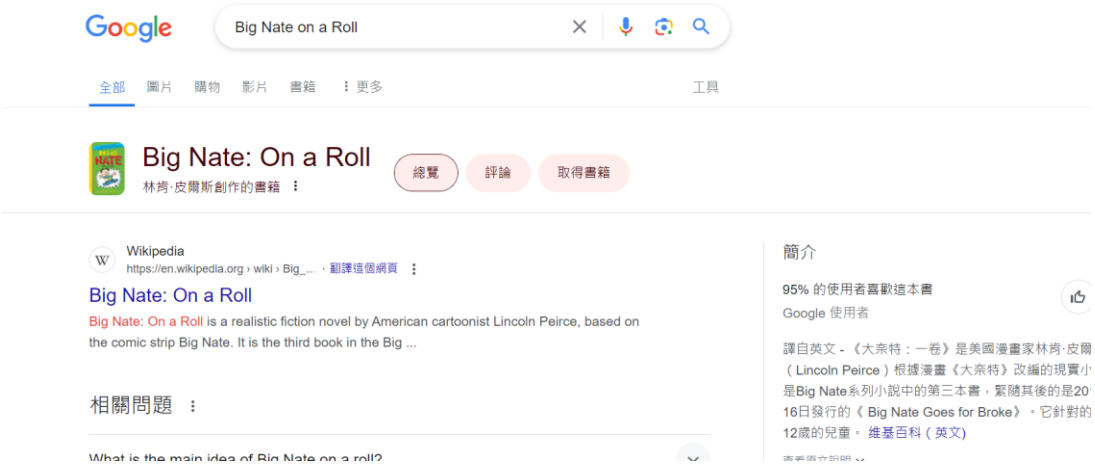
1. user-based




- First, select a random user and show ten of the books the user has read. Next, we will show the book recommended to the user.

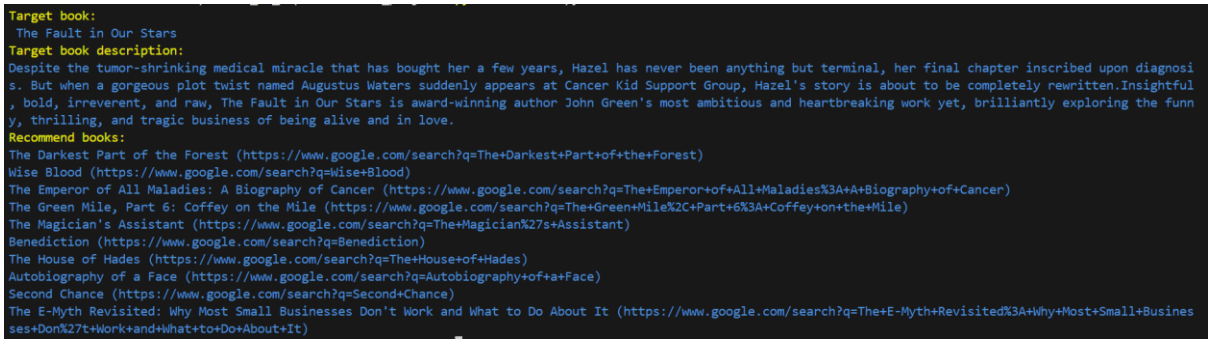

Big Nate on a Roll | Big Nate Strikes Again | In the Kingdom of Ice: The Grand and Terrible Polar Voyage of the USS Jeannette | When the Game Was Ours | □□□□□□□□□ 15

- We will also show the book title and their corresponding title if they have.

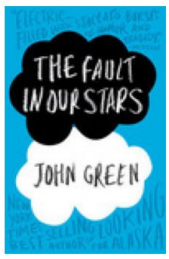



- We will provide the url of google search page for each book. The url is printed behind the recommended book.

2. content based

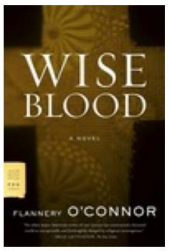



- First we will choose a target book, and print its description. Next, we will show the recommended book.

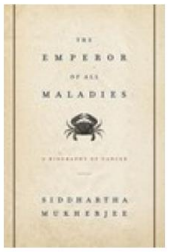


Rating: 4.2

Rating: 3.9

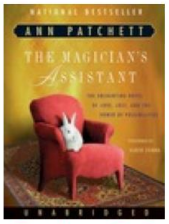


Rating: 3.9



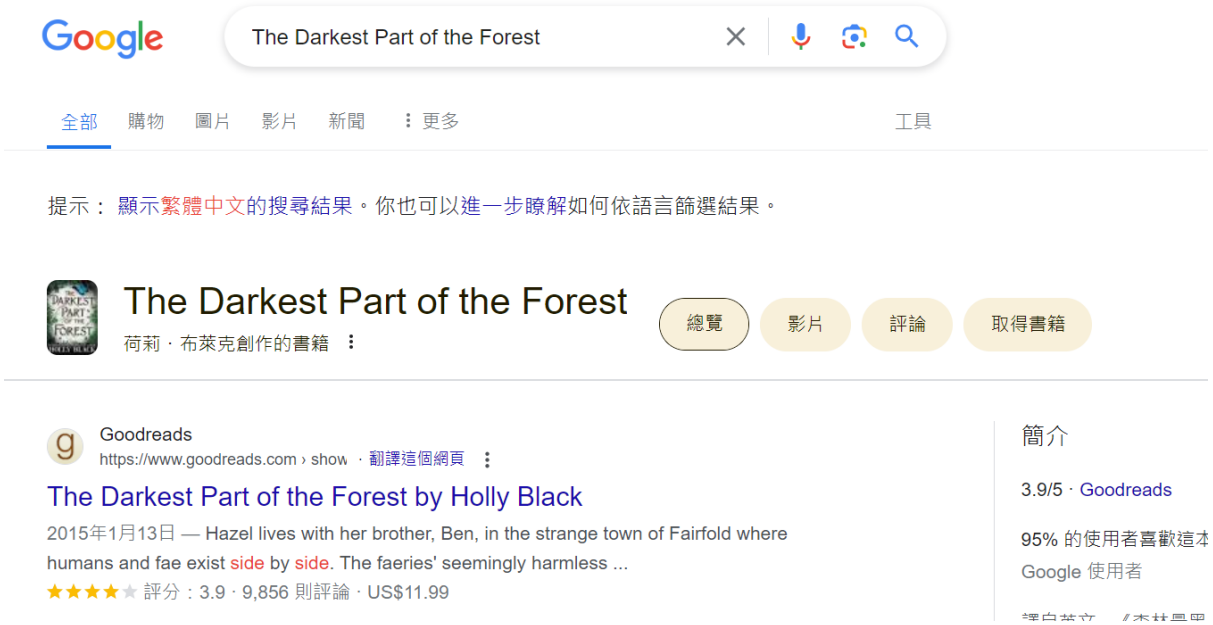
Rating: 4.3


Rating: 4.6



Rating: 3.6

- Here is the cover of the recommended books and their ratings.




- We will provide the url of google search page for each book. The url is printed behind the recommended book.

# Conclusion

From the results, we can see that our recommendation systems can roughly infer the readers' preferences. Since the datasets we have do not contain recently user records, however, if we can solve the problem and use this system in the place such as library, or online books stores, and so on, we think it is a good way to let the users know the potential books that he or she may want to read, moreover, we have two ways to to recommendation, so we hope using these two methods, we can further explore users' preference.
The above is the whole of our final project report, thank you.

# Reference

[1]Content-Based
https://www.kaggle.com/code/mechatronixs/recommend-project-content-based-recommendation
[2]User-Based
https://www.kaggle.com/code/mechatronixs/recommendation-project-3-user-based-recommendation/notebook
[3]TF-IDF Reference
https://zh.wikipedia.org/zh-tw/Tf-idf
[4]Cosine Similarity
https://en.wikipedia.org/wiki/Cosine_similarity
[5]Collaborating Filtering
https://realpython.com/build-recommendation-engine-collaborative-filtering/

# Contributions

黃芷柔: content-based, evaluation
李宜蓁: find information, demo code
劉詠: user-based