Algorithms being used:

Recommendation Algorithm:

The recommendation algorithm will recommend books to users without typing in specific keywords, which is essentially the whole point of the algorithm. This algorithm needs to identify the tags on each book in the database in order to return the tiers. It will group the books that have the most tags in common, and list them after the user selects the book from the dropdown menu. We are using a cosine similarity algorithm which measures the angle between the two vectors and then uses the equation above to measure how close they are two each other.

Sorting Algorithm:

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}},$$

For our data structure we are using Lists from python. The reason that we are using lists is because lists provide us with the ability to add elements as we go. This feature is very useful due to the flexibility it allows since part of our algorithm searches the books entered, and collects all the tags as they go through the books. The flexibility plays a part in the role of the structure because the user can input an infinite amount of books. Secondly, with the use of list we are able to combine multiple lists together in order to create a resultant pseudo hash-table. For example, if we want to see how many times a certain tag was found; we could have one list for the tags generated from the data of each book, then a separate list for the count of the occurrence of each tag.

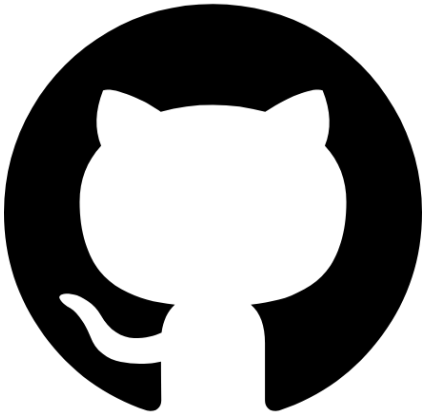
Some Screenshots

1. The Great Gatsby
2. Of Mice and Men
3. Great Expectations
4. Macbeth
5. Othello
6. To Kill a Mockingbird
7. Hamlet
8. The Adventures of Huckleberry Finn
9. Julius Caesar
10. The Awakening
11. Ethan Frome
12. The Scarlet Letter
13. Death of a Salesman
14. One Flew Over the Cuckoo's Nest
15. A Separate Peace
16. The Crucible
17. Frankenstein

Recommending Alpha

Software Engineering Spring 2019

Demo Date: March 26, 2019



https://github.com/vedantadhobley/SoftwareEngineeringProject2019

What does it do:

This project is designed to allow a user to search up a book that they want to find similar books for. The user will input a book and the algorithms will first narrow down their searches, and then show a list of similar books. There are two main algorithms, a searching algorithm which helps a user narrow down their searches using a drop down menu, and a recommendation algorithm which displays the list of recommended books in tiers. The tiers will display books that have the greatest similarities first and books that are good matches, but less similar in the bottom tier brackets.

Who it is for:

This project was designed for people of all ages. Parents can use this to find books for their children. Avid readers can use this to find all kinds of books that match their interests. Teachers may use it to find better books for their students to read. There is no limit on who can use this application.

Where will it be used:

This project will be created so that it can be used in libraries, book stores, and wherever else someone wants to implement a book searching helper in their department. It can be installed in school computers for kids who need information for their research papers. It can be used to help parents find books that help their children learn. It can also be used by tutors looking to find books for students they are assisting in older academic levels as well.

List of Features with Descriptions:

So far, the search, database (static, local) and retrieval for the recommender have been implemented abstractly.

We've begun work on creating the object classes in order to store this information logically. The recommendation algorithm has just been finished and is ready to be used.

The website is developed for the user to input his or her books into. The algorithms will soon be applied to the website and it will be given a fresh look.

System Requirements:

Backend

This development feature regards retrieval and object creation. Testing for the backend will require more effort. Using a sample set of books and estimating return values, we can adjust our algorithm multipliers (for different tag types) in order to return books that are most like the books we've inputted. With repetition we should be able to enter any list of books and receive a list of books which has strong correlation to those entered.

Frontend

For frontend, we are implementing using Bootstrap to make a user-friendly and dynamic site. From the frontend site itself, the system is accessing the database to check for book's existence and if it exists then the system accesses the backend part to run the recommendation algorithms. The Search subsystem accesses the database and finds out all the similar books to recommend. At database level, the system communicates to Evaluate sub-system to display the recommended users on the frontend to the user.

Group 16 Information

Group Members:

Vedanta Dhobley: vjd41@scarletmail.rutgers.edu

(Team Captain)

Avani Bhardwaj: ab1572@scarletmail.rutgers.edu

Shazidul Islam: si194@scarletmail.rutgers.edu

Akshat Shah: avs91@scarletmail.rutgers.edu

Kutay Kerimoglu: kk851@scarletmail.rutgers.edu

Alan Patel: akp122@scarletmail.rutgers.edu

Anthony Matos: amm720@scarletmail.rutgers.edu

Joel Cruz: jc2125@scarletmail.rutgers.edu