Recommendation system

Specifications sheet

Supervised by:

Prof ASSKLOU Abdelwahed

Prof BERRADA Ismail

LABZAE Kawtar

KABLY Malak

Introduction:

Our recommendation system is designed to provide book recommendations to users. Indeed, the user will have the ability to input a keyword, and in response, our system will generate a curated list of relevant books. Additionally, our system will also offer personalized recommendations for each one of the users based on their history, past ratings, and other relevant information.

For our recommendation system, the chosen system type is a hybrid approach with a strong focus on content-based recommendation and limited incorporation of collaborative filtering.

So, by implementing the content-based approach, we will analyze comprehensively the various book attributes, such as title, author, genre, and descriptions. By leveraging the characteristics of the books themselves, we can identify similarities and patterns that will enable us to recommend similar books.

About the collaborative filtering techniques, we will utilize them to identify users with similar reading patterns and preferences.

Dataset:

Source: The dataset is obtained from the Google Books API (base_url: https://www.googleapis.com/books/v1/volumes).

Description: The dataset contains book metadata obtained from the Google Books API based on a specified query. The dataset provides detailed information about each book, encompassing the following features:

- ✓ *Title:* The title of the book (string).
- ✓ *Authors:* The authors of the book (list of strings).
- ✓ *Summary:* The summary or description of the book (strings).
- ✓ *Language:* The language in which the book is written (string).
- ✓ *Publisher:* The name of the book's publisher (string).
- ✓ *Page Count:* The number of pages in the book (integer).
- ✓ *Publication Year***:** The year in which the book was published(integer).
- ✓ *Price:* The price of the book (float).

So, as we said, the dataset is collected by querying the Google Books API, specifying the desired criteria, such as the search term and the maximum number of results. The API responds with book information in the JSON format matching the query. The code provided uses the requests library to retrieve the data and processes it to extract the relevant metadata and store it in instances of the Book class.

Data Structures Design:

Class LinkedList

Constructor:

- *__init__():* Initializes a new instance of the LinkedList class.

Attributes:

- *head:* The head node of the linked list.

Methods:

- *append(data):* Appends a new node with the provided data to the end of the linked list.
- *extend(my_list):* Extends the current linked list by appending all nodes from another linked list.
- *remove(data):* Removes the first occurrence of a node with the provided data from the linked list.
- *is_empty():* Checks if the linked list is empty.
- *remove_duplicates():* Removes duplicate elements from the linked list.

Class Stack:

Constructor:

- *__init__():* Initializes a new instance of the Stack class.

Attributes:

- *linked_list:* An instance of the LinkedList class used to implement the stack.

Methods:

- *append(data):* Pushes a new element onto the stack.
- *remove*(): Removes and returns the top element from the stack.
- *extend(my_list):* Extends the current stack by pushing all elements from another stack.
- *is_empty():* Checks if the stack is empty.

Class Book:

Constructor:

- *__init__():* Initializes a new instance of the Book class with the given title, authors, summary, language, publisher, page_count and publication_year

Attributes :

- *Id (int):* The ID of the book.
- *Title (str):* The title of the book.
- *Authors (list):* A list of authors of the book.
- *Summary (str) :* A summary or description of the book.
- *Language (str):* The language in which the book is written.
- *Publisher (str)*: The publisher of the book.
- *page_count (str) :* The total number of pages in the book.
- *publication_year*: (str) The year of publication of the book.
- *Price (int):* The price of the book.
- *Likes (int):* The number of likes received for the book.
- *Dislikes (int)*: The number of dislikes received for the book.
- *Readers (list)*: A linked list of readers who have read the book.
- *Feedbacks (list)*: A linked List of feedback that the users had added to the book

Methods :

- *get_id():* Returns the ID of the book**.**
- *add_reader(user):* Adds a reader to the book's list of readers and adds the book to the user's readbook list.
- *print_info()***:** Prints the information about the book**.**

Class User:

Constructor:

- *__init__(self, name):* Initializes a new User object with the given name.

Attributes:

- *Name (str)*: The name of the user.
- R*eadbook (linked list)*: An instance of the LinkedList class to store the books the user has read.
- *Likebook (linked list):* An instance of the LinkedList class to store the books the user has liked.
- *Dislikebook(linked list)*: An instance of the LinkedList class to store the books the user has disliked.
- *Historic (stack):* An instance of the Stack class to store the user's search history.

Methods:

- *set_name(name):* Sets the name of the user.
- *Like(book_id):* Adds a like to the book with the specified book_id and appends it to the user's liked books.
- *dislike(book_id):* Adds a dislike to the book with the specified book_id and appends it to the user's disliked books.
- *add_feedback_book(id_book, feed_back):* Adds feedback to the book with the specified id_book.
- *add_book(book_id):* Adds the book with the specified book_id to the user's read books.
- *remove_book(book_id):* Removes the book with the specified book_id from the user's read books.
- *search(key_word):* Searches for books based on the specified key_word and appends them to the user's search history.
- *new_recommender():*Provides book recommendations based on genre for a user without a search history.
- *enter_description(text):* Searches for books based on the specified description and appends the description to the user's search history.
- *historical_recommender():* Provides book recommendations based on the user's historical data.
- *filter_search(keyword, page_count=range(0, 10000000), price=range(80, 1001), likes=20, language="fren", pub_year=0):* Filters search results based on various parameters and appends the filtered books to the user's search history.
- *reset():* Resets the user's read, liked, disliked books, and search history.
- *guide():* Provides an introduction to the ReadQuest system and describes each method available for the use.

Outcome:

Our recommendation system is designed to provide personalized book recommendations to users based on their preferences and past engagement with similar items. Our system aims to achieve the following:

Minimum successful result:

- A search functionality: Implement a search feature that allows users to enter keywords to search for books within the system.
- Recommendations: The system considers the attributes of books and the user's history to provide personalized book recommendations.
- Basic user interaction: Implement methods within the User class to enable basic interactions, such as liking a book, disliking a book, and adding a book. Those actions will be recorded in the user's profile and can help in improving the accuracy of future recommendations.

Ideal – Maximum result:

- Book rating and feedback: The user will be able to rate books they have read. They can also provide ratings and feedback, such as reviews or comments, and share their opinions about the books.
- A search function with filters: Enhance the search functionality by allowing users to apply filters to refine their search results. Users can specify additional criteria such as book ratings, price range, or genre to narrow down their search and receive more tailored recommendations.
- Recommendations with advanced techniques: Enhance the recommendation engine by incorporating advanced techniques such as collaborative filtering. This approach expands the recommendation scope beyond content-based filtering.
- Backup Database: As an ideal solution, incorporate a backup database in case the Google Books API does not have relevant data or experiences connectivity issues. The backup database can be in the form of a .db or .csv file, which can be accessed locally.