User’s Manual

Book Recommendation System

Nadine Thomas, Kacie Myers, and Gracie Lovell

Introduction

This project allows users to input a book they love and receive a list of recommendations similar to the imputed book. The program analyzes book reviews to generate a hybrid recommendation system based on the following:

- An aggregation of book description, genre, and reviews
- User ratings were used to produce a weighted sentiment analysis, giving books with a 4 or higher rating a boosted weight

The system takes two original CSV files:

1. amazon_book_reviews.csv
2. books_data.csv

These files are combined using SQL, processed in Python, and analyzed to produce book recommendations.

All SQL and Python files are provided. Users do not need to write any code - only to follow the instructions to run it

Requirements

Software

- Python 3.9 or later
- SQL database software (MySQLWorkbench preferred)
- Python editor (VS Code preferred)

Python Libraries

Install the required libraries by running the following command in a terminal:

```
pip install pandas numpy nltk scikit-learn sqlalchemy
```

Additional download for sentiment analysis:

```
import nltk
nltk.download('vader_lexicon')
```

Data Files

- amazon_book_reviews.csv - contains book ratings and review text
- books_data.csv - contains book details
- databasesetup.sql - sql code used to combine two csv files
- main.py - python code that does sentiment analysis and provides recommendations

Running the System

Step 1: Run SQL File

1. Open SQL program
2. Open the file: databasesetup.sql
3. Run the script

This script does the following:

- Creates two tables
- Load book info and review data into respective tables

After the script runs, you will have a single database with two tables: one containing rating, review text, and book title and the other containing all the book information. The two tables are connected through a shared book.id key

Step 2: Run Python File

1. Open a terminal or Python editor
2. Run the following command:'

   python main.py

3. The first run of this file is expected to take ~20 minutes. This will cache all of the book profiles into a .pkl file called book_recommender_cache.pkl. From now on, succeeding runs should only take a couple of seconds.

This file does the following:

- Connects to SQL database to import data
- Merges two tables into one dataframe
- Pre-processes text
- Weighs text by user rating
- Builds a recommendation model using TF-IDF
- Calculates book similarity using cosine similarity
- Launches an interactive interface that generates a list of the top 10 recommended books

Getting Recommendations

With the Python script, you can request recommendations using:

recommend_books(“Book Title You LOVE Here”)

You will receive output similar to the following:

Recommended Books:

1. *Book A*
2. *Book B*
3. *Book C*
4. *Book D*
5. *Book E*
6. *Book F*
7. *Book G*
8. *Book H*
9. *Book I*
10. *Book J*

Recommendations are based on:

- Reviews: The text from all available reader reviews is aggregated.
- Categories & Description: The book's official categories (genre) and publisher description are added.
- Weighting: Content from highly positive reviews (4 or 5 stars) is given double the weight to ensure that the keywords associated with a good reader experience have a stronger influence.

Troubleshooting

Issue	**Fix**
SQL file fails	Make sure CSV paths inside databasesetup.sql are correct
Python cannot connect to SQL	Update database path in main.py
VADER errors	Run *nltk.download(“vader_lexicon”)*
No recommendations shown	Double check book title is spelled correctly

Summary

This system is a Content-Based Book Recommendation Engine that processes book reviews and metadata to suggest books with similar themes and topics.

Key Functionality

- Data Acquisition: Connects directly to the SQL database using Python and executes two SQL queries to import filtered book and review data.
- Book Profile Creation: Aggregates review text, descriptions, and categories into a single Book Profile for every book.
- Rating-Based Weighting: Uses the explicit ≥ 4 star review scores to boost the influence of positive reviews on the Book Profile, ensuring the system prioritizes content associated with positive reader experiences.
- Model Building: Builds the core recommendation model using TF-IDF (Term Frequency-Inverse Document Frequency) to quantify the descriptive keywords of each book's profile.
- Similarity Calculation: Calculates book-to-book similarity using Cosine Similarity, identifying books that are most similar in content.
- Output: Outputs a list of 10 recommended books based on their content similarity to the source book, while also filtering for diversity (excluding the same author) and quality (minimum review count).

Deployment and Usage

Because the core processing logic is contained within a single Python file (main.py) and data retrieval is handled internally, users only need to:

1. Ensure SQL Setup: Run any necessary database setup scripts (e.g., to create tables or load initial data).
2. Run main.py: Execute the Python file.

The system handles everything else automatically, including caching the built model for fast loading on subsequent runs.