Book Recommendations Using RAG

Rohan Arora

DATS 4001: Data Science Capstone

Abstract

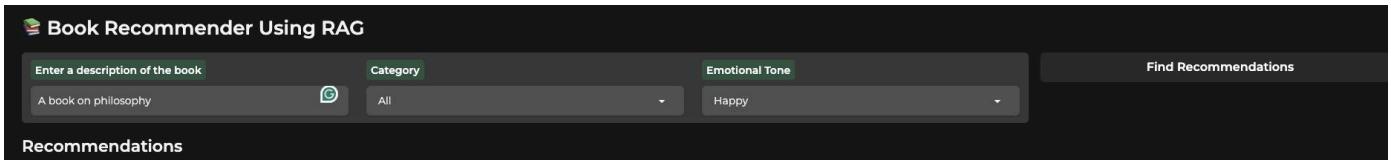
This project introduces a content-based book recommendation system that uses large language models to suggest titles based on book descriptions and emotional tone, rather than user history. The system integrates zero-shot genre classification, sentiment analysis, and vector similarity search, and is built using Python tools such as LangChain, Hugging Face, and Gradio. Drawing on data from Goodreads and Open Library, it recommends books through a simple web interface.

Video Presentation with Demonstration of Application

I. Introduction

With millions of new books published each year, readers often struggle to find novels that match their interests. Recommendation systems on sites like Amazon or Goodreads aim to solve this, but these systems rely on collaborative filtering, which depends on user ratings, reviews, and the individuals' purchasing data. These systems fail when data is sparse or when books lack user feedback. A content-based approach offers an alternative by focusing on the content of the books themselves, rather than user behavior. This project follows this approach using large language models (LLMs). Using natural language processing techniques, the system analyzes book descriptions and emotional tone to match users with titles that match their interests. The recommendation engine combines vector-based similarity searches, zero-shot classification, and sentiment analysis to provide meaningful suggestions without requiring user history.

This system is built in Python, using packages like LangChain for document handling, Hugging Face Transformers for classification, and Gradio to create an interactive web interface. Users begin by entering a short book description, selecting a genre, and choosing a desired emotional tone. The system uses this input to filter and rank books from a pre-processed dataset.



It first narrows down the dataset using zero-shot genre classification, then applies emotion-based filtering using sentiment analysis. Finally, it runs a vector similarity search between the user's description and the remaining candidates to find the most semantically similar titles. Unlike systems based on popularity and user history, the model evaluates both what the book is about

and how it feels. Emotion classification helps match with moods such as sadness, hope, or excitement. The system then returns the top eight most relevant books, each accompanied by key metadata, such as title, author, and description. This setup enables a more nuanced book discovery process that goes beyond surface-level features to capture a book's content and quality.

II. Data

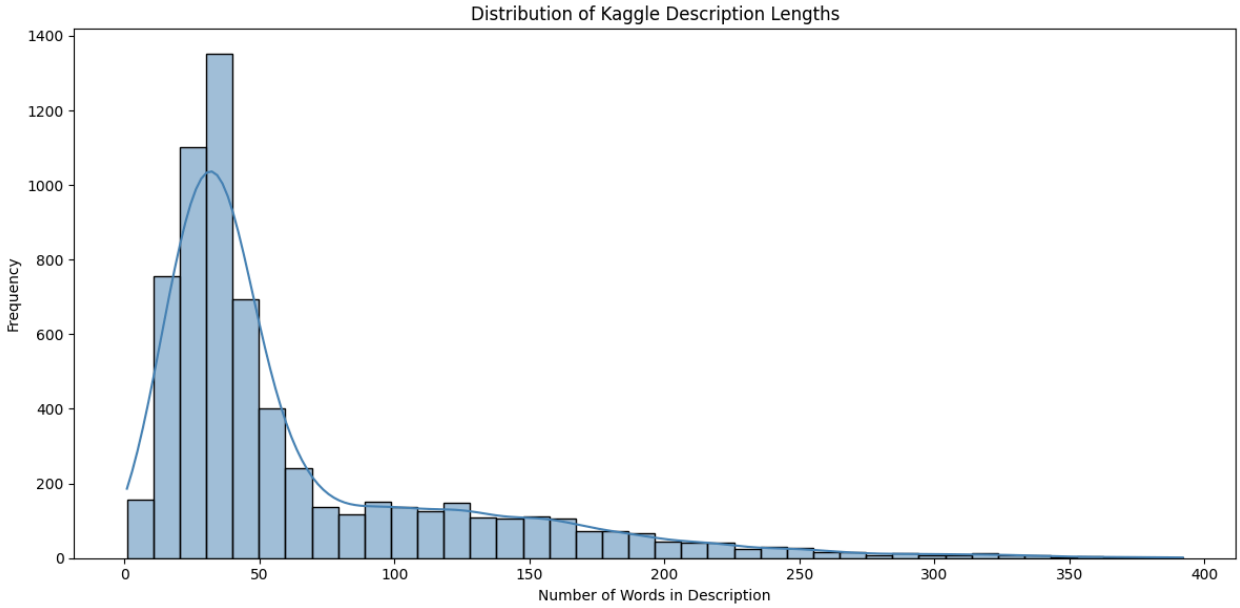
A. Sources

This project draws on two sources to create a dataset of books: a Kaggle dataset scraped from the Goodreads API and the Open Library API. Together, these sources provide a blend of structured and unstructured text, which supports both the machine learning pipeline and the user interface. The Kaggle dataset includes 6,810 books containing fields such as ISBN-13, title, subtitle, authors, categories, description, published year, average rating, and number of pages. These features provide a solid foundation for evaluating basic content and bibliographic details. Still, the descriptions in this dataset can be brief or missing, which limits their use for content-based modeling. To add to this existing dataset, the Open Library API was used to retrieve 26,266 books along with richer metadata. This API also provides access to edition counts, ebook availability, and preview links, adding depth to the dataset.

B. Data Cleaning and EDA

Before building the recommendation system, both datasets required significant cleaning to ensure consistency and quality in the suggestions. An initial inspection of the Kaggle dataset revealed that 262 books did not have descriptions, which are necessary for the recommendation system, so these entries were removed. After analyzing the distribution of description lengths, most entries were short, with a length under 250 words and a mean of 66 words. However, descriptions under 20 words were often vague or lacked meaningful detail. Filtering out these

descriptions left 5,595 books. Lastly, the dataset splits titles into two different fields: title and subtitle. To improve clarity, the fields were aggregated into a single string. This step preserved important context; for example, combining “I Am That” with its subtitle, “Talks with Sri Nisargadatta Maharaj”, helps the reader and model understand the subject matter better.



A similar process was applied to the Open Library data, starting with removing books with descriptions, which was a sizable chunk of the data - 12,719 books. Compared to the Kaggle dataset, the descriptions were much longer and more descriptive, with a mean length of 114 words. Like the Kaggle dataset, books with fewer than 20 words were excluded due to the lack of detail. This filtering left 8,751 books for the merged datasets, which were merged on the unique identifier, ISBN-13. By combining two datasets, this provided the model with more information that can be used to acquire accurate book recommendations.

III. Model Setup

The model pipeline integrates three core components: zero-shot genre classification, emotion-based sentiment analysis, and vector-based similarity search. Each stage filters or ranks

books to help return recommendations that match the user's description, preferred genre, and desired emotional tone.

A. Genre Classification

The initial step narrowed down the large variety of category labels, totaling 9,099 distinct categories, using a combination of rule-based mapping and zero-shot classification. First, the most common genres, such as biography, autobiography, and history, were manually categorized into fiction, nonfiction, children's fiction, and children's nonfiction. For books with multiple categories or ultra-specific genres, like children of physicians, a zero-shot classification model (facebook/bart-large-mnli) was used to assign them to the four categories listed above. This model evaluates the likelihood of a category belonging to a given set of categories without

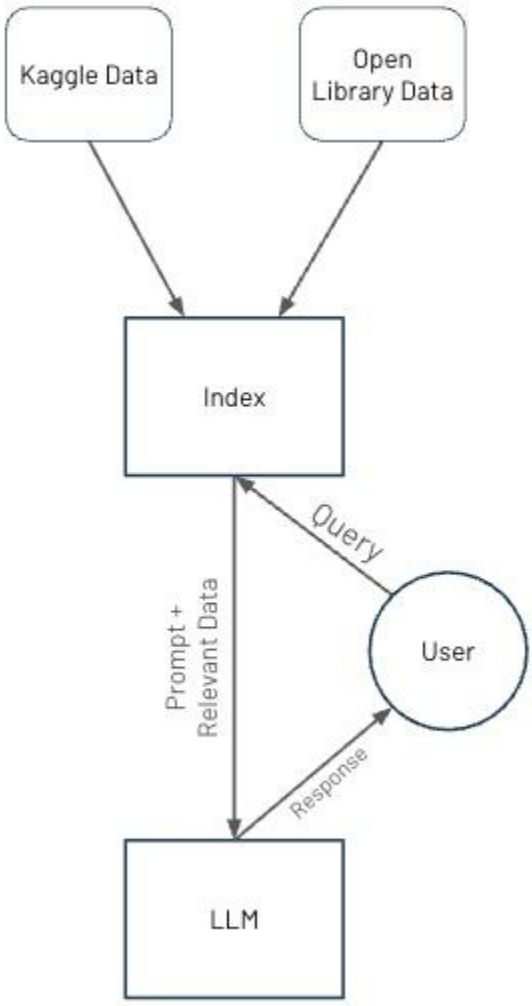
requiring retraining. It correctly predicted the genre 89% of the time when tested against the existing labeled examples.

B. Sentiment/Emotion Classification

To further refine results by tone, the system uses the emotion-english-distilroberta-base model to analyze the emotional content in each book description. After testing this model against the entire book description, it yielded inaccurate results, as the model would often focus on one or two sentences in the description for its analysis. To address this, descriptions were split into sentences, truncated based on token length, and passed through the classifier to identify seven emotions: anger, disgust, fear, joy, sadness, surprise, and neutral. The model output scores for each emotion, and the highest score across all sentences were retained for each label.

C. Vector Embedding and Similarity Search

The system then generated vector embeddings from the book descriptions to support content matching. Vector embeddings are a numerical representation of text in a high-dimensional space, where similar texts are positioned closer to each other. To create these embeddings, a simplified text file was created containing only the ISBN-13 and description variables. This smaller file was passed into a LangChain-based embedding pipeline using the OpenAI Embeddings API. LangChain handles this through a structured pipeline that splits the documents into manageable chunks, processes them via a



transformer-based language model, and generates dense 1536-dimensional vectors. Each vector captures features of the text, such as thematic content, genre-related vocabulary, and emotional undertones, allowing the system to capture nuanced similarities even when explicit keywords differ. These vectors were stored in a vector database, enabling fast similarity searches across all of the embedded documents. This workflow follows the retriever and embedding concepts defined in LangChain's documentation: embeddings act as a projection of meaning into a vector space, while the receiver uses similarity metrics to return the most relevant documents given a query vector.

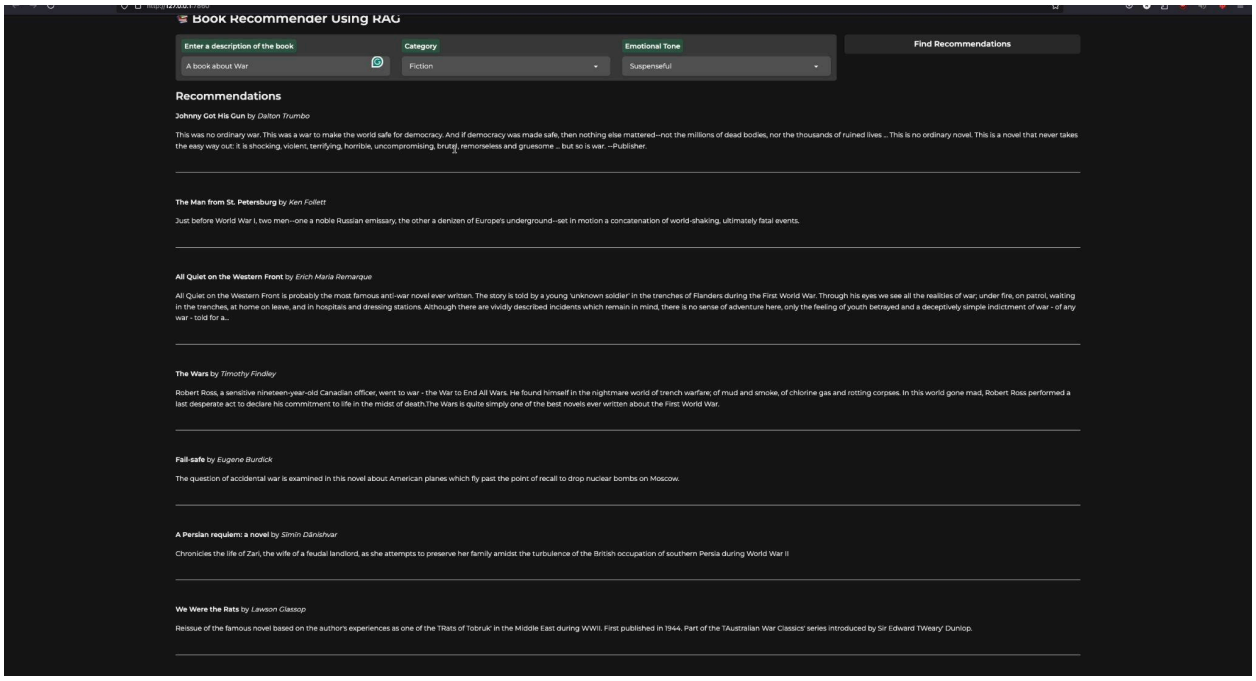
When a user inputs a custom query or mentions a specific book, that input is embedded using the same process. The system then computes the cosine similarity between the user's input vector and the precomputed book vectors to rank books based on semantic closeness. Cosine similarity compares the angle between vectors, making it ideal for detecting shared meaning. For

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

example, if a user searches for a "gritty detective thriller," the system may give a book described as a "dark mystery" if the underlying semantic vectors are close. The top matches are then reconnected to the full dataset using ISBN-13 as the key, and the genre and emotion filters are applied. Ensuring that only books that match the user's selected category and tone are included in the recommendations.

IV. Results

The final system offers an intuitive dashboard for book recommendations. Users enter a sort description of the type of book they are looking for, select a genre, and choose an emotional tone. In return, the system displays eight recommended books that align with the user's inputs across all three dimensions: content, category, and tone. The recommendations are displayed with relevant metadata, the title, author(s), and descriptions.



However, this system does have limitations. Since it only uses the descriptions and not full book content, recommendations are based solely on how they are described, which can lead to surface-level matches when descriptions are vague or poorly written. Emotion classification also depends on sentence-level context and can misinterpret descriptions with shifting or ambiguous tone. These constraints highlight the importance of high-quality descriptions and

suggest opportunities for future refinement, such as incorporating full-text analysis or additional metadata like user tags or themes.

V. Bibliography

Trivedi, Ayushi. (2024, October 4). *Top 6 Books on Retrieval Augmented Generation (RAG).* Analytics Vidhya.

https://www.analyticsvidhya.com/blog/2024/10/books-on-rag/

Castillo, D. J. (n.d.). *7k Books with Metadata* [Data set]. Kaggle.

https://www.kaggle.com/datasets/dylanjcastillo/7k-books-with-metadata

Dhapre, M. (2024, February 2). *Book Recommendation using Retrieval Augmented Generation.* Medium.

https://medium.com/@mrunmayee.dhapre/book-recommendation-using-retrieval-augmented-generation-52965b71ed16

Jonathandika. (n.d.). *llm-recommender-system*. GitHub.

https://github.com/Jonathandika/llm-recommender-system

Open Library. (n.d.). *Developer Center / APIs.* Open Library.

https://openlibrary.org/developers/api