Literary Lens Project Report

Team Members : Tayisiya Chernenko, Abby East

Repo : https://github.com/TayisiyaChernenko/NLPBooksRecs#

1 LiteraryLens Project Overview

LiteraryLens is a web application designed to provide readers with deeper insights into books using data from the Goodreads dataset. The platform features automatic review summarization powered by HuggingFace's BART transformer model, enabling users to quickly grasp the overall sentiment and key takeaways from a large number of reviews. To further enhance interpretability, the system employs the log-odds ratio with Dirichlet priors method to identify and rank the top ten most distinctive words associated with positive and negative reviews for each book. These findings are then presented through an interactive word cloud visualization. The data was preprocessed, ensuring a sufficient sample of English reviews per book, to yield the best quality summarization for users without sacrificing computational efficiency.

The application is built on a modern tech stack, with a responsive React frontend that delivers a smooth user experience and a Dockerized PostgreSQL database for efficient data storage and retrieval, with the larger dataset in Amazon S3. By combining NLP-driven analysis with engaging visualizations, LiteraryLens offers a unique and informative lens into the collective voice of readers, helping users discover books more effectively and make informed reading choices.

2 Natural Language Processing

2.1 Data Pre Processing

The data preprocessing pipeline is designed to clean and structure raw Goodreads review data to prepare it for downstream analysis like topic modeling and summarization. It begins by merging review text with relevant book metadata, such as titles, authors, and ratings. The review texts are then filtered to include only English-language content, stripped of unwanted characters, and standardized in format. Reviews are grouped by book, and when necessary, a representative sample is taken to reduce memory usage while preserving content diversity. This preprocessed dataset is then saved for use in summarization, log odds ratio comparisons, and topic modeling, ensuring that all subsequent steps are built on clean, consistent input.

2.2 Review Summarization

Transformers are a type of deep learning model designed to process and understand sequential data, such as language, by using a mechanism called self-attention. Self-attention enables the model to weigh the importance of each word in context. Unlike earlier models that processed data in order (like RNNs), transformers can look at all parts of the input simultaneously, allowing them to capture complex relationships between words regardless of their position in a sentence. The model uses an encoder to understand the input and, in some cases, a decoder to generate output.

LiteraryLens utilizes the pre-trained Hugging Face’s BART model. BART was developed as a hybrid of two models: BERT and GPT. BART integrates the bidirectional encoding capabilities of BERT with the autoregressive decoding strengths of GPT. This architecture allows BART to comprehend the context of a sentence and generate coherent text. Its pre-training process involves intentionally corrupting input data through techniques like token masking and sentence permutation, then training the model to reconstruct the original text. This method encourages a deep understanding of linguistic structure and meaning. Then, BART can be fine-tuned on specific tasks such as summarization, which our project utilized, as well as other tasks such as translation and question answering.

2.3 Top Review Sentiments

To get the top ten words associated with positive and negative reviews, the data was split by ratings before applying log odds ratios with Dirichlet priors. This is a useful statistical method for identifying the most distinctive words associated with different categories. By comparing the frequency of words in each class while accounting for overall word distribution and smoothing via Dirichlet priors, this method highlights words that are not just frequent but disproportionately characteristic of one class over another. This helps reveal words that are strongly indicative of sentiment, such as "excellent" for positive reviews or "terrible" for negative ones, while avoiding misleading results caused by rare or overly common terms.

3 Architecture

3.1 React JS

React is a JavaScript library developed by Facebook. It enables developers to create reusable UI components, which makes code more organized and easier to maintain. React’s virtual DOM improves performance by efficiently updating only the parts of the page that change, rather than reloading the entire page. Given that our website has static components which stay on the page consistently such as book recommendations, while others dynamically update given a user’s search, this aspect of React optimizes our front end. We also utilized the React-Wordcloud library, which is supported for React 18.0.

3.2 Postgres

PostgreSQL is an open-source relational database system known for its stability, ACID compliance, and support for advanced features like JSONB and full-text search. It uses SQL for querying and supports indexing, joins, and transactions, making it suitable for complex data structures and large-scale applications. Literary Lens stores data quickly accessed by users in Postgres tables Summaries and Books, and queries information via Book Titles, as well as average ratings in descending order.

3.3 Docker

Docker is a containerization platform that packages applications and their dependencies into lightweight, portable containers. When used for Postgres, Docker allows developers to quickly spin up isolated PostgreSQL database instances using official Docker images, define settings via environment variables, and persist data using Docker volumes.

3.4 Node-Postgres and Express

The node-postgres (pg) library is a PostgreSQL client for Node.js that allows interaction between a Node.js application and a Postgres database. When used with Express, it enables developers to handle user input from HTTP requests, query the database, and return results efficiently. A connection is established using a Client or Pool from the pg module, which connects to the Postgres server with credentials and configuration details. In an Express route, user input is accessed via req.query and passed into SQL queries using parameterized syntax to prevent SQL injection. The query is executed, and the resulting JavaScript objects are sent back to the client as a JSON response. This setup provided our project with a secure and scalable way to perform dynamic searches for books and retrieve data.

3.5 Amazon S3

Amazon S3 is used as cloud storage to manage large-scale data more efficiently during processing and deployment. After preprocessing Goodreads reviews (such as filtering, cleaning, and merging with book metadata), the processed data is uploaded to S3 to avoid repeatedly handling the original large files locally. This setup allows the backend or any collaborators to access preprocessed data directly from S3 as needed. Additionally, by saving outputs like topic modeling results or summarization data back to S3, the project maintains a centralized and scalable way to serve content to the frontend without requiring the backend to recompute results on the fly.

4 Self Scoring, Contributions, and Lessons Learned

Abby East

Self-scoring:

80 points - significant exploration beyond baseline
30 points - Innovation or Creativity: implemented log odds test with dirichlet priors by hand, not imported library
10 points - highlighted complexity - designed data pipeline from pre processing to external storage in s3, utilized in front end and back end
10 points - discussion of lessons learned and potential improvements - learned:

Contributions:

Data Preprocessing

- Source Data: Goodreads .json.gz files containing millions of user reviews and corresponding book metadata..
- Language Filtering: Applied the langdetect library to keep only English-language reviews for consistent and accurate analysis.
- Review Threshold: Filtered dataset to include only books with at least 50 reviews to ensure robust, representative insights.
- Merging: Merged review data with book metadata (e.g., title, author, average rating) to create a comprehensive dataset for analysis.
- Storage: Saved the cleaned and structured dataset in CSV/JSON formats and uploaded it to AWS S3 for centralized access and integration.

Log Odds Ratio Analysis

- Goal: Identify words that are statistically distinctive for each book compared to the rest of the corpus.
- Method: Manually implemented log odds ratio computation using Dirichlet priors to compare word usage in a book's reviews against all others.
- Output: Generated lists of high-impact, distinctive words for each book, separated by sentiment (e.g., positive vs. negative reviews).
- Storage: Stored the output as JSON-like files in S3, optimized for downstream use in the web app's word analysis features.

Review Summarization (Abstractive)

- Tool: Used the facebook/bart-large-cnn model from Hugging Face's transformers library for abstractive text summarization.
- Goal: Provide concise natural-language summaries capturing the core themes and sentiments of user reviews for each book.
- Approach:
    - Combined up to 500 English reviews per book, capped at the model's input size by sampling sentences.

    - Generated 1–2 sentence summaries using the BART model with fixed length constraints to ensure clarity and consistency.
    - Focused summarization on the top 10 most-reviewed books due to resource constraints.
- Use Case: Allows users to quickly understand common feedback without reading full reviews; summaries are displayed in the front-end interface.

Lessons learned:

I learned a lot about using NLP technologies while working on this project. I learned how to process text data. The entire process was new to me, from tokenization to language detection. My background is in mathematics, so I am familiar with numerical data; processing language is entirely new to me. I also learned how to integrate and evaluate different NLP models. Although I only ended up using two models, while working on the project, I tested several, including LDA and BertTopic for keyword and topic identification and Seq2seq for review summarization. By testing a handful of models, I learned how to identify which was better for our unique dataset and problem statement.

Overall, the project taught me many skills that will be applicable in real-world scenarios once I enter the job market. Processing large data, utilizing cloud computing resources, and adapting language models to niche problems are all skills that I look forward to using both on the job and in my future personal projects.

Tayisiya Chernenko

Self-scoring:

80 points - significant exploration beyond baseline: Researched BART models and alternatives for review summarization.
30 points - Innovation or Creativity: Demonstrated unique approaches with displaying the data.
10 points - highlighted complexity : Designed architecture, and handled user and data errors.
10 points - discussion of lessons learned and potential improvements
10 points - exceptional visualization/repo : Made expandable Wordclouds and organized repo directories, with a README.md file in each directory specifying its purpose.

Contributions:

Efficient Data Storage and Access

- Goal: Transform JSON formatting into structured tables with the SQL database Postgres, and query them safely.
- Approach:
    - Created a Postgres database within a docker-compose file to aid with quick instantiaton.
    - Developed relational models as tables in Postgres

    - Formatted data into match the models and loaded the database
    - Created REST API endpoints using Express, handling HTTP GET requests, extracting query parameters, validating input, and returning JSON responses.
    - Enabled clean error handling and input validation to improve API robustness
    - Implemented the data access layer using the node-postgres pg connection pool, writing SQL queries and encapsulating them in async functions.
    - Utilized routes on the React frontend with the native fetch API

Log Odds Ratio Analysis Display

- Goal: Identify distinctive vocabulary per book compared to all other reviews, and display it in a clear manner.
- Method :
    - Parsed the words and log odds values by converting to usable form
    - Scaled the log odds values to maintain the distance between words with only positive values to match the visualization restrictions.
    - Convered the data into wordclouds, with the sizing of the word matching the relevance for a visually intuitive sign of importance per word.

Review Summarization and Log Odds Research

- Goal : Research and plan the tools and methods for review summarization
- Proposed :
    - HuggingFace BART or LCF-BERT
    - Split the reviews by positive and negative sentiment prior to NLP analysis.
    - Display log odds with WordClouds
    - Generate one summary, the way Amazon does for its product reviews, in order to minimize user effort to attain insight.

Website Display (Front End)

- Goal : Make insights easily available with a visually appealing and intuitive website.
- Approach:
    - Used React-Vite to build reusable, optimized components
    - Displayed 3 key sections : Review Summarization, log odds for sentiment, and recommended books by rating.
    - Connected the frontend to postgres via the node-postgres pg library.

Architecture Design

- Goal: Determine the over-arching architecture of the project, reflected in the chosen Github directories
- Method : Researched and selected the best options for the full stack pipeline.

Lessons Learned:

One lesson I learned from this project was the importance of a data processing pipeline. We filtered the reviews for English words, and found that our top words were at times spam, such as a user repeatedly linking their own review page for more engagement. This problem highlighted the need for proper data processing, and in the future we would consider adding a spam detection step to the pipeline in order to limit these problems. In the future, we could also add a per-genre breakdown of top books using NLP.

Doing this project allowed me the opportunity to dive more into transformer models like BART, and the differences between the current open-source solutions available at the moment. It was a fun learning experience, and I learned more about how to introduce data to users in a way that is easily digestible, informative, and coherent using NLP alongside web-dev. I will be able to talk about this work in future interviews and show off our website.