Big Data Project Report

Spark your Imagination

Team Members

1. Madhuri Mahalingam (mm13575)
2. Swarali Dabhadkar (sd5664)
3. Kartikey Sharma (ks7154)

Dashboard Link: https://sparkyourimagination.streamlit.app/
Github Repo: https://github.com/KartikeySharma/SparkYourImagination

Dataset details

The datasets were collected in late 2017 from *goodreads.com*, where the data was scraped from users' public shelves. User IDs and review IDs are anonymized.
There are three groups of datasets: (1) meta-data of the books, (2) user-book interactions (users' public shelves) and (3) users' detailed book reviews. These datasets can be merged together by joining on book/user/review ids.
Basic Statistics of the Complete Book Graph:

- 2,360,655 books (1,521,962 works, 400,390 book series, 829,529 authors)
- 876,145 users; 228,648,342 user-book interactions in users' shelves (include 112,131,203 reads and 104,551,549 ratings)

Abstract

This project develops a book recommendation and analysis system, integrating sentiment analysis, collaborative and content-based filtering, k-means clustering, and exploratory data analysis (EDA). Utilizing the Alternating Least Squares (ALS) algorithm, the system personalizes recommendations based on user interactions, while Locality Sensitive Hashing (LSH) identifies similarities in book content. Sentiment analysis on user reviews adds depth to understanding reader preferences. K-means clustering categorizes books and authors, uncovering patterns and relationships between genres. The EDA aspect offers insights into literary trends and authorship. This multifaceted system serves as a dynamic tool for readers, authors, and publishers, enriching the literary landscape with data-driven insights.

Introduction

In a world saturated with literary choices, finding the right book can be overwhelming for readers. Our project is motivated by the need to simplify this discovery process, providing a clear path through the vast array of available books. We aim to create a system that not only guides readers to their next favorite read but also offers insights into the evolving trends of the literary world. These trends may help authors or publishing houses in targeting the right audiences based on genres and writing styles.

Problem Statement

1. Temporal analysis of goodreads datasets, including genre and author trends over time
2. K-means clustering analysis of books based on genre-tagging
3. Sentiment analysis on book reviews based on different factors
4. Content based and collaborative recommendation system

We first derive insights from various data exploration techniques and go on to build a recommendation system. The trend and sentiment analysis would help authors and publishing houses in gaining insights on user engagement, and the recommendation system aims to make it easier for readers to pick their next book based on previous likings.

Methodology

Temporal Analysis

Starting with user history, we did some visualization for a random user. Each horizontal line represents a single book, and the dots on the line indicate the start and finish dates of reading.

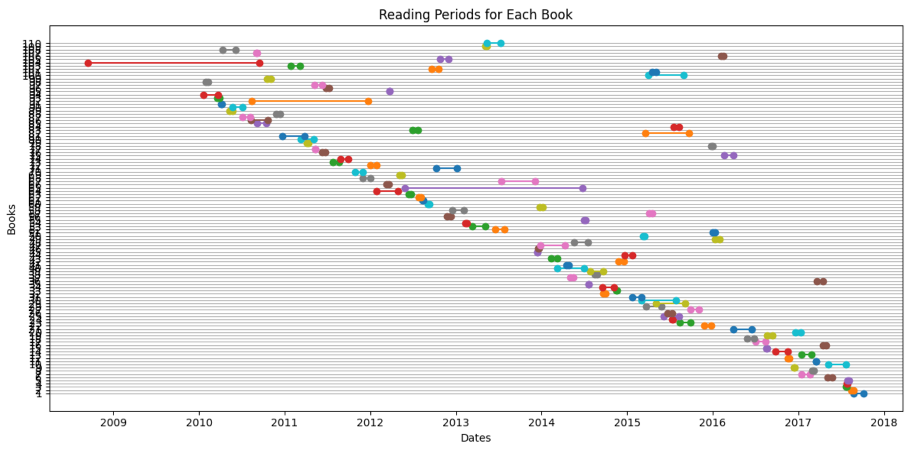


Fig. Book periods for a random user

Genre-based analysis

(Note: We sampled 1M records out of the 229M records dataset)
There is significant variability in average ratings over time for all genres. This could be due to various factors, including changes in reader preferences, the introduction of new authors or influential books, or shifts in cultural trends. It is challenging to compare genres directly due to the overlap. Still, some genres, such as history or non-fiction, may have less extreme variance compared to genres like comics or romance.

Sharp spikes or drops in ratings could indicate data anomalies, such as a few books with very high or low ratings skewing the average, or they could represent moments when a particularly influential book was published.

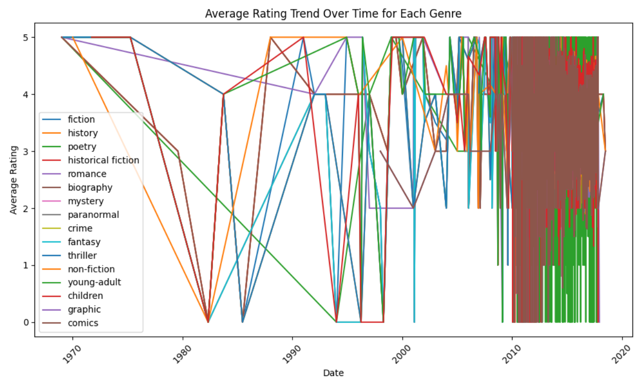


Fig: Average Rating Trend (1000000 data points)

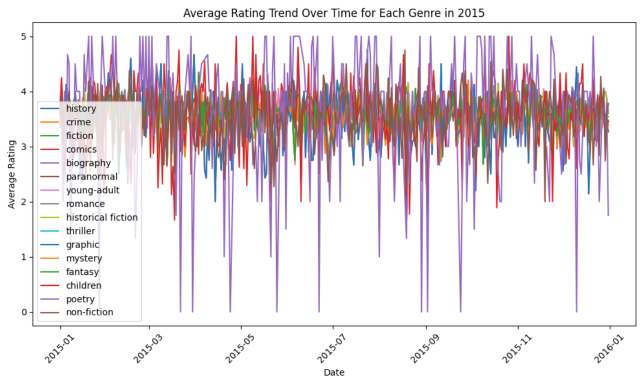


Fig. Average Rating Trend for 2015 (all genres)

Sampling the data for the top 3 genres based on a number of books, we can analyze that while there are fluctuations, there doesn't appear to be a long-term upward or downward trend in the average ratings for any of the genres, indicating stable popularity.

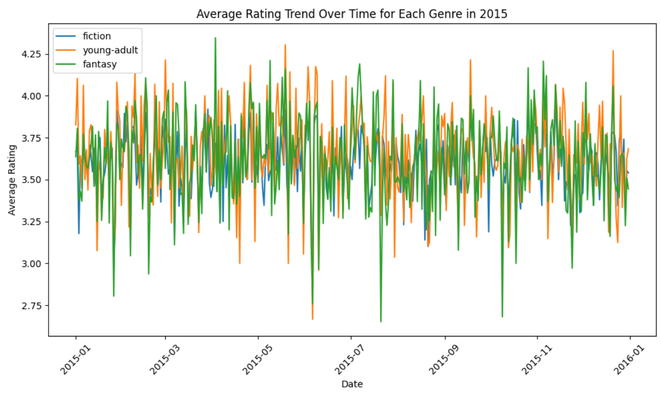


Fig. Average Rating Trend for top 3 genres

We also performed EDA on the distribution of average ratings and total number of books based on genre. The visualizations are shown below.

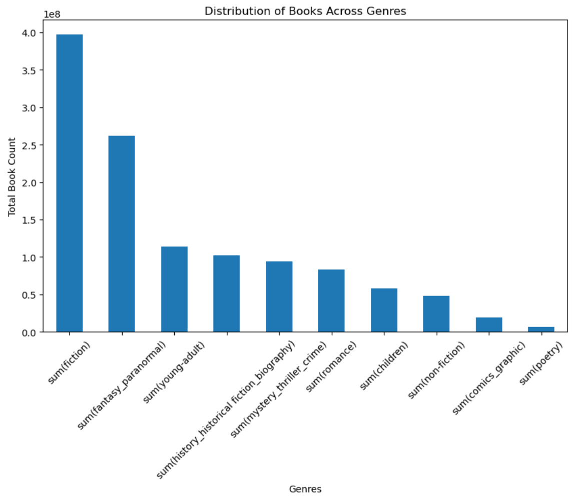


Fig. Distribution of books across each genre

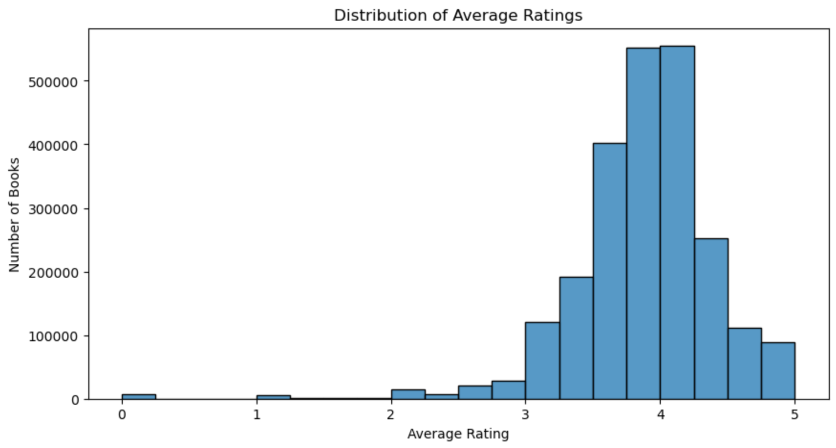


Fig. Average rating distribution

- “Fiction” genre has the maximum number of books.
- The average rating distribution is approximately a skewed Gaussian distribution and the maximum no of ratings exists between 3.75 to 4.25.

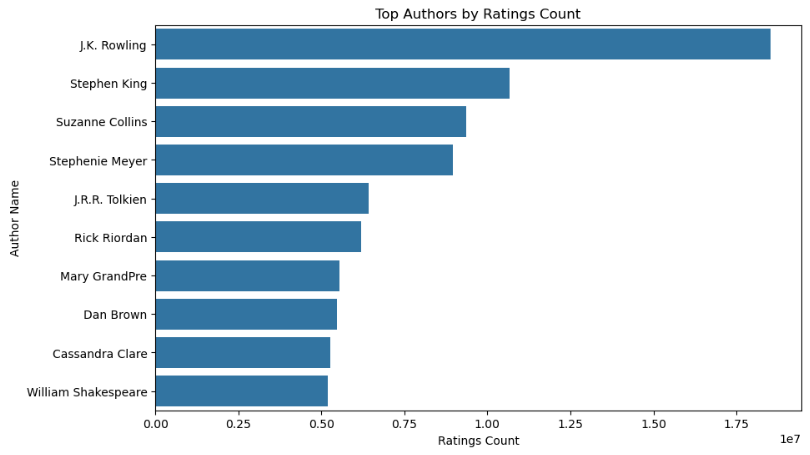


Fig. Top authors

The popularity difference between JK Rowling and Stephen King is relatively significant compared to other authors. One of the reasons for this could be because Stephen King has a niche genre of horror-based fiction, which obviously will have a niche audience compared to people who enjoy reading the Harry Potter series.

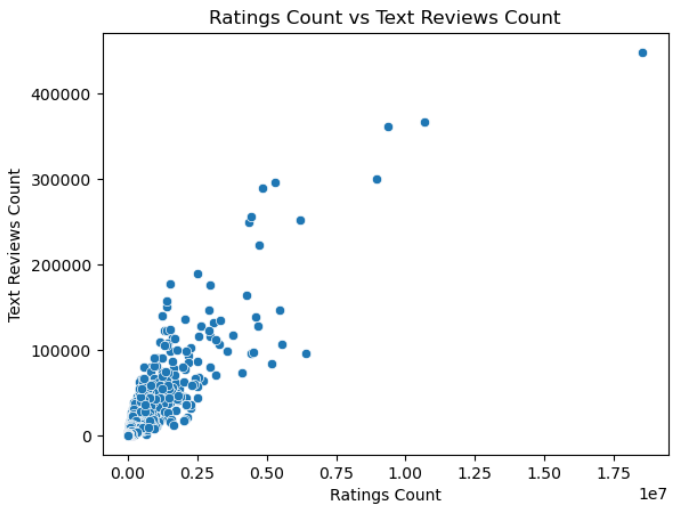


Fig. Rating vs Text Reviews count

There's a general trend that as the 'Ratings Count' increases, the 'Text Reviews Count' also tends to increase. This suggests that more popular books, which get more ratings, also tend to get more text reviews.

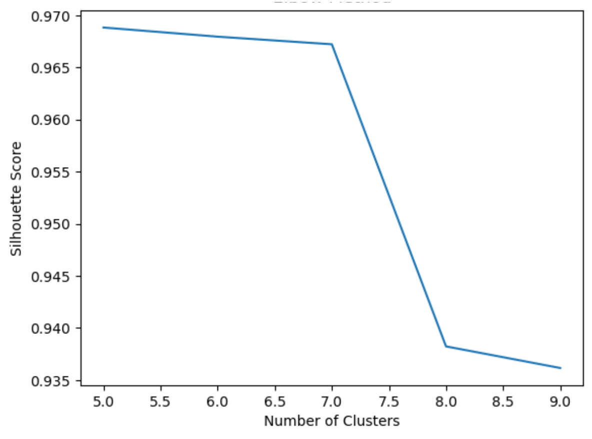
The data points are denser at the lower end of both axes, indicating more books with fewer ratings and reviews. This is typical for book data as only a few books become popular.

K-Means Clustering for Genres

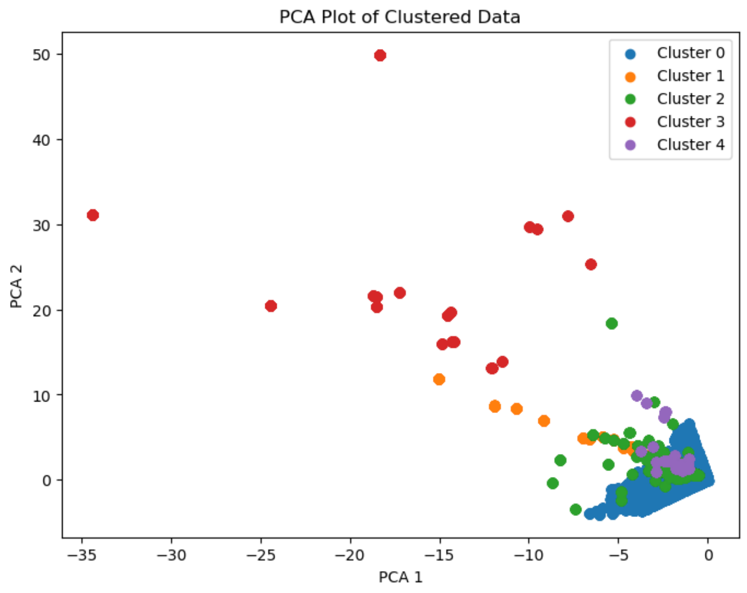
We performed K-means clustering in order to better understand reader preferences and inform our book categorization strategy.
We followed the below steps:

1. **Preprocess the data by scaling**
   First, we used a VectorAssembler to combine selected columns, excluding the first column, into a single vector column named "features." The assembled features are then standardized using a StandardScaler, where the input features are scaled to have zero mean and a standard deviation of 1. The resulting scaled features are stored in a new column called "scaled_features." This sequence of operations is crucial for preparing the data before applying machine learning algorithms, as it ensures that features are appropriately structured and normalized, enhancing the performance and interpretability of subsequent machine learning models. This preprocessing step is particularly beneficial when features in the dataset have different scales or units, as it promotes more effective and reliable model training.

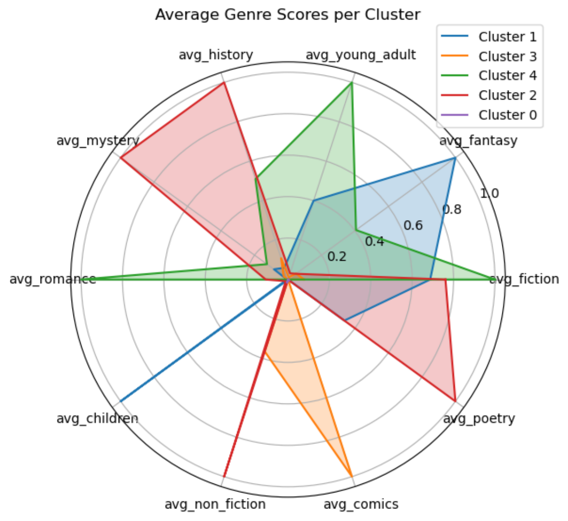
2. **Use Silhouette method to get an optimal “k” value**
   The objective is to identify the optimal number of clusters for the given data using the elbow method. The code iteratively fits k-means models with varying cluster numbers (ranging from 5 to 9) to a pre-scaled dataset. For each iteration, the Silhouette Score, a metric assessing clustering quality, is calculated using the ClusteringEvaluator. These scores are then stored in a list. The subsequent visualization using Matplotlib portrays the Silhouette Scores against the number of clusters, aiding in the identification of the optimal cluster count by locating the "elbow" point on the plot. This comprehensive approach enables a data-driven determination of the most suitable clustering configuration, crucial for understanding the inherent structure within the dataset.



3. **PCA for dimensionality reduction**
   Next we utilized PySpark's machine learning and visualization capabilities to perform dimensionality reduction and visualize clustered data. It first converts a PySpark DataFrame into a Pandas DataFrame for ease of visualization. Then, it applies Principal Component Analysis (PCA) with two components to reduce the dimensionality of the data. The resulting PCA features are extracted and plotted in a 2D scatter plot, where each point represents a data instance colored by its assigned cluster. This plot provides a visual representation of the clustered data in a reduced-dimensional space, aiding in the interpretation and analysis of the clustering results.



4. **Spider chart visualization**
   The spider chart represents the average scores of books across various genres for five different clusters. Each spoke of the chart represents a different genre, and the distance from the center represents the average score in that genre for the cluster.

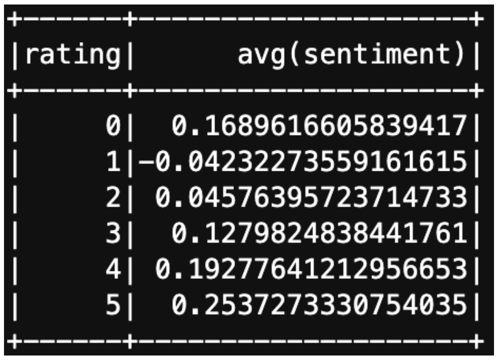


Sentiment Analysis

This analysis was performed using the TextBlob library, known for its natural language processing capabilities.

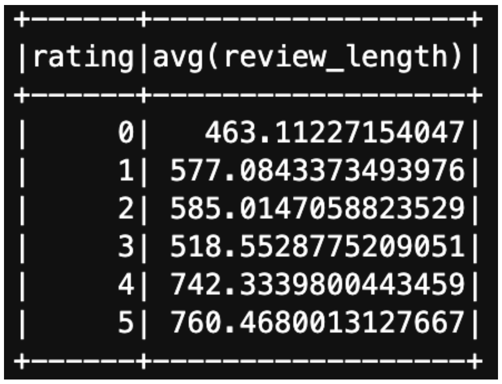
We created a UDF called 'sentiment_udf.' This UDF leverages TextBlob's 'sentiment.polarity' method to calculate sentiment polarity scores. These scores range from -1 (indicating very negative sentiment) to 1 (indicating very positive sentiment), with 0 being neutral. The results are stored as strings within the DataFrame.

Firstly, we explore 'Sentiment Analysis by Rating.' It's evident that higher-rated reviews, such as those with a rating of 5, tend to express more positive sentiment. Conversely, lower-rated reviews, like those with a rating of 1, convey more negative sentiment. This aligns with our intuition that readers are more likely to express positivity for books they enjoyed.



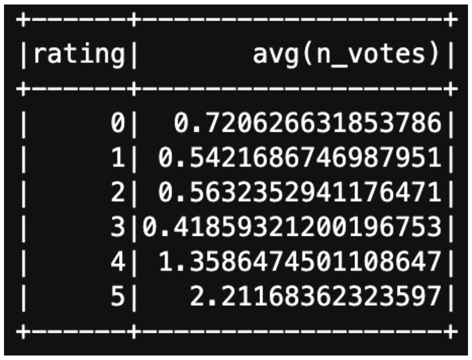
```
+------+--------------------+
|rating|      avg(sentiment)|
+------+--------------------+
|     0|  0.1689616605839417|
|     1|-0.04232273559161615|
|     2| 0.04576395723714733|
|     3|  0.1279824838441761|
|     4| 0.19277641212956653|
|     5|  0.2537273330754035|
+------+--------------------+
```

Next, we uncover 'Sentiment Analysis by Review Length.' Longer reviews are associated with higher ratings, notably with a rating of 5. This implies that readers who rate books more positively tend to provide more detailed and descriptive feedback. Longer reviews often indicate a deeper level of engagement and enthusiasm.

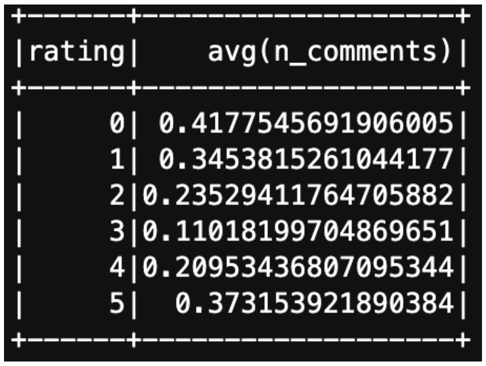


```
+------+------------------+
|rating|avg(review_length)|
+------+------------------+
|     0|   463.11227154047|
|     1| 577.0843373493976|
|     2| 585.0147058823529|
|     3| 518.5528775209051|
|     4| 742.3339800443459|
|     5| 760.4680013127667|
+------+------------------+
```

Finally, we delve into 'Sentiment Analysis by Engagement Metrics.' Here, we observe that higher-rated reviews receive more reader engagement, with the average number of votes and comments increasing significantly. Reviews with a rating of 5, in particular, stand out, as they tend to attract the most votes and comments, indicating heightened reader interaction.



```
+------+------------------+
|rating|      avg(n_votes)|
+------+------------------+
|     0|  0.720626631853786|
|     1| 0.5421686746987951|
|     2| 0.5632352941176471|
|     3|0.41859321200196753|
|     4| 1.3586474501108647|
|     5|   2.21168362323597|
+------+------------------+
```

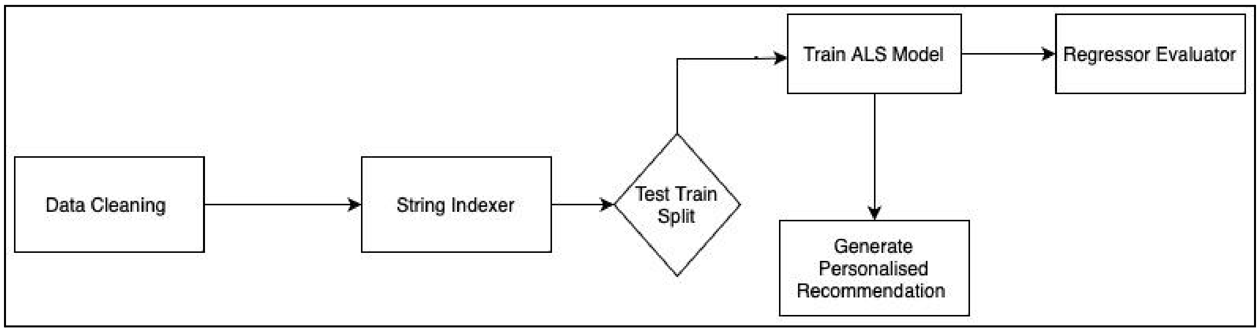


```
+------+------------------+
|rating|   avg(n_comments)|
+------+------------------+
|     0| 0.4177545691906005|
|     1| 0.3453815261044177|
|     2|0.23529411764705882|
|     3|0.11018199704869651|
|     4|0.20953436807095344|
|     5|  0.373153921890384|
+------+------------------+
```

In conclusion, this sentiment analysis provides a comprehensive understanding of user sentiments, review length patterns, and reader engagement dynamics on Goodreads. These insights hold the potential to shape book recommendations and engagement strategies, ultimately enhancing user experiences and fostering positive interactions within the Goodreads community. Leveraging the power of PySpark and TextBlob, this analysis efficiently processes vast data, enabling data-driven decisions for the platform's continuous growth and improvement.

Recommendation System

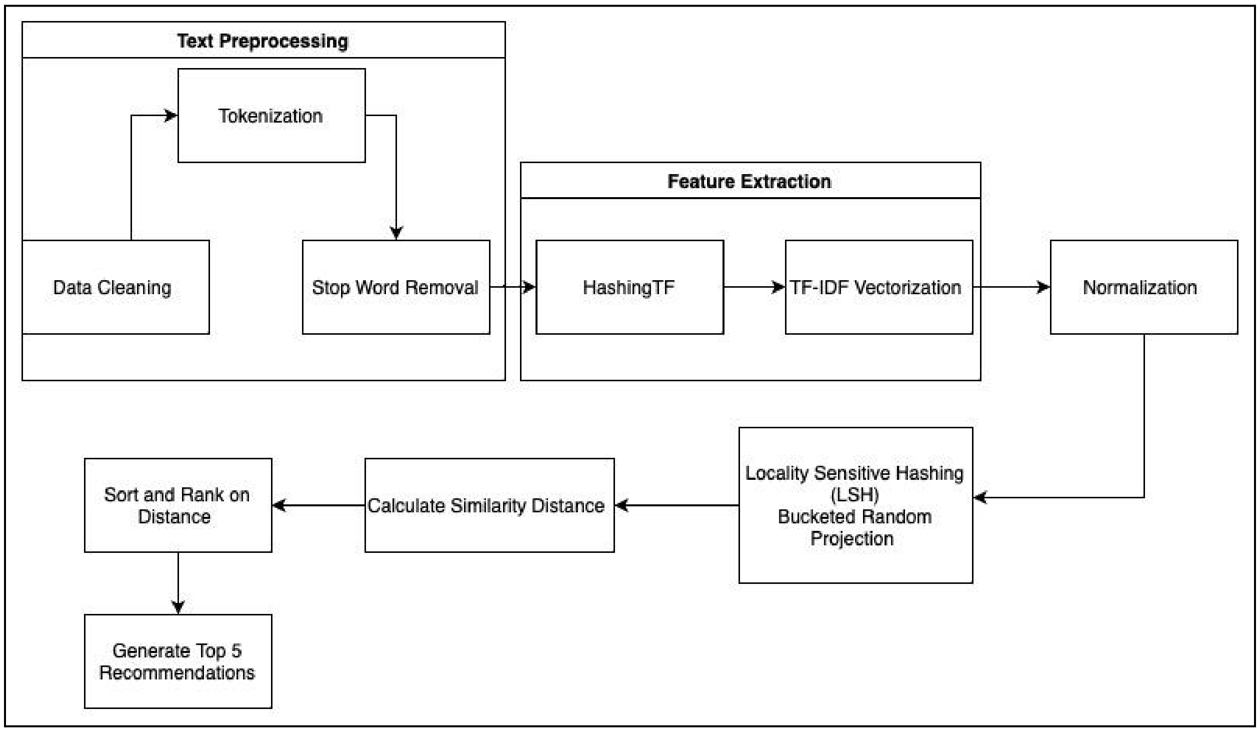
1. Collaborative Recommendation System Using ALS



Collaborative filtering provides personalized recommendations based on a user's past behavior and the behavior of similar users. It is effective in capturing complex user preferences and patterns without the need for explicit book features.

The Collaborative Filtering system was realized through the Alternating Least Squares (ALS) algorithm within PySpark's MLlib. ALS is a matrix factorization technique that decomposes the user-item interaction matrix into latent factors for users and items (books in this case), facilitating the prediction of missing entries (unrated books).
Key to this process is the `StringIndexer`, which converts string identifiers (user and book IDs) into numerical values suitable for the ALS algorithm. The performance of the ALS model was quantified using the Root Mean Square Error (RMSE), which was calculated to be 1.61645. This metric indicates the average deviation of the predicted ratings from the actual ratings on a scale of 1-5, suggesting a moderate level of accuracy.

2. Content-Based Recommendation System Using LSH


Text Preprocessing
Tokenization
Data Cleaning
Stop Word Removal
Feature Extraction
HashingTF
TF-IDF Vectorization
Normalization
Locality Sensitive Hashing (LSH) Bucketed Random Projection
Calculate Similarity Distance
Sort and Rank on Distance
Generate Top 5 Recommendations

The Content-Based system, on the other hand, focused on analyzing book descriptions. The initial step involved preprocessing the text data, where descriptions were first 'tokenized' (splitting text into individual words) and then 'stop words' (commonly used words with little semantic value) were removed. This was followed by feature extraction using HashingTF

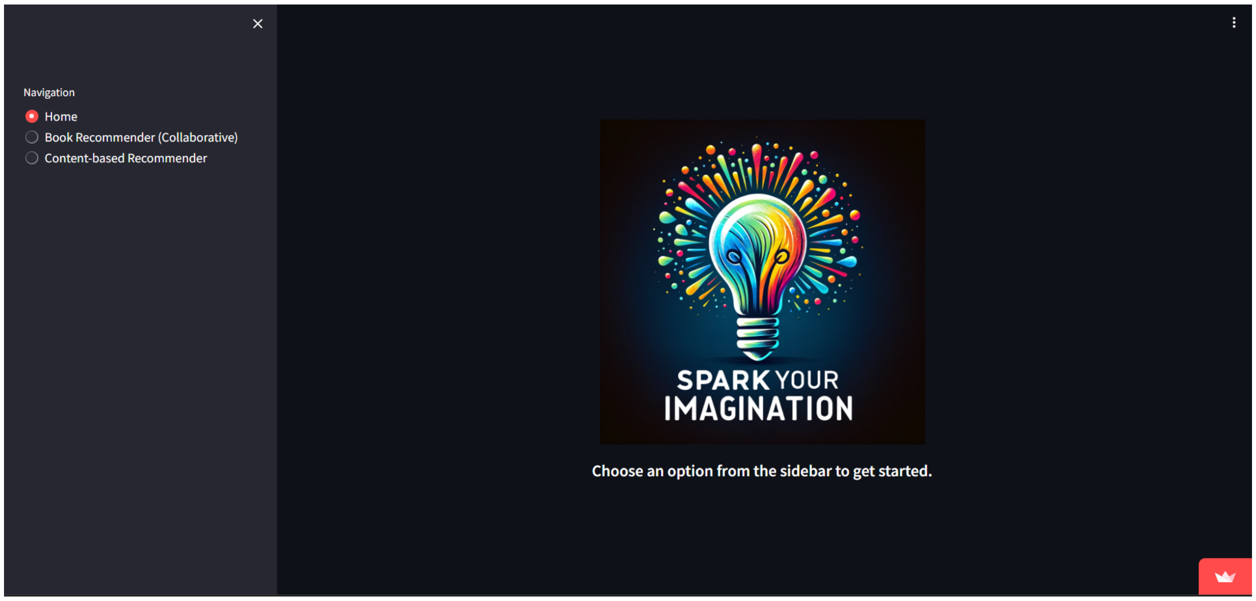
(Hashing Term Frequency) and IDF (Inverse Document Frequency), effectively transforming text data into numerical form while considering the importance of words within the dataset. These features were then normalized to ensure uniformity in scale.
For similarity computation, the Bucketed Random Projection LSH technique was applied, an efficient method for approximating nearest neighbors in high-dimensional data. LSH works by hashing input items in such a way that similar items map to the same "buckets" with high probability, thereby significantly reducing the complexity of similarity computations. We chose Bucketed Random Projection for its efficiency and effectiveness with high-dimensional textual data. It reduces computational complexity while maintaining the integrity of content similarities, making it an excellent choice for large datasets like ours.
The model was configured with specific parameters like `bucketLength` and `numHashTables`, critical in determining the granularity and accuracy of the hash-based approximations.
Approximate similarity joins were performed using LSH to identify books with similar descriptions. LSH is efficient for large-scale datasets and provides a way to approximate nearest neighbors searches without exhaustive pairwise comparisons. This method leverages the content (descriptions) of the books, making it effective in recommending similar books based on their textual content, regardless of user interaction data. The system provided the top 5 similar books for each book based on their descriptions. This approach is particularly useful for discovering similar books and providing recommendations based on book content.

The two systems utilize different data sources and methodologies, offering complementary strengths. The collaborative system excels in personalization based on user behavior, while the content-based system focuses on the characteristics of the books themselves.

The combination of these two systems can potentially offer a comprehensive recommendation engine, balancing personalized suggestions with content relevance.

Streamlit

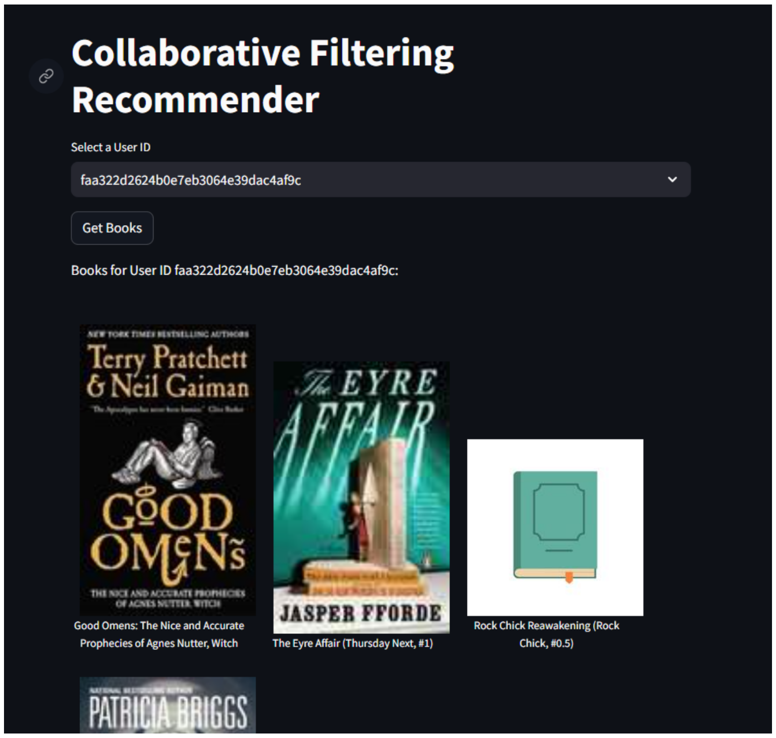
Dashboard Link: https://sparkyourimagination.streamlit.app/



Choose an option from the sidebar to get started.

Fig. Dashboard Home

We utilized the Streamlit framework to build an interactive web application for book recommendations and visual representation of book data.



Collaborative Filtering Recommender

Select a User ID

faa322d2624b0e7eb3064e39dac4af9c

Books for User ID faa322d2624b0e7eb3064e39dac4af9c:

Good Omens: The Nice and Accurate Prophecies of Agnes Nutter, Witch

The Eyre Affair (Thursday Next, #1)

Rock Chick Reawakening (Rock Chick, #0.5)

Fig. Collaborative Recommendation page

We showcase book recommendations based on collaborative filtering and content-based methods. It leverages Streamlit for its ease of creating user interfaces and interactive elements.

The collaborative filtering section allows users to select a user ID and generates top book recommendations based on that user's reading history.

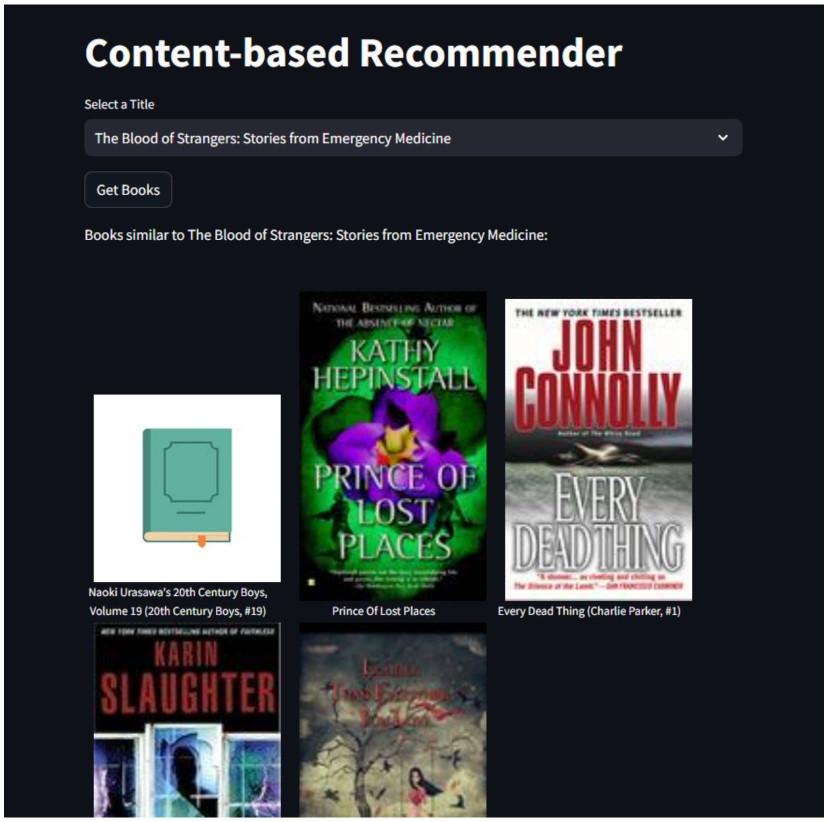

Content-based Recommender
Select a Title
The Blood of Strangers: Stories from Emergency Medicine
Get Books
Books similar to The Blood of Strangers: Stories from Emergency Medicine:
Naoki Urasawa's 20th Century Boys, Volume 19 (20th Century Boys, #19)
Prince Of Lost Places
Every Dead Thing (Charlie Parker, #1)

Fig. Content-based recommendation page

On the other hand, the content-based recommendation segment assists users in finding books similar to a selected title. Both sections display book covers alongside their titles, enhancing the visual appeal and aiding users in making informed choices.

The recommendation output is dynamically generated based on the user's selection, providing tailored and relevant book suggestions and enhancing the overall user experience for users interacting with the application.

Future Scope

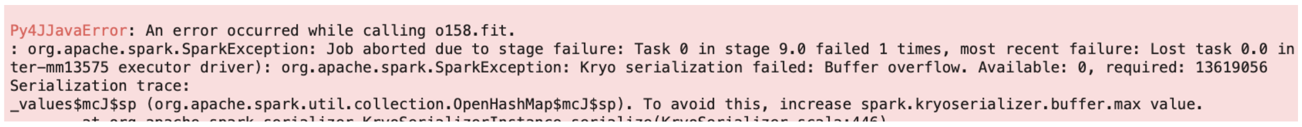
1. **GPT Integration**
    a. Integrating GPT models will enable the system to engage users in natural language interactions, empowering them to converse and articulate preferences for more nuanced book recommendations.
    b. It will help analyze conversational cues, context, and user queries to offer personalized and contextually relevant book suggestions. It will elevate the user experience by providing tailored recommendations aligned with individual preferences and tastes**.**
2. **Real-Time Analysis with Kafka**
    a. Implementing Kafka for real-time data streaming allows the system to capture, process, and derive immediate insights from dynamic, streaming Goodreads data.
    b. This ensures that the recommendation engine stays abreast of evolving user behavior, trending books, and new releases, facilitating up-to-the-minute recommendations that reflect current interests and preferences.
3. **Scalability and Data Handling**
    a. Preparing the system for increased data volumes involves optimizing infrastructure and data processing methods.
    b. We can use scalable solutions such as distributed computing or cloud-based architectures to ensure seamless handling of growing datasets, enabling efficient data processing, storage, and retrieval even as the user base expands.
4. **Comprehensive Recommendation System**
    a. The system's evolution into a comprehensive recommendation engine involves the integration of advanced recommendation algorithms, user feedback loops, and diverse content sources.
    b. The updated system will offer refined, adaptive, and diverse book recommendations, continuously learning from user interactions, ensuring a more enriching and personalized user experience.

kafka

Limitations

1. **Sampled Data**
    a. Due to resource constraints, specific datasets have been sampled rather than using complete data. (Interactions dataset has 229M records)
    b. This sampling might have impacted the accuracy of the insights derived, or models developed, limiting the depth of analysis and recommendations provided.
2. **Limited Data Availability**
    a. The available dataset spans only until 2017, which might restrict the system's ability to capture recent trends or changes in user behavior.
    b. Newer data could significantly influence the accuracy and relevance of recommendations, especially in a dynamic domain like book preferences.
3. **JupyterHub Resource Constraints**

    Many buffer errors and timeouts were happening all the time, which is one reason why we had to sample our dataset and only consider a small part of it for demonstration purposes.



```
Py4JJavaError: An error occurred while calling o158.fit.
: org.apache.spark.SparkException: Job aborted due to stage failure: Task 0 in stage 9.0 failed 1 times, most recent failure: Lost task 0.0 in
ter-mm13575 executor driver): org.apache.spark.SparkException: Kryo serialization failed: Buffer overflow. Available: 0, required: 13619056
Serialization trace:
_values$mcJ$sp (org.apache.spark.util.collection.OpenHashMap$mcJ$sp). To avoid this, increase spark.kryoserializer.buffer.max value.
```

Fig. Kryo Serialization buffer overflow error

Conclusion

Why is it Big Data?

This project, currently managing 100 GB of diverse book data, aligns with big data characteristics and holds potential for further expansion into a quintessential big data venture. The volume itself is significant, necessitating Apache Spark's distributed computing framework for efficient processing, a key attribute of big data initiatives. The variety of data, from structured metadata to unstructured text in reviews and descriptions, exemplifies big data's multifaceted nature, requiring complex data integration and transformation techniques.

Looking ahead, incorporating real-time streaming data, potentially from sources like Goodreads, could introduce a velocity aspect, elevating the project's big data profile. Integrating Kafka for real-time data handling would demand systems capable of high-throughput and low-latency processing, essential for dynamic analytics. The project's complexity is already evident in its use of advanced analytical methods such as sentiment analysis and K-means clustering, showcasing the computational intensity and sophistication typical in big data environments.

Facing scalability, data quality, and privacy challenges, the project mirrors big data's inherent complexities while promising impactful insights into reader preferences and market trends. This potential expansion into real-time data processing and the ongoing challenges and opportunities highlight the project's alignment with big data principles and its capacity to evolve into a more comprehensive, real-time analytical system.