

Documenting the Creation of a Book Recommendation System – Personal Project

Symbol Okonkwo

Introduction

The trajectory of developing a book recommendation system was propelled by a significant shift in my career trajectory. The transition from healthcare to the realm of technology, with a specific focus on becoming a data analyst, culminated in the conception of a project that seamlessly intertwines my technical aspirations with my love for reading. This endeavour manifested as the amalgamation of my pursuit of data analytics expertise and the practical application of crafting a book recommendation system using the versatile Python programming language.

Problem Statement

The crux of this project originates from a shared challenge - the difficult task of selecting the next book from a vast sea of literary options. As a passionate reader, I have personally experienced the problem that comes with this decision-making process. Leveraging my adeptness in coding, I identified an opportunity to mitigate this dilemma through technology. The choice of Python as the principal tool emerged as the strategic means to address this issue.

Objectives

At its core, my objective in embarking on this project was twofold - to enhance and develop my technical skills and simultaneously create a practical and effective book recommendation system.

Data Collection and Preprocessing

The journey to procure a suitable dataset led me through the intricacies of Goodreads. Unfortunately, the unavailability of developer keys steered me toward uncharted waters. Negotiating these obstacles, the avenue of web scraping emerged as the logical conduit, culminating in the selection of Octoparse as the optimal tool. With unwavering diligence, I meticulously curated an expansive catalogue of over 750 books, spanning diverse genres, authors and publishers within the Goodreads domain. Exploiting Octoparse's capabilities, I meticulously collected important attributes for each book, safeguarding data fidelity.

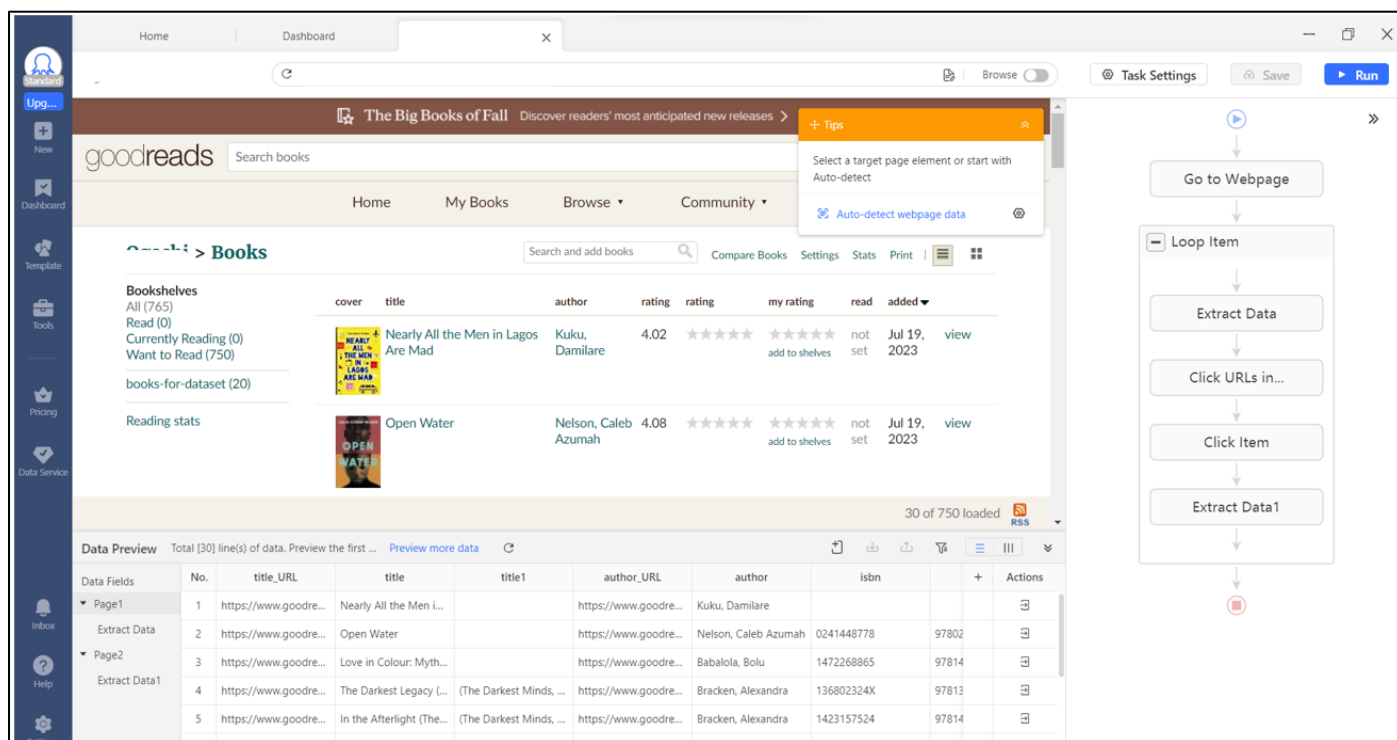


Figure 1: Web Scraping Workflow Using Octoparse

In this endeavour, I employed the Octoparse web scraping tool to curate a comprehensive dataset of over 750 books. Figure 1 above showcases a snapshot of the web scraping workflow that I designed within Octoparse. This workflow was tailored to navigate through Goodreads' book listings, extracting relevant attributes for each book. While the web scraping process presented its own set of challenges, it ultimately facilitated the creation of a valuable dataset.

Exploratory Data Analysis (EDA):

To shape the framework of the recommendation system, I delved into exploratory data analysis (EDA) within the realm of Google Colab. The insights derived from the analysis of genre distributions and prolific authors played a transformative role in sculpting the system's architecture. Notably, an epiphany arose through the analysis of word counts in titles and descriptions, fundamentally shaping the algorithm's methodology for assessing relevance.

Genre Distribution Analysis:

One of the initial steps in the exploratory analysis was the examination of genre distribution among the collected books.

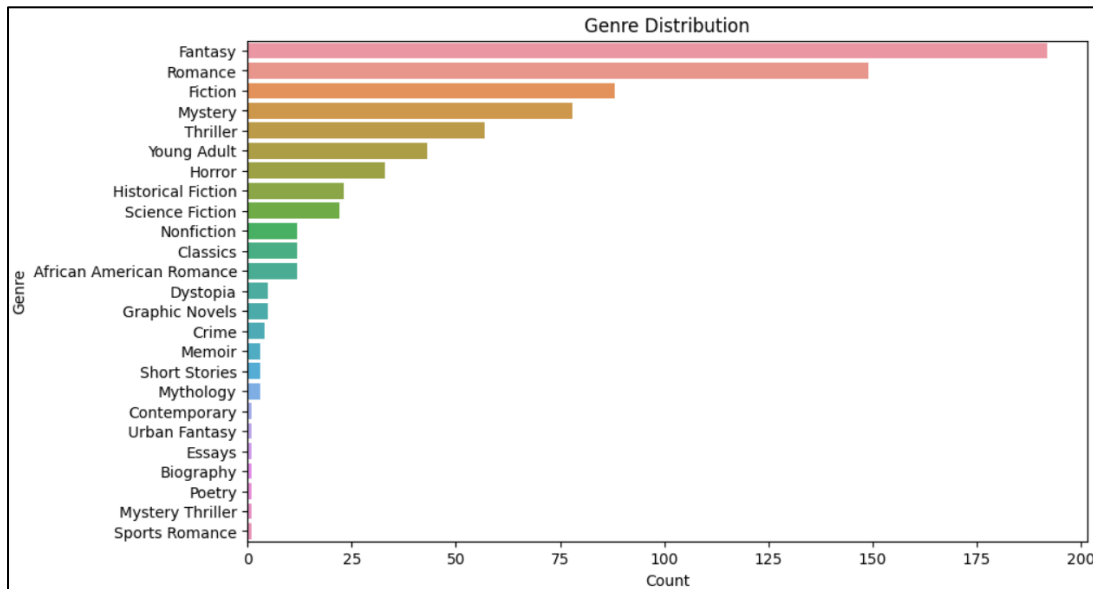


Figure 2: Genre Distribution of Books

The visualisation in Figure 2 illustrates the distribution of book genres within the dataset. This analysis highlighted the most prevalent genres and aided in understanding the landscape of genres for recommendation.

Prolific Authors Impact:

Another critical facet of the EDA was understanding the contribution of prolific authors to the dataset.

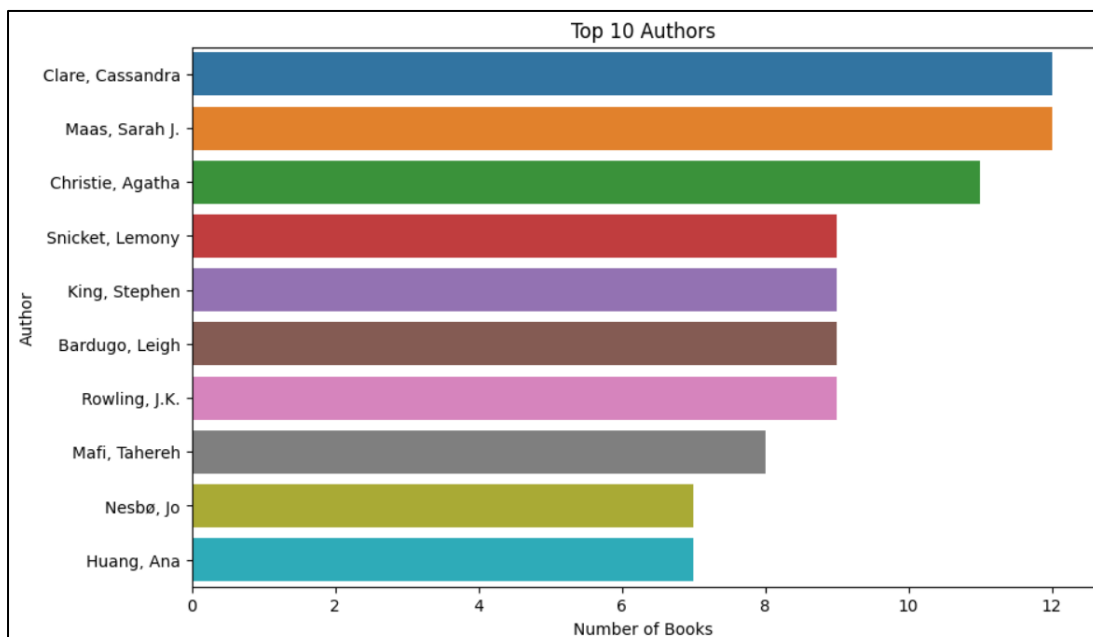


Figure 3: Top 10 Authors by Book Count

In Figure 5, I was able to delve deeper into the analysis by visualising the distribution of title word counts. This visualisation aids in understanding title length patterns and potential relevance to recommendation scores.

Choosing a Recommendation Algorithm

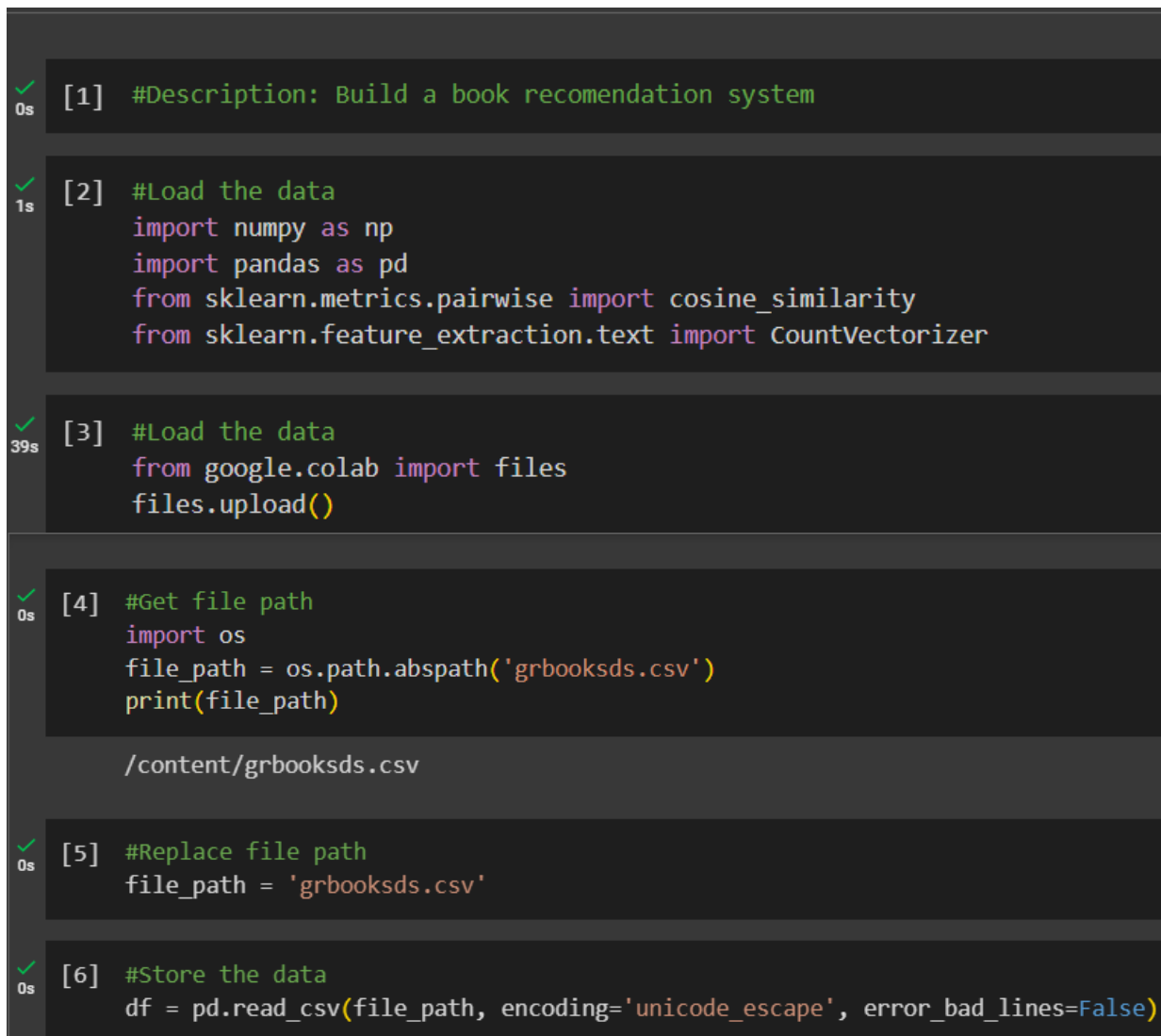
The crossroads of recommendation algorithms demanded a meticulous evaluation of collaborative and content-based filtering approaches. Electing a content-based approach, rooted in cosine similarity, was propelled by pedagogical intent. This avenue harmonised seamlessly with the manipulation of textual data, allowing a focused exploration of the project's technical intricacies.

Implementation

The implementation of the chosen content-based recommendation algorithm was orchestrated meticulously through the prism of Python. Capitalising on libraries such as pandas, scikit-learn's CountVectorizer, and cosine_similarity, the orchestration process was executed with precision. A pivotal augmentation was the incorporation of user input for book titles, thus enhancing user engagement and system usability.

Implementing the book recommendation system involved a step-by-step process that combined data preprocessing, algorithm selection, and user interaction. The following code snippets and explanations provide insights into the key components of the implementation.

Loading and Processing Data:



```
[1] #Description: Build a book recommendation system

[2] #Load the data
import numpy as np
import pandas as pd
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.feature_extraction.text import CountVectorizer

[3] #Load the data
from google.colab import files
files.upload()

[4] #Get file path
import os
file_path = os.path.abspath('grbooksds.csv')
print(file_path)

/content/grbooksds.csv

[5] #Replace file path
file_path = 'grbooksds.csv'

[6] #Store the data
df = pd.read_csv(file_path, encoding='unicode_escape', error_bad_lines=False)
```

Figure 6: Loading and Preprocessing Data

Figure 6 demonstrates the process of importing necessary libraries and loading the dataset to perform initial data preprocessing. These steps were essential for preparing the data for subsequent analysis and recommendation generation.

Creating Combined Features:

```
[ ] #Create list of columns to keep
columns=['title', 'author', 'genre', 'publisher']

[ ] #Create a function to combine the important/relevant columns
def combine_features(data):
    features = []
    for i in range(0, data.shape[0]):
        features.append( data['title'][i] + ' '+data['author'][i] + ' '+ data['genre'][i] + ' '+ data['publisher'][i])

    return features

[ ] #Create a column to store the combined features
df['combined_features']=combine_features(df)

#Show the data
df
```

Figure 7: Creating Combined Features

Figure 7 showcases the creation of combined features by concatenating relevant columns such as title, author, genre, and publisher. This process lays the foundation for content-based analysis and recommendation calculation.

Calculating Cosine Similarity:

```
[ ] #Convert the text from the new column to a matrix of word counts
cm = CountVectorizer().fit_transform(df['combined_features'])

[ ] #Get the cosine similarity matrix from the count matrix
cs = cosine_similarity(cm)

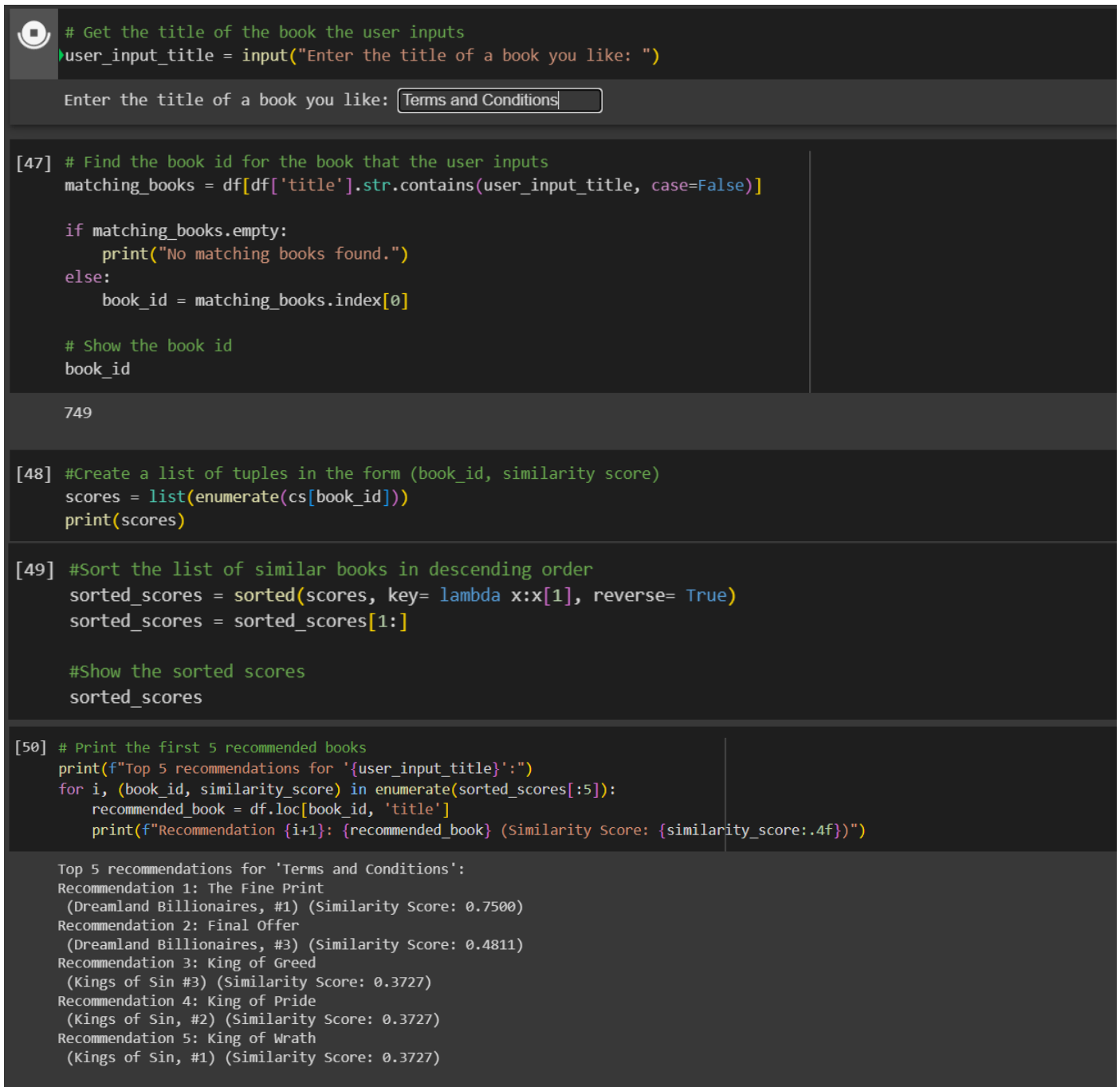
#Print the scores
print(cs)
```

```
[[1.         0.         0.14285714 ... 0.08908708 0.         0.07715167]
 [0.         1.         0.         ... 0.         0.         0.         ]
 [0.14285714 0.         1.         ... 0.08908708 0.07715167 0.15430335]
 ...
 [0.08908708 0.         0.08908708 ... 1.         0.48112522 0.48112522]
 [0.         0.         0.07715167 ... 0.48112522 1.         0.75         ]
 [0.07715167 0.         0.15430335 ... 0.48112522 0.75         1.         ]]
```

Figure 8: Calculating Cosine Similarity

Here, the text data from the combined_features column is transformed into a matrix of word counts using CountVectorizer. Figure 8 illustrates the calculation of cosine similarity scores based on the combined features. This step was pivotal for determining the similarity between books and generating meaningful recommendations.

User Interactions and Recommendations:



```
# Get the title of the book the user inputs
user_input_title = input("Enter the title of a book you like: ")

Enter the title of a book you like: Terms and Conditions

[47] # Find the book id for the book that the user inputs
matching_books = df[df['title'].str.contains(user_input_title, case=False)]

if matching_books.empty:
    print("No matching books found.")
else:
    book_id = matching_books.index[0]

# Show the book id
book_id

749

[48] #Create a list of tuples in the form (book_id, similarity score)
scores = list(enumerate(cs[book_id]))
print(scores)

[49] #Sort the list of similar books in descending order
sorted_scores = sorted(scores, key= lambda x:x[1], reverse= True)
sorted_scores = sorted_scores[1:]

#Show the sorted scores
sorted_scores

[50] # Print the first 5 recommended books
print(f"Top 5 recommendations for '{user_input_title}':")
for i, (book_id, similarity_score) in enumerate(sorted_scores[:5]):
    recommended_book = df.loc[book_id, 'title']
    print(f"Recommendation {i+1}: {recommended_book} (Similarity Score: {similarity_score:.4f})")

Top 5 recommendations for 'Terms and Conditions':
Recommendation 1: The Fine Print
(Dreamland Billionaires, #1) (Similarity Score: 0.7500)
Recommendation 2: Final Offer
(Dreamland Billionaires, #3) (Similarity Score: 0.4811)
Recommendation 3: King of Greed
(Kings of Sin #3) (Similarity Score: 0.3727)
Recommendation 4: King of Pride
(Kings of Sin, #2) (Similarity Score: 0.3727)
Recommendation 5: King of Wrath
(Kings of Sin, #1) (Similarity Score: 0.3727)
```

Figure 9: User Interaction and Recommendation Generation

Figure 9 provides insight into the user interaction aspect of the system. Users provide a book title they like [Terms and Conditions], and the system then finds matching books, calculates similarity scores, sorts the scores, and generates the top 5 recommendations based on user input.

Evaluation

System evaluation enveloped both quantitative and qualitative measurements. While quantitative metrics were hindered by the lack of real user data, qualitative insights collected from peers, combined with the alignment of recommendations to user profiles on Goodreads. Through these things, I was able to carry out a comprehensive assessment of system performance.

Challenges Faced

Throughout this developmental journey, I encountered a variety of challenges. Spanning the range from circumventing data collection obstacles and addressing missing values to understanding the intricacies of web-scraping, overcoming these challenges not only refined my problem-solving skills but also gave the project technical dimension with depth and substance.

Lessons Learned

On reflection, the project highlights my adaptability as a core trait. The intricate difficulties that come with real-world data shows the crucial significance of accurate data preprocessing. In addition, the project clearly displays the efficiency in structured planning as well as the solid commitment to continuous learning.

Conclusion and Future Work

In conclusion, the development of the book recommendation system marks a significant step in my journey into the realm of data analytics and technology. By integrating my passion for reading with newfound coding skills, this project stands as a testament to the collaboration between technical proficiency and personal interests. Through meticulous data preprocessing, algorithm selection, and user interaction, I've been able to create a functional and interactive tool that offers users tailored book recommendations.

Key Takeaways:

Personal Growth: This project allowed me to bridge the gap between my aspirations as a data analyst and my love for literature. It showcased the practical application of data analytics in a domain that resonates with my personal interests.

Personal Growth: This project allowed me to bridge the gap between my aspirations as a data analyst and my love for literature. It showcased the practical application of data analytics in a domain that resonates with my personal interests.

Technical Mastery: The implementation process deepened my understanding of data preprocessing, cosine similarity calculations, and user input integration. The hands-on experience honed my skills and provided insights into the complexities of recommendation systems.

Problem Solving: Overcoming challenges such as data collection roadblocks and missing values underscored the importance of creative problem-solving. Adapting to unexpected hurdles became a valuable learning experience.

Future Directions:

While the current iteration of the book recommendation system demonstrates its feasibility and functionality, several avenues for further improvement and expansion warrant exploration:

Expanding the Dataset: To further enhance the recommendation accuracy and diversity, one of the immediate future steps is expanding the size and diversity of the dataset. A larger dataset encompassing a wider range of genres, authors, and time periods could lead to more nuanced and accurate recommendations.

Incorporating User Ratings: Introducing user ratings and feedback could enhance the accuracy of recommendations. A collaborative filtering approach could be considered to leverage collective preferences.

Hybrid Recommendation Techniques: Combining content-based and collaborative filtering methods could lead to more robust recommendations. A hybrid approach could harness the strengths of both techniques while mitigating their individual limitations.

Machine Learning Enhancements: Exploring machine learning algorithms like matrix factorisation or deep learning architectures could potentially refine the recommendation process, especially with larger datasets.

Enhancing User Interface: Creating a user-friendly interface, such as a website or application, could democratise access to the recommendation system, making it more widely accessible to book enthusiasts.

Testing with Real Users: Conducting user testing with a diverse group of individuals and gathering feedback could provide valuable insights into user experiences and preferences, aiding in further system refinement.

Exploring Diverse Data Sources: Expanding the dataset to include more diverse sources, such as user-generated reviews and external APIs, could enrich the recommendation system's content and adaptability.

In essence, the book recommendation system project has been a gratifying learning experience that bridges my career transition and literary passions. While the current system serves as a foundation, the future holds

opportunities for innovation and growth, allowing the system to evolve into a more sophisticated and user-centric tool.

References

The trajectory of this project was enriched by a diverse spectrum of resources, spanning online tutorials and courses, programming forums, YouTube tutorials, and immersive engagements with tools including Octoparse, Google Colab, pandas, and scikit-learn.