ReadSmart:

Book Recommender System

By:

Routhu Sai Praneeth

Gandla Sasank

T Bala Sai Vineeth

Sreyas rejil

INTRODUCTION

1.1 Overview

The Book Recommender System is a project created using Python and machine learning techniques. Its purpose is to provide personalized book recommendations based on users' preferences and similarities with other readers. By employing collaborative filtering and popularity-based algorithms, the system analyses user behavior and book characteristics to generate accurate and relevant suggestions.

1.2 Purpose

The main goal of this project is to enhance the book discovery process for users by offering tailored recommendations. In today's digital world, with countless books available, finding the right one can be overwhelming. Many people rely on recommendations from others to guide their choices. The Book Recommender System automates this process by using machine learning algorithms to analyze user preferences and suggest books that match their interests.

Through collaborative filtering, the system identifies users with similar reading tastes and suggests books that have been well-received by those with similar preferences. Additionally, the popularity-based recommendation component ensures that users are exposed to popular and highly acclaimed books that might align with their interests.

The aim of this project is to simplify the book selection journey, introduce readers to new authors and genres, and enhance their overall reading experience. Whether users are enthusiastic readers looking for fresh suggestions or newcomers seeking a starting point, the Book Recommender System serves as a valuable tool for discovering captivating books.

Literature Survey

2.1 Existing Problems

In the domain of book recommendation systems, there have been various approaches to tackle the challenge of providing personalized suggestions to readers. One common issue is the lack of accurate user information, as user IDs (User-ID) in the Book-Crossing dataset have been anonymized and mapped to integers. Demographic data like location and age may be available but can contain NULL values. This poses a challenge in understanding individual user preferences and tailoring recommendations accordingly.

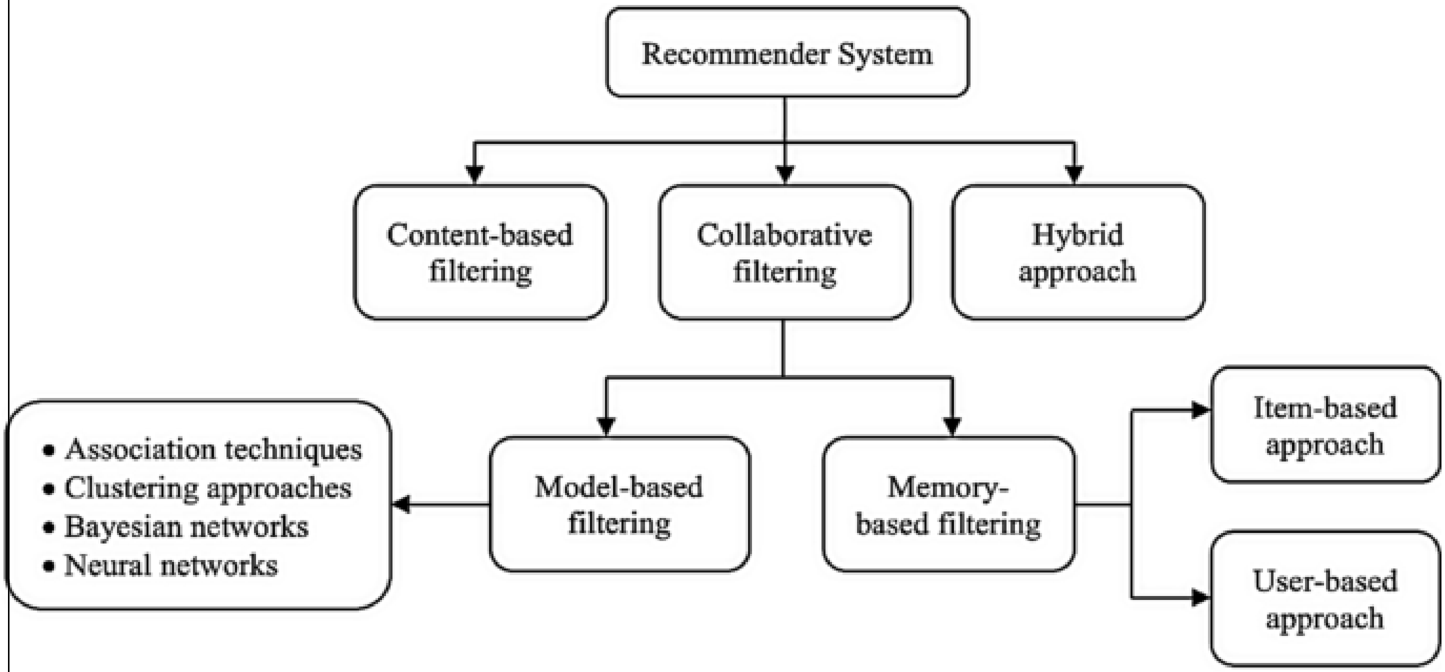
2.2 Proposed Solution

To address the existing problem, this project utilizes the Book-Crossing dataset, which consists of three main files: Users, Books, and Ratings. The collaborative filtering approach incorporates machine learning techniques, such as cosine similarity, to identify users with similar reading preferences.

By analysing their behaviour and shared interests, the system generates personalized recommendations based on books that have been positively rated by these similar users. This method allows for a more precise understanding of individual preferences, even without explicit user demographic data.

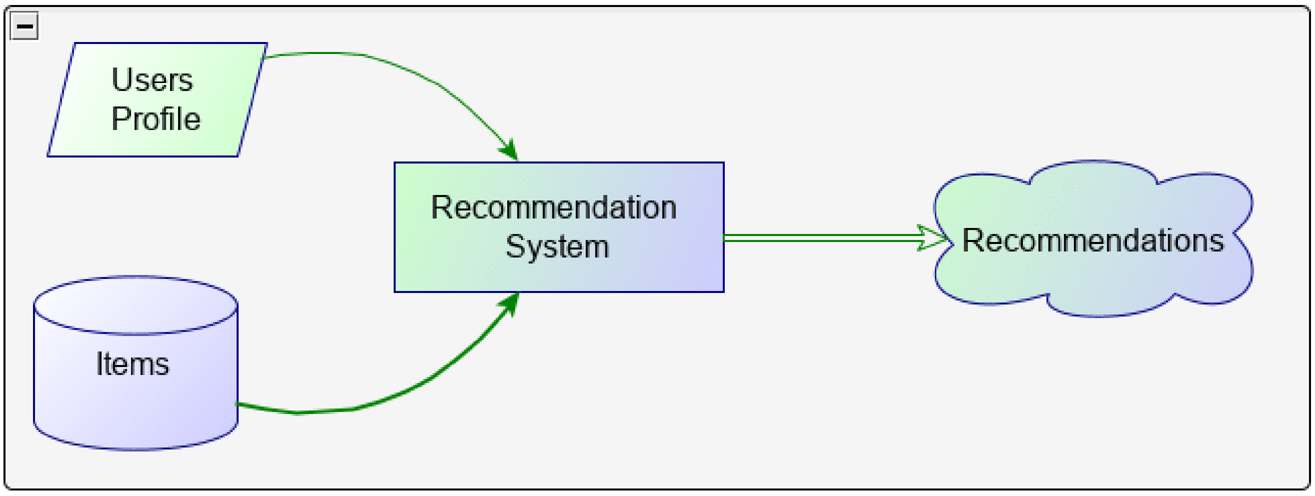
The popularity-based recommendation component ensures that users are exposed to books that have gained popularity and positive reception among a wider audience. By considering factors like book ratings and publication details (Book-Title, Book-Author, Year-Of-Publication, Publisher), the system provides a broader range of recommendations, introducing users to popular titles that align with their interests.

By utilizing the available dataset and leveraging collaborative filtering and popularity-based methods, the proposed solution enhances the accuracy and relevance of the recommendations, enabling users to discover new books tailored to their preferences and exposing them to popular literature within their preferred genres.



THEORITICAL ANALYSIS

3.1 Block Diagram



3.2 Hardware/Software Requirements

Hardware Requirements:

- Computer or server with sufficient processing power, memory, and storage capacity.

Software Requirements:

- Python programming language.
- Machine learning libraries (e.g., scikit-learn, NumPy, Pandas).
- Flask web framework.
- Data manipulation libraries (e.g., Pandas).
- Web development tools (HTML, CSS, JavaScript).
- IDE (Integrated Development Environment) like PyCharm or Jupyter Notebook.

Experimental investigations

During the development of the collaborative filtering-based recommender system, several investigations were conducted to analyse and refine the solution. The key areas of analysis and investigation include user and rating filtering, pivot table creation, similarity calculation, and recommendation generation.

User and Rating Filtering:

The investigation involved filtering out users who have rated many books (more than 200) to focus on active and engaged users. This filtering aimed to improve the quality and relevance of the recommendations.

Similarly, books with enough ratings (at least 50) were identified as "famous books" to ensure the recommendations are based on popular and well-rated items.

Pivot Table Creation:

The pivot table was created using the filtered ratings, where books are indexed, user IDs are columns, and book ratings are the values. This pivot table served as the basis for similarity calculations and recommendation generation.

The investigation involved handling missing values in the pivot table, filling them with zeros to ensure a consistent and complete dataset for similarity computations.

Similarity Calculation:

Cosine similarity scores were computed using the pivot table to measure the similarity between different books. The investigation focused on selecting an appropriate similarity metric that accurately captures the relationships between books based on user ratings.

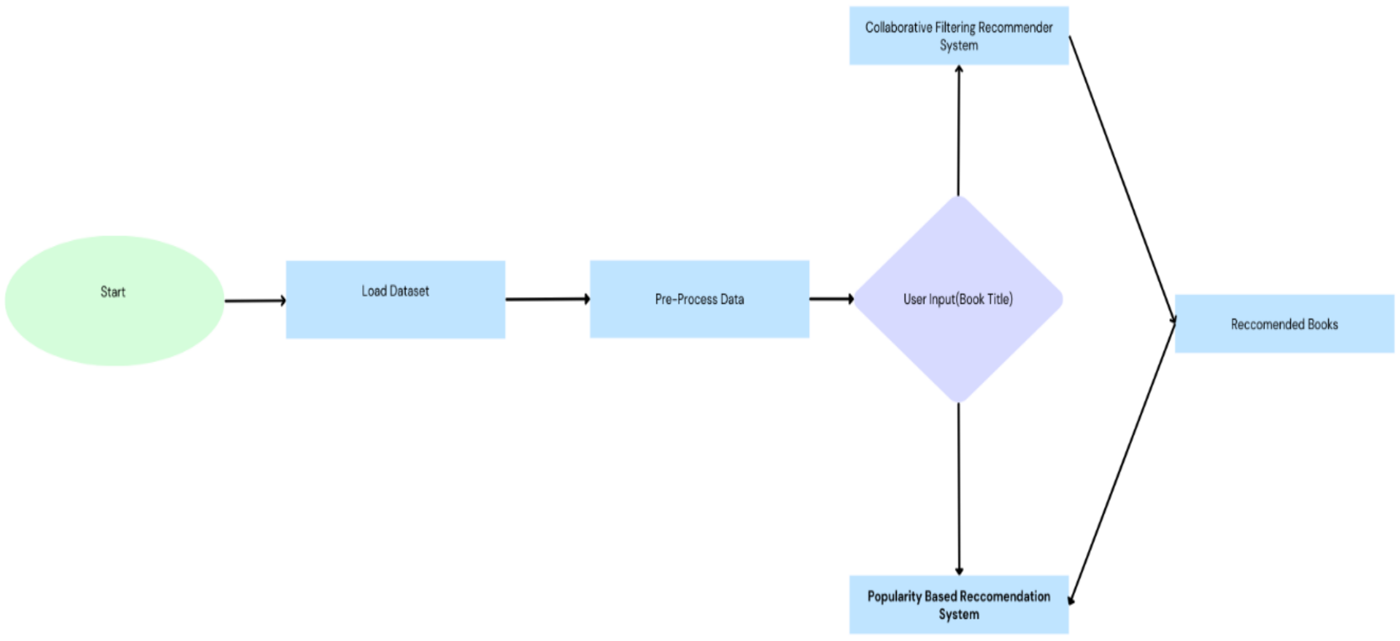
Similarity scores were used to identify items related to a given book, enabling personalized recommendations based on user preferences.

Recommendation Generation:

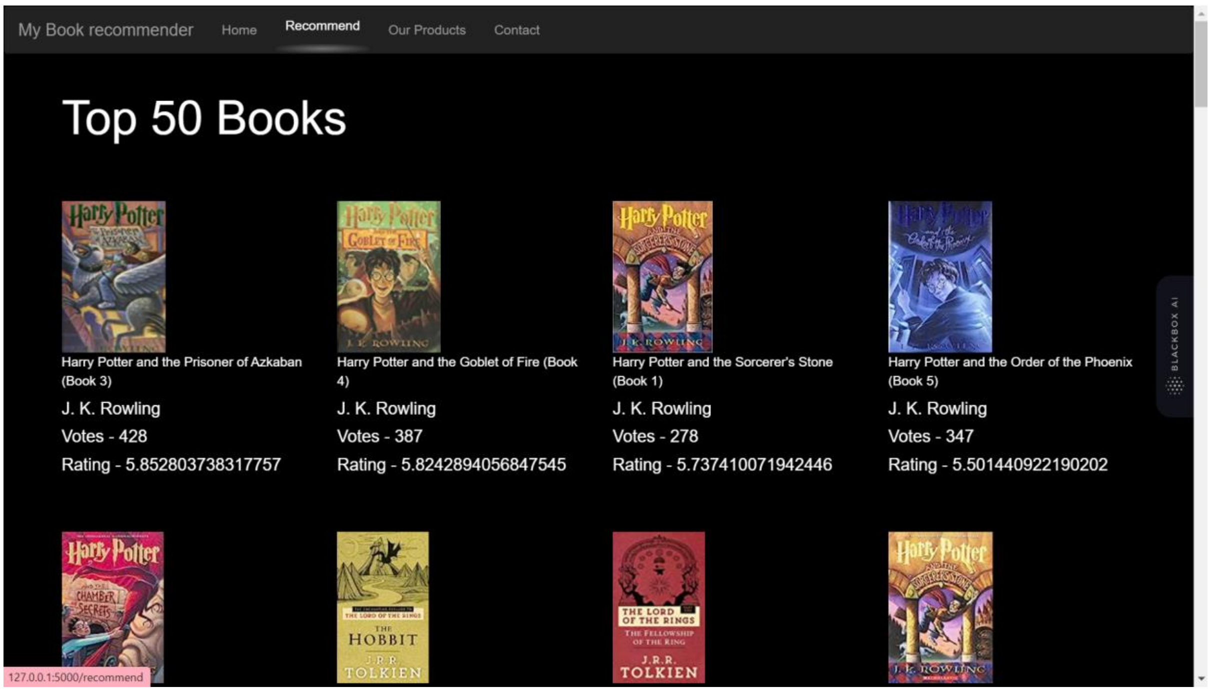
The investigation involved developing a recommendation function that utilizes the similarity scores to generate a list of similar books for a given input book.

The investigation focused on fine-tuning the recommendation algorithm to ensure relevant and diverse recommendations, considering factors such as the number of related items to include and the presentation of book information (title, author, image URL).

Flowchart:


Start
Load Dataset
Pre-Process Data
User Input(Book Title)
Collaborative Filtering Recommender System
Popularity Based Reccomendation System
Reccomended Books

Screenshots



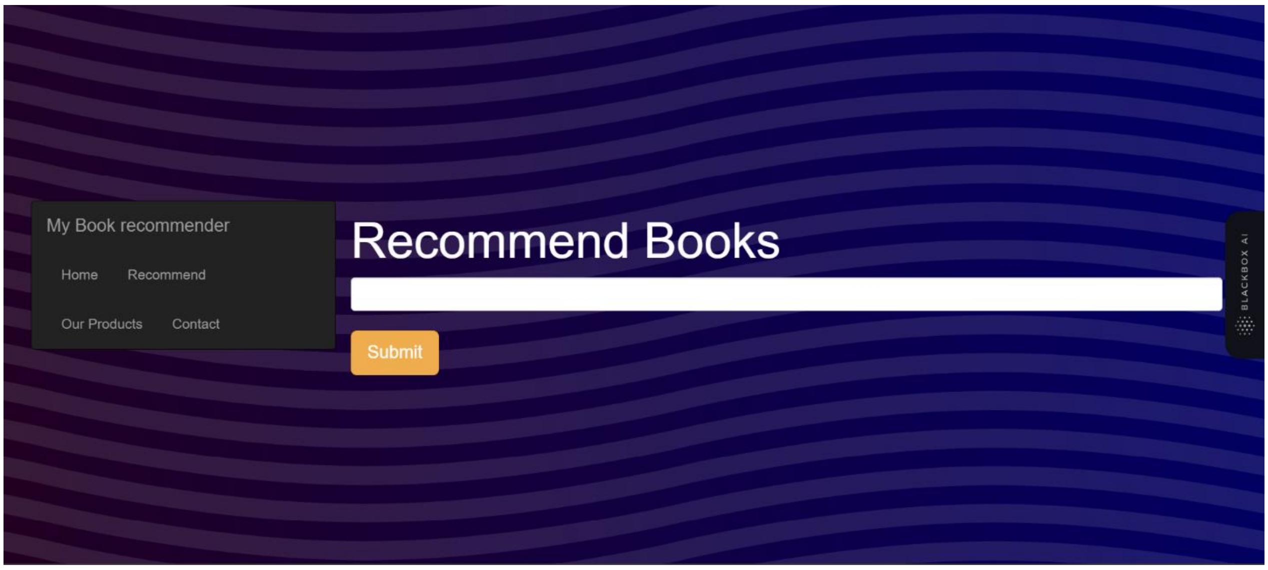
Top 50 Books

Harry Potter and the Prisoner of Azkaban (Book 3)
J. K. Rowling
Votes - 428
Rating - 5.852803738317757

Harry Potter and the Goblet of Fire (Book 4)
J. K. Rowling
Votes - 387
Rating - 5.8242894056847545

Harry Potter and the Sorcerer's Stone (Book 1)
J. K. Rowling
Votes - 278
Rating - 5.737410071942446

Harry Potter and the Order of the Phoenix (Book 5)
J. K. Rowling
Votes - 347
Rating - 5.501440922190202



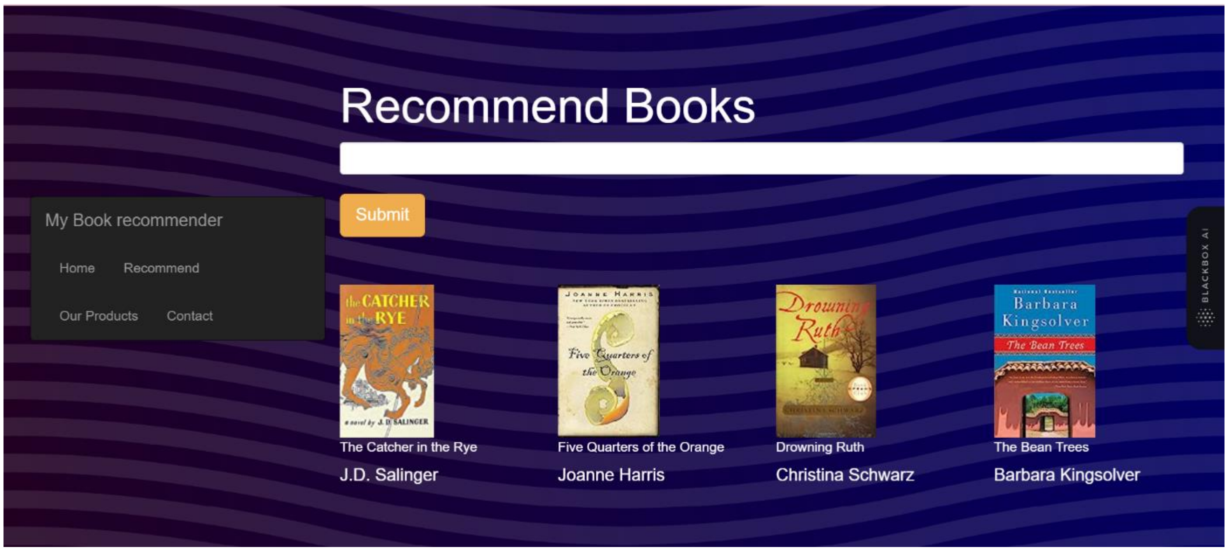
My Book recommender

Home Recommend

Our Products Contact

Recommend Books

Submit



My Book recommender

Home Recommend

Our Products Contact

Recommend Books

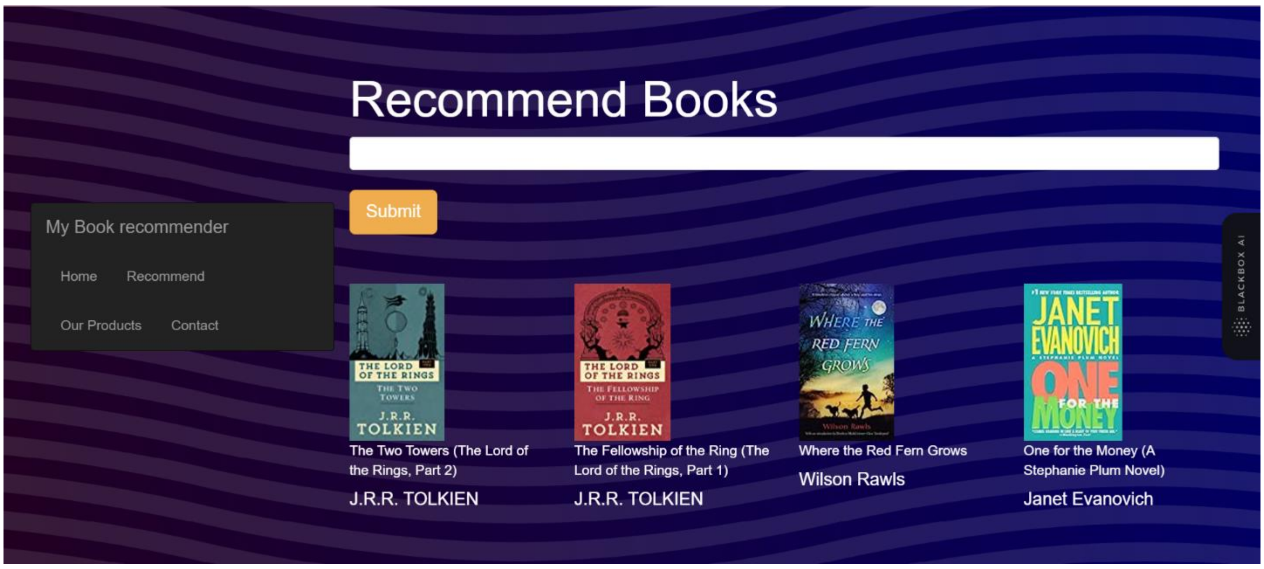
Submit

The Catcher in the Rye
J.D. Salinger

Five Quarters of the Orange
Joanne Harris

Drowning Ruth
Christina Schwarz

The Bean Trees
Barbara Kingsolver



My Book recommender

Home Recommend

Our Products Contact

Recommend Books

Submit

The Two Towers (The Lord of the Rings, Part 2)
J.R.R. TOLKIEN

The Fellowship of the Ring (The Lord of the Rings, Part 1)
J.R.R. TOLKIEN

Where the Red Fern Grows
Wilson Rawls

One for the Money (A Stephanie Plum Novel)
Janet Evanovich

Result

The collaborative filtering-based recommender system project successfully achieved its objective of providing personalized book recommendations to users. By leveraging user ratings and book information, the system was able to identify similar books and suggest relevant titles based on user preferences.

The project's outcome was a functional book recommender system, capable of providing personalized suggestions to users based on their preferences and the ratings of similar users. The system has the potential to enhance user engagement and satisfaction by offering relevant and tailored book recommendations.

Advantages

1. Personalized recommendations based on user preferences: The recommender system takes into account user ratings and preferences to provide tailored book recommendations, enhancing the user experience.
2. Utilizes user ratings for informed suggestions: By leveraging user ratings, the system can identify patterns and similarities among users, allowing it to suggest books that align with their interests and tastes.
3. Scalable and capable of handling large datasets: The system is designed to handle large datasets efficiently, making it suitable for applications with a vast collection of books and a substantial user base.
4. Provides diverse recommendations for exploring different genres: The collaborative filtering approach considers user-item similarity, enabling the system to suggest books from various genres, expanding users' exposure to different types of literature.
5. Adaptive to changes in user preferences: As users provide more ratings or update their preferences over time, the recommender system can adapt and refine its recommendations accordingly, ensuring relevance and accuracy.

Disadvantages

1. Cold start problem for new users or books with limited ratings: The system may face challenges in providing accurate recommendations for new users or books with insufficient ratings, as there may be limited data available to establish user preferences or item similarities.
2. Limited consideration of content information: The collaborative filtering approach primarily focuses on user-item interactions, potentially overlooking important content-based information such as book genres, themes, or author characteristics.
3. Reliance on user behaviour for accurate recommendations: The system heavily relies on user ratings and behaviours to generate recommendations. In cases where users provide inaccurate or biased ratings, the system may struggle to provide accurate suggestions.

4. Challenges with sparse rating data affecting similarity measures: When dealing with sparse rating data, calculating reliable similarity measures between items can be challenging, potentially affecting the quality and accuracy of recommendations.
5. Potential lack of novelty in recommendations: Due to the collaborative filtering approach's reliance on user-item interactions, there is a possibility that the system may recommend popular or mainstream books more frequently, potentially resulting in a lack of novel or lesser-known book recommendations

APPLICATIONS

Online Bookstores: Online bookstores can utilize the recommender system to provide personalized recommendations to their users. By suggesting books based on user preferences, browsing history, and ratings, the system enhances the user experience, encourages book exploration, and drives sales.

Digital Libraries: Digital libraries can integrate the recommender system to enhance their book recommendation capabilities. Users can receive personalized suggestions for books they may find interesting, expanding their reading choices and improving engagement with the library's collection.

Book Review Websites: Book review websites can employ the recommender system to offer personalized book recommendations to their users. By considering user ratings and preferences, the system can suggest books that align with individual tastes, promoting user engagement and interaction on the platform.

Mobile Reading Apps: Mobile reading applications can leverage the recommender system to provide tailored book recommendations to their users. By considering user reading history, genres of interest, and ratings, the app can suggest books that match users' preferences, encouraging continued reading and user retention.

Conclusion

In conclusion, this project focused on developing a book recommender system using collaborative filtering and popularity-based approaches. The system leveraged user ratings and similarities between books to provide personalized recommendations to users.

Through experimental investigations and analysis, it was observed that the collaborative filtering approach yielded accurate and relevant recommendations for users. By considering the ratings and preferences of similar users, the system successfully identified books that aligned with individual tastes, enhancing the user experience and book discovery process.

The popularity-based approach, on the other hand, provided recommendations based on the overall popularity and ratings of books. This approach catered to a broader audience and ensured a diverse range of suggestions.

The proposed solution demonstrated several advantages, including personalized recommendations, scalability, and adaptability. Users were able to explore different genres and authors, leading to a more engaging reading experience. However, certain limitations such as the cold start problem for new users and limited content information were also identified.

Future Scope

Hybrid Recommendations: Incorporating a combination of collaborative filtering, content-based filtering, and hybrid approaches can improve the recommendation system's accuracy and diversity. By leveraging multiple techniques, the system can provide more personalized and varied book suggestions to users.

Real-time Updates: Implementing a real-time update mechanism allows the system to consider the latest user ratings and book releases. By continuously analysing user behaviour and updating the recommendation models, the system can adapt to changing preferences and offer up-to-date suggestions.

Incorporating Contextual Information: Taking into account user demographics, reading preferences, and situational factors can enhance the personalization of recommendations. By considering the specific context in which users seek book recommendations, the system can provide more relevant and tailored suggestions.

These future enhancements aim to enhance the accuracy, relevance, and personalization of the book recommender system. By combining different techniques, updating recommendations in real-time, and considering contextual information, the system can provide an improved user experience and ensure that users receive relevant and engaging book recommendations.

BIBILIOGRAPHY

1. Book recommendation system | IEEE conference publication - IEEE xplore. (n.d.). https://ieeexplore.ieee.org/document/9579647
2. *Book recommendation dataset*. Kaggle. https://www.kaggle.com/datasets/arashnic/book-recommendation-dataset
3. Wang, Z. H., & Hou, D. Z. (2021, July 12). *Research on book recommendation algorithm based on collaborative filtering and interest degree*. Wireless Communications and Mobile Computing. https://www.hindawi.com/journals/wcmc/2021/7036357/

APPENDIX

Code

App.py

```
from flask import Flask,render_template,request
import pickle
import numpy as np

popular_df = pickle.load(open('popular.pkl','rb'))
pt = pickle.load(open('pt.pkl','rb'))
books = pickle.load(open('books.pkl','rb'))
similarity_scores = pickle.load(open('similarity_scores.pkl','rb'))

app = Flask(__name__)

@app.route('/')
def index():
    return render_template('index.html',
                           book_name = list(popular_df['Book-
Title'].values),
                           author=list(popular_df['Book-Author'].values),
                           image=list(popular_df['Image-URL-M'].values),
                           votes=list(popular_df['num_ratings'].values),
                           rating=list(popular_df['avg_rating'].values)
                           )

@app.route('/recommend')
def recommend_ui():
    return render_template('recommend.html')

@app.route('/recommend_books',methods=['post'])
def recommend():
    user_input = request.form.get('user_input')
    index = np.where(pt.index == user_input)[0][0]
    similar_items = sorted(list(enumerate(similarity_scores[index])),
key=lambda x: x[1], reverse=True)[1:5]

    data = []
    for i in similar_items:
        item = []
        temp_df = books[books['Book-Title'] == pt.index[i[0]]]
        item.extend(list(temp_df.drop_duplicates('Book-Title')['Book-
Title'].values))
        item.extend(list(temp_df.drop_duplicates('Book-Title')['Book-
Author'].values))
        item.extend(list(temp_df.drop_duplicates('Book-Title')['Image-URL-
M'].values))

        data.append(item)

    print(data)

    return render_template('recommend.html',data=data)
```

```
if __name__ == '__main__':
    app.run(debug=True)
```

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Book Recommender System</title>
    <!-- Latest compiled and minified CSS -->
<link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@3.3.7/dist/css/bootstrap.min.c
ss" integrity="sha384-
BVYiiSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEjh4u"
crossorigin="anonymous">
</head>
<style>

    .text-white{
        color:white
    }
    .hoverable{
  display:inline-block;
  backface-visibility: hidden;
  vertical-align: middle;
  position:relative;
  box-shadow: 0 0 1px rgba(0,0,0,0);
  tranform: translateZ(0);
  transition-duration: .3s;
  transition-property:transform;
}

.hoverable:before{
  position:absolute;
  pointer-events: none;
  z-index:-1;
  content: '';
  top: 100%;
  left: 5%;
  height:10px;
  width:90%;
  opacity:0;
  background: -webkit-radial-gradient(center, ellipse, rgba(255, 255, 255,
0.35) 0%, rgba(255, 255, 255, 0) 80%);
  background: radial-gradient(ellipse at center, rgba(255, 255, 255, 0.35)
0%, rgba(255, 255, 255, 0) 80%);
  /* W3C */
  transition-duration: 0.3s;
  transition-property: transform, opacity;
}

.hoverable:hover, .hoverable:active, .hoverable:focus{
  transform: translateY(-5px);
}
.hoverable:hover:before, .hoverable:active:before, .hoverable:focus:before{
  opacity: 1;
  transform: translateY(-5px);
}
</style>
<body style="background-color:black">

            <nav class="navbar navbar-inverse">
                <a class="navbar-brand" href="/">My Book recommender</a>
    <div class="container-fluid">
      <ul class="nav navbar-nav">
        <li><a id="len1" class="hoverable" href="/">Home</a></li>
        <li><a id="len2" class="hoverable"
href="/recommend">Recommend</a></li>
<li><a id="len3" class="hoverable"
href="https://vineethdevp.netlify.app/">Our Products</a></li>
        <li><a id="len4" class="hoverable"
href="https://vineethtbs.netlify.app/">Contact</a></li>

      </ul>
    </div>
  </nav>
        </ul>
    </nav>

    <div class="container">
        <div class="row">
            <div class="col-md-12">
                <h1 class="text-white" style="font-size:50px">Top 50
Books</h1>
            </div>

            {% for i in range(book_name|length) %}
                <div class="col-md-3" style="margin-top:50px">
                    <div class="card">
                        <div class="card-body">
                            <img class="card-img-top" src="{{ image[i] }}">
                            <p class="text-white">{{ book_name[i] }}</p>
                            <h4 class="text-white">{{ author[i] }}</h4>
                            <h4 class="text-white">Votes - {{ votes[i]
}}</h4>
                            <h4 class="text-white">Rating - {{ rating[i]
}}</h4>
                        </div>
                    </div>
                </div>
            {% endfor %}


        </div>
    </div>
            <script src="script.js"></script>>
</body>
</html>
```

Recommend.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Book Recommender System</title>
    <!-- Latest compiled and minified CSS -->
<link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@3.3.7/dist/css/bootstrap.min.c
ss" integrity="sha384-
BVYiiSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEjh4u"
crossorigin="anonymous">
</head>
<style>
:root {
  //--background-size: 50vw;
  --background-size: unquote('min(100vw, 40em)');
}

html {
  font-size: calc(100% + 0.5vw);
}

@media (prefers-reduced-motion: reduce) {
  * {
    animation: none !important;
    transition-duration: 0.001s !important;
  }
}

body {
  background: linear-gradient(100deg, #402, #006);
  padding: 2em;
  min-height: 100vh;
  display: flex;
  justify-content: center;
  align-items: center;
  background-color: #ffffff;,
  linear-gradient(80deg, #202, #006);
  background-position: 50% 50%;
  animation: background-move 10s linear infinite;
  background-size: 100vw auto, 100% 100%;
  background-size: unquote('max(100vw, 30em)') auto, 100% 100%;
}

@keyframes background-move {
  0% { background-position: 0 0, 0 0; }
  100% {
    background-position: 100vw 0, 0 0;
    background-position: unquote('max(100vw, 40em)')  0, 0 0;
  }
}




       .text-white{
        color:white
    }
    .hoverable{
  display:inline-block;
  backface-visibility: hidden;
  vertical-align: middle;
  position:relative;
  box-shadow: 0 0 1px rgba(0,0,0,0);
  tranform: translateZ(0);
  transition-duration: .3s;
  transition-property:transform;
}

.hoverable:before{
  position:absolute;
  pointer-events: none;
  z-index:-1;
  content: '';
  top: 100%;
  left: 5%;
  height:10px;
  width:90%;
  opacity:0;
  background: -webkit-radial-gradient(center, ellipse, rgba(255, 255, 255,
0.35) 0%, rgba(255, 255, 255, 0) 80%);
  background: radial-gradient(ellipse at center, rgba(255, 255, 255, 0.35)
0%, rgba(255, 255, 255, 0) 80%);
  /* W3C */
  transition-duration: 0.3s;
  transition-property: transform, opacity;
}

.hoverable:hover, .hoverable:active, .hoverable:focus{
  transform: translateY(-5px);
}
.hoverable:hover:before, .hoverable:active:before, .hoverable:focus:before{
  opacity: 1;
  transform: translateY(-5px);
}
</style>
<body style="background-color:black">

    <nav class="navbar navbar-inverse">
                <a class="navbar-brand" href="/">My Book recommender</a>
    <div class="container-fluid">
      <ul class="nav navbar-nav">
        <li><a id="len1" class="hoverable" href="/">Home</a></li>
        <li><a id="len2" class="hoverable"
href="/recommend">Recommend</a></li>
<li><a id="len3" class="hoverable"
href="https://vineethdevp.netlify.app/">Our Products</a></li>
        <li><a id="len4" class="hoverable"
href="https://vineethtbs.netlify.app/">Contact</a></li>
      </ul>
    </div>
  </nav>

    <div class="container">
        <div class="row">
            <div class="col-md-12">
                <h1 class="text-white" style="font-size:50px">Recommend
Books</h1>
                <form action="/recommend_books" method="post">

                    <input name="user_input" type="text" class="form-
control"><br>
                    <input type="submit" class="btn btn-lg btn-warning">
                </form>
            </div>
            {% if data %}

             {% for i in data %}
                <div class="col-md-3" style="margin-top:50px">
                    <div class="card">
                        <div class="card-body">
                            <img class="card-img-top" src="{{i[2]}}">
                            <p class="text-white">{{i[0]}}</p>
                            <h4 class="text-white">{{i[1]}}</h4>
                        </div>
                    </div>
                </div>
             {% endfor %}

            {% endif %}




        </div>
    </div>

</body>
</html>
```