

Final Report on Advanced Book Search & Recommender APIs

Names

May 15, 2024

Abstract

This report presents a detailed development summary of the Advanced Book Search and Recommendations system. The system was created with the aim of providing fast, relevant, and personalized recommendations and search algorithms to all manner of bookstores, libraries, and other facilities. This report details the research, development, testing, and deployment stages of the project.

1 Introduction

The digital age has revolutionized the way we interact with literature, providing unprecedented access to a vast array of books through online platforms. However, despite the convenience of digital bookstores, users often encounter challenges in navigating the plethora of available titles, finding books that match their interests, and ensuring the authenticity of transactions, particularly in the realm of used book sales. Addressing these challenges, our project focuses on the development of a used book sales system powered by blockchain technology. By integrating blockchain, we aim to enhance transparency, security, and trust in the used book marketplace, mitigating concerns related to counterfeit books and fraudulent transactions. Furthermore, by incorporating content-based search algorithms, our system offers personalized recommendations to users, guiding them towards books that align with their tastes and preferences. This report provides a comprehensive overview of our project, detailing the objectives, methodology, design, development, testing, deployment, and future prospects of our innovative solution. Through this endeavor, we seek to revolutionize the used book marketplace, fostering a seamless and trustworthy experience for book enthusiasts worldwide.

2 Problem statement

The domain of online bookstores, encompassing both new and used books, constitutes a significant segment of the e-commerce industry. However, the current landscape of online book sales platforms, particularly in the realm of used books, presents several challenges that hinder user experience and trustworthiness. Existing solutions often lack robust mechanisms for facilitating efficient search and discovery processes, resulting in user frustration and suboptimal outcomes. Users encounter difficulties in navigating through the vast array of available titles, and concerns regarding the authenticity and provenance of books persist, casting doubt on the reliability of transactions.

Inadequate search algorithms exacerbate the problem, failing to provide personalized recommendations tailored to individual user preferences. As a result, users struggle to find relevant titles amidst the extensive catalog, hampering their overall experience. Furthermore, the management of bookstores faces challenges in efficiently handling inventory and ensuring the quality and authenticity of used books.

To address these challenges, our project proposes a comprehensive solution that leverages blockchain technology and advanced search algorithms. By integrating blockchain, we aim to enhance transparency, security, and trust in the used book marketplace. Blockchain ensures immutable records of transactions, mitigating concerns related to counterfeit books and fraudulent activities. Additionally, our solution focuses on improving the performance of search algorithms to provide users with personalized recommendations, facilitating smoother transactions and enhancing user satisfaction.

In summary, our project seeks to revolutionize the way users interact with online bookstores by addressing the inefficiencies and challenges prevalent in the current landscape. By providing a robust and user-friendly platform for buying and selling used books, we aim to foster a seamless and trustworthy experience for both book enthusiasts and bookstore management. Used references are BookSwap.lk and UsedBooks.lk

3 Literature Review

4 Methodology

Prior to proposal presentation, we created system block diagrams for how the various systems are to function within the system.

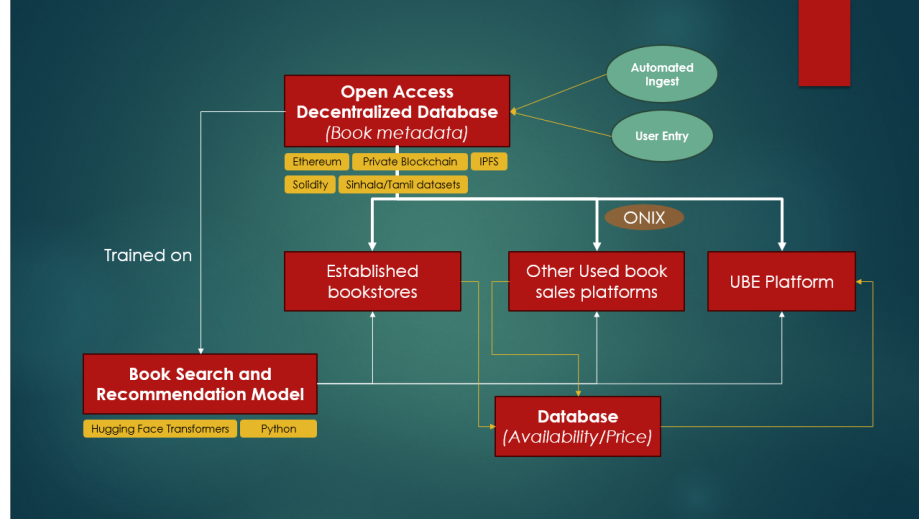


Figure 1: Block Diagram for Proposed System

As per our proposal, we identified 4 key deliverables necessary to succeed in our project, namely:

- Book Data Database
- Recommendation System APIs
- Search APIs
- Demo applications

4.1 Development

Development was split into three phases. Namely,

- API Creation
- Demo Application Creation
- Testing & Integration

4.1.1 API Creation

We chose to run our API on Flask, for its performance and ease of use. Since we simply needed to deploy the model and have a few routes to describe the functionalities of the API, and to serve the models, we felt that a micro-framework like Flask would be the best.

The API contains the following routes.

`/predict`

`/models`

`/predict` is a HTTP POST endpoint, and is used to serve the Recommender models themselves. This accepts two parameters, `title`, and `model`. `Title` is the name of the book for which we want recommendations. `Model` gives shorthand for the various models able to be served by this API. For now, we can serve `distilbert`, `distilbert_v2`, `bert`, `tf_idf`, and `word2vec` models.

`/models` is a HTTP GET endpoint, and simply returns a list of available models that can be used by the client to request recommendations.

Recommender Models

We tried various approaches in creating the recommender models. First, we tried traditional ML techniques to create a model that can recommend books based on similarities in content to other books. Our first model, `tf_idf`, ended up being our most stable and useful model.

TF-IDF

`tf_idf` utilizes Term Frequency - Inverse Document Frequency to generate vector embeddings for the various fields used to compare books. Term Frequency-Inverse Document Frequency (TF-IDF) is a numerical statistic used in information retrieval and text mining to reflect the importance of a term in a document relative to a collection of documents (corpus). It is commonly used as a weighting factor in various text analysis tasks such as document classification, information retrieval, and text similarity calculation. In this case, we utilized TF-IDF to generate embeddings for the book titles and descriptions. This model also processes language codes, genres, and author data, but utilizes different methodologies to convert this data into embeddings. For genres, we

use Multi-Label Binarization, and a hashing algorithm to compute embeddings for the author field. The libraries implementing these algorithms were taken from **sklearn**.

For this model, we utilized Cosine Similarity as our measure of similarity, as it provided the best recommendations as per (limited) user testing. Given below is the feature extraction portion of the TF-IDF Model.

Listing 1: Feature Extraction using traditional ML

```
class FeatureExtractor:
def extract_features(self, books_df_processed):
    vectorizer = TfidfVectorizer()

    # Count of unique authors
    count_of_unique_authors = books_df_processed['
        author'].nunique()

    hasher = FeatureHasher(n_features=
        count_of_unique_authors, input_type='string')
    mlb = MultiLabelBinarizer()

    # Hash the authors
    # author_features = hasher.transform(
        books_df_subset['author'])

    # Binarize the genres column
    binarized_genres = mlb.fit_transform(
        books_df_processed['genres'])

    # One-hot encode the language_code
    books_df_subset = pd.get_dummies(
        books_df_processed, columns=['language_code'])

    # Vectorize the title column
    title_features = vectorizer.fit_transform(
        books_df_subset['title'])

    # Vectorize the description column
    description_features = vectorizer.fit_transform(
        books_df_subset['description'])

    # Composite feature Vector
    composite_feature_vector = hstack([
        binarized_genres, title_features,
        description_features])

return composite_feature_vector
```

* Note that the Author embeddings are suppressed in this code, and it is suppressed from the model because the data in the original dataset is not clean enough, which resulted in improper recommendations when using that column.

Bert & DistilBert

BERT is a language model based on the transformer architecture, released in 2018 by researchers at Google. We intended to use BERT for generating embeddings for the data, as transformer based pre-trained models provide significant advantages over traditional ML methodologies for extracting richer relationships between text tokens.

While this was the idea in theory, we were unable to actually train any of the BERT based models, as the hardware available to us was not sufficient to train the model in any reasonable length of time. The best machine available at our disposal was a laptop equipped with an AMD Ryzen 9 6900HX processor, 16 Gigabytes of RAM, and a AMD Radeon 6700S GPU. BERT / DISTILBERT is not optimized for the AMD architecture, and caused massive issues during training.

Word2Vec

Word2Vec is another traditional ML approach to generating embeddings. We used Word2Vec on all the available fields (title, description, author, genres), and the result was not satisfactory. The resultant model produced matches that simply matched strings within the other titles, and thus did not capture any semantic meaning behind the words themselves.

We left the model in the system, for testing and evaluation purposes.