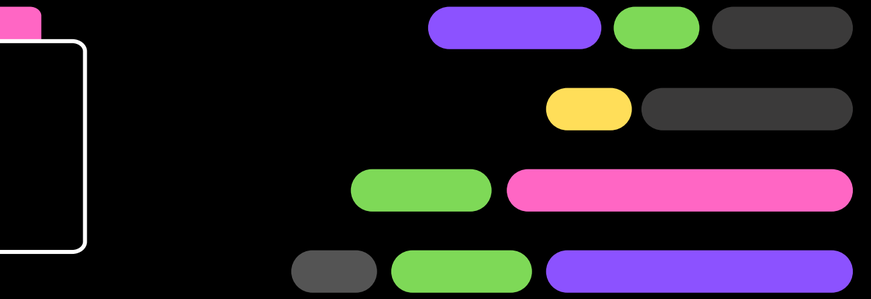
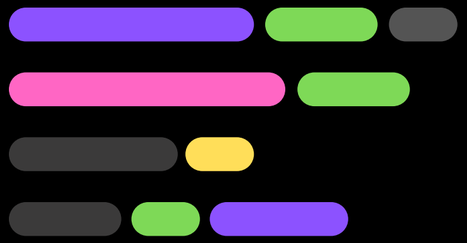
Book Recommendation System

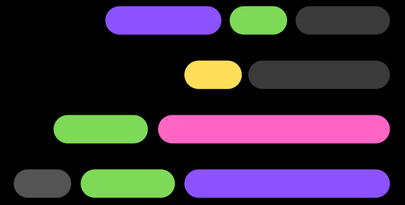
Using IR & ML





Presented by: Kartik Mittal
Vansh Prajapati

TABLE OF CONTENTS



01

INTRODUCTION

Our project, Intelligent Book Recommendation System, helps users find books based on their preferred genre using a hybrid approach. We combine **BM25**, **Cosine Similarity**, and **XGBoost** to deliver accurate and diverse recommendations. The system features a simple **Streamlit** interface and uses **NDCG** to evaluate result quality.

02 Our Project

1. Data Set
2. Text Preprocessing
3. BM25- Based Recommendation
4. Cosine Similarity Based Recommendation
5. XGBoost Based Recommendation
6. Combining Recommendations
7. Evaluation with NDCG
8. Streamlit Web App
9. Code

DataSet

We used a curated dataset named `books_df.csv`, containing key book-related information:

- **Title**, **Author**, **Main Genre**, **Rating**, and **URLs**
- Text fields were cleaned and normalized for processing
- Genre data was encoded for use in machine learning model.
  This dataset forms the foundation for generating meaningful recommendations.

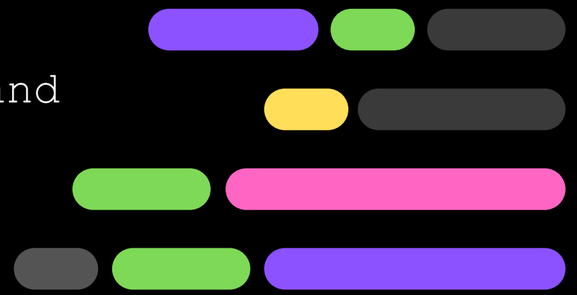
Link of the Dataset:
https://www.kaggle.com/datasets/chhavidhankhar11/amazon-books-dataset

Text Preprocessing

To ensure consistent and meaningful input, we applied standard NLP preprocessing steps using **NLTK:**

- **Tokenization:** Splitting text into individual words
- **Stopword Removal:** Removing common, non-informative words
- **Lemmatization:** Converting words to their base form
- **Cleaning:** Removing punctuation and non-alphanumeric characters

This step prepares the data for accurate similarity scoring and model training.

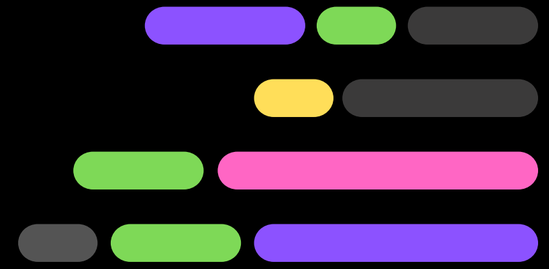


BM25-Based Recommendation

We use the **BM25Okapi** algorithm from the `rank_bm25` library to rank books by relevance to a user's genre input.

- Constructs a corpus using preprocessed **Title**, **Author**, and **Genre**
- Tokenizes and scores relevance based on keyword matches
- Generates a **BM25 score** for each book
- Recommends top-N books with the highest scores

BM25 is effective for capturing keyword importance and contextual relevance.



Cosine Similarity-Based Recommendation

We use **TF-IDF** Vectorization and **Cosine Similarity** to measure textual similarity between genres.

- Converts the **Main Genre** column into TF-IDF vectors
- Computes **cosine similarity** between user input and all book genres
- Identifies and ranks books with genres most similar to the user’s query
- Returns top-N books with highest similarity scores

This method captures genre-based semantic closeness between books.

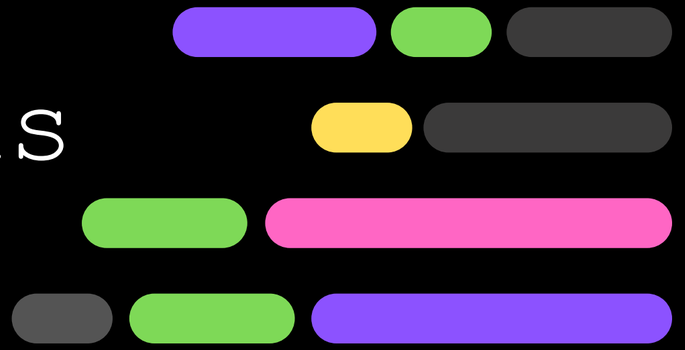
XGBoost-Based Recommendation

- We use **XGBoost**, a powerful machine learning classifier, to predict book genres based on ratings.

- **Input**: Book data
- **Target**: Encoded genres using `LabelEncoder`
- Trained the model to classify books into genres
- Predicted genres are matched against user input
- Top-rated books from the predicted genre are recommended

- This adds a learning-based perspective to complement the IR models.

Combining Recommendations

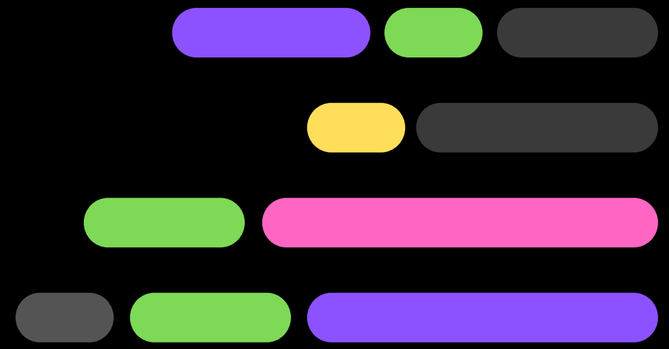


To enhance recommendation quality, we combine results from all three models:

- Merge outputs from **BM25**, **Cosine Similarity**, and **XGBoost**
- Compute a **combined score** (average of BM25 and cosine similarity scores)
- Sort books by this score and select top-N recommendations
- Ensures a balance of **relevance**, **semantic similarity**, and **learning-based insights**

This hybrid approach increases diversity and robustness of suggestions.

Evaluation with NDCG



We evaluate recommendation quality using Normalized Discounted Cumulative Gain (**NDCG**):

- Measures how well the recommended books match the user's preferred genre
- **True Relevance**: 1 if the book's genre matches the input, else 0
- **Predicted Relevance**: 1 for recommended books, else 0
- Higher **NDCG score** indicates better-ranked, more relevant suggestions

NDCG provides a fair way to assess the effectiveness of ranked recommendations.

Streamlit Web App

We built an interactive frontend using Streamlit to make the system user-friendly.

- **Input**: User enters a preferred book genre
- **Output**: Top recommendations from BM25, Cosine, and XGBoost
- **Displays**: Combined results and NDCG score
- Clean, responsive interface for real-time interaction

Streamlit allows quick deployment and easy exploration of recommendations.

Code Snippet

```
iimport pandas as pd
import numpy as np
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
from rank_bm25 import BM25Okapi
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
import warnings
import json
import streamlit as st
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import ndcg_score
from xgboost import XGBClassifier
```

```
warnings.filterwarnings("ignore")
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet‘)

# Preprocessing function
def preprocess(text):
    if pd.isna(text):
        return ''
    tokens = word_tokenize(text.lower())
    tokens = [w for w in tokens if w.isalnum()]
    tokens = [w for w in tokens if w not in
stopwords.words('english')]
    lemmatizer = WordNetLemmatizer()
    return " ".join(lemmatizer.lemmatize(w) for w in tokens)
```

```
# Load dataset from script folder
def load_data():
    try:
        books = pd.read_csv("books_df.csv")
        return books
    except Exception as e:
        print(f"Error loading dataset: {e}")
        return None

# Create BM25 corpus for a subset of books
def create_bm25_corpus_subset(books_subset):
    books_subset['full_text'] = (books_subset['Title'].fillna('') + ' ' +
books_subset['genres_str'].fillna('') + ' ' +
books_subset['Author'].fillna('')).apply(preprocess)
    bm25_corpus = books_subset['full_text'].apply(str.split).tolist()
    bm25 = BM25Okapi(bm25_corpus)
    return books_subset, bm25
```

```
# Get recommendations based on genre input using BM25
def get_bm25_recommendations(books, user_genre, top_n=5):
    genre_books = books[books['Main Genre'].str.lower().str.contains(user_genre.lower(),
na=False)]
    if genre_books.empty:
        return pd.DataFrame()

    # Create BM25 corpus and model for the filtered genre books
    genre_books, bm25 = create_bm25_corpus_subset(genre_books)

    query_text = user_genre
    tokens = preprocess(query_text).split()
    q_scores = bm25.get_scores(tokens)

    genre_books['BM25_score'] = q_scores
    top_recommendations = genre_books.sort_values('BM25_score', ascending=False).head(top_n)
    return top_recommendations[['Title', 'Author', 'Main Genre', 'Rating', 'URLs',
'BM25_score']]
```

```
# Get recommendations based on genre input using cosine similarity
def get_cosine_similarity_recommendations(books, user_genre, top_n=5):
    tfidf = TfidfVectorizer(stop_words='english')
    tfidf_matrix = tfidf.fit_transform(books['Main Genre'])
    cosine_sim = cosine_similarity(tfidf_matrix, tfidf_matrix)
    genre_indices = books[books['Main Genre'].str.lower().str.contains(user_genre.lower(),
na=False)].index
    if genre_indices.empty:
        return pd.DataFrame()
    sim_scores = list(enumerate(cosine_sim[genre_indices[0]]))
    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)
    top_indices = [i[0] for i in sim_scores[1:top_n+1]]
    top_recommendations = books.iloc[top_indices][['Title', 'Author', 'Main Genre',
'Rating', 'URLs']]
    top_recommendations['cosine_sim_score'] = [sim_scores[i+1][1] for i in
range(len(top_indices))]
    return top_recommendations
```

```
# Add XGBoost
def add_xgboost(books):
    books['Rating'] = books['Rating'].fillna(0)
    label_encoder = LabelEncoder()
    books['Encoded Genre'] = label_encoder.fit_transform(books['Main Genre'])
    X = books[['Rating']]
    y = books['Encoded Genre']
    model = XGBClassifier()
    model.fit(X, y)
    return model, label_encoder
```

```
# Get recommendations based on genre input using XGBoost
def get_xgboost_recommendations(books, model, user_genre, top_n=5):
    books['Rating'] = books['Rating'].fillna(0)
    label_encoder = LabelEncoder()
    books['Encoded Genre'] = label_encoder.fit_transform(books['Main Genre'])
    X = books[['Rating']]
    predicted_genres = model.predict(X)
    books['Predicted Genre'] = label_encoder.inverse_transform(predicted_genres)
    genre_books = books[books['Predicted
Genre'].str.lower().str.contains(user_genre.lower(), na=False)]
    if genre_books.empty:
        return pd.DataFrame()
    top_recommendations = genre_books.sort_values('Rating', ascending=False).head(top_n)
    return top_recommendations[['Title', 'Author', 'Main Genre', 'Rating', 'URLs']]
```

```
## Combine recommendations from BM25, cosine similarity, and XGBoost
def combine_recommendations(bm25_recommendations, cosine_recommendations,
xgboost_recommendations, top_n=5):
    combined = pd.concat([bm25_recommendations, cosine_recommendations,
xgboost_recommendations])
    combined['combined_score'] = combined[['BM25_score', 'cosine_sim_score']].mean(axis=1)
    combined = combined.sort_values('combined_score', ascending=False).head(top_n)
    return combined[['Title', 'Author', 'Main Genre', 'Rating', 'URLs']]

# Calculate NDCG
def calculate_ndcg(books, recommendations, user_genre, top_n=5):
    if recommendations.empty:
        return 0.0  # Return 0.0 if no recommendations found

    true_relevance = np.zeros(len(books))
    predicted_relevance = np.zeros(len(books))
```

```
# Mark true relevance (1 if the book belongs to the user's genre, 0 otherwise)
    for idx, book in books.iterrows():
        if user_genre.lower() in str(book['Main Genre']).lower():
            true_relevance[idx] = 1

 # Mark predicted relevance (1 if the book is in recommendations, 0 otherwise)
    recommendation_indices = recommendations.index
    predicted_relevance[recommendation_indices] = 1

    # Calculate NDCG
    ndcg = ndcg_score([true_relevance], [predicted_relevance])
    return ndcg
```

```
# Streamlit frontend
st.title("Book Recommendation System")
books = load_data()
if books is not None:
    books["genres_str"] = books["Main Genre"].fillna("").str.lower().str.replace(r"\s+", "_", regex=True)
    user_genre = st.text_input("Enter your preferred genre:")
    if st.button("Get Recommendations"):
        bm25_recommendations = get_bm25_recommendations(books, user_genre)
        cosine_recommendations = get_cosine_similarity_recommendations(books, user_genre)
        model, label_encoder = add_xgboost(books)
        xgboost_recommendations = get_xgboost_recommendations(books, model, user_genre)

        if bm25_recommendations.empty and cosine_recommendations.empty and xgboost_recommendations.empty:
            st.write("No books found for the specified genre.")
        else:
            combined_recommendations = combine_recommendations(bm25_recommendations, cosine_recommendations,
xgboost_recommendations)
            ndcg = calculate_ndcg(books, combined_recommendations, user_genre)
            st.write("Combined Recommendations:")
            st.dataframe(combined_recommendations)
            st.write(f"NDCG Score: {ndcg}")
```

Output Screenshots

Book Recommendation System

Enter your preferred genre:

Travel

Get Recommendations

Combined Recommendations:

	Title	Author	Ma
7925	Eyewitness Travel Phrase Book French (EW Travel Guide Phrase Books)	DK	Tra
7887	India The Journey - A Travel Book on India	MRM Publications	Tra
7909	Lonely Planet India (Travel Guide)	Lonely Planet	Tra
7926	Lonely Planet Australia (Travel Guide)	Andrew Bain	Tra
7923	Insight Guides Poland (Travel Guide with Free eBook) (Insight Guides Main Series)	Insight Travel Guide	Tra

NDCG Score: 0.5381333221539772

Book Recommendation System

Enter your preferred genre:

Politics

Get Recommendations

Combined Recommendations:

	Title	Author	Main Genre	Rating	URLs
4216	On Palestine	Ilan Pappe	Politics	4.6	https://www.amazon.in/Palestine
4343	The Social Contract	Jean-Jacques Rousseau	Politics	4.5	https://www.amazon.in/Social-C
4254	WHY BHARAT MATTERS	S. JAISHANKAR	Politics	4.6	https://www.amazon.in/WHY-BH
4268	On Palestine	Ilan Pappe	Politics	4.6	https://www.amazon.in/Palestine
4250	Why Bharat Matters	S. Jaishankar	Politics	4.6	https://www.amazon.in/Why-Bha

NDCG Score: 0.5624614471656796

03 Conclusion

- We developed a **Book recommendation system** combining **BM25**, **Cosine Similarity**, and **XGBoost**
- The system delivers accurate, diverse, and personalized book suggestions
- Evaluated using **NDCG** to ensure relevance and quality of recommendations
- Integrated with a clean, interactive **Streamlit web app** for real-time user interaction
- This approach balances **information retrieval** and **machine learning**, enhancing recommendation effectiveness