SIX WEEK RESEARCH INTERNSHIP REPORT

NIT KURUKSHETRA, HARYANA

Submitted By:

MAUSAM KUMAR GIRI (211563)

BATCH: 2021-2025

Department of Computer Science and Engineering School of Engineering and Technology

CENTRAL UNIVERSITY OF HARYANA

ACKNOWLEDGEMENT

Contents

1. Introduction

1.1 Background of the Internship

In recent years, the proliferation of digital content has necessitated the development of sophisticated recommendation systems. These systems have become integral in providing personalized content to users, thereby enhancing user experience and engagement. During my six-week research internship, I had the opportunity to delve into the field of recommendation systems, focusing specifically on cross-domain recommendations. This involves leveraging data from one domain (books) to make recommendations in another domain (movies).

1.2 Objectives of the Internship

The primary objective of my internship was to work on a cross-domain recommendation system that can predict movies based on a user's reading habits. This was achieved by training a RoBERTa model on a dataset of books. The specific objectives were:

1. To understand the theoretical foundations of recommendation systems and cross-domain recommendations.
2. To gain hands-on experience with Natural Language Processing (NLP) techniques, Multilabel Classification techniques, particularly using the RoBERTa model by utilizing transformers.
3. To collect and pre-process a dataset of books and movies, providing corresponding movie recommendations.
4. To train the RoBERTa model on the books dataset.
5. To evaluate the performance of the cross-domain recommendation system.
6. Develop web application to provide the recommendation to the users through an interactive user space.

1.3 Overview of the Report

This report provides a comprehensive account of my internship experience and the development of the book-movies cross-domain recommendation system. The report is structured as follows:

- **Introduction**: An overview of the internship, its objectives, and the structure of the report.
- **Institution Profile**: A description of the institution where the internship was conducted, including the research department and team.
- **Literature Review**: A review of existing literature on recommendation systems, cross-domain recommendations, and the role of NLP, particularly the RoBERTa model, in these systems.
- **Project Description**: Details of the project objectives, scope, and timeline.
- **Technical Background**: An overview of the relevant machine learning, NLP concepts, Multilabel Classification techniques and the RoBERTa model, and the datasets used.
- **Methodology**: The methods used for data collection, pre-processing, model training, and system development.
- **Implementation**: A detailed account of the system architecture, development tools, and the step-by-step implementation process.
- **Evaluation**: The metrics used to evaluate the system, performance analysis, and comparison with other models.
- **Discussion**: A discussion of the findings, implications, and limitations of the study.
- **Conclusion**: A summary of the work done, key takeaways, and recommendations for future work.
- **References**: A list of all the books, articles, papers, and online resources referenced in the report.

Institution Profile

1.4 Overview of Institution

National Institute of Technology (NIT), Kurukshetra

The National Institute of Technology (NIT) Kurukshetra is a premier technical institute in India, known for its commitment to excellence in technical education and research. Established in 1963, NIT Kurukshetra has consistently been at the forefront of imparting quality education and fostering innovation.

Vision of the Institute

NIT Kurukshetra envisions being a role model in technical education and research, responsive to global challenges. The institute aims to shape future leaders and innovators who can address the evolving needs of society through cutting-edge technologies and research.

Mission of the Institute

- **M1:** To impart quality technical education that develops innovative professionals and entrepreneurs.
- **M2:** To undertake research that generates cutting-edge technologies and futuristic knowledge, focusing on socio-economic needs.

1.5 Department & Mentor

Department of Computer Science & Engineering

The Department of Computer Science & Engineering at NIT Kurukshetra has a rich history of academic excellence and innovation. The department began offering a B.Tech. Programme in Computer Engineering in 1987 with an initial intake of 30 students, which has since grown to 210. In 2006, the department introduced a B.Tech. program in Information Technology (IT) with an intake of 140 students.

The department also offers two M.Tech. programs in Computer Engineering and Cyber Security, and has been awarding Ph.D. degrees since 2002. To date, 56 Ph.D.s have been awarded, with 31 currently in progress.

The faculty specialize in areas such as Distributed Computing, Software Engineering, Computer Networks, Database and Data Mining, Natural Language Processing, Information and Cyber Security, and Image Processing.

Mentor

Assistant Professor Dr. Vikram Singh

Dr. Vikram Singh, an esteemed faculty member in the Computer Engineering Department at NIT Kurukshetra, served as my mentor during this internship. Dr. Singh holds a Ph.D. from NIT Kurukshetra, an M.Tech from JNU, New Delhi, and a B.Tech from UIT, RGPV Bhopal. His extensive experience spans both industry and academia, with three years in the industry (2009-2012) and over a decade in academia (2012 onwards).

Dr. Vikram Singh's teaching interests include Database Systems, Data Mining & Data Warehouse, Information Science (Retrieval & Web Search), and Human-Computer Interaction (HCI). His current research focuses on Exploratory Data Analytics, Human-Information-Interaction, Neural Models for Interactive Information Search, Large Language Models (LLM), Soft Data Analytics & Evaluation (Sentiment, Micro-Emotions/Micro-expression), Social Network Science & Applications, and the application of ML/DL & AI in real-world scenarios. He has an impressive record of 75 research publications, including 14 journal papers and 61 international conference papers as of July 2023.

I would like to express my sincere gratitude to Dr. Vikram Singh for his invaluable guidance and support throughout my internship. His expertise and mentorship were instrumental in the successful completion of my project. Working under his supervision has been a profoundly enriching experience, providing me with deep insights into the field of recommendation systems and the practical application of machine learning techniques.

2. Literature Review

Introduction to Recommendation Systems

A recommendation system, also known as a recommender system, is an information filtering technology that assists users in finding relevant items or content based on their preferences, interests, and past behaviour. It is widely used in various domains, including e-commerce, entertainment, social media, and more. The primary goal of a recommendation system is to provide personalized recommendations that enhance user experience, engagement, and satisfaction.

Recommendation systems leverage advanced algorithms and techniques to analyse large datasets, such as user profiles, item attributes, and historical interactions. These systems strive to understand user preferences and interests, as well as the characteristics of items or content, in order to make accurate and relevant recommendations .

Evaluation of recommendation systems is essential to measure their performance and effectiveness. Metrics like precision, recall, accuracy, and mean average precision are commonly used to evaluate the quality of recommendations and compare different algorithms.

2.1 Fundamental steps to the Recommender System

(i). **Define the objective:** Identify the purpose of your recommendation system. Determine what you want to recommend and to whom, whether it's products, movies, articles, or any other items.

(ii). **Pre-process data:** Collect relevant data about users and items. Cleanse and pre-process the collected data. This involves handling missing values, removing noise, normalizing data, and transforming it into a suitable format for analysis.

(iii). **Implement personalization:** Customize recommendations for individual users by incorporating user-specific features and preferences. Personalization can significantly enhance the user experience and increase engagement.

(iv). **Select recommendation algorithm:** Choose an appropriate recommendation algorithm based on your objective and available data. Common algorithms include collaborative filtering, content-based filtering, matrix factorization, and deep learning-based models.

(v). **Incorporate feedback:** Continuously gather feedback from users to enhance the recommendation system. Feedback can be collected through explicit ratings, implicit feedback (e.g., clicks, purchase history), or user surveys. Incorporate this feedback to refine the model and improve recommendations.

2.2 Types of Recommendation Systems:

(i). **Content-based Filtering:** This type of recommendation system uses the characteristics and attributes of items to make recommendations. It analyses user preferences and recommends items that are similar to the ones the user has liked in the past. For example, if a user has shown interest in action movies, the system will recommend other action movies .

(ii). **Collaborative Filtering:** Collaborative filtering considers the preferences and behaviour of multiple users to make recommendations . It identifies similarities in user preferences and recommends items that users with similar tastes have liked. There are two main types of collaborative filtering:

(a) **User-based Filtering:** This approach finds users with similar preferences and recommends items to users based on similarity to other users.

(b) **Item-based Filtering:** This approach identifies similar items based on user preferences and recommends items that are similar to the ones a user has liked.

(iii). **Hybrid Recommendation Systems:** Hybrid systems combine multiple recommendation techniques to provide more accurate and diverse recommendations. For example, a hybrid system may use both content-based and collaborative filtering approaches to leverage the advantages of both methods.

2.3 Challenges and Issues

This proposal's coverage of conceptual work has identified the issues in existing research work. The main purpose of following details is to highlight the potential strategy for mentioned issues.

Functional Issues:

Functional issues refer to the problems associated with the functionality of recommendation system algorithms. These issues can affect the effectiveness and usability of the recommender system.

(i). **Long Tail**: The system usually recommends popular items neglecting unpopular ones. On the other hand, items generally have different levels of exposure to users. Hence, the recommendation system may be skewed towards the particular items favored. In these settings, training new RSs from interaction data available by the previous model makes a feedback loop that usually affects the diversity in recommendations also.

(ii). **Cold Start Problem**: With the entry of a new user or item, the system isn't aware of the user preferences (lack of historical interaction with the user)/ Item's rating (lack of feedback/rating from consumers), so it becomes difficult for the systems to suggest accurate recommendations with limited user/item knowledge (or minimal interaction). A cold start problem can be problematic when it comes to multiple domains because a service in one domain needs to communicate with a service in another. Session based recommendation systems alleviate cold start problems by providing personalized recommendation based on the user's current session or browsing behavior.

(iii). **Shieling attack:** Shieling attack is a type of attack where a malicious user profile and item description are injected to alter the review and rating decision of the recommender system. Such an attack alters the recommender process to promote and demote a particular product.

Non-Functional Issues

Non-functional issues in recommendation systems are not directly related to the functional requirement of the system but impact the overall performance, usability, and reliability of the system.

(i). **Performance:** The recommendation system requires significant computational resources when dealing with large and complex information.

(ii). **Scalability:** With the explosive growth of data, the recommendation system should be able to provide timely and relevant recommendations.

(iii). **Accuracy:** The performance of the recommendation system on user-item interaction is evaluated based on different accuracy measures such as *MAP, MRR, Precision, and Recall.*

(iv). **Data Sparsity:** It refers to the situation where the available data about user interaction with an item is sparse.

3. Cross-Domain Recommendation

Cross Domain Recommendation transfers knowledge across domains based on similarity of user and item. It combines information (for e.g., review and rating) from multiple source domains and transfers it to target domain, primarily to overcome the drawback of single domain recommendation system (SDRS). SDRS is unable to capture the full spectrum of user's interest and evolving preferences. Generally, domains are defined at four levels: *Attribute level, Type level, Item level, and System level*. In each level, different information sets are available, and CDRS often utilizes information from different levels of a domain to handle Cold-Start problems in recommendation. Few existing works highlight the needs of CDRS to alleviate data sparsity issues as well.

In recent years, Multilayer Perceptron (MLP), a Neural Network ecosystem, is used to learn non-linear mapping functions across domains. In these, MLP takes the user latent factor in source domain as input and user latent factor in target domain as output. Different approaches, such as Knowledge transfer, Feature Engineering, Hybrid model, Meta and Transfer learning have also been implemented to develop the Cross-domain recommendation system for different application domains. For example: Imagine an online platform that recommends both books and movies. A user who frequently rates science fiction books highly may have their preferences transferred to the movie domain. The system might use the user's book ratings to recommend popular science fiction movies or use shared features like the genre of books and movies to suggest titles that match their interests.

Example Process:

1. **Collect Data**: Gather user ratings and reviews for books.
2. **Analyze Preferences**: Identify the user's interest in science fiction.
3. **Transfer Knowledge**: Use this preference to recommend science fiction movies.
4. **Refine Recommendations**: Continuously adjust recommendations based on new data from both books and movies.

3.1 Current approaches of Cross-domain Recommendation System:

Cross-domain recommendation aims to provide personalized recommendations across multiple domains. There are several approaches used in developing such a system.

(i). Hybrid Approaches

(ii). Shared Nearest Neighbor

(iii). Cross-domain Collaborative Filtering

(iv). Joint Factorization Model

3.2 Working Steps of Cross-domain Recommendation System (CDRS)

(i). Cross-domain recommendation system collects data on user-item interaction from multiple domains [36]. Data may be selected according to the match of user profile, and item attribute information.

(ii). Common techniques like collaborative filtering, content-based methods, and matrix factorization are used to extract latent features from data

(iii). Transfer learning technique is used for latent features mapping across multiple domains

(iv). Finally, the recommendation model is used to generate recommendation from multiple domains.

Key Concepts

1. **Enhanced Data Utilization**: By integrating data from various domains (e.g., movies and books), cross-domain systems can make more informed recommendations. For example, if a user frequently rates sci-fi books highly, the system might suggest sci-fi movies based on this related interest.
2. **Cold-Start Problem Mitigation**: New users or items with insufficient data can benefit from existing data in related domains. For instance, a new user's book preferences might be used to recommend movies even before they have provided ratings in the movie domain.
3. **Domain Definitions**: Domains can be defined in various ways, such as broad categories (e.g., movies vs. books) or more specific subcategories (e.g., textbooks vs. novels). Understanding the context and nature of these domains is crucial for effective recommendations.

3.3 Motivation

Traditional RS tends to exploit user data and user-item interaction property for the personalized recommendation. The key challenges such as data sparsity and cold-start problem should be addressed that could trigger the development of modern recommendation systems.

CDRS enhances the user experience and satisfaction by leveraging data from multiple domains instead of a single domain, thereby addressing the limitation of conventional methods. The motivation behind the CDRS is to resolve the various issues that are unsolvable through a single-domain recommendation system. The objective of the CDRS tends to utilize information in the target domain generated by the source domain. It resolves the identified issues more accurately as compared to the SDRS because it transfers the item features from a richer domain to a sparser domain. It uses various techniques such as matrix factorization with regularization and users & items side information to resolve data sparsity problems.

3.4 Classification of Cross-Domain Recommendations

Cross-domain recommendations can be classified based on several factors:

1. **Single-Domain vs. Multi-Domain:** Single-domain recommendation system utilizes the information from one platform due to which it suffers from data sparsity and cold-start problems. In multi-domain data and information is utilized from multiple platforms. In multiple domains, data is trained on the source domain to perform well on the target domain. Multi-domain recommendation system helps to leverage the data-sparsity and cold-start problem.
2. **User-Based vs. Item-Based Transfer:** User-based transfer involves using data from the same users across different domains. Item-based transfer leverages similarities between items across domains. For example: Using a user's book ratings to recommend movies (user-based) vs. using the genre similarities between books and movies (item-based).
3. **Feature-Level vs. Model-Level Transfer:** Feature-level transfer involves sharing features across diverse domains. The features of users and items are extracted from the source domain using a certain filtering algorithm and on the behalf of similarity these features are mapped across diverse domains. Model-level transfer uses domain adaptation techniques to train the model on the source domain in order to generate the relevant and accurate recommendation on the target domain. For example: Sharing user demographic features (feature-level) vs. using a book recommendation model to help recommend movies (model-level).

3.5 Cross Domain Recommendation System Building Blocks

Data Collection Techniques

The study of different building blocks of CDRS includes a spectrum of techniques. These techniques primarily begin with the task of data acquisition or extraction applied to the source and target domain. The main objective of the data collection task is to acquire the relevant data objects with features to deliver enhanced performance.

Feature Embedding or Extracting latent feature

This is the process of identifying common features between both domains and then creating a feature mapping between these domains. Before the mapping, the embedding process takes place where features of the item and user transform into a latent space identified for each domain uniquely. Different techniques such as the *content-based method, collaborative filtering, and hybrid model* are used to train particular domains on behalf of their features to improve recommendation performance.

Mapping and transfer learning across domains

Mapping and transfer learning techniques are used to leverage knowledge from the source domain and transfer it to the target domain. In embedding, features of user and item attributes are **transformed** into the latent factor. Various deep learning and domain adaptation techniques are employed to fine-tune the model on the source domain to perform well in the target domain

Recommendation Generation and Evaluation Approaches

Cross-domain recommendation system makes recommendations in target domains based on knowledge learned from multiple source domains. Different ranking algorithms are used to rank the product of the user's preference. Evaluating recommendation system performance is essential to determine their effectiveness in providing relevant and personalized recommendations to users. The performance of the recommendation systems is evaluated by different accuracy measures such as Recall, Precision, F1-Score, etc.

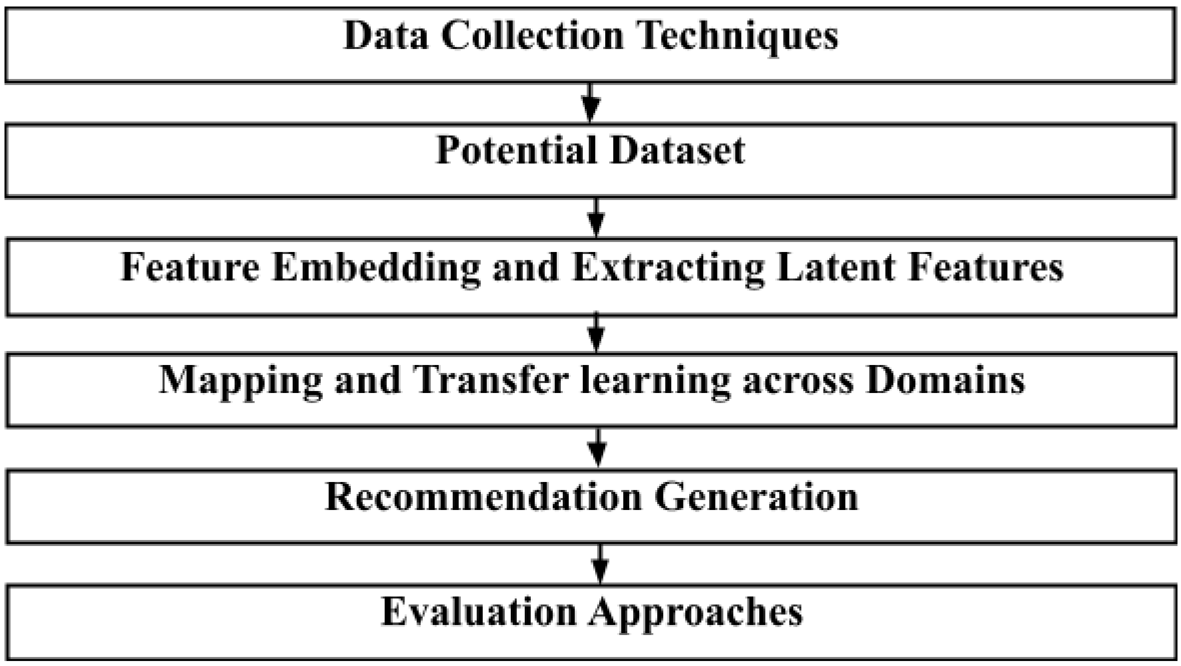

Data Collection Techniques
Potential Dataset
Feature Embedding and Extracting Latent Features
Mapping and Transfer learning across Domains
Recommendation Generation
Evaluation Approaches

Figure 1: CDRS Building Block

3.6 Problems and Challenges in Cross-Domain Recommendations

Cross-domain recommendation systems are complex and face various challenges when attempting to provide recommendations across different domains. In cross-domain recommendation systems, authors have investigated 3 main issues: The first challenge is to understand the correlation or similarity between items (e.g., products, content, services) in different domains. This involves determining how items in one domain are related to or can be mapped to items in another domain. To address this issue, techniques such as content-based and collaborative filtering, matrix factorization, or deep learning methods can be employed. By building a comprehensive understanding of item similarity across domains, you can better align user preferences in one domain with potential choices in another.

We define the cross-domain recommendation problem as follows:

(i). **Mapping across different domains:** Cross-domain recommendation systems must tackle the challenge of mapping user preferences and item characteristics across disparate domains. Each domain may have its unique set of features, making it challenging to find common ground for recommendations. For instance, mapping a user's interest in books to their preferences in movies can be non-trivial due to the distinct nature of these domains.

(ii). **Establishing content-based relations:** To enable cross-domain recommendations, it's essential to establish content-based relationships between items. This involves identifying and quantifying similarities and associations between items from different domains based on their features. However, this can be difficult because items in different domains may have diverse attributes, making it challenging to find meaningful correlations.

(iii). **Negative knowledge transfer:** When building a cross-domain recommendation system, transferring knowledge from a source domain to a target domain is common. However, negative knowledge transfer can occur when the characteristics or preferences that worked well in the source domain do not apply to the target domain. This can degrade recommendation performance as irrelevant information is carried over.

(iv). **Data sparsity:** Data sparsity is a common problem in cross-domain recommendation systems. In scenarios, where one domain has significantly fewer users and items than another, the system may struggle to generate meaningful recommendations. This issue can be particularly challenging when there are insufficient interactions and user-item data in the smaller domain.

(v). **Diverse domain characteristics:** Different domains often exhibit distinct item types, user behaviours, and contextual properties. These disparities can hinder the correlation of items across domains. For example, attempting to recommend restaurants based on a user's music preferences might not be straightforward, given the disparity in the nature of these domains.

(vi). **Data leakage:** Combining data from multiple domains may create security and privacy concerns. Even when data is anonymized, it may still be possible to re-identify individuals or organizations if the data contains unique identifiers or if other publicly available data can be cross-referenced with the combined dataset. Different domains might have varying levels of security protocols and data protection measures. Sharing data between these domains can expose vulnerabilities, as one domain's weaker security could compromise the entire dataset's integrity.

(vii). Cold start problem: For new users, with little historical information it is difficult to infer the user's preference. Traditional collaborative filtering methods rely on historical user and item interactions but without this data, it is difficult to generate recommendations across domains for new users and items.

Natural Language Processing in Recommendation Systems

Natural Language Processing (NLP) has transformed recommendation systems by enabling them to analyze and interpret textual data, such as user reviews, product descriptions, and social media interactions. By leveraging NLP techniques, recommendation systems can offer more accurate and contextually relevant suggestions.

Key NLP Techniques in Recommendation Systems:

1. **Text Embeddings:**
   - **Definition:** Text embeddings are numerical representations of textual data that capture semantic meaning. They convert words, phrases, or entire documents into fixed-dimensional vectors, where similar texts are represented by similar vectors.
   - **Models:**
     - **Word2Vec:** Generates word vectors by analyzing the context of surrounding words in a sentence. It can be trained using either the Continuous Bag of Words (CBOW) model or the Skip-Gram model.
     - **GloVe (Global Vectors for Word Representation):** Creates word embeddings by leveraging global statistical information from a corpus, based on the frequency of word co-occurrences.
     - **BERT (Bidirectional Encoder Representations from Transformers):** A transformer-based model that understands the context of words by considering their surrounding words in both directions, enabling it to capture more nuanced meanings.
   - **Applications:** Text embeddings can be used to capture the thematic essence of book descriptions, movie plots, or product reviews. For instance, embeddings can help identify books with similar genres or themes based on their descriptions, leading to more relevant recommendations.

2. **Sentiment Analysis:**
   - **Definition:** Sentiment analysis involves determining the emotional tone or sentiment expressed in a text. It can classify text as positive, negative, or neutral and can even detect specific emotions like joy, anger, or sadness.
   - **Techniques:**
     - **Lexicon-Based:** Utilizes predefined lists of sentiment-associated words, aggregating their scores to determine the overall sentiment of a text.
     - **Machine Learning-Based:** Involves training models on labeled datasets to predict sentiment, using algorithms such as Logistic Regression, Naive Bayes, or Support Vector Machines (SVM).
     - **Deep Learning-Based:** Employs neural networks, like LSTM (Long Short-Term Memory) or Transformers, to capture more nuanced sentiment patterns in text.
   - **Applications:** Sentiment analysis can be applied to user reviews to gauge the sentiment toward specific books, movies, or products. For example, if a user consistently leaves positive reviews for science fiction books, the system can recommend more books in that genre, considering the user's positive sentiment.
3. **Topic Modeling:**
   - **Definition:** Topic modeling is a technique used to uncover hidden thematic structures within a large collection of documents. It identifies recurring topics that co-occur frequently, providing insights into the document's content and structure.
   - **Techniques:**
     - **Latent Dirichlet Allocation (LDA):** A generative probabilistic model that assumes documents are mixtures of topics and topics are mixtures of words. It assigns probabilities to words and documents to infer the distribution of topics.
     - **Non-Negative Matrix Factorization (NMF):** A matrix decomposition technique that factors the term-document matrix into two lower-dimensional matrices, representing topics and their importance in documents.
     - **Latent Semantic Analysis (LSA):** Analyzes relationships between documents and terms by performing singular value decomposition (SVD) on the term-document matrix.
   - **Applications:** Topic modeling can identify common themes in product descriptions or user reviews, grouping similar items together. For example, it can cluster books into categories like mystery, romance, or fantasy based on their content, improving the relevance of recommendations for users with specific genre preferences.

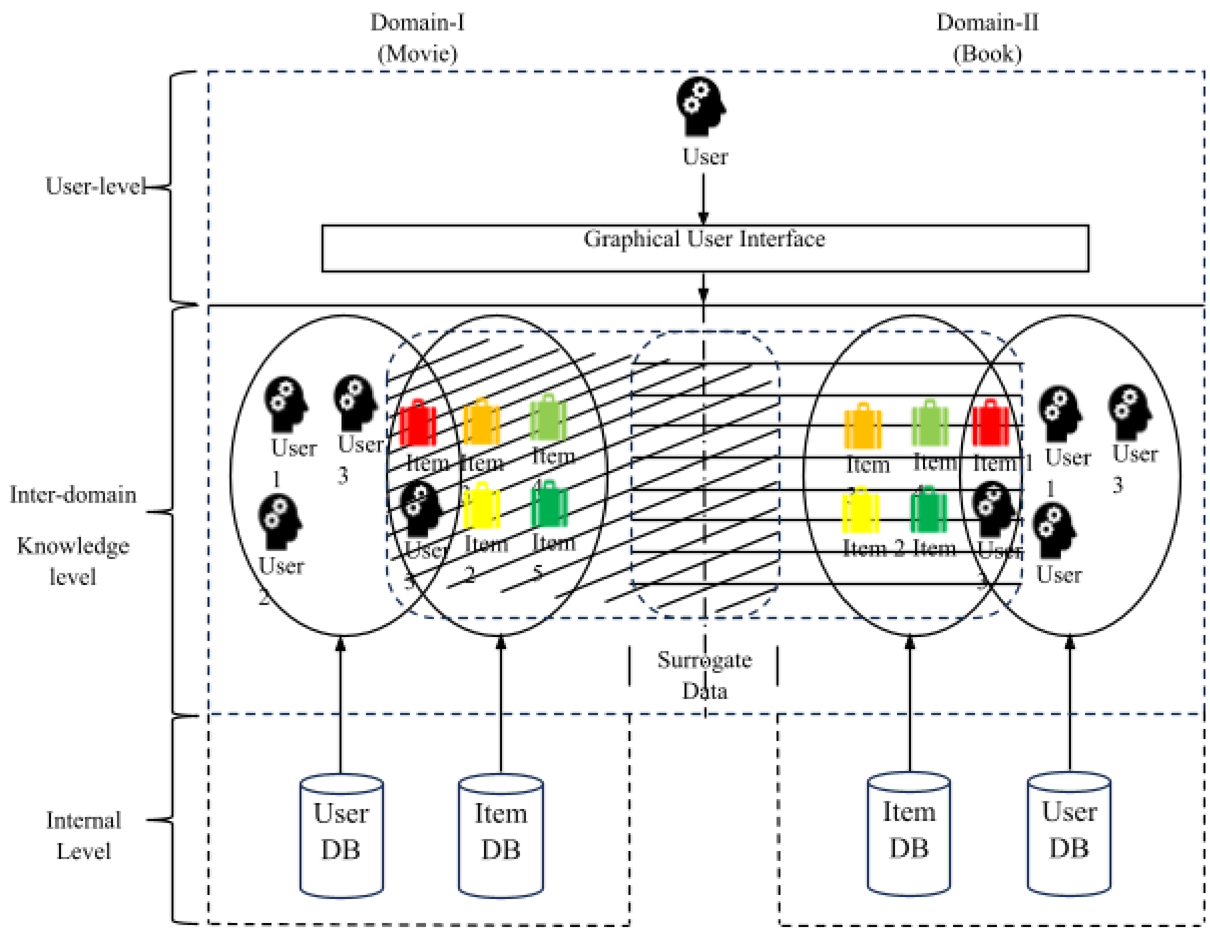


Figure 2: A Systematic view of the generic CDRS system

4. Project Description

4.1 Project Objectives

The primary objective of this project was to develop a book-movies cross-domain recommendation system. The specific objectives included:

1. **Data Collection and Preprocessing:**
    - To gather a comprehensive dataset of books, including their names, descriptions, and genres.
    - To preprocess the dataset by converting book genres to vectors representing 18 major genres and extending the model's understanding with a genre dataset containing 191 subgenres.
2. **Model Training:**
    - To train a RoBERTa model on the books dataset to understand the semantic relationships between books and movies.

    - To utilize multilabel classification with the Simple Transformers library to predict movie recommendations based on user reading habits.
3. **System Evaluation:**
    - To evaluate the performance of the recommendation system using appropriate metrics such as precision, recall, and F1-score.
4. **Web Application Development:**
    - To develop a functional web application that provides users with movie recommendations based on their book choices.
    - To implement a user-friendly interface for inputting book preferences and receiving recommendations.

4.2 Roberta Model Overview

RoBERTa (Robustly optimized BERT approach) is an advanced language representation model that builds on the foundation of BERT (Bidirectional Encoder Representations from Transformers). It enhances BERT by utilizing more data, larger batch sizes, and longer training sequences, leading to improved performance.

Key Features of RoBERTa

1. **Pre-training:**
    - **Process:** RoBERTa is pre-trained on a large corpus of text using a masked language model (MLM) objective. In this process, random tokens in the input are masked, and the model learns to predict the masked tokens based on the context provided by the surrounding tokens.
    - **Benefits:** This pre-training enables RoBERTa to capture rich linguistic features and general language patterns, making it highly effective for various NLP tasks.
2. **Fine-tuning:**
    - **Process:** After pre-training, RoBERTa is fine-tuned on specific tasks with smaller, task-specific datasets. Fine-tuning involves further training the pre-trained model to adapt it to the nuances of the new task.
    - **Example in the Project:** RoBERTa is fine-tuned on the book dataset to predict genres and recommend movies. This fine-tuning process adapts RoBERTa's general language understanding to the specific domain of book descriptions.
3. **Architecture:**
    - **Components:** RoBERTa retains the transformer architecture of BERT, consisting of multiple layers of self-attention and feed-forward neural networks. Each layer captures different aspects of the input text, allowing the model to build a comprehensive understanding.

    - **Example in the Project:** In processing book descriptions, the multi-layer transformer architecture allows RoBERTa to capture complex relationships and dependencies between words, enhancing the accuracy of genre predictions.

4. **Performance:**
    - **Achievements:** RoBERTa has demonstrated state-of-the-art performance on various NLP benchmarks, surpassing BERT on multiple tasks. Its robust training procedure and architecture contribute to its superior performance.
    - **Example in the Project:** RoBERTa's advanced capabilities enable the recommendation system to accurately classify book genres and make reliable cross-domain recommendations.

Application of RoBERTa in the Project

1. **Generating Embeddings:**
    - **Process:** RoBERTa generates dense vector embeddings for book descriptions. These embeddings capture the semantic content of the descriptions, providing a rich representation for further processing.
    - **Benefits:** The embeddings help the system identify similar books and relevant movie recommendations, enhancing the overall accuracy and relevance of the recommendations.
2. **Multilabel Classification:**
    - **Process:** The model is fine-tuned to handle multilabel classification, predicting multiple genres for each book based on its description.
    - **Benefits:** This capability allows the system to provide detailed and accurate genre predictions, which are crucial for effective cross-domain recommendations.
3. **Improving Recommendation Accuracy:**
    - **Process:** By leveraging RoBERTa's advanced language understanding capabilities, the system can make more accurate and relevant movie recommendations based on the user's reading habits.
    - **Benefits:** Users receive personalized and high-quality recommendations, enhancing their overall experience and satisfaction with the system.

Role of RoBERTa Model in Recommendation Systems

RoBERTa (Robustly optimized BERT approach) is a state-of-the-art NLP model developed by Facebook AI. It is an optimized version of BERT (Bidirectional Encoder Representations from Transformers) that improves performance by training with more data and computational resources.

RoBERTa has demonstrated superior performance in various NLP tasks due to its robust training procedure and larger training dataset.

Applications of RoBERTa in Recommendation Systems:

1. **Generating Item Embeddings:**
   - RoBERTa can generate high-quality embeddings for item descriptions, capturing their semantic meaning.
   - These embeddings can be used to find similar items and enhance content-based recommendations.
   - Example: Generating embeddings for book descriptions to recommend similar books.
2. **Modeling User Preferences:**
   - RoBERTa can analyse user reviews and textual interactions to model user preferences accurately.
   - It helps in understanding the context and nuances of user opinions.
   - Example: Analysing user reviews of books to infer preferences and recommend relevant movies.
3. **Multilabel Classification:**
   - RoBERTa can be used for multilabel classification tasks, which are common in recommendation systems.
   - It can predict multiple labels for items, such as genres or categories from the description provided to the model.
   - Example: Predicting multiple genres for books and using this information for cross-domain recommendations.

In this project, the RoBERTa model was utilized to understand the relationships between books and movies by training it on a dataset of book descriptions and genres. The model's ability to generate meaningful embeddings and perform multilabel classification was key to developing an effective cross-domain recommendation system.

4.3 Project Experimental Setup

<u>Tools and libraries required:</u>

Data Processing:

- Reading and Analyzing Data:
  - **pandas** for data manipulation
  - **numpy** for numerical operations
- Text Preprocessing:
  - Normalization, Tokenization, and Lemmatization:

        - **nltk** library:
            - Stopwords Removal: Using **stopwords** from nltk.corpus
            - Tokenization: Using **word_tokenize** from nltk.tokenize
            - Lemmatization: Using **WordNetLemmatizer** from nltk.stem
    - Noise Removal: Regular expressions (**re** module)
- Non-English Data Removal:
    - **langdetect** library:
        - Language Detection: Using detect from langdetect
        - Seed Initialization: DetectorFactory.seed = 0 to ensure consistent results

Data Handling:

- **json**: For reading genre data.
- **pickle**: For saving and loading the trained model.
- **pandas**: For reading and handling CSV data.
- **numpy**: For numerical operations.
- **math**: To compute the training and testing data split.

Model Training:

- **Model Type**: simpletransformers library is used for training a multi-label classification model based on **RoBERTa**.
- **Model Details:**
    - RoBERTa Model: Utilizes **MultiLabelClassificationModel** from simpletransformers for handling multiple labels per book.
    - Arguments: Model training involves specifying parameters such as learning rate, batch size, and number of epochs.

Recommendation Generation:

- **Feature Scaling and Similarity Computation:**
    - **Feature Scaling:** StandardScaler from sklearn.preprocessing is used to standardize feature vectors.
    - **Recommendation Model:** NearestNeighbors from sklearn.neighbors is used to find the nearest neighbors based on cosine similarity between book vectors.

Web App

Server-Side (API Creation)

- **Flask**:
    - **Framework**: Flask is used to create the web server and handle API requests.
    - **Components**:
        - Flask: The core Flask class for creating the application.
        - CORS: flask_cors is used to handle Cross-Origin Resource Sharing, allowing the client-side app to communicate with the server.
        - configure_routes: Custom module for setting up API endpoints and routing.
    - **Data Handling**:
        - pandas: Used for data manipulation and handling.
        - numpy: Utilized for numerical operations.
        - pickle and json: For model loading/saving and handling JSON data.
    - **Endpoints**:
        - request: To handle incoming requests.
        - jsonify: To format responses as JSON.

Client-Side

- **React**:
    - **Library**: React is used for building the user interface of the web application.
- **TypeScript**:
    - **Language**: TypeScript provides static typing for JavaScript, enhancing code quality and maintainability in the React application.
- **TailwindCSS**:
    - **Utility-First CSS Framework**: TailwindCSS is used for styling the application with utility classes, allowing for rapid UI development.
- **Styled-Components**:
    - **CSS-in-JS**: styled-components is used to apply component-level styles, enabling scoped and dynamic styling within the React components.

4.4 Data Collection and Preprocessing

Book Dataset:

- **Source:** The "Goodreads Best Books Ever" dataset was collected from a publicly available source.
- **Composition:** This dataset comprises 52,478 entries of various books, with 49,927 unique book titles after removing duplicates.
- **Fields/Columns:**
    - **Retained**: title, rating, description, language, isbn, genres, numRatings
    - **Dropped**: series, author, characters, bookFormat, edition, pages, publisher, publishDate, firstPublishDate, awards, ratingsByStars, likedPercent, setting, bbeScore, bbeVotes, price

Movie Dataset:

- **Source:** The "IMDB Movies Dataset," available on Kaggle.
- **Composition:** This dataset contains 10,178 entries representing various movies.
- **Fields/Columns:**
    - **Retained**: names, score, genre, overview, orig_title
    - **Dropped**: date_x, crew, status, orig_lang, budget_x, revenue, country

Genre Data:

- **Source:** Additional genre data was collected to extend the model's understanding.
- **Composition:** This dataset consists of 18 major genres and 191 subgenres, which were used to represent book genres in a more granular manner.

Book Dataset Details

The primary focus of the book dataset is on the following fields:

- **Title:** The name of the book.
- **Rating:** The average rating given to the book by readers.
- **Description:** A detailed description of the book's content.
- **Language:** The language in which the book is written.
- **ISBN:** A unique identifier for the book.
- **Genres:** The genres associated with the book.
- **Number of Ratings:** The number of ratings the book has received.

Movie Dataset Details

The movie dataset focuses on the following fields:

- **Names:** The title of the movie.
- **Score:** The IMDB score of the movie.
- **Genre:** The genres associated with the movie.
- **Overview:** A brief description of the movie's plot.
- **Original Title:** The original title of the movie.

4.5 Algorithm for the Book-Movie Recommendation System

Input:

- P_target: Target book
- N_rec: Number of recommendations to return

Output:

- Rec(P_target, N_rec): List of recommended movies

1. **Data Preprocessing**
    - Loads the books and movies datasets into Pandas DataFrames.
    - Preprocesses two datasets: books_df and movies_df.
    - Removes irrelevant columns and rows with missing values.
    - Extracts the primary genre from the genre column in movies_df.

2. **Text Cleaning**
    - **Clean and normalize text:**
        - Remove stop words, punctuation, and special characters using regular expressions.
        - Convert text to lowercase.
        - Separate concatenated words and numbers.
        - Remove patterns related to numbers, dates, and specific words.
    - **Filter non-English content:**
        - Implement a language detection mechanism (e.g., using a language detection library) to identify non-English content.
        - Remove books and descriptions written in languages other than English.
    - **Ranking:**

        Books

        - Calculate a weighted score based on rating and numRatings.

- Sort books by the calculated weighted_score.

Movies

- Calculate a weighted score based on score and revenue.
- Sort movies by the calculated weighted_score.

3. **Feature Extraction**
    - Load genre mapping data from book_genres and store it in major_genres.
    - Create a dictionary from major_genres where the key is 'genre' and the value is 'includes' containing sub genres.
    - **Feature Mapping**
        - **Function:** map_feature(feature_list, target)
        - Convert the feature_list and the value corresponding to target in feature_dictionary to sets (sft and mft).
        - If target is in sft or the intersection of sft and mft is not empty, return 1. Otherwise, return 0
        - **Feature Mapping Function:**
            - f(feature_list, target) = 1 if $\text{target} \in \text{sft}$ or $\text{sft} \cap \text{mft} \neq \varnothing$
            - f(feature_list, target) = 0 otherwise
    - Apply the map_feature function to the 'genres' column of books_dataframe and store the result.
4. **Label Vectorization**
    - Create an empty vector of length equal to the number of features in book_features.
    - For each feature in book_features:
        - If the corresponding value in the row is 1, set the corresponding element in the vector to 1. Otherwise, set it to 0.
    - Store the resulting vector in a column 'labels' in the 'books' dataset containing genres information.
5. **Model Training**
    - **Data Preparation:**
        - Split the dataset containing book descriptions and genre labels into training and testing sets (70%/30%).
        - Prepare the data for the model by converting descriptions to text format and labels to numerical vectors.

- **Model Configuration:**
  - Define hyperparameters for training the model, such as the number of epochs, batch size, train size, learning rate, etc.
  - Specify the number of output labels based on the number of features (genres) extracted earlier.
- **Model Training:**
  - Train a pre-trained model (e.g., Roberta) on the prepared training data for multi-label classification.
- **Model Evaluation:**
  - **LRAP** (Label Ranking Average Precision): This metric measure how well the model ranks relevant genres higher than irrelevant ones, crucial for multi-label tasks.
  - **Binary Cross Entropy Loss** (eval_loss): This measures the model's ability to distinguish between genres a book belongs to and those it doesn't. Lower loss indicates better performance.

6. **Recommendation Generation**

- **Genre Prediction for Movies:**
  - Utilize the pre-trained multi-label classification model (trained on book descriptions and genres) to predict genre labels for movie descriptions.
  - Store the predicted genre labels for each movie in a dedicated column within the movie dataset.
- **Nearest Neighbor Model Training:**
  - Convert the predicted genre labels (represented as vectors) into a numerical array suitable for the Nearest Neighbors model.
  - Train the Nearest Neighbors model using the cosine similarity metric to find movies with similar genre profiles.
- **Recommendation:**
  - **Input**: User's preferred book genre vector (obtained from their book selections).
  - **Process:**
    - Use the Nearest Neighbors model to find the top-n movies with genre profiles closest to the user's preferred book genres (based on cosine similarity).

- **Post-processing:**
    - Sort the recommended movies based on a weighted score (potentially considering factors like movie ratings or popularity) stored in the movie dataset.
- **Output:**
    - Return the sorted list of recommended movies to the user.

4.6 Web application

Backend Server: Flask

Flask is utilized as the backend server for this web application due to its lightweight and flexible framework, which facilitates rapid development and seamless integration with Python-based machine learning models. Its simplicity in setting up RESTful APIs enables efficient handling of user interactions and model deployment.

Model and Data

- **Model**: Cross-Recommender-Model
- **Books Data**: Dataset containing book details.
- **Movies Data**: Dataset containing movie details.
- **Genre Labels**: A list of genres and sub-genres for classification.

Response Objects

```
@dataclass
class Movie:
    name: str = ''
    description: str = ''
    genre: str = ''
    classified_genre: str = ''


@dataclass
class Book:
    idx: int = -1
    name: str = ''
    description: str = ''
    genre: str = ''
    classified_genre: str = ''
```

API Routes

Route: /

- **Description**: Displays an overview of the dataset.
- **Returns**:
    - List of books in the Books Dataset.
    - List of movies in the Movies Dataset.
    - List of genres/labels considered.

Route: /api/records

- **Description**: Provides the count of records in the datasets.
- **Function**: Computes the total number of records.
- **Returns**:
    - Length of the books dataset.
    - Length of the movies dataset.

Route: /api/books

- **Description**: Retrieves a paginated list of books.
- **Arguments**: page_number (optional)
- **Function**: Fetches books based on the page number, initially providing 10 books and increasing the number on subsequent requests.
- **Returns**: List of Book objects with details from the Books Dataset.

Route: /api/recommend

- **Description**: Provides movie recommendations based on selected books.
- **Arguments**: book_ids (List of Book IDs)
- **Function**: Uses the Cross-Recommender-Model to find and return the top 5 recommended movies based on the nearest neighbor algorithm.
- **Returns**: List of Movie objects from the Movies Dataset.

Route: /api/movies

- **Description**: Displays a random selection of movies.
- **Returns**: List of 10 random Movie objects with details from the Movies Dataset, suitable for the initial user screen.

Route: /api/bookFilter

- **Description**: Filters books based on major genres.
- **Arguments**: filter_index (Index of Major Genres)
- **Function**: Creates a genre vector based on selected major genres and applies the nearest neighbor algorithm to find books that closely match the selected genres.
- **Returns**: Filtered list of Book objects from the Books Dataset.

Client Side: React JS

The client-side code for the web application is developed using React JS, a popular JavaScript library for building user interfaces. React's component-based architecture allows for the creation of dynamic and interactive web applications, making it an ideal choice for developing a responsive user experience.

Key Components:

- **HomePage Component**
    - **Purpose:** Displays an overview of available books and movies, as well as genre labels.

    - **Features**
        - Fetches and displays data from the / route.
        - Provides users with initial insights into the datasets and available genres.

- **BooksList Component**
    - **Purpose:** Renders a paginated list of books.
    - **Features:**
        - Fetches book data from the /api/books route.
        - Implements pagination to load more books dynamically as the user scrolls or navigates through pages.

- **MoviesList Component**
    - **Purpose:** Displays movie recommendations based on selected books.
    - **Features:**
        - Sends book IDs to the /api/recommend route to get movie recommendations.
        - Shows a list of recommended movies to the user.

- **BookFilter Component**
    - **Purpose:** Allows users to filter books based on selected genres.
    - **Features:**
        - Sends genre filter indices to the /api/bookFilter route.
        - Displays books that match the selected genre filters.

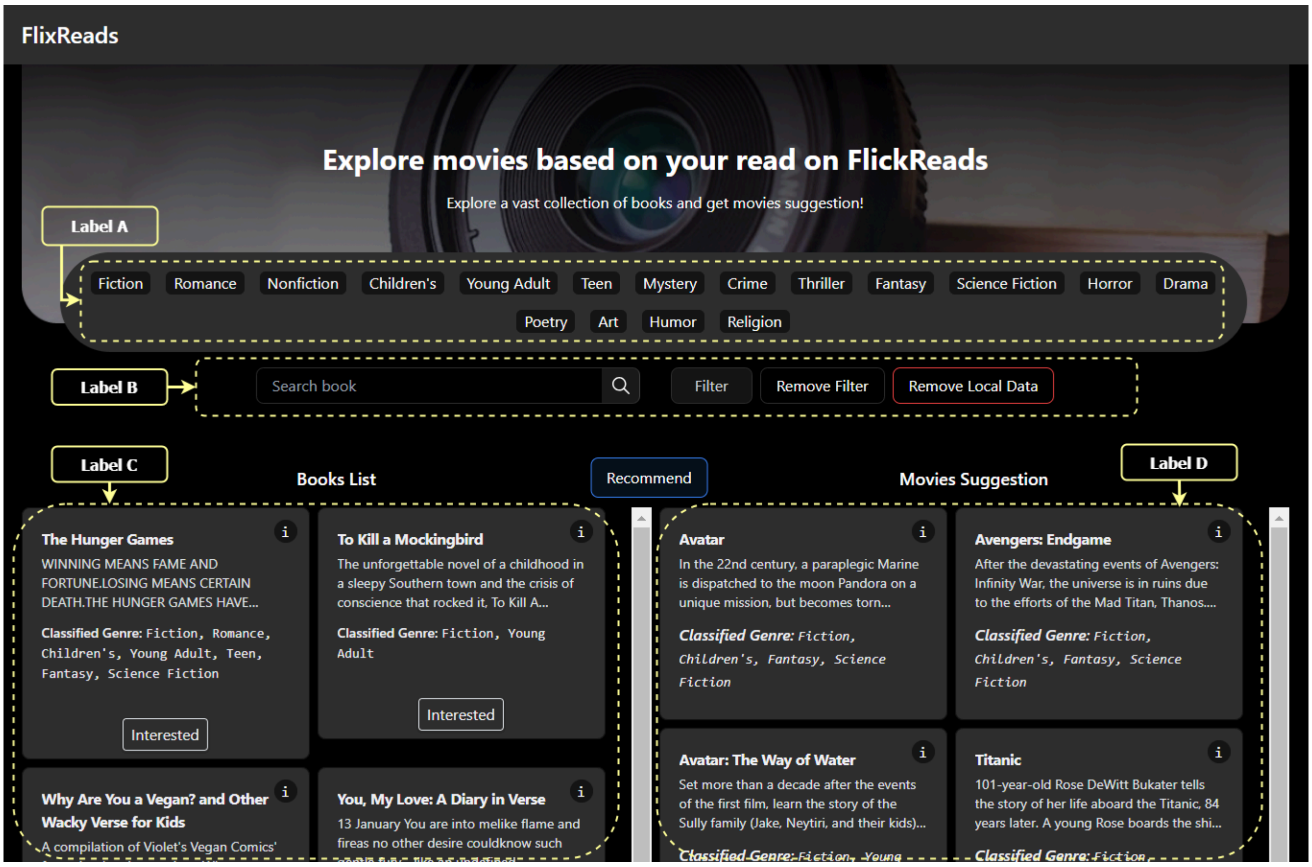
Data Handling:

- **State Management:** Utilizes React's **useState** and **useEffect** hooks for managing and updating component state based on API responses.
- **API Integration:** Makes asynchronous requests to the Flask backend using **fetch** to retrieve and send data.
- **Pagination:** Implements logic to handle pagination and dynamic loading of book data.

User Interface:

- **Design**: Emphasizes a user-friendly and responsive design, ensuring compatibility across various devices and screen sizes.
- **Interactivity**: Includes interactive elements such as buttons and filters to enhance user engagement and provide a seamless experience.

- **Styling**: Utilizes **Tailwind CSS** for utility-first styling, enabling rapid and flexible design customization. **Styled-Components** are used for creating reusable and encapsulated components with scoped CSS, enhancing maintainability and consistency in styling.



The Hunger Games
WINNING MEANS FAME AND FORTUNE.LOSING MEANS CERTAIN DEATH.THE HUNGER GAMES HAVE...
Classified Genre: Fiction, Romance, Children's, Young Adult, Teen, Fantasy, Science Fiction

To Kill a Mockingbird
The unforgettable novel of a childhood in a sleepy Southern town and the crisis of conscience that rocked it, To Kill A...
Classified Genre: Fiction, Young Adult

Why Are You a Vegan? and Other Wacky Verse for Kids
A compilation of Violet's Vegan Comics'

You, My Love: A Diary in Verse
13 January You are into melike flame and fireas no other desire couldknow such

Avatar
In the 22nd century, a paraplegic Marine is dispatched to the moon Pandora on a unique mission, but becomes torn...
Classified Genre: *Fiction, Children's, Fantasy, Science Fiction*

Avengers: Endgame
After the devastating events of Avengers: Infinity War, the universe is in ruins due to the efforts of the Mad Titan, Thanos....
Classified Genre: *Fiction, Children's, Fantasy, Science Fiction*

Avatar: The Way of Water
Set more than a decade after the events of the first film, learn the story of the Sully family (Jake, Neytiri, and their kids)...
Classified Genre: *Fiction, Young*

Titanic
101-year-old Rose DeWitt Bukater tells the story of her life aboard the Titanic, 84 years later. A young Rose boards the shi...
Classified Genre: *Fiction*

Book - Movie Cross Domain Recommendation Web App Dashboard

[Label A] : Label filters to find the books
[Label B]: Search Input and Action button to perform book filtering and removing user data
[Label C]: Shows the list of all available book in our dataset
[Label D]: Show the Recommended movies based on selected book

5. Accuracy Measures

LRAP

Label Ranking Average Precision (LRAP) evaluates the ability of a model to rank relevant labels higher than irrelevant ones for each sample in a multi-label classification problem. Unlike traditional metrics, LRAP assesses how well the model prioritizes true labels over false ones by measuring the average precision of predicting labels across all samples. It essentially answers the question of what percentage of higher-ranked labels were actually true labels, thus providing insight into the quality of label ranking.

LRAP extends the concept of average precision to the context of label ranking. While average precision evaluates the accuracy of predictions based on the presence of labels, LRAP focuses on the order of labels, making it particularly suited for multi-label scenarios where the ranking of relevant labels is crucial. This metric helps in understanding how effectively the model ranks relevant labels compared to irrelevant ones.

In multi-label ranking problems, LRAP is valuable because it directly addresses the goal of prioritizing labels associated with each sample. By focusing on the ranking of labels rather than just their classification, LRAP ensures that the model's ability to deliver relevant and well-ordered label predictions is accurately measured, thereby enhancing the overall effectiveness of the recommendation system.

Cosine Similarity

In our recommendation system, Cosine Similarity was chosen due to its effectiveness in handling sparse, high-dimensional binary vectors that represent genres for books and movies. By measuring the angle between vectors rather than their magnitude, Cosine Similarity is well-suited for comparing genre vectors that are often sparse, containing many zeros. This metric normalizes the vectors, ensuring that the comparison focuses on the direction of the vectors, which is crucial for accurately identifying items with similar genre profiles regardless of the number of genres they possess.

Other metrics like Average Precision, Recall, and NDCG are generally used to evaluate the relevance and ranking quality of recommendations, but they require a predefined set of ground truth or user preferences to compute. These metrics assess the accuracy and completeness of the recommendations based on known relevant items, which are not available in our system. Without ground truth, it becomes challenging to determine what constitutes a "relevant" or "ideal" recommendation, rendering these metrics impractical for evaluating our model’s performance.

By relying on Cosine Similarity, we leverage a metric that effectively measures similarity based on the orientation of genre vectors, making it a robust choice for our multi-label classification task. This approach allows us to generate relevant recommendations based on genre alignment without needing explicit relevance judgments, which are essential for other traditional evaluation metrics.

6. Conclusion

As the volume of online data is increasing rapidly, it becomes essential to develop a recommendation tool that will provide the recommendation of user preference. Traditional recommendation systems that operate on single domains suffer from data sparsity and cold-start problems. These challenges are addressed by integrating the data of a book and movie domain. The RoBETa model captures the complex and semantic pattern of relationships across diverse domains. Implementing a multilabel classification model using a single transformer enhances the system's ability to predict user preferences and generate accurate recommendations. Evaluation metrics such as LRAP and cosine similarity improve the precision of recommendation compared to the traditional method used in single-domain recommendation. The evolution of a user-friendly web environment enables users to receive the preferred movie recommendation based on their book reading habits. In conclusion, the integration of diverse domains provides a solution to the problems of recommendation systems. We may pay attention to a user-centric application in the future.

7. Bibliography

1. Simple Transformers Library: https://simpletransformers.ai/docs/multi-label-classification/
2. Multi-Label Classification: A Survey (or) A Gentle Introduction to Multi-Label Classification (references can be found in Simple Transformers documentation)
3. Goodreads Best Books Ever dataset: https://www.kaggle.com/datasets/jealousleopard/goodreadsbooks
4. IMDB Movies Dataset: https://www.kaggle.com/datasets/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews
5. Label Ranking Average Precision Score: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.label_ranking_average_precision_score.html