DNN based movie recommendation system with explainable reasoning

M.A.D. Ruwanthilake.

218540F

Master of Science in Artificial Intelligence

Department of Computational Mathematics
Faculty of Information Technology

University of Moratuwa
Sri Lanka

April 2023

DNN based movie recommendation system with explainable reasoning

M.A.D. Ruwanthilake.

218540F

Thesis/Dissertation submitted in partial fulfillment of the requirements for the degree Master of Science

Department of Computational Mathematics
Faculty of Information Technology

University of Moratuwa
Sri Lanka

April 20

DECLARATION

I declare that this thesis is the result of my independent effort and has not been previously presented in any form for the purpose of obtaining another degree or diploma from any university or tertiary institution. Acknowledgment has been provided within the text for any information or findings derived from the published or unpublished works of others, and a comprehensive list of references is included.

Signature: Date: 04 July 2024

The above candidate has carried out research for the Master's thesis/dissertation under my supervision. I confirm that the declaration made above by the student is true and correct.

Name of Supervisor: Dr. Thushari Silva

Signature of the Supervisor: Date: 04 July 2024

ABSTRACT

A recommender system gives confidential private preferences to its users by evaluating users' historical data and behaviors. Since the early 1980s, the recommender system has been used as a popular symbolic AI technology. It comes under the broad umbrella of the Expert system. However, with the latest developments, The use of machine learning techniques and algorithms in recommender system development has been a natural way to increase prediction accuracy and address issues with data sparsity and cold start. Users and service providers can both benefit from recommender systems.

Frequently, visitors are directed to recommended articles that they were not expecting, which can lead to confusion. As a result, explainability becomes to be evaluated in direct relation to how well the recommendation system operates. Enhancing efficiency, transparency, and customer happiness leads to increased user loyalty. The recommendations are supported by justifications for the approaches. Hence, with the rapid growth of the demand for recommender systems, its landscape is evolving into explainable recommender systems for better quality. Since then explainable recommender systems become a research challenge. Some techniques for generating explanations of recommendations are provided by the academia. The Recommender system is depending more and more on these machine learning algorithms due to the recent success of machine learning models and algorithms. Explainable Artificial Intelligence is a new field that aims to produce high-quality suggestions and clear explanations of the outcomes while offering transparent learning techniques. This has opened a new era of recommender systems, revealing more complex relationships between users and products, offering highly sophisticated data visualizations, and discovering vast amounts of information in contextual, virtual, demographic, and textural data. Therefore, this thesis reports on the development of a machine learning-based recommender system solution with explainable reasoning.

DEDICATION

To my parents, my family, and all my teachers for their genuine motivation and collaboration towards the success of my educational life … !!!

ACKNOWLEDGMENT

I sincerely thank my adviser, Dr. Thushari Silva, for her unwavering guidance and support during my master's program. Her expertise and patience have been invaluable to me and have been necessary for me to finish my thesis.

I would especially want to thank the Department of Computation Mathematics staff members. They played a vital for the success this research.

I am genuinely thankful to my family and teachers for their courage and support during this research study. Without their motivation and support, I would not have been able to complete this study.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

Abbreviation	Description
AI	Artificial Intelligence
GUI	Graphical user interface
NLP	Natural language processing
DNN	Deep Neural Network
ML	Machine Learning
IoT	Internet of Things
RS	Recommender system
WSGI	Web Server Gateway Interface
CLI	Command Line Interface
CPU	Central processing unit
GPU	Graphics processing unit
IDE	Integrated development enviorment
VS	Visual Studio
SQL	Structured Query Language
NLTK	Natural Language Toolkit
UI	User Interface
HTML	Hyper Text Markup Language
CSS	Cascading Style Sheet
REST	Representational State Transfer
API	Apllication Programing Interface
DB	Data Base
FE	Front End
BE	Back End
FFN	Feed Forward Network

ORM	Object Relational Mapping
SVD	Singular Value Decomposition
PCA	Principal Component Analysis
SGD	Stochastic gradient Decent
RNG	Random Number generator
BST	Behavioral Sequence Transformer
RMSE	Root Mean Squared Error
MAE	Mean Absolute Error
KNN	K Nearest Neighbors
Std	Standard deviation
DDPG	deep deterministic policy gradient

CHAPTER 1

INTRODUCTION

1.1 Prolegomena

With the internet facilities and IoT devices, data are overloaded these days. The exponential growth of data and data sources caused to introduce difficulty in analyzing and organizing basic data. This opened new doors to a new era where recommendation systems were born [2]. Artificial intelligence-based personalized services assist in addressing the issue of information overload, which facilitates consumer decision processes and improves customer experience [3]. Recommendation systems give opinions to users based on their profile and history. There are many items for which a system can provide recommendations such as books, movies, friends, touring services, research articles, etc. presence of a recommendation system in a particular area may save time for users and provide better matching for them. However, many recommendation systems are unable to reason out how they reach the opinion of which user is interested [1]. Most machine learning-based recommendation systems behave like a black box and they are unable to explain how they reach the recommended option.

Recommendations frequently result in confusion since the user does not anticipate receiving them. As a result, explainability becomes a crucial component of the recommendation system's quality metrics. Providing evidence to back up the explanations and arguments for the recommendations enhances productivity, transparency, user happiness, and trust. Getting an explanation or a rationale is difficult no matter how important they are. Explainable Artificial Intelligence is a young area that aims to produce high-quality suggestions and clear explanations of the outcomes while offering transparent learning techniques [4]. This research aims to develop a DNN-based movie recommendation system with explainable reasoning. The goals, background and motivation, issue description, suggested solution and thesis structure are all presented in this chapter.

1.2 Objectives

This project aims to design and develop a Deep neural Network (DNN) based movie recommendation application with explainable reasoning. Therefore, the following objectives are defined.

- A Critical review of the literature related to the DNN-based recommendation systems with explainable reasoning and defined the research problem.
- In-depth study of techniques used in deep neural network-based recommendation systems with explainable reasoning.
- Design and develop a DNN-based movie recommendation system with explainable reasoning.
- Evaluate the proposed solution for its effectiveness.

1.3 Background and Motivation

The field of "explainable artificial intelligence" focuses on creating intelligent systems that can communicate and reason out their results to people in an acceptable manner. Traditional recommender systems use several techniques and some of them are based on artificial intelligence concepts. More recently, recommender systems have benefited from the use of various AI approaches, which have improved user happiness and the overall user experience. Recommendations made from AI technologies are of a better quality than those from conventional methods. This has ushered in a new era of the recommender domain, revealing more intricate links between people and goods, offering increasingly sophisticated data visualizations, and unearthing vast amounts of information in contextual, virtual, demographic, and textural data. This thesis will address the most current and innovative theoretical and practical contributions to the subject. The use of fuzzy methods, active learning, transfer learning, deep learning, and various neural networks, computer vision, image processing, natural language

processing, reinforcement learning, and genetic algorithms are just a few of the advances artificial intelligence has brought to recommendation systems.

The recommender system is designed to facilitate and improve the social process of adopting recommendations from others when one has limited personal knowledge or expertise with the possibilities [8]. Many methods for developing recommendation systems have emerged recently. In this thesis, various recommender systems are examined taking into account their benefits and drawbacks: content-based, collaborative filtering, knowledge-based, deep neural network-based, and hybrid models.

1.4 Problem in Brief

Artificial intelligence-based recommendation systems are now widely used in industry. 60% of YouTube videos are watched by users based on recommendation [6]. However, many of these recommendation systems have a common weakness in the ability to explain recommended results. This leads to trust issues about the system among users. Therefore, this research is focused on developing techniques to provide better recommendations with explanations based on machine learning approaches.

1.5 DNN based movie recommender with explainable reasoning

These independent processes work together to produce an outcome of the system. In the beginning system asked for user inputs from the user via a graphical interface. This information will be gathered and stored in the system. Once adequate data is gathered it processes this data and creates its imaginary ratings and well as induced data sets to perform recommendations. Explicit user ratings and system-generated ratings will be used for the prediction. Once this is done, the system is ready to make recommendations. When a user makes a query regarding a movie, the system now generates a recommendation item list using a machine learning-based technique. A deep neural network is used to do this prediction and it is more flexible for the user inputs. While doing the prediction, the system considers product and user embeddings context embeddings, and additional product features separately and merges results to

provide the final recommendation list. To provide better recommendations user profile information and product information are critical. If they are not accurate, cannot expect quality accurate recommendations from the system.

1.6 Thesis Structure

There are eight chapters in this thesis. An introduction and summary of the research are provided in Chapter 1. Evolutions and challenges in the recommender systems are discussed in Chapter 02. Technology advancements in recommender systems are discussed in Chapter 03. An approach to an explainable recommender system is discussed in Chapter 04. Design details are presented in chapter 05. Implementation of the proposed recommender system is discussed in Chapter 06. An evaluation of the proposed solution is presented in Chapter 07. Further improvements Conclusion; highlights the research findings and challenges are discussed in chapter 08.

1.7 Summary

In this Chapter, have discussed an overview of the thesis with its aims and objectives of it. The chapter also presented the structure of the thesis, which consists of several chapters. Gives a brief background about the research problem. In the next chapter, chapter 2 is going to immerse deep into the Evolutions and challenges in the Recommender systems.

CHAPTER 2

EVOLUTIONS AND CHALLENGES IN THE RECOMMENDER SYSTEMS

2.1 Introduction

Chapter 1, presented the overall picture of this thesis covering the research problem and the essentials of the solution. This chapter gives a comprehensive literature review in the area of Recommender Systems (RS). The literature review has been structured to cover the basis of recommender systems, major developments, and future directions. Here it summarizes the research challenges in RS and defines our research problem.

2.2 Basis of recommender systems

A recommender system is characterized as a method for helping consumers make decisions in intricate information settings [9]. Recommender systems offer tailored, unique content and service suggestions to users, addressing the issue of information overload that they often face.

Recommendation generation is a process. Three main phases of this process can be identified. The information-gathering phase is the initial stage. In order to create a user profile or model, this gathers important user data. This profile is later used to do recommendation prediction. Important data obtained by this procedure includes the characteristics of the user, their actions, and the content of the sites they have accessed. Recommendation systems cannot function accurately until user profiles have been constructed. Recommender systems rely on several forms of input, including improved user reviews. Different forms of feedback are evident, including hybrid, implicit, and explicit feedback. Users can rate the system using an interface provided by the system when providing explicit feedback. But, in implicit feedback, the system automatically rates without user intervention. Combining these two methods can have hybrid feedback as well [6]. The learning step comes next. Using the input obtained during

the information-gathering phase, it filters and utilizes the user's features in this phase by using learning algorithms.

The final phase is the recommendation or prediction phase. It suggests or forecasts the kinds of products that the user would find appealing. There are many techniques used in this phase and let's look at some of them in this chapter. [6].

2.3 Major developments in recommender systems

The development of explainable recommender systems has focused on providing transparent and interpretable recommendations, helping users understand the reasons behind the recommendations they receive. Traditional recommender systems use several techniques and some of them are based on artificial intelligence concepts This section explores several recommender systems, taking into account their benefits and drawbacks: content-based, collaborative filtering, knowledge-based, deep neural network-based, and hybrid models.

2.3.1 Recommendations based on content of items

Content-based recommendation systems operate by evaluating an item's content and making predictions about its utility based on the user's past profile activities. As an example, if the item represents a movie, the system will compare various information about the movie with the user profile to predict its utility [2]. A filtering system that compares the qualities of the items to the user profile is called a content-based recommender system. This kind of system is learning about the user’s preferences through profile matching and dynamically updating the user profile as well. If the user is like to certain properties it will remember that information when making a recommendation in the future. Another notable point about this method is usually recommends similar items that are previously interesting to the user. The system is capable of extracting item properties from documents or descriptions as structured data or unstructured data. Hence, item representation plays a key role in the performance of the system [3].

The content-based recommendation has several advantages and disadvantages as well. Because the content-based suggestion is predicated on the product representation, it is independent of the user. Hence, this kind of system isn't affected by data sparsity problems. The content-based recommender systems can recommend new items to users without system cold-start [3]. A further significant benefit is that it can offer a concise justification for the suggested outcomes. Transparency is a great benefit, but to achieve this user profile information is seriously affected. Inaccurate or missing user profile data has a significant negative influence on the recommendation system's accuracy. The fact that the feature representation of the products is somewhat hand-engineered is another drawback. The accuracy of the recommender system greatly depends on the hand-engineered characteristics, and it demands a great deal of subject expertise. Also, this model has limited ability to expand user's existing interests. Content-based recommendation systems always recommend similar items to users. This may lead to overspecialization in recommendation. Content-based filtering is used in several famous recommendation systems like News Dude, LIBRA, and Pandora [5].

2.3.2 Collaborative filtering-based recommender systems

Collaborative filtering-based recommendation systems use users' personal, historical records to predict the utility of a product based on other users' ratings. Hence, this method can address some of the limitations present in Content-based recommender systems. To provide suggestions, collaborative filtering compares users and objects at the same time [3]. The embeddings can be learned automatically in this method and don't depend on hand-engineered features. The basic assumption under this method is that users with similar interests will consume similar items. Collaborative filtering techniques highly rely on information provided by users who have similar interests [1]. There are two main types of Collaborative filtering techniques. The first generation of collaborative filtering is memory-based and evaluates user and item correlations using heuristic methods [17]. Model-based Collaborative filtering builds a model for prediction based on machine learning or data mining techniques. In this method, additional information such as tags, locations,

reviews, etc. used other than the rating matrix. This model is a good selection for recommendation systems if these additional data can be present with items. Matrix factorization is used in Model-based collaborative filtering recommendation techniques [15]. Which is useful to reduce the dimensions of the user-item rating matrix significantly [3].

Recommendation systems based on collaborative filtering have numerous noteworthy benefits. One of the biggest advantages is that there is no requirement for specific domain knowledge. Because collaborative filtering-based recommendation is based on product rating scores [20]. As a result, knowledge about the product is not necessary. Another advantage is users may receive surprising recommendations that may be desirable for them. This surprise factor plays a key role in the trust of users in the recommendation system [6]. Despite collaborative filtering having overcome several issues faced by content-based filtering recommendations, still there are some drawbacks. One of the biggest issues with collaborative filtering-based recommendation is the cold start problem [7]. The rating of a newly added individual or item in the system is uncertain since there isn't enough data available to make a forecast. Sparsity is another well-known drawback of this method. When a user evaluates very few items in the system, it leads to a sparse matrix of missing rate values and an inability to locate successful neighbors [1]. Another notable drawback is the gray sheep problem. Users with unusually diverse preferences might not find many other users who match them. It may be very difficult to make recommendations to such a user. The collaborative filtering-based recommendation is used in several famous recommendation systems like Ringo, GroupLens, and Amazon.com [6].

2.3.3 Systems with knowledge-based recommendations

In knowledge-based recommendation systems, recommendations are based on pre-existing information or rules about user preferences and item features. Knowledge-based recommender systems maintain their knowledge base, in contrast to collaborative filtering-based and content-based recommendation strategies [3]. The deployment of knowledge-based recommender systems may be divided into many crucial phases. The knowledge representation is the most important one. This

knowledge base is constructed by analyzing user's history records, previous problems, constraints, corresponding solutions, and user reviews and feedback. Knowledge-based recommender systems are especially helpful in fields like education, e-commerce, and expert systems when explicit knowledge about the products is accessible. The system builds a profile for each user, capturing their preferences, constraints, and requirements at the user profiling.

Knowledge in the knowledge base and user profiles are referred to by the system when a new recommendation problem is encountered. To find similarities between items and users, it is required more structured and organized representation. This indicates that highly specialized subject knowledge is needed for knowledge-based recommendation systems, and keeping a knowledge base is always expensive. However, the cold start problem associated with other techniques is not present for knowledge-based recommendation. Another advantage of this method is users can impose constraints on recommendation results [13]. Also, may struggle to capture complex user preferences or recommend items outside the scope of the available knowledge.

2.3.4 Deep Neural Networks based recommendation systems

A deep neural network (DNN)- based recommender system is a type of recommendation system that leverages deep learning techniques to model the complex relationships between users and items. Neural networks are rarely used in recommendation systems because the recommendation task requires ranking rather than classification, despite the fact that deep learning techniques have shown great success in many fields, including computer vision, natural language processing, speech recognition, and classifications [3]. Unlike traditional collaborative filtering or content-based approaches, DNN-based systems can automatically learn intricate patterns and representations from large amounts of data, allowing for more accurate and personalized recommendations. With technological advancement, more data is now available, and more user-generated comments, tags, reviews, photos, and videos are present for any items. Since the recommendation of music items, movies, and videos are not very successful with previously mentioned methods, now deep

learning techniques are getting attention. In particular, the recommendation of multimedia items such as images, videos, and music items using deep neural networks gives promising results [1].

Different types of neural networks can be used for the implementation of recommendation systems such as multi-layer perception, convolution neural networks, recurrent neural networks, and Graph neural networks [3]. In explainable recommendation systems are now widely used neural networks [21]. Recommended results are supported with some artifact results which helps the user to understand why results were recommended. These artifacts might be user reviews or comments. Hence, user interaction and feedback are essential when using this method. However, DNN-based recommender systems also come with challenges, such as the need for large amounts of data, computational resources, and expertise in deep learning techniques for training and tuning complex models.

2.3.5 Hybrid recommendation systems

The Hybrid recommendation is another method used in recommendation systems that is focused on overcoming limitations present in the methods above mentioned. Hybrid recommendation systems seek to address the drawbacks and biases of single methods by combining the best features of many methodologies. It integrates two or more disparate recommendation methods [1]. The most often used hybrid approach is the mix of content-based and collaborative filtering. Hybrid suggestion seeks to address the drawbacks of earlier methods. There are several approaches to hybridization. The basic idea behind hybridization is that a mixture of algorithms will yield recommendations that are more accurate and practical than those of a single algorithm by making up for the inadequacies of that algorithm [6].

There are several ways to combine recommendation techniques in hybrid systems. It can use weighted averages to combine different algorithms or techniques. Another possible way is to combine features extracted from different algorithms or techniques into a single space and the training recommender system with it. The output

of one recommender algorithm could provide input to another recommender technique inside the same system in a cascade manner.

The Hybrid recommendation system has overcome obstacles such as sparsity problems and cold start concerns by incorporating elements from other recommendation systems or depending on their logic. There are several ways of combining approaches to achieve hybridization. By combining the output of many recommendation systems, weighted hybridization creates a recommendation list and uses a weighted score computation to provide predictions. Based on user reactions weights will be adjusted automatically. In Switch hybridization, the system will swap recommendation techniques based on the user's heuristic reflection [6]. This method is useful when evidence is not solid for recommendation results. Cascade hybridization creates an order of preferences between several products through an ongoing refining process. One recommendation strategy's result is iteratively improved upon by another recommendation method. Netflix and Cinmatch are popular recommendation systems that are built based on hybrid techniques [14].

2.4 Future trends of the recommender system

In this section, discuss the future direction and challenges of the recommender systems with the following sub-topics. Reinforcement learning technology usage in recommender systems, concept drift, and privacy-preserving and security-related concerns.

2.4.1 Reinforcement learning based recommender systems

Using a recommender system involves interacting with it through a succession of states and actions, all of which are determined by reinforcement learning. In the realm of recommender systems, reinforcement learning (RL) has drawn growing attention as a means of enhancing sequential decision-making procedures and raising user happiness and involvement. RL techniques enable recommender systems to learn optimal decision policies by interacting with users and receiving feedback over time. Unlike traditional recommender systems, which frequently focus on predicting users' preferences at a certain time point, reinforcement learning model-based recommender

systems aim to maximize user involvement and satisfaction over the long run. With the advancement of deep deterministic policy gradient (DDPG) and deep Q-network, which offer the benefit of concurrently addressing short- and long-term incentives, deep reinforcement learning has recently attracted more interest [10].

2.4.2 Understanding and responding to concept drift in recommender systems

Users' past records are weighted equally in traditional recommender systems because they operate under the assumption that user preferences would remain mostly constant over time. Nonetheless, user preferences shift over time due to the steady development of unique preferences, life experiences, and effects influenced by popularity. This effect, known as notion drift, is frequently observed in streams of Big Data [11]. Older records that a user accumulates over time may not match the user's most recent requests. Prediction accuracy is indiscriminately compromised by using all available data, and recommender systems that ignore this risk see a decline in performance. To tackle this problem, time-aware recommender systems were created [12].

2.4.3 Privacy-preserving and security concerns

Users are becoming increasingly concerned about their privacy as recommender systems are used more and more in a variety of application domains. On the other side, this makes consumers hesitant to provide the system with genuine information and preferences, which negatively affects the recommender system's effectiveness [3]. Also, most recommender systems are acting as black boxes to users, and hence their trust is something systems should earn. To ensure that recommender systems receive accurate user feedback without reluctance, privacy and security issues should be addressed. Leaking private data to outside parties has more severe effects on users and one incident may be enough to destroy the credibility of the system.

The current generation of recommendation systems should improve to provide better explanations and high-quality results. Concept drift detection and reactions accordingly, usage of reinforcement learning like technologies in recommender systems, imbalanced data handling, privacy-preserving, and secured systems are key challenges to address in the near future.

2.5 Problem definition

Artificial intelligence-based recommendation systems are now widely used in industry. However, many of these recommendation systems have a common weakness in the ability to explain recommended results. This leads to trust issues about the system among users. Therefore, here it is focused on developing techniques to provide better recommendations with explanations based on machine learning approaches.

2.6 Summary

This chapter presented our literature in the area of RS by covering many research papers with very high citation records. Here it summarizes the challenges in the field and also defines the research problem as explainability in recommender systems remains unsolved. This chapter also identified machine learning as a potential technology for recommendation systems. However, it should assist with other techniques to address explainability issues in RS. Chapter 3 presents our in-depth study of ML techniques by highlighting their reliance on explanability in RS.

CHAPTER 3

TECHNOLOGY ADOPTED

3.1 Introduction

This chapter describes the technologies, tools, and frameworks that have been used to develop a DNN-based movie recommendation system called DreX Movie Buddy. Especially looks into programming languages, Frameworks, tool kits, NLP libraries, and other development tools that have been selected for this research study. The effectiveness, ease of usage, and capabilities of each technology are discussed in a brief overview of the next sections.

3.2 Programming Laguages

In this research project, used several programming languages depending on our requirements and best fit. Python programming language used for back-end implementation and model development. JavaScript is used for web-based front-end development.

3.2.1 Python

Python is a well-organized high-level programming language that is very versatile and simple to learn. No compilation is required since it is an interpreted programming language. Since Python is very supportive and rich with library facilities, Python is used as the main language of the program throughout the whole solution. Python's simplicity, versatility, and large ecosystem make it a popular choice for this kind of research project. Many guidelines are available, ease of learning curve, ease of using data structures and speed of interpretations are a few more reasons to select Python as the programming language. Combined with its powerful features and libraries, makes it an excellent tool for building everything from simple scripts to complex applications. The performance of execution and debugging facilities are also considered when selecting Python as the programming language.

3.2.2 JavaScript

JavaScript is a very lightweight interpreted programming language to develop web application frontends. Developing front ends with JavaScript is efficient and time-saving. JavaScript is used to develop the DReX Movie Buddy front end. It has very rich capabilities for web applications and browsers.

3.3 Libraries

Several machine learning and data processing libraries and tools are used to develop our recommender system called DReX Movie Buddy.

3.3.1 NumPy Library

One of the core libraries in the python which allows users to manipulate various data structures. It has rich capabilities to support multi-dimensional arrays and matrices. NumPy offers a comprehensive mathematical functional framework for users to work easily.

3.3.2 Pandas Library

Python provides the foundation for the open-source Pandas data analysis and manipulation package. It provides operations and data structures in a very adaptable way. It is a very powerful library that is always used in the data science and machine learning world.

3.3.3 TensorFlow Library

The Google Brain team created the open-source software library TensorFlow for artificial intelligence and machine learning. Although it may be used for many other tasks, its main focus is on deep neural network training and inference.

3.3.4 Flask Fraework

Flask is a micro web framework written in Python and is suitable for rapid development and prototyping. Easy to use and has no database associated with it. Flask depends on the Jinja template engine, WSGI tool kit, and Click CLI tool kit.

3.4 Computing hardware

Development of the models and testing was performed on the Google Cloud platform, especially on Google Colab. No specific hardware requirement is present for this research work. All required modules and libraries were installed as required in these environments.

3.4.1 Google Colab

Our system's models were developed using the Google Cloud platform. Google Colab is an online Jupyter Notebook environment that provides free Google Cloud platform access to users. It empowers users to train and model machine learning and artificial intelligence models and systems rapidly. Collab is used as an online development tool with rich CPU and GPU support. Sharable files and storage using Google Drive is an added benefit. Debugging is very easy in Colab. So it is very good for rapid prototyping. Also, it came up with various pre-installed Python libraries. New module installing also very easy.

3.4.2 Visual Code

Very rich supportive IDE and used as an offline development and editing tool. This is used as the source code editor for the entire lifespan of the research. Various plugins and supportive libs are there for VSCode.

3.5 Data management and pre processing

Pandas and NumPy Python libraries were mainly used for data preprocessing of the developing DReX movie buddy recommendation system. Also, Python inbuilt data structures and algorithms are used for data manipulation and data management in this project. Lists. Dictionaries and tuples are some of them. These data structures provide efficient data handling throughout the study period of the research work in an organized manner.

The user's historical data is stored in a SQLite database and data management and query handling are performed with SQLAlchemy. These data can be extracted to data files in a normalized way such that user personal data is not compromised. Extractable data does not have any sensitive or private data of the user.

3.6 Techniques for recommendation generation

When users are interested in movies, it is better to have recommendations based on their interests and tastes. In our research, a system is built that is capable of providing recommendations based on user input. Various techniques are combined with a deep neural network used to make the recommendation.

A deep neural network is used to make recommendations as the main technique. The model training and evaluation were done with the data set. Python library called "surprise" is used to make the recommendations under different algorithms to compare results. The Surprise library package gives a rich framework to develop recommender systems with greater control than traditional algorithms.

3.7 Techniques for explainable reasoning

The Natural language processing toolkit (NLTK) of Python is used for explanation generation. There are several things to consider in the explanation generation module. One important thing is getting the sentimental meaning of the reviews. It is also required to identify them as a good review or bad review. Another thing is to generate explanations for predicted items. NLTK consists of a series of libraries, which provide all kinds of language processing features like classifications, tagging, tokenization, semantic reasoning, and parsing.

3.8 User Interface

The User Interface of the DReX Movie Buddy recommendation system was developed using Python Flask, Bootstrap, and JavaScript. Flask is a lightweight web framework featuring RESTFUL APIs. Which enables back-end and front-end communication of the system to be more comprehensive. Bootstrap was used to create responsive web

user interfaces to interact with the system. JavaScripts are used to enable utility functionalities of the UI and HTML and CSS does the styling.

3.9 Summary

In this chapter, it is discussed the various technologies used in this research study and the development of the DReX Movie Buddy recommendation system. The details about the Python and JavaScript programming languages and why selected for this project is also presented. Then discussed the libraries and frameworks used for the implementation. Pandas, NumPy, Flask, and TensorFlow library details were presented. The hardware infrastructure and its usage were discussed in detail as well. Then techniques used for recommendation generation and techniques used for explanation generation are presented.

The next chapter presents our approach for the DReX movie buddy recommender system in detail.

CHAPTER 4

AN APPROACH TO MOVIE RECOMMENDER SYSTEMS

4.1 Introduction

Chapter 3 presents technology adapted for developing recommendation solutions with explainable reasoning. This chapter presents a Deep Neural network-based recommendation system approach with explainable reasoning. Named **DReX**Movie Buddy, an acronym for **D**NN-based **Re**commendation with E**x**plainable Reasoning for Movies. This chapter presents the DReX Movie Buddy approach by covering our hypothesis, input, output, process, features, and users. Among others, the process specifically identifies the task within DReX Movie Buddy which provides the insight into design of the solution.

4.2 Hypothesis

Inspired by the practical usage of recommendation systems, artificial intelligence concepts are giving promising results. However, many of these recommender systems suffer from one common weakness which is explainability. A deep neural network model is hypothesized for a movie recommendation system with explainable reasoning to improve efficiency, transparency, user satisfaction, and user loyalty. The system was tested through a series of experiments and algorithms described in subsequent chapters.

4.3 Input

DReX Movie Buddy Recommendation system takes user queries about movies as input. In addition to that, explicit feedback on the rating of movies is also taken as input to the system. The system. Explicit feedbacks contain ratings of the movies and the system will use them to generate implicit feedback to the system.

Once user profiles are constructed system takes user queries as input to make recommendations.

4.4 Output

The proposed system generates two sets of outputs for a given user query. First, it provides possible recommendation movies for the user with a weighted rating score. It may appear as a list of movies and the DReX movie buddy frontend displays them with original movie posters. In the second set, provide explainable reasons why this item is recommended to the user. This explainable reasoning also consists of several top reviews about the movie by similar users. Explainable reasoning is also present via the graphical user interface of the system. In addition to that, main actors and characters are also displayed if the user needs to get more information about them.

4.5 Process

Motivated by existing explainable recommendation models, this methodology seeks to surpass their constraints and enhance the clarification of suggested outcomes. The DReX Movie Buddy recommendation system has several key steps. This method proposes to use two separate independent processes for recommendation and explanation. Once the recommendation generation process has finished and the outputs have been processed separately, by the explanation generation phase begins. Recommendations are based on a deep neural network.

First user has to log in to the DReX Movie Buddy system using the graphical user interface. User authentication is performed using Auth0 standard API and users are allowed to log in using their social media accounts such as Google and Facebook. Once the user logs into the system first time it gives a chance to rate a series of movies. These ratings are saved in the user historical database. User-rated data sets have been collected and processed for the system. It grows with the system.

The data preparation step is aimed at preprocessing data and identifying qualified datasets for the model training. At this step, some linguistic data are combined to get better meaning. Once data is ready model is trained and saved as pickle object files to load into our DReX Movie Buddy system. In the same way, an explanation generation

model is prepared as well. Once adequate data preparation is done, the system creates its weighted average ratings on data sets to perform recommendations. Explicit user ratings and system-generate ratings will be used for the prediction. Once this phase is done system is ready to make recommendations.

When a user makes a query regarding a movie, the system now generates a recommendation movie list using a DNN-based technique. A Deep neural network is used to do this prediction and it is more flexible for the user inputs. While doing the prediction system considers movie and user embeddings context embeddings, and additional movie features separately and combines results to provide the final recommendation list. To provide better recommendations user profile information and product information are critical. If they are not accurate, the user cannot expect quality accurate recommendations from the system.

4.6 Features

DReX Movie buddy recommender system approach has several key features to highlight. First, it is very personalized. User profiles are available in the system. Never expose user preferences to any other. Users can log in using their social media accounts such as Google and Facebook at the sign-up activity.

It gives freedom to the user to rate movies according to their perception rather than depending on third-party ratings. It not only provides significant recommendation movies but also provides explainable reasonings for each item. You can get more details about recommended items by using show me option in the GUI. Other than the basic details of the movie, the system displays the actors and main characters of the movie as well. It helps users to get more details about the movie and its background. Explanations are generated based on information available and some reviews supporting the movie display. Reviews are categorized and only positive reviews are given as output. This approach of recommendation generation and explanation generation is implemented independently and reasoning may describe why it was selected in a user-friendly manner. The DReX system concept can be expanded to support different types of products and training is essential.

4.7 Users

Anyone who wants to use a recommender system to assist in finding the best movies for them can use this system. The system guarantees their personal details are not exposed to the outside world and are only used for recommendations. Researchers may also use this system for further development.

4.8 Summary

Our approach to the DReX Movie Buddy system was discussed in this chapter. Inputs, outputs, and processes of the system are defined. More importantly, major tasks have been identified, like recommendations generation, movie rating, review rating, and explanations generation within the system. The next chapter describes the design of the DReX movie buddy system with reference to the process identified here.

CHAPTER 5

DESIGN

5.1 Introduction

In the previous chapter, the approach for this research is described. This chapter extensively dives into the design of the research implementation using the discussed technologies and the approach.

5.2 Architecture overview of DReX Movie Buddy system

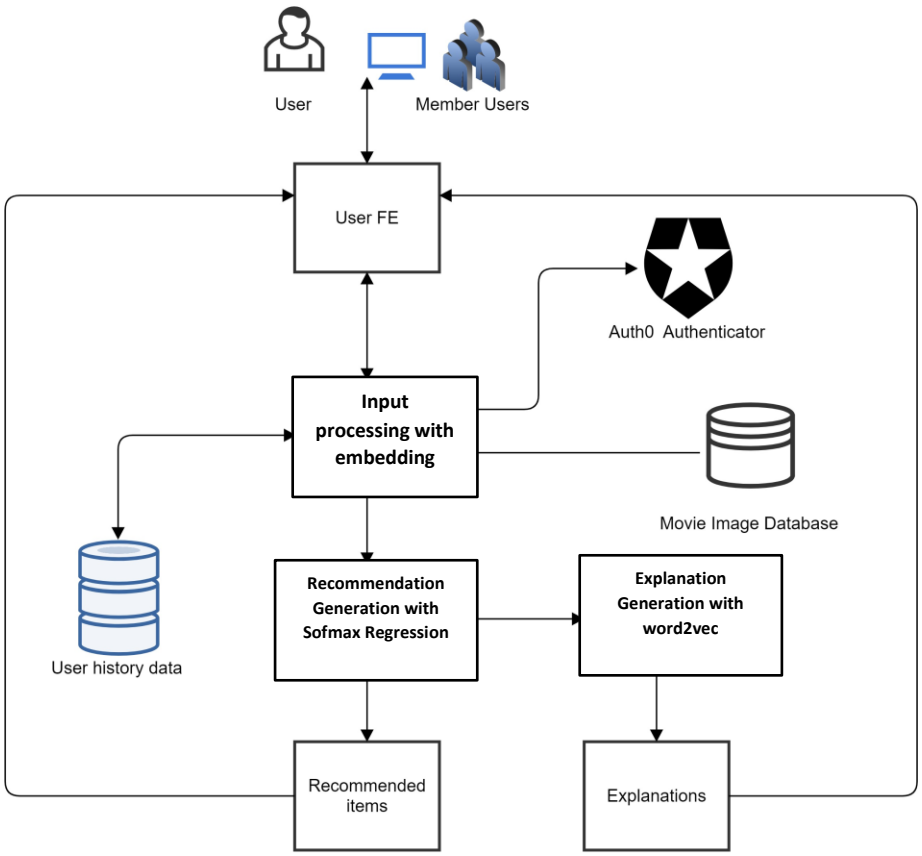

Figure 5.1: DRex Movie Buddy system architecture overview

As shown in the architecture diagram above DRex Movie Buddy system has several key modules named user frontend, Input processing module, recommendation generation module, and explanation generation module. The user frontend is the landing page of the system and it allows users to log into the system. Also, it displays all outputs as well. The input processing unit handles three major sub-tasks authentication connection handling, Movie images/posters interface handling, and user rating data handling with database. The recommendation generation module generates a recommended movie list for a given user input. The explanation generation module generates reasonings for the recommended results.

5.3 Graphical User interface

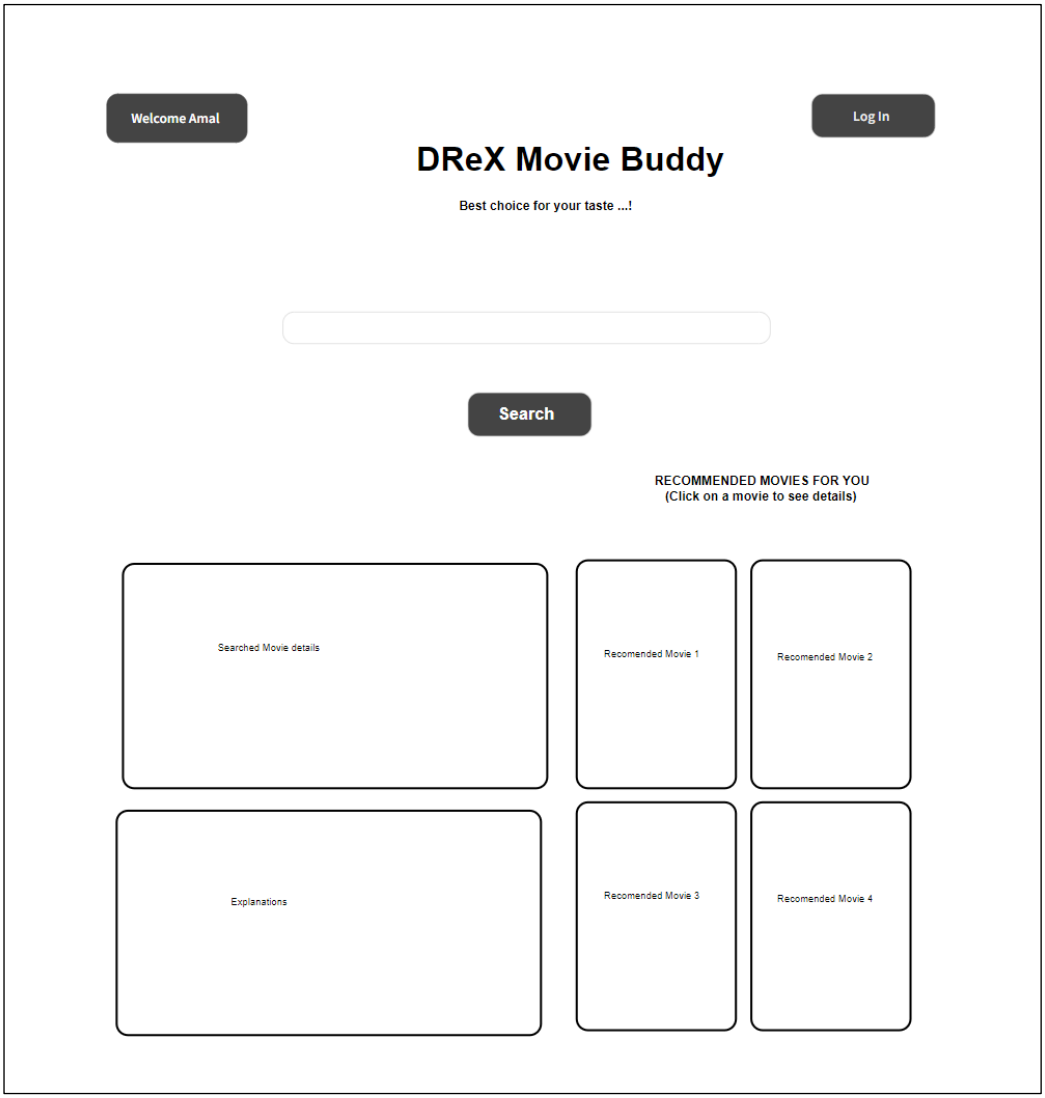

Figure 5.2: DRex Movie Buddy GUI wire frame design

The landing page of the DReX Movie Buddy system gives a login option and search option for guest users. Guest user details are not tracked and do not have personalized recommendations for them. But they could get an idea about the system before signing up. When a user first time sign-up they are directed to the movie rating page where they can specify their personal rating about the default set of movies.

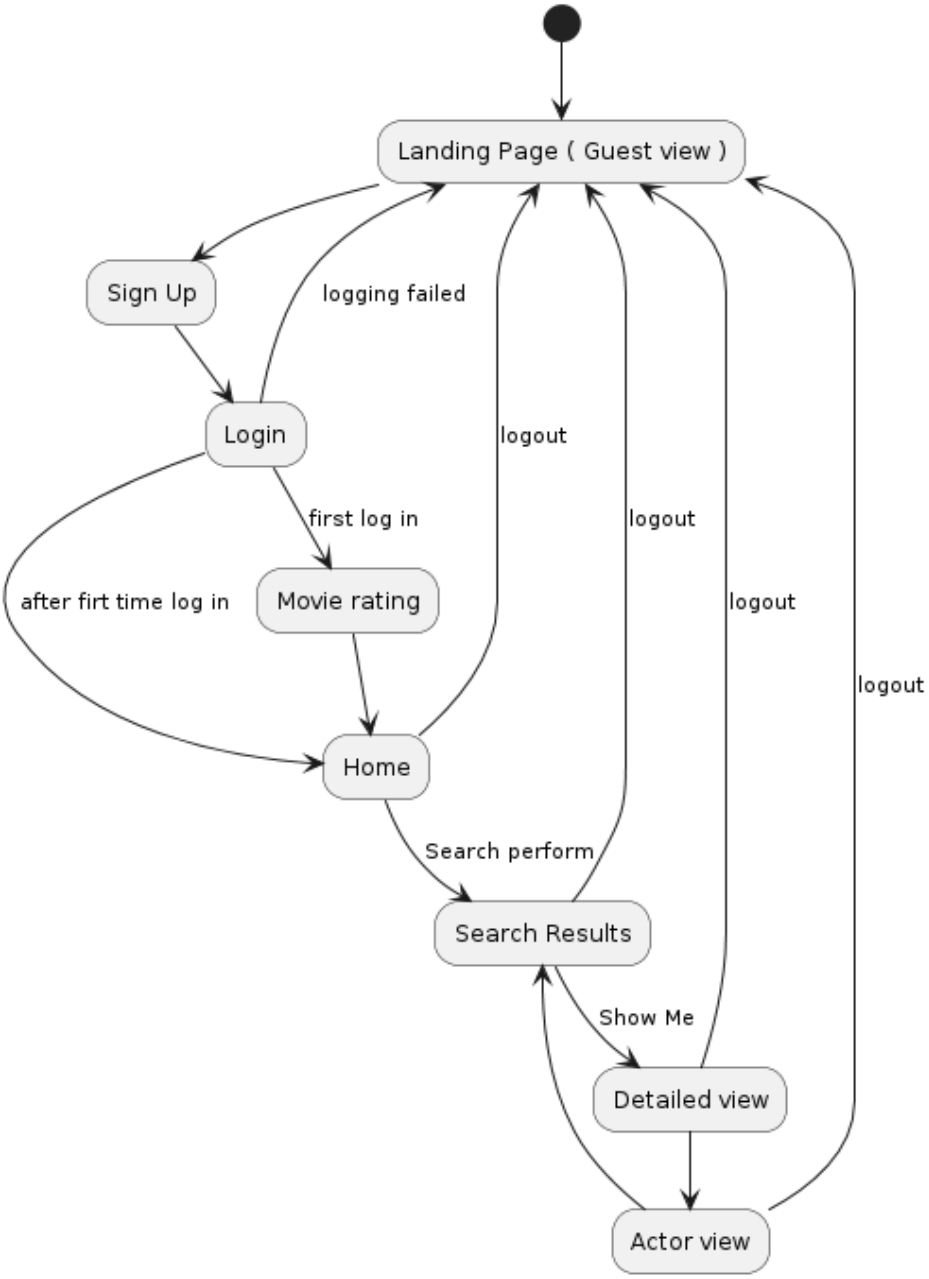

Figure 5.3: DRex Movie Buddy GUI activity diagram

5.4 Input processing module

Input processing of the DReX Movie Buddy system performs three major tasks for the system. It handles the connection with the Auth0 authentication server. When a user logs in or signup it will connect to the Auth0 authentication server through REST API and get the access token. Instead of user name and password users can use their social media accounts for the sign-up as it is facilitated by Auth0 standard API.

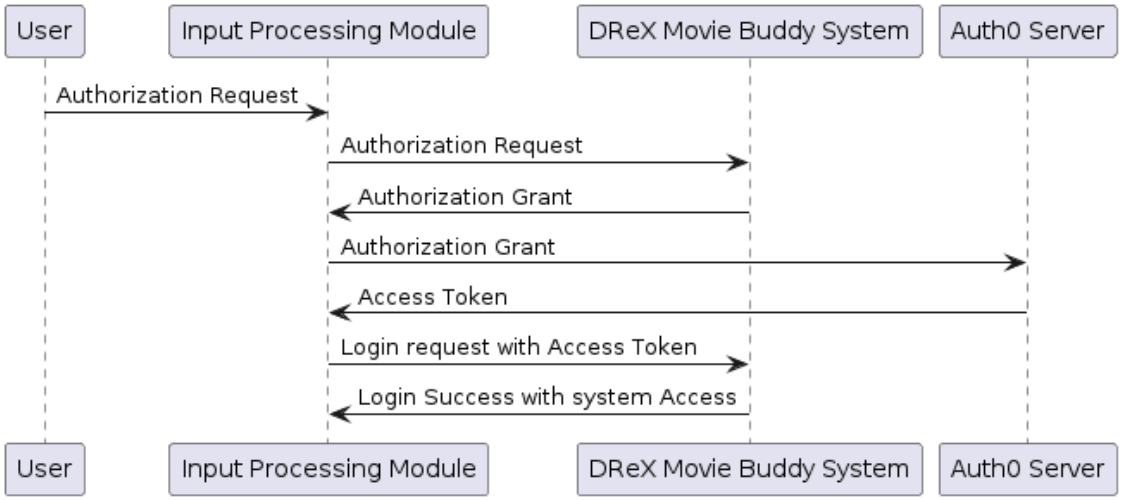

Figure 5.4: Authentication work flow

The second important task performed by the input processing module is handling movie image- database server connection. It is essential to show the original movie posters to the users when they are interested in them. Also casting members and movies have some branded posters that more likely bind with the user's mind. In this system, the system displays original movie posters as in other professional applications. To get movie posters and Cast member posters the system uses standard movie poster database API.

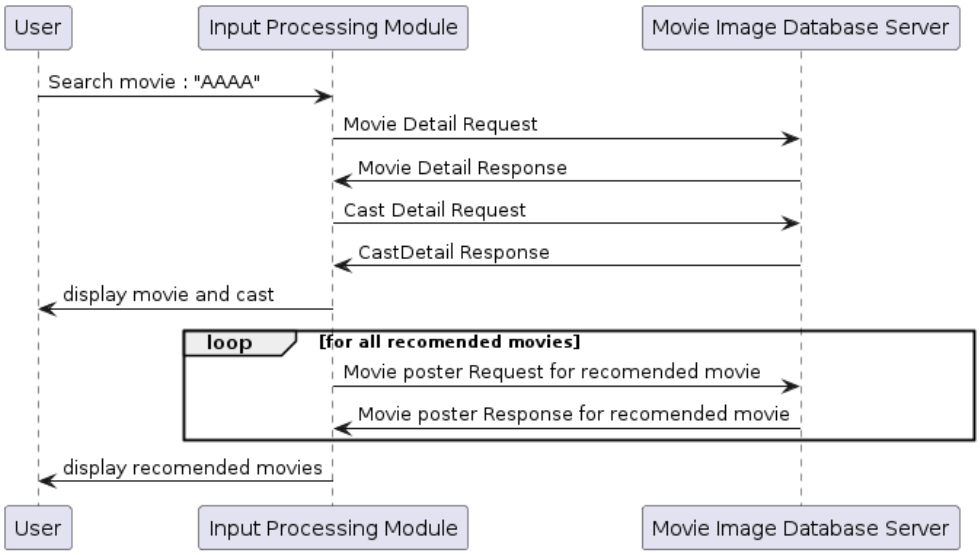

Figure 5.5: Movie Poster image Database work flow

The third major activity performed by the input processing module is handling the SQLite database connection to store user rating history. SQLite database was selected due to inbuilt support with Python. Also, it is very convenient to use without any database server requirements. When a user completes a rating, data is saved into the database with the support of SQLAlchemy ORM. SQLAlchemy ORM has rich database management activity support with Python.

5.5 Recommendation generation module

The recommendation generation process uses a Deep neural network (DNN) model. A Deep neural network (DNN) model provides better flexibility to incorporate features present in the movie record dataset. There for DNN model can merge with collaborative filtering and content-based recommendation models to have better results. Since the input layer of the network has the flexibility to accept more details movie features and user features can be captured and it helps to identify the specific interests of the user. Hence, the systems can improve the relevance of

recommendations. The recommendation may improve with context awareness of the rating. Also, more item features improve the taste of recommendation to the user. Therefore, separate inputs can be provided to the system for additional movie features and movie ratings. The results of each will merge to predict the final recommendation.

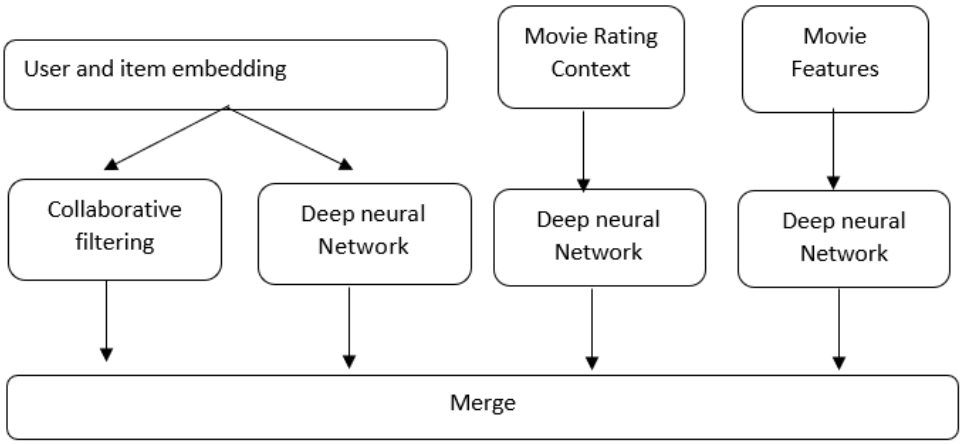

Figure 5.6: Recommendation generation module overview

Recommended results are generated using several well-known prediction algorithms and evaluated their performances through the system. The system has been designed such that it can load the recommendations generation module into the system separately. This enables us to compare results with different algorithms.

Users are watching movies sequentially and the system can have a list of movies a particular user has watched. Let's take this watch list as $S(m) = \{m_1, m_2, m_3, \cdots m_n\}$ Also, they rate watched movies, and a rating of the list is also, needed for the system. Rating of the watched movies take as $R(r) = \{r_1, r_2, r_3, \cdots r_n\}$. The system finds a probability that the user watching a targeted movie and prepares an output recommended movies list. Every input feature is embedded into a fixed-size, low-dimensional vector by the embedding layer [18]. User profile models with a user ID, occupation, age group, and sex. The movie is modeled with a set of genres, titles, movie ID, and plot keywords.

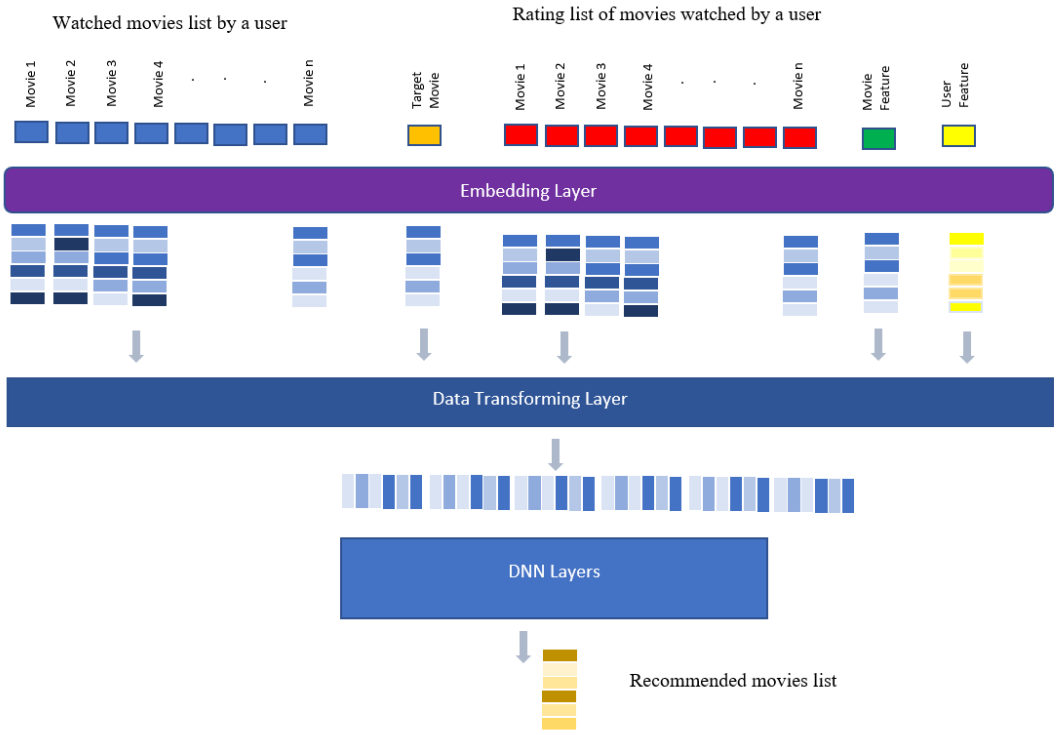

Figure 5.7: Recommendation generation process overview

The Attention score is computed at the transforming layer for inputs consisting of reduced dimensions. Input consists of query (Q) and keys (K) of dimension d_q and values (V) of dimension d_v. Attention function is defined as follows.

$$AF(Q,K,V) = SoftMax\left(\frac{Q.K^T}{\sqrt{d_q}}\right).V$$

Multi-head attention can be used with dot product attention score and it consists of multiple ads instead of a single layer.

$$MultiHead(Q,K,V) = Concat(H_1,H_2,H_3,\cdots H_h)W^0$$

$$H_i = AF(QW^Q, KW^K, VW^V)$$

To avoid overfitting the system uses Dropout and LeakyReLU and FFN (feed forward network). It helps to learn more meaningful features as well.

$$S^0 = LayerNorm\left(S + DropOut\left(MH(S)\right)\right)$$

$$F = LayerNorm\left(S^0 + DropOut\left(Leaky\,\mathrm{Re}\,LU(S^0W_1 + b_1)W_2 + b_2\right)\right)$$

$$Y = FFN(F)$$

Where W_1, W_2, b_1 and b_2 are learnable weighted parameters. LayerNorm is the standard normalization layer.

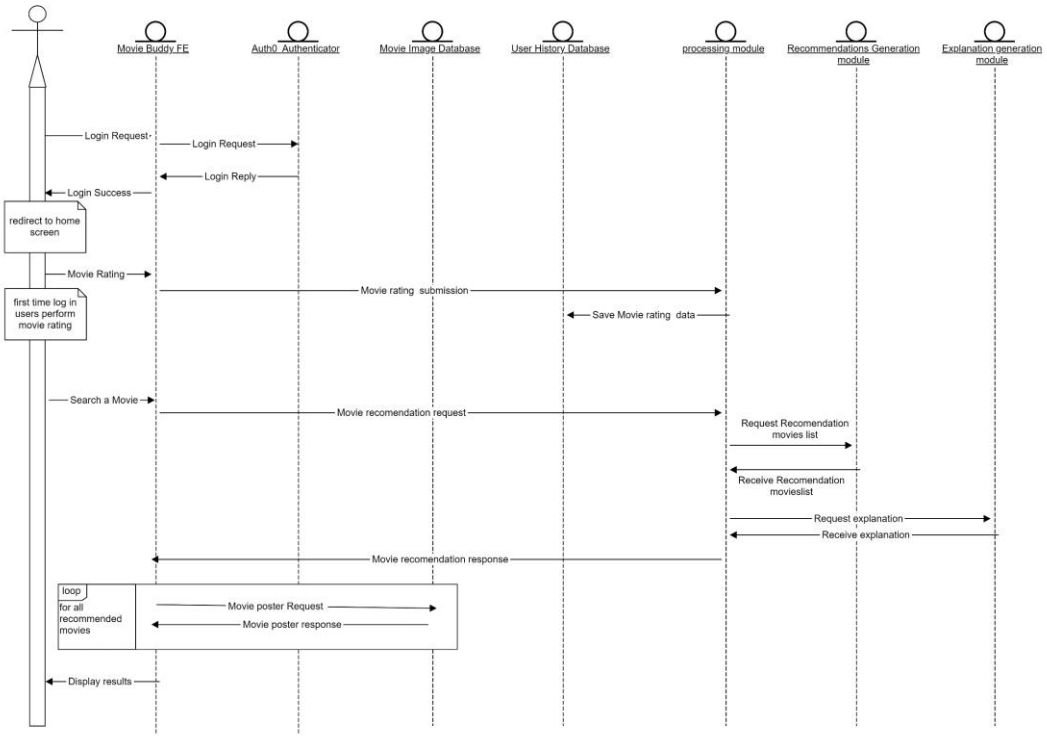

Figure 5.8: Recommendation generation work flow

5.6 Explanations generation module

Explanations generation is independent of recommendation and triggered after a complete recommendation. Several word embedding features of the movie data set are used to be embedded together. A word representation method called word embedding enables machine learning algorithms to comprehend words with comparable meanings. The word2vec model, which Google introduced, is a well-known method for creating word embeddings for improved word representation. Once trained, this two-layered neural network can identify words that are synonymous and recommend further words for sentences that are only partially formed. Natural Language Toolkit (NLTK) has built-in support for Word2Vec and Doc2Vec with the Gensim module. Based on this the system generates reasoning for recommended results.

In addition to that the system gives some good reviews from similar users provided for the movie. To show these reviews, the system needs to have a way to categorize them as good and bad reviews. A separate module is used for this and presents a set of positive reviews with reasoning and altogether considered as the explanation.

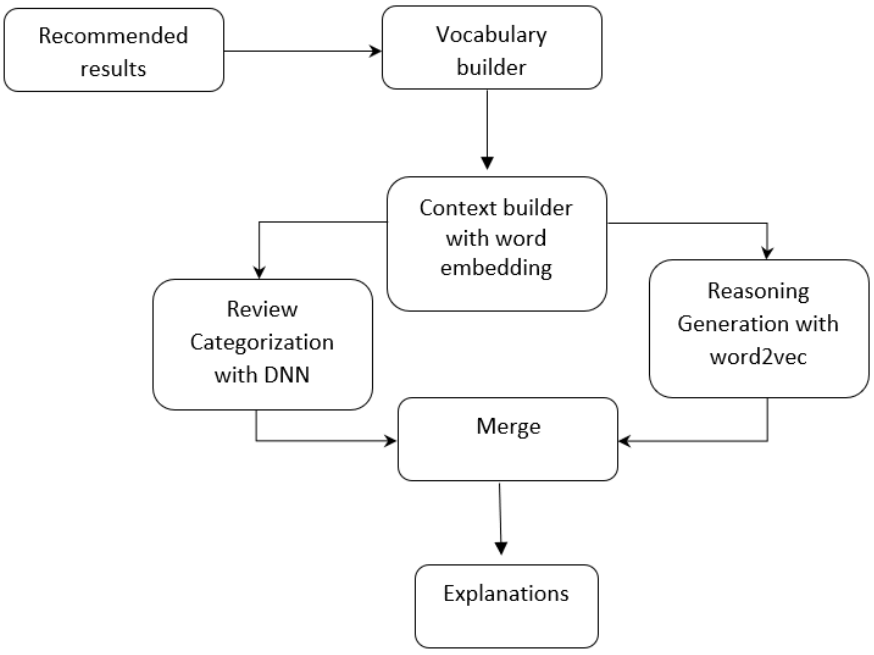

Figure 5.9: Explanation generation module overview

5.7 Summary

The design of the DReX movie Buddy system discussed in this chapter allows us to develop the system smoothly and efficiently way. The high-level modules of the system and their main tasks are defined as well. GUI wireframe design is also included and message flows and sequence diagrams are included to have better clarity. The next chapter describes the implementation of the DReX movie Buddy with reference to the design presented here.

CHAPTER 6

IMPLEMENTATION

6.1 Introduction

This chapter provides a detailed overview of the implementation of the DReX Movie Buddy system. This chapter focuses on the practical aspects of the development including data pre-processing, implementation of modules, testing of module functionalities, training of language models, and loading.

6.2 Data format and processing

The data selected in this research study is a famous movie lens data set. It has two versions. A small data set contains about 100000 ratings for 9000 movies from 600 users while a large data set contains 33 Million ratings for 86000 movies from 330000 users. Since processing power is limited, A small data set is selected and developed for the system. However, it is capable of running large data sets as well if required resources are available. In Addition to movie data, the system uses the NLTK brown data package and NLTK word2vec_sample data package for language processing.

There are some movie records which does not have a significant vote count and they also affect the rating if considered them. To prepare our movie chart it is needed to rate them with a rating value. A weighted average formula is used to calculate the weighted rating of movies. There are two major parameters in defining weighted ratings. The first one is the average of the rating across all records. The second one is the minimum vote count required to be included in the chart. Since the data set has vote_average and vote_count, can get these parameters from them to formulate a new weighted rating. The weighted rating formula can be defined as follows.

$$Ws = \left(\frac{v}{v+m}\right).R_i + \left(\frac{m}{v+m}\right).\mu_{all}$$

v is the number of preferences for the considered movie.

m is the minimum number of preferences which required to be in the rating chart. If votes are less than m it will be ignored and not take into the chart.

R_i is the average rating of the movie

μ_{all} is the average rating of all movies

W_s is the weighted score rating of the movie.

6.3 Implementation of Graphical User Interface

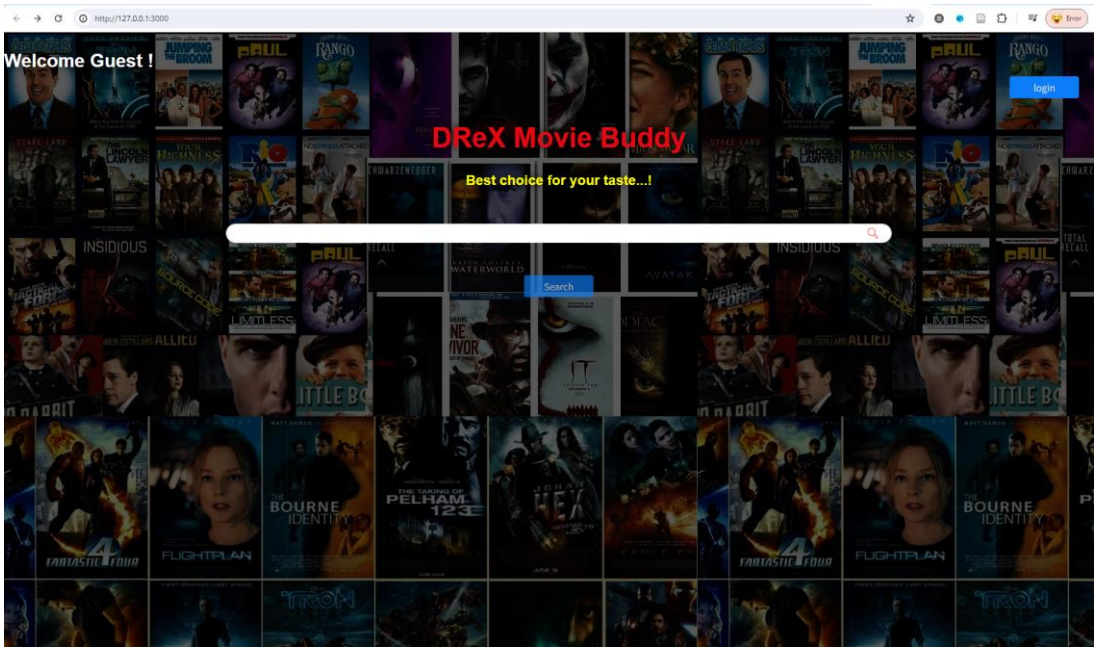

Figure 6.1: DReX Movie Buddy GUI – Landing page

Frontend of the DReX Movie Buddy system was developed using Bootstrap, JavaScript, HTML, CSS, etc. It is a web application and all outputs are presented using GUI.

```python
@app.route("/callback", methods=["GET", "POST"])
def callback():
    token = oauth.auth0.authorize_access_token()
    session["user"] = token
    myuser = oauth.auth0.userinfo()
    suggestions = get_auto_comlete_options()
    return render_template('home.html',suggestions=suggestions, user=myuser.nickname )

@app.route("/logout")
def logout():
    session.clear()
    return redirect(
        "https://" + env.get("AUTH0_DOMAIN")
        + "/v2/logout?"
        + urlencode(
            {
                "returnTo": url_for("home", _external=True),
                "client_id": env.get("AUTH0_CLIENT_ID"),
            },
            quote_via=quote_plus,
        )
    )

@app.route("/")
@app.route("/home",methods=["GET", "POST"])
def home():
    suggestions = get_auto_comlete_options()
    return render_template('home.html',suggestions=suggestions, user="")

@app.route("/rating",methods=["GET", "POST"])
def rating_profile():
    if request.method == 'POST':
        rec_movies = request.form['rec_movies']
        rec_posters = request.form['rec_posters']
        rec_movies = make_list_obj(rec_movies)
        rec_posters = make_list_obj(rec_posters)
        movie_cards = {rec_posters[i]: rec_movies[i] for i in range(len(rec_posters))}
        return render_template('rating.html', movie_cards=movie_cards)
    else:
        return render_template('rating_home.html')


@app.route("/saverating",methods=["POST"])
def saverating():
    rec_movies = request.form['rec_movies']
    rec_scores = request.form['rec_scores']
    rec_movies = make_list_obj(rec_movies)
    rec_scores = make_list_obj(rec_scores)
    return redirect(url_for('home'))
```

Figure 6.2: DReX Movie Buddy Back end code segment

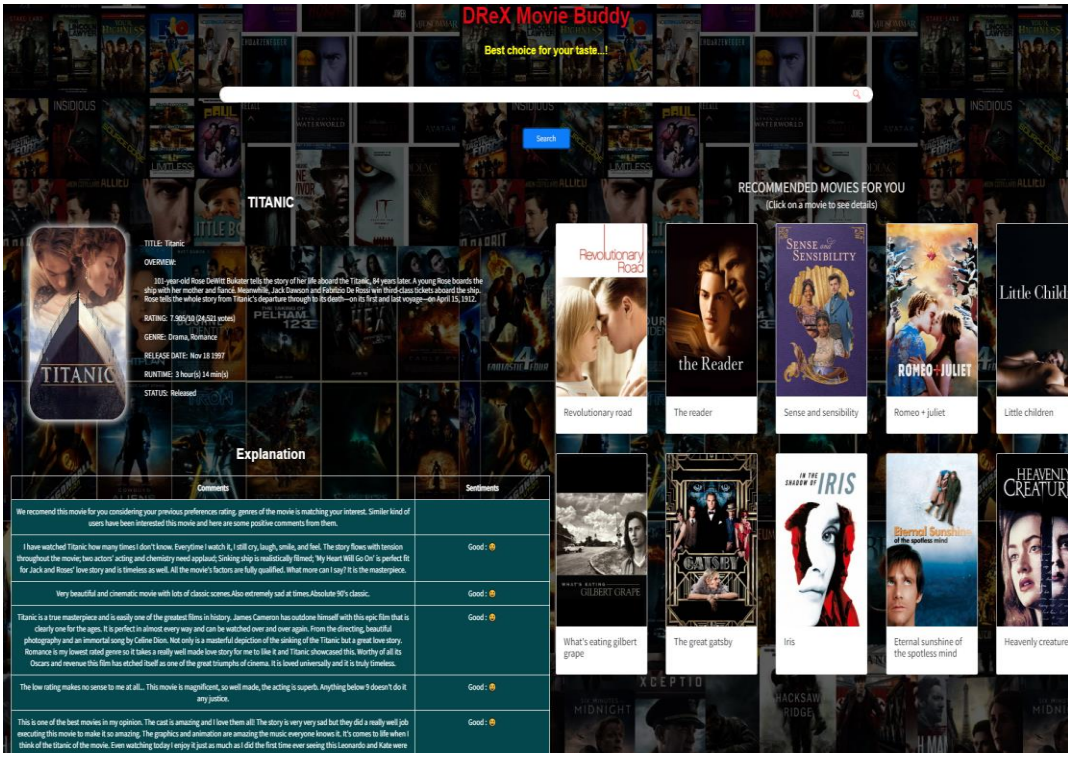

Figure 6.3: Sample recommended results display in GUI.

6.4 Implementation of Input processing module

The input processing module has performed three major tasks and facilitates a smooth run for the system. Python flask is used to implement several functionalities while API calls use JavaScript. First, the system has been registered with the Auth0 service provider to get authentication service API.

6.4.1 Auth0 Connection implementation

When a user is logging in the system directs to Auth0 API to authenticate the user. Then user has already signed up with the DReX Movie Buddy system they just log in using the username and password. They have other options to log in with a Google account.

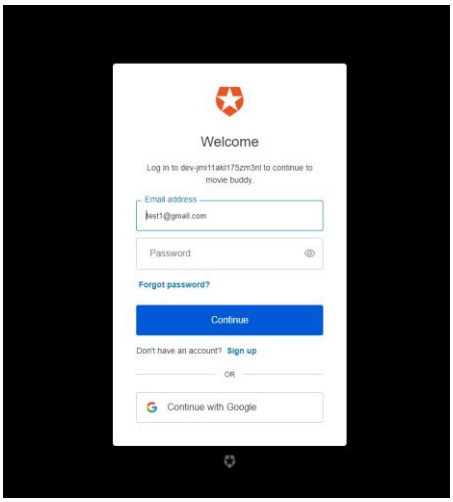

Figure 6.4: DReX Movie Buddy logging options

```python
oauth.register(
    "auth0",
    client_id=env.get("AUTH0_CLIENT_ID"),
    client_secret=env.get("AUTH0_CLIENT_SECRET"),
    client_kwargs={
        "scope": "openid profile email",
    },
    server_metadata_url=f'https://{env.get("AUTH0_DOMAIN")}/.well-known/openid-configuration'
)

@app.route("/login", methods=["GET", "POST"])
def login():
    return oauth.auth0.authorize_redirect(
        redirect_uri=url_for("callback", _external=True)
    )

@app.route("/callback", methods=["GET", "POST"])
def callback():
    token = oauth.auth0.authorize_access_token()
    session["user"] = token
    myuser = oauth.auth0.userinfo()
    suggestions = get_auto_comlete_options()
    return render_template('home.html',suggestions=suggestions, user=myuser.nickname )

@app.route("/logout")
def logout():
    session.clear()
    return redirect(
        "https://" + env.get("AUTH0_DOMAIN")
        + "/v2/logout?"
        + urlencode(
            {
                "returnTo": url_for("home", _external=True),
                "client_id": env.get("AUTH0_CLIENT_ID"),
            },
            quote_via=quote_plus,
        )
    )
```

Figure 6.5: Auth0 connection code segment

6.4.2 Movie image database Connection implementation

The movie image database provides service to users and applications to query movie-related details via REST API. This service is used to get the movie posters of the recommended movies. In that way, the system can present the original poster of the movies when gives a recommendation. The following code segment is used to retrieve movie posters from the service.

```
function get_movie_posters(arr,my_api_key){
  var arr_poster_list = []
  for(var m in arr) {
    $.ajax({
      type:'GET',
      url:poster_search_base_url+my_api_key+'&query='+arr[m],
      async: false,
      success: function(m_data){
        arr_poster_list.push(original_poster_base_url+m_data.results[0].poster_path);
      },
      error: function(){
        alert("Invalid Request!");
        $("#loader").delay(500).fadeOut();
      },
    })
  }
  return arr_poster_list;
}
```

Figure 6.6: Movie poster retrieving code segment

6.4.3 User history data store implementation

User ranking data is saved in a SQLite database table and derived user-specific charts also saved in there as well. SQLAlchemy framework is used to manage databases with Python applications. The SQLAlchemy ORM provides an API to manipulate database table schema creations on the SQLite database. It provides rich capabilities for all simple to complex object-relational mapping.

```
app.config["SQLALCHEMY_DATABASE_URI"]='sqlite:///users.db'
db = SQLAlchemy ( )
db.init_app(app)

class MovieUsers ( db.Model):
    id = db.Column( db.Integer, primary_key = True )
    name = db.Column ( db.String(200), nullable =False)
    date_created = db.Column ( db.DateTime, default=datetime.utcnow)
    last_login = db.Column ( db.DateTime, default=datetime.utcnow)

    def __repr__(self):
        return '<Name %r>' % self.id

class MovieRating ( db.Model):
    id = db.Column( db.Integer, primary_key = True )
    user_id = db.Column ( db.Integer, nullable =False)
    movie_id = db.Column ( db.Integer, nullable =False)
    rating = db.Column ( db.Float(2), nullable =False)

    def __repr__(self):
        return '<Name %r>' % self.id

class UserGenresChart ( db.Model):
    id = db.Column( db.Integer, primary_key = True )
    user_id = db.Column ( db.Integer, nullable =False)
    genre_name = db.Column ( db.String(200), nullable =False)
    rating = db.Column ( db.Float(2), nullable =False)

    def __repr__(self):
        return '<Name %r>' % self.id

class UserDirectorsChart ( db.Model):
    id = db.Column( db.Integer, primary_key = True )
    user_id = db.Column ( db.Integer, nullable =False)
    director_name = db.Column ( db.String(400), nullable =False)
    rating = db.Column ( db.Float(2), nullable =False)

    def __repr__(self):
        return '<Name %r>' % self.id

class UserCastChart ( db.Model):
    id = db.Column( db.Integer, primary_key = True )
    user_id = db.Column ( db.Integer, nullable =False)
    cast_name = db.Column ( db.String(400), nullable =False)
    rating = db.Column ( db.Float(2), nullable =False)

    def __repr__(self):
        return '<Name %r>' % self.id

class UserPlotKeyWordsChart ( db.Model):
    id = db.Column( db.Integer, primary_key = True )
    user_id = db.Column ( db.Integer, nullable =False)
    plot_key_word = db.Column ( db.String(400), nullable =False)
    rating = db.Column ( db.Float(2), nullable =False)
```

Figure 6.7: Historical user data table creation code segment

6.5 Implementation of recommendation generation module

Python surprise library package gives rich capabilities to implement recommendation systems and, in this research, it is used. It has several well-known algorithms and they can be directly trained with our interested data sets. One of the famous algorithms is SVD which is popularized by Simon Funk [18]. This algorithm is used for recommendation generation.

Singular Value Decomposition is a matrix factorization method that generalizes the eigenvectors [15]. SVD is a similar method for principal component analysis (PCA). However, it is a more generalized theory than PCA [19]. Because PCA is assumed input is always a square matrix. But in SVD it is not. The basic SVD formula can be presented as follows.

$$M = U.\Sigma.V^*$$

M is the $m \times n$ input matrix want decompose.

U is $m \times m$ left singular matrix. Columns of U contain eigenvectors of $M.M^*$ matrix. U matrix should satisfied $U.U^* = I_m$ condition.

Σ is a $m \times n$ diagonal matrix containing eigen values.

V is $n \times n$ left singular matrix. Columns of V contain eigenvectors of $M^*.M$ matrix. V matrix should satisfied $V.V^* = I_n$ condition.

The following is an estimate of the anticipated rating for the user.

$$\hat{r}_{ui} = \mu + b_i + b_u + \left(q_i\right)^T.p_u$$

$\hat{r}_{ui}$ is the estimated rating for user u for item i.

μ is the mean of all ratings.

b_u is the base line rating of the user. Zero for unknown users.

b_i is the base line rating of the item. Zero for unknown items.

q_i is the item factor. Zero for unknown items.

p_u is the user factor. Zero for unknown users.

Following error function minimization drive by stochastic gradient decent (SGD).

$$E = \sum_{r_{ui} \in R_{train}} \left(r_{ui} - \hat{r}_{ui} \right)^2 + \lambda \left(b_u^2 + b_i^2 + \left\| q_i \right\|^2 \left\| p_u \right\|^2 \right)$$

The following steps are performed over the all training set repeated n_epoch times. All baselines are initialized to 0. User factors and item factors initialize according to a normal distribution.

$$b_u \leftarrow b_u + \gamma \left(e_{ui} - \lambda b_u \right)$$

$$b_i \leftarrow b_i + \gamma \left(e_{ui} - \lambda b_i \right)$$

$$p_u \leftarrow p_u + \gamma \left(e_{ui} . q_i - \lambda p_u \right)$$

$$q_i \leftarrow q_i + \gamma \left(e_{ui} . p_u - \lambda q_i \right) \quad \text{where } e_{ui} = r_{ui} - \hat{r}_{ui} .$$

The SVD algorithm has several parameters in surprise library implementation. The following parameters are used for the model training.
DNN model setup and training for the prediction. This DNN model is based on behavior sequence transformer architecture. Underlying neural network layers capture user activity history as a sequence to make recommendations.

```
HIDDEN_NETWORK  = [256, 128]
DROPOUT_RATE    = 0.1
NUMBER_OF_NODES = 3

def DReXDNNmodel():
    input_data = generateInputData()
    DNN_features, other_features = convertToFeatures( input_data,
                                                      False,
                                                      False,
                                                      False )

    output_layer = layers.MultiHeadAttention(
                            num_heads=NUMBER_OF_NODES,
                            key_dim=DNN_features.shape[2],
                            dropout=DROPOUT_RATE )(DNN_features, DNN_features)

    output_layer = layers.Dropout(DROPOUT_RATE)(output_layer)
    normalized_layer = layers.Add()([DNN_features, output_layer])
    normalized_layer = layers.LayerNormalization()(normalized_layer)
    leaky_layer = layers.LeakyReLU()(normalized_layer)
    leaky_layer = layers.Dense(units=leaky_layer.shape[-1])(leaky_layer)
    leaky_layer = layers.Dropout(DROPOUT_RATE)(leaky_layer)
    DNN_features = layers.Add()([normalized_layer, leaky_layer])
    DNN_features = layers.LayerNormalization()(DNN_features)
    features = layers.Flatten()(DNN_features)


    if other_features is not None:
        features = layers.concatenate(
            [features, layers.Reshape([other_features.shape[-1]])(other_features)]
        )

    for num_units in HIDDEN_NETWORK:
        features = layers.Dense(num_units)(features)
        features = layers.BatchNormalization()(features)
        features = layers.LeakyReLU()(features)
        features = layers.Dropout(DROPOUT_RATE)(features)

    outputs = layers.Dense(units=1)(features)
    model = keras.Model(inputs=input_data, outputs=outputs)
    return model

model = DReXDNNmodel()
model.compile(
    optimizer=keras.optimizers.Adagrad(learning_rate=0.01),
    loss=keras.losses.MeanSquaredError(),
    metrics=[keras.metrics.MeanAbsoluteError()],
)
train_dataset = getTrainData()
history = model.fit(train_dataset, epochs=5)
test_dataset = getTestData()
err = model.evaluate(test_dataset, verbose=0)
```

Figure 6.8: DNN based recommendation generation module sample code segment

6.6 Implementation of explanation generation module

The explanation generation performs two major tasks in the system. The first one generates explanations based on user history data and recommended movies. The second one analyzes reviews and presents a set of positive reviews and comments from similar users.

When a user signs up and logs in to the system first time, the system asks to perform a movie rating. A Default set of 20 movies is presented for rating and these ratings are stored in a SQLite database. Basic rating data only contain movie ID, user ID, rating, and time stamp. Based on these ratings the system implicitly generates four types of data.

The first one is preparing a chart of genres of rated movies with a weighted rating. Each genre of rated movies is sorted with the weighted rate as described for movies in the data preprocessing section.

The second one is the preparing top cast chart. This chart contains the actors and actresses' names of rated movies. It is also sorted with a weighted rate which is calculated as mentioned in the above sections.

The third one is the preparing director chart. This chart contains the directors' names of rated movies. It is also sorted with a weighted rate which is calculated as mentioned in the above sections.

The fourth one is preparing a chart of interested users' plot keywords. This chart contains plot keywords of rated movies. It is also sorted with a weighted rate. Each plot keyword of rated movies is sorted with a weighted rate as described for movies in the data preprocessing section.

Anytime users can rate additional movies and all four charts change with the impact of new ratings. These rating charts are internally maintained in the system for all users and data is stored in SQLite database. When a user performs a search activity, for each recommended movie the system generates explanations based on these charts. Combinations of four types of reasoning are produced as an output considering the best fit for the user.

Python Gensim library module is used to calculate similarities between recommended movie features and user-specific chart items. Soft cosine similarity scores of the word2vec model have been used to identify similar features of the recommended movie. The following parameters have been used for the Gensim word2vec model.

```
params = {
    'alpha'       : 0.05,
    'vector_size' : 100,
    'window'      : 5,
    'epochs'      : 5,
    'min_count'   : 5,
    'sample'      : 1e-4,
    'sg'          : 1,
    'hs'          : 0,
    'negative'    : 5,
}
```

Figure 6.9: word2vec model parameters

Review categorization is performed using the library of natural language toolkit (NLTK). A review data file is used to train the model and store it as a binary file to use in the DReX Movie Buddy system.

```
import pandas as pd
import numpy as np
import nltk
import pickle
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn import naive_bayes
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_auc_score,accuracy_score

dataset = pd.read_csv(review_file_data_path,sep = '\t', names =['Reviews','Comments'])

stop_word_set = stopwords.words('english')
vectorizer = TfidfVectorizer(use_idf = True,lowercase = True, strip_accents='ascii',stop_words=stop_word_set)
review_dataset = vectorizer.fit_transform(dataset.Comments)
review_category = dataset.Reviews
review_dataset_train, review_dataset_test, review_category_train, review_category_test =
                        train_test_split(review_dataset, review_category, test_size=0.20, random_state=42)
model = naive_bayes.MultinomialNB()
model.fit(review_dataset_train,review_category_train)
```

Figure 6.10: Review categorization model training code segment

6.7 Summary

This chapter, discussed details of the implementation of the DReX Movie Buddy system. Details about the GUI implementation are presented in early sections and details of the implementation of all modules defined in the design chapter are explained. In practical aspect of the implementation and deployment covered and sample code segments and sample screenshots of the system are attached in this chapter.

In the next chapter, results and evaluation are discussed.

CHAPTER 7
EVALUATION

7.1 Introduction

This evaluation chapter aims to assess the performance, accuracy, and quality of the DReX Movie Buddy recommendation system. The evaluation methodology, metrics, and overall objectives of the evaluation are discussed in this chapter.

7.2 Evaluated dataset

Two datasets are used for evaluation. Datasets from group lens containing about 100000 ratings and 1M movie ratings for about 6000 movies from 400 users. These datasets are evaluated and fine-tuned with models. For review classification, a review data file is used that contains about 7000 reviews. For explanation reasonings, the system used “stopwords” of NLTK and text8 datasets.

7.2.1 Data format

Movie data file further explored about their data distribution and other details.

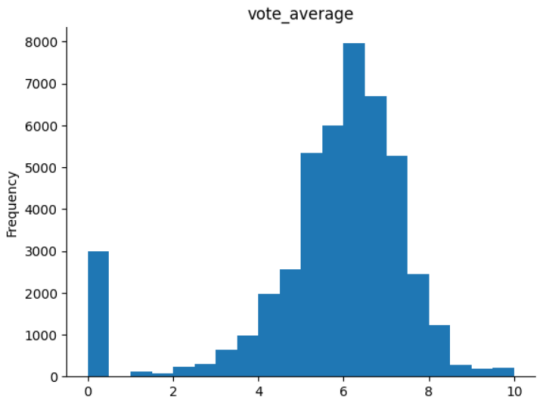

Figure 7.1: vote average distribution of movie data

The following table contains columns and sample data found in the movie data file which used in this research analysis.

Table 7.1 : Sample movie data record

Column name	Sample Data
color	Color,
director_name	James Cameron
num_critic_for_reviews,	723
duration,	178
director_facebook_likes	0
actor_3_facebook_likes	855
actor_2_name	Joel David Moore
actor_1_facebook_likes	1000
gross	760505847
genres	Action|Adventure|Fantasy|Sci-Fi
actor_1_name	CCH Pounder
movie_title	Avatar
num_voted_users	886204
cast_total_facebook_likes	4834
actor_3_name	Wes Studi
facenumber_in_poster	0
plot_keywords	avatar|future|marine|native|paraplegic
movie_imdb_link	http://www.imdb.com/title/tt0499549/?ref_=fn_tt_tt_1
num_user_for_reviews	3054
language	English
Country	USA
content_rating	PG-13
Budget	237000000
title_year	2009
actor_2_facebook_likes	936
imdb_score	7.9
aspect_ratio	1.78
movie_facebook_likes	33000

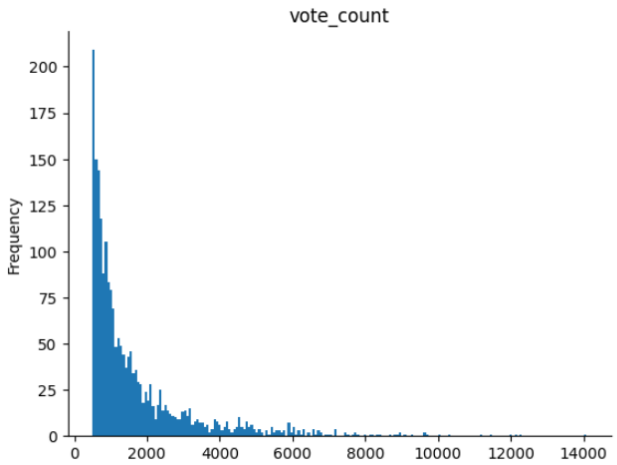

Figure 7.2: vote count distribution of movie data

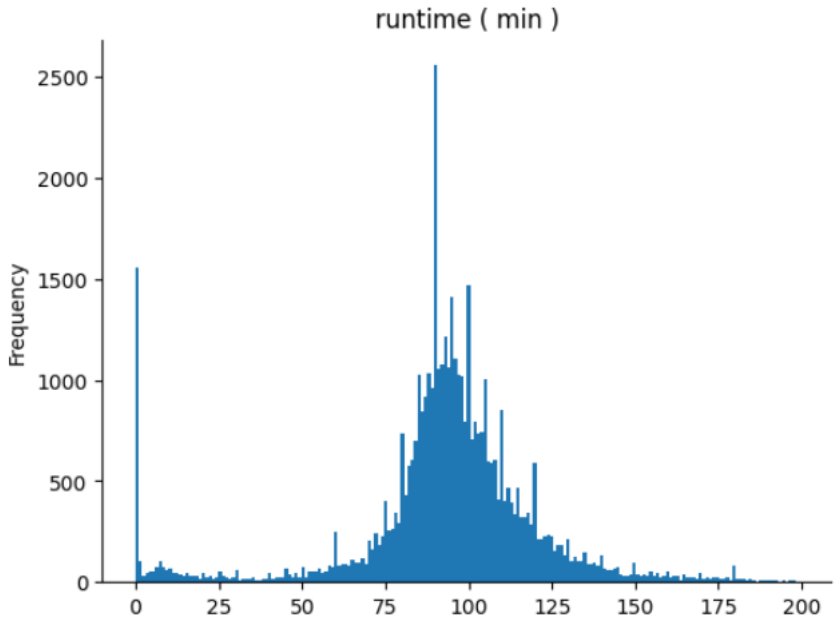

Figure 7.3: Runtime (in minitues) distribution of movie data

7.2.2 Data preprocessing

First, prepare a clean data set by removing garbage values and preparing a clean data set which is significant. Genres of the movie data set have some Jason file format which is not good for us to process. Genre name is replaced with an array of strings.

```
0        [{'id': 16, 'name': 'Animation'}, {'id': 35, '...
1        [{'id': 12, 'name': 'Adventure'}, {'id': 14, '...
2        [{'id': 10749, 'name': 'Romance'}, {'id': 35, ...
3        [{'id': 35, 'name': 'Comedy'}, {'id': 18, 'nam...
4                          [{'id': 35, 'name': 'Comedy'}]
                                    ...
45461    [{'id': 18, 'name': 'Drama'}, {'id': 10751, 'n...
45462                       [{'id': 18, 'name': 'Drama'}]
45463    [{'id': 28, 'name': 'Action'}, {'id': 18, 'nam...
45464                                                   []
45465                                                   []
```

Figure 7.4: Before process genres of movie data

```
movie_data['genres'] = movie_data['genres'].fillna('[]').apply(literal_eval).apply(lambda x: [i['name'] for i in x] if isinstance(x, list) else [])
```

Figure 7.5: genres modification

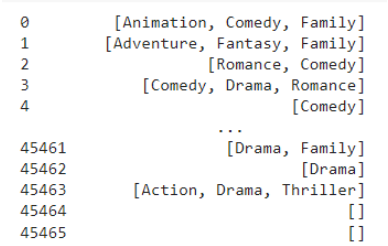

Figure 7.6: After process genres of movie data

7.3 Evaluation Metrics

A recommender system may be assessed using a variety of measures. However, it should have predictive metrics and ranking metrics for the successful evaluation of a such system. Also, it is very important to have the Top-K parameter definition for a recommendation system [14]. In this system, it is focused on the top 10 recommended movies which means our Top-K parameter is 10.

Predicting metrics are used to indicate correctly the system recommends movies for a given user. There are several predictive metrics but here consider one quality metric and one quantitative metric. F1 Score is a well-known quantitative measurement that indicates the degree of overlap between predicted movies and the ground truth. It considers both precision and recall. The F1 score is a very common one for classification tasks or activities. However, it can give a good guess in the recommender system domain as well. The original score can be defined in the following formula.

$$F_{\beta} = \frac{(1+\beta^2) \times precision\,at\,K \times recall\,at\,K}{\beta^2 \times precision\,at\,K + recall\,at\,K}$$

F_1 formula can be derived when $\beta = 1$ as follows.

$$F_1 = \frac{2 \times precision\,at\,K \times recall\,at\,K}{precision\,at\,K + recall\,at\,K}$$

Precision and recall of a recommender system is defined with respect to the Top-K parameter, in this case is 10.

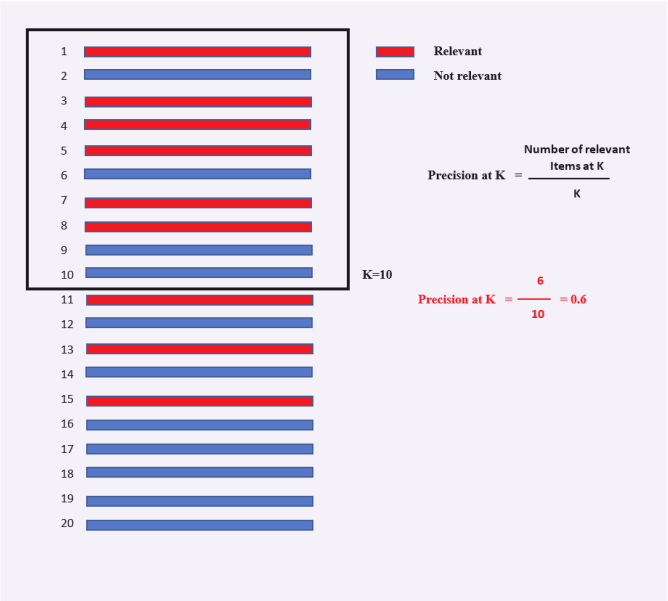

Figure 7.7: Precision

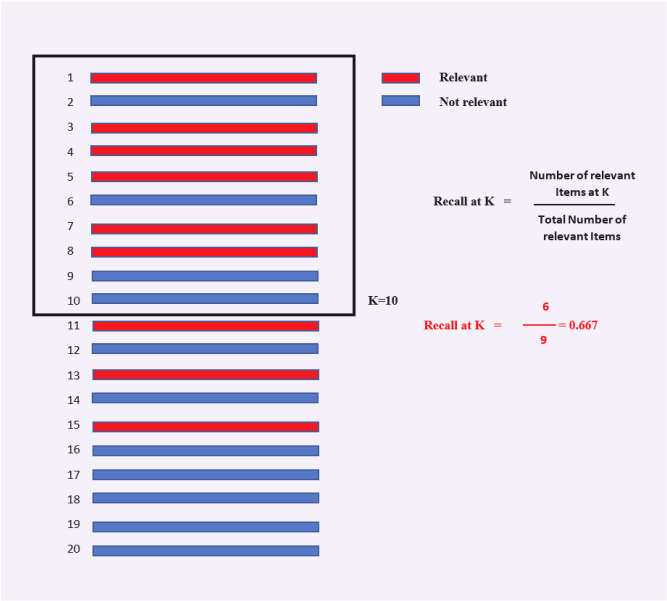

Figure 7.8: Recall

Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE) metrics are evaluated for each algorithm used in the system. Following classic formulas used to calculate RMSE and MAE.

$$RMSE = \sqrt{\frac{y_i - y_p}{n}}$$

$$MAE = \frac{|y_i - y_p|}{n}$$

y_i is the real value.

y_p is the estimated value.

n is the number of items.

SVD algorithm and KNN algorithm used for predictions and their error values evaluate as follows.

Table 7.2 : Parameters of SVD algorithm

Parameter name	Description	Value
n_factors	The number of factors.	100
n_epochs	The number of iterations of the SGD procedure.	20
biased (bool)	Whether to use baselines (or biases).	True
init_mean	The mean of the normal distribution for factor vectors initialization.	0
init_std_dev	The standard deviation of the normal distribution for factor vectors	0.1
lr_all	The learning rate for all parameters.	0.005
reg_all	The regularization term for all parameters.	0.02
lr_bu	The learning rate for bu. Takes precedence over lr_all if set.	none
lr_bi	The learning rate for bi. Takes precedence over lr_all if set.	none
lr_pu	The learning rate for pu. Takes precedence over lr_all if set.	none
lr_qi	The learning rate for qi. Takes precedence over lr_all if set.	none
reg_bu	The regularization term for bu. Takes precedence over reg_all if set.	none
reg_bi	The regularization term for bi Takes precedence over reg_all if set.	none
reg_pu	The regularization term for pu. Takes precedence over reg_all if set.	none
reg_qi	The regularization term for qi. Takes precedence over reg_all if set.	none
random_state	Determines the RNG that will be used for initialization.	none
verbose	If True, prints the current epoch.	True

```
from surprise import SVD,Reader,Dataset
from surprise.model_selection import KFold
from surprise.model_selection import cross_validate
from collections import defaultdict
reader = Reader()

def calculate_fitness_SVD ( ratings ):
    data = Dataset.load_from_df(ratings[['userId', 'movieId', 'rating']], reader)
    algo = SVD()
    cross_validate(algo, data, measures=['RMSE', 'MAE'], cv=5, verbose=True)
```

Figure 7.9: SVD evaluation for RMSE and MAE

```
Evaluating RMSE, MAE of algorithm SVD on 5 split(s).

                  Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Mean    Std
RMSE (testset)    0.9067  0.8894  0.9011  0.8992  0.8878  0.8968  0.0072
MAE (testset)     0.6960  0.6844  0.6936  0.6910  0.6852  0.6900  0.0045
Fit time          1.53    1.60    1.54    1.52    1.46    1.53    0.05
Test time         0.12    0.11    0.12    0.30    0.12    0.15    0.07
```

Figure 7.10: RMSE and MAE values of SVD for 5 splits.

```
from surprise import SVD,Reader,Dataset
from surprise.model_selection import KFold
from surprise.model_selection import cross_validate
from collections import defaultdict
reader = Reader()

def calculate_fitnes_KNN ( ratings ):
    dataKNN = Dataset.load_from_df(ratingsKNN[['userId', 'movieId', 'rating']], readerKNN)
    algoKNN = KNNBasic()
    cross_validate(algoKNN, dataKNN, measures=['RMSE', 'MAE'], cv=5, verbose=True)
```

Figure 7.11: KNN evaluation for RMSE and MAE

```
Evaluating RMSE, MAE of algorithm KNNBasic on 5 split(s).

                  Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Mean    Std
RMSE (testset)    0.9633  0.9675  0.9692  0.9678  0.9685  0.9673  0.0021
MAE (testset)     0.7424  0.7431  0.7451  0.7418  0.7455  0.7436  0.0015
Fit time          0.11    0.15    0.18    0.15    0.15    0.15    0.02
Test time         1.43    1.41    1.45    1.41    1.43    1.43    0.02
```

Figure 7.12: RMSE and MAE values of KNN for 5 splits.

Accuracy of the review classification also evaluated.

```
stop_word_set = stopwords.words('english')
vectorizer = TfidfVectorizer(use_idf = True,lowercase = True, strip_accents='ascii',stop_words=stop_word_set)
review_dataset = vectorizer.fit_transform(dataset.Comments)
review_category = dataset.Reviews
review_dataset_train, review_dataset_test, review_category_train, review_category_test=train_test_split(review_dataset,
                                                                                                        review_category,
                                                                                                        test_size=0.20,
                                                                                                        random_state=42)

model = naive_bayes.MultinomialNB()
model.fit(review_dataset_train,review_category_train)
```

```
accuracy_score(review_category_test, model.predict(review_dataset_test))*100
```

```
97.47109826589595
```

Figure 7.13: Review Classification accuracy.

The explanation generation module used the Gensim library and Google's word2vec model. The performance of this module is also evaluated with the num_trees parameter to fine-tune the accuracy of the system. The below image shows how the accuracy was fine-tuned with the parameter at the cost of initialization time.

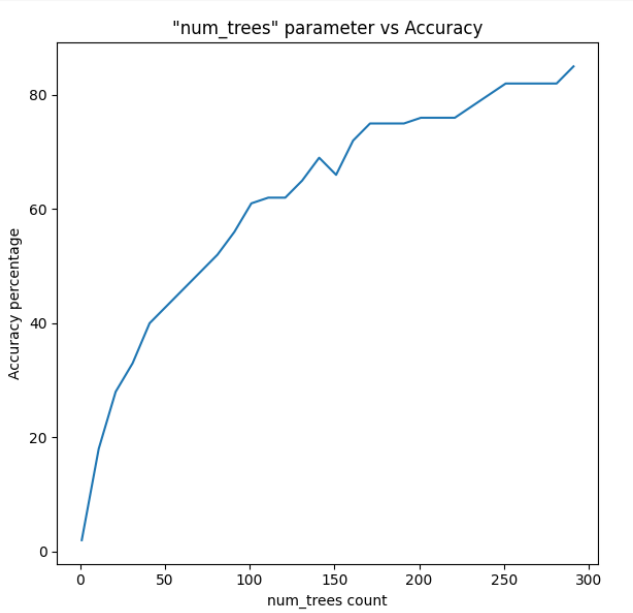

Figure 7.14: explanation generation module accuracy fine tuning.

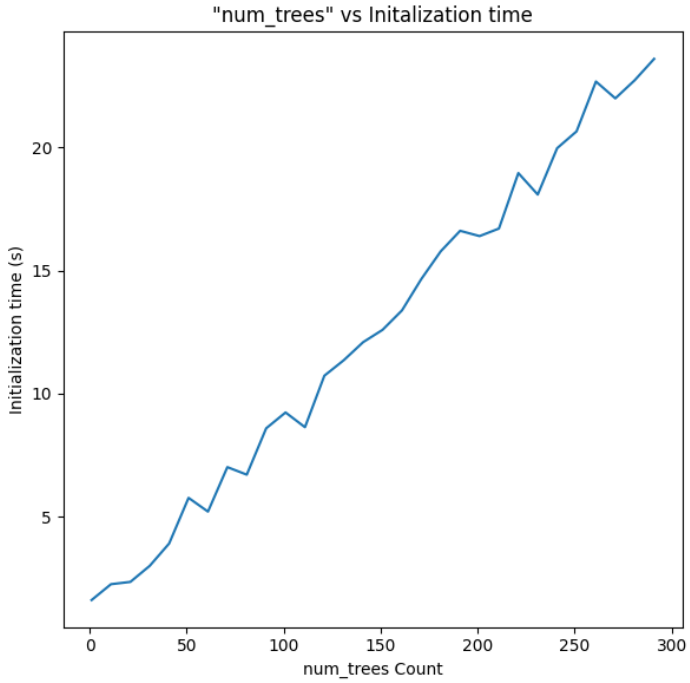

Figure 7.15: explanation generation module initialization time cost.

7.4 Comparison of model performances

The comparative analysis evaluates the behavior of the DReX Movie Buddy system with different models and variations. Fine-tuning the system performance with various parameters and optimized models are used for the system implementation.

Table 7.3: Finetuned model performance comparison.

	SVD model	**KNN Model**	**DReX model**
RMSE	0.8968	0.9673	0.9099
MAE	0.6900	0.7436	0.7599

By comparing the performances of models, it is observed comparative performance achieved with the DReX model with respect to the other standard algorithms and used that as the final model for the system development.

7.5 Discussion of Findings

Based on the evaluation results obtained, it is evident that the deep neural network-based recommend system gives promising results. If it is integrated with natural language processing techniques, can have realistic reasoning and explanation as well. That will improve the transparency and trust in the user's mind. Such a system may have a long run and can have a personalized service for loyal customers. Since everyone has a very busy lifestyle, it is good to have an AI-assisted personal system to recommend very good movies matching their taste without wasting time watching trailers and plots. Since time is priceless if users can have a system like DReX Movie Buddy it will be popular soon.

7.6 Summary

This chapter presents how the evaluation process of the system with evaluation metrics. Evaluation of the recommendation module and evaluation of the explanation module are discussed in detail. The performance of the different algorithms is compared and the best-fit selection process is described. Also, discussed how can be finetuned the performance of the explanation generation module with parameters.

CHAPTER 8

CONCLUSION AND FURTHER WORK

In this research project, a deep neural network-based recommender system is developed with explainable reasoning called the DReX Movie Buddy system. It used the surprise library package for recommendation generation and the Gensim library package with the Google word2vec model for explanation generation. The system presented with a nice web-based GUI and it gives promising results. It showcases the potential to address the requirement for the best movie recommendation system for new-generation users who are having very busy lives. When they have very limited free time they can enjoy the best movies that match their choice rather than wasting that free time to find movies matching their taste.

Through the implementation and evaluation, it can be observed that the recommendation module plays a key role in achieving accurate results. If the system has a higher number of user base its results are more accurate. Also, note that when a user is searching for some movie with the system that doesn't mean the user knows the movie very well. Sometimes this might be the first time that a user wants to know more details about the movie. So the system provides some more details with the GUI so users get more compatible.

Another important thing is that data from movies are also important to the system's performance. However, this data set did not have many details about the movie plot, other than a few plot keywords. If a plot summary is there with movie data, may have more dimensions and results may be enriched.

There is a new dawn in the artificial intelligence domain with the introduction of new impressive systems like Chat-GPT, Google Bard, etc. A new era is on the horizon once these technologies evolve. Everything in life is getting bound with AI technology and movie recommendations also be exposed to these technologies soon. Personalized confidential service might attract more customers than other systems if

the system has better transparency with reasoning. In such cases, mobile applications will provide immense potential to establish a user-friendly movie recommendation system. Hence, this system can be further developed as a mobile application as well.

In conclusion, this research project successfully developed a deep neural network model-based movie recommendation system with explainable reasoning. The system demonstrates promising results by recommending excellent movies matching the user's choice. The findings of the research lay a foundation for further advancement of movie recommendation systems with artificial intelligence technologies.

REFERENCES

[1] Amina SAMIH, Abderrahim GHADI, Abdelhadi FENNAN, (2021) “ExMrec2vec: Explainable Movie Recommender System based on Word2vec”, (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 12, 8

[2] A. Samih, A. Adadi, and M. Berrada, (2019), “Towards a knowledge based Explainable Recommender Systems ”in Proceedings of the 4th International Conference on Big Data and Internet of Things, Rabat Morocco, p.1-5

[3] Qian Zhang · Jie Lu · Yaochu Jin, (2021), “Artificial intelligence in recommender systems”, Complex & Intelligent Systems (2021), P:439–457

[4] J. Lamy, B. Sekar, G. Guezennec, J. Bouaud, B. Séroussi, (2019), “Explainable artificial intelligence for breast cancer: a visual case-based reasoning approach”, Science Direct, 94 :42-53.

[5] E .Negre, (2018), Recommender systems: a categorization, Online :https://interstices.info/les-systemes-de-recommandation-categor isation/ , 2018 (Journal Online Sources style).

[6] F. Isinkaye, Y. Foljimi and B.Ojokoh ,(2015), “Recommendation systems : Principles, methods and evaluation”, Egyptian Information Journal,pp.

[7] Kelong Mao∗, Jieming Zhu∗, Jinpeng Wang, Quanyu Dai, Zhenhua Dong Xi Xiao, Xiuqiang He, (2021), SimpleX: A Simple and Strong Baseline for Collaborative Filtering, arXiv:2109.12613v1 [cs.IR] 26 Sep 2021.

[8] Resnick P, Varian HR. Recommender system’s. Commun ACM 1997;40(3):56–8. http://dx.doi.org/10.1145/245108.24512.

[9] Rashid AM, Albert I, Cosley D, Lam SK, McNee SM, Konstan JA et al. Getting to know you: learning new user preferences in recommender systems. In: Proceedings of the international conference on intelligent user interfaces.

[10] Zheng G et al (2018) DRN: a deep reinforcement learning framework for news recommendation. Proc World Wide Web Conf 2.

[11] Harries M, Horn K (1995) Detecting concept drift in financial time series prediction using symbolic machine learning. In: Proceedings of the 8th Australian Joint Conference on Artificial Intelligence,

[12] Yin H, Cui B, Chen L, Hu Z, Zhou X (2015) Dynamic user modeling in social media systems. ACM Trans Inf Syst.

[13] A. Samih, A. Ghadi, A. Fennan, DEEP GRAPH EMBEDDINGS IN RECOMMENDER SYSTEMS: A SURVEY, Journal of Theoretical and Applied Information Technology 15th August.

[14] F. Isinkaye, Y. Foljimi and B.Ojokoh , “Recommendation systems : Principles, methods and evaluation”, Egyptian Information Journal,

[15] Ruslan Salakhutdinov and Andriy Mnih. (2008) “Probabilistic matrix factorization”. URL: https://papers.nips.cc/paper/3208-probabilistic-matrix-factorization.pdf.

[16] Sheng Zhang, Weihong Wang, James Ford, and Fillia Makedon. (1996) “Learning from incomplete ratings using non-negative matrix factorization.” URL: https://www.siam. org/meetings/sdm06/proceedings/059zhangs2.pdf.

[17] Thomas George and Srujana Merugu.(2005) “A scalable collaborative filtering framework based on co-clustering”.

[18] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, Illia Polosukhin (2017), “Attention Is All You Need” NIPS online : https://sifter.org/~simon/journal/20061211.html

[19] Simon funk, (2006), NetFlix journal- Netflix prize, online : https://sifter.org/~simon/journal/20061211.html

[20] Ruslan Salakhutdinov and Andriy Mnih. (2007) “Probabilistic Matrix Factorization”, NIPS

[21] Qiwei Chen, Huan Zhao, Wei Li, Pipei Huang, Wenwu Ou (20019) “Behavior Sequence Transformer for E-commerce Recommendation in Alibaba”, Alibaba Search & Recommendation Group