Universidade Federal de Juiz de Fora

Instituito de Ciências Exatas

Bacharelado em Ciência da Computação

CineMatch: A self-adaptive architecture for a content recommendation system

Vitória Natália Caetano

JUIZ DE FORA

JULHO, 2024

CineMatch: A self-adaptive architecture for a content recommendation system

Vitória Natália Caetano

Universidade Federal de Juiz de Fora

Instituito de Ciências Exatas

Departamento de Ciência da Computação

Bacharelado em Ciência da Computação

Orientador: José Maria Nazar David

JUIZ DE FORA

JULHO, 2024

CineMatch: A self-adaptive architecture for a content recommendation system

Vitória Natália Caetano

MONOGRAFIA SUBMETIDA AO CORPO DOCENTE DO INSTITUITO DE CIÊNCIAS EXATAS DA UNIVERSIDADE FEDERAL DE JUIZ DE FORA, COMO PARTE INTEGRANTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE BACHAREL EM CIÊNCIA DA COMPUTAÇÃO.

Aprovada por:

JUIZ DE FORA
4 DE JULHO, 2024

Dedico este trabalho ao meu pai, que, sob muito sol, fez-me chegar até aqui, na sombra.

Resumo

Sistemas de recomendação são cruciais em ambientes digitais, especialmente em serviços de *streaming* de filmes, através dos quais eles aumentam o engajamento do usuário pelas sugestões de conteúdo personalizadas. Sistemas tradicionais frequentemente enfrentam desafios com preferências dinâmicas dos usuários e o problema de início frio (cold start), onde dados limitados sobre novos usuários ou itens prejudicam o desempenho. Esta pesquisa introduz o CineMatch, uma arquitetura de recomendação autoadaptativa que alterna a recomendação dinamicamente entre um modelo de rede neural (Keras) e um modelo de vizinhos mais próximos (KNN) com base em métricas de RMSE (Erro Quadrático Médio Raiz) em tempo real. Ao aproveitar os pontos fortes de ambos os modelos, o CineMatch mantém alta precisão e se adapta às preferências evolutivas dos usuários e aos novos conteúdos. O objetivo desta pesquisa é desenvolver uma estrutura híbrida e adaptativa em formato modular. Para isso, implementaremos uma aplicação web destinada a analisar a viabilidade do estudo e aprimorar a eficiência das recomendações. A aplicação será projetada para maximizar a adaptabilidade e a escalabilidade no tratamento de grandes volumes de dados. Incorporando um *loop de feedback*, o CineMatch aprende continuamente com as interações dos usuários para otimizar as estratégias de recomendação. Esta pesquisa detalha o desenvolvimento, implementação e avaliação do CineMatch, demonstrando sua eficácia em fornecer recomendações de filmes precisas. Uma aplicação web que utiliza o conjunto de dados MovieLens foi desenvolvida para apresentar a viabilidade da solução e refinar as capacidades do sistema sob diversos cenários de usuários, destacando a adaptabilidade e precisão desta arquitetura.

Palavras-chave: Sistemas autoadaptativos, Sistemas de recomendação, Recomendações de filmes, Aprendizado de Máquina, Modelos híbridos de recomendação, análise em tempo real, personalização do usuário.

Abstract

Recommender systems are crucial in digital environments, especially in movie streaming services, where they enhance user engagement through personalized content suggestions. Traditional systems often face challenges with dynamic user preferences and the cold start problem, where limited data on new users or items impair performance. This research introduces CineMatch, a self-adaptive recommendation architecture that dynamically switches between a neural network model (Keras) and a K-nearest neighbors (KNN) model based on real-time RMSE (Root Mean Square Error) metrics. By leveraging the strengths of both models, CineMatch maintains high accuracy and adapts to evolving user preferences and new content. The goal of this project is to develop a hybrid and adaptive structure in a modular format. To achieve this, we will implement a web application designed to analyze the feasibility of the study and enhance the efficiency of recommendations. The application will be crafted to maximize adaptability and scalability in handling large volumes of data. Incorporating a feedback loop, CineMatch continuously learns from user interactions to optimize recommendation strategies. This research details the development, implementation, and evaluation of CineMatch, demonstrating its effectiveness in providing accurate movie recommendations. A web application using the MovieLens dataset was developed to test and refine the system's capabilities under diverse user scenarios, showcasing the adaptability and accuracy of this architecture.

Keywords: Self-adaptive systems, Recommendation systems, Movie recommendations, Machine Learning, Hybrid recommendation models, real-time analytics, user personalization.

Acknowledgements

I want to express my heartfelt gratitude to some truly special individuals who have played a significant role in my journey. First and foremost, I owe a tremendous debt of thanks to Professor José Maria David. His unwavering support, invaluable guidance, and constant encouragement have been instrumental in helping me navigate the challenges of my work. Professor David not only shared his wisdom and knowledge but also motivated me to persist and strive for excellence.

I also want to extend my warm appreciation to my dear family and close friends. Their belief in me has been a steady source of strength. There were times when self-doubt loomed large, but their faith in my abilities served as a powerful driving force. They provided the emotional support and encouragement I needed to push through difficulties and keep moving forward.

So, from the depths of my heart, I want to say a profound "thank you" to all those who have been part of this journey with me. Your support, mentorship, and unwavering belief in me have not only been pivotal in my personal and professional growth but have also filled my journey with warmth and inspiration.

“While one may encounter many defeats,

one must not be defeated.“

Maya Angelou

Contents

List of Figures

List of Tables

List of Abbreviations

ALS	Alternative Least Squares
DCC	Departamento de Ciência da Computação
KNN	K-nearest neighbors
MAE	Mean Absolute Error
RMSE	Root Mean Square Error
RS	Recommendation System
SVD	Singular Value Decomposition
UFJF	Universidade Federal de Juiz de Fora

1 Introduction

This chapter aims to provide an introduction to the issues that will be addressed in this work. This will be achieved by presenting the theme, providing contextualization, describing the problem and its underlying reasons, outlining the general and specific goals of the project, as well as describing the methodology used in this work.

1.1 Theme Presentation

In the evolving field of software development, the capacity for systems to adapt autonomously to changing conditions has become increasingly important. This need is particularly acute in environments characterized by dynamic user preferences and rapid data evolution, such as movie recommendation systems (ROY; DUTTA, 2022). Traditional static models often struggle to maintain accuracy over time or require frequent manual updates to stay relevant. Self-adaptive systems, however, are designed to overcome these challenges by continuously monitoring and adjusting their behavior without human intervention, ensuring sustained performance and relevance (BISWAS; HASAN, 2023).

Self-adaptive systems are not just theoretical constructs but are increasingly employed in practical applications where user engagement and satisfaction are directly impacted by the system's ability to adapt. These systems incorporate mechanisms that allow them to perceive changes in the environment or their performance and make adjustments in real-time to meet predefined objectives (SALEHIE; TAHVILDARI, 2009). For example, in the context of movie recommendations, where a user's interest may quickly evolve, a self-adaptive system can modify its recommendation strategies based on ongoing user feedback and interaction patterns, thus maintaining a high level of personalization and accuracy (SUNNY et al., 2017).

The introduction of self-adaptive capabilities into movie recommendation systems can significantly enhance user experience by providing more accurate, personalized, and effective suggestions. By leveraging a hybrid model that intelligently switches between different algorithms, based on real-time performance assessments, the system not only addresses the common challenges of recommendation systems, such as the cold start problem and scalability but also introduces a fresh method of maintaining relevance in a rapidly changing environment (CHENG et al., 2009).

This research aims to explore this innovative approach by developing and evaluating a self-adaptive architecture that dynamically selects the most effective recommendation strategy based on continuous performance metrics. Such an approach is particularly suited to the domain of movie recommendations, where the diversity of content and user preference requires a powerful system design.

1.2 Contextualization

The digital landscape today sees a pervasive use of recommendation systems across various platforms, notably in the realm of movie streaming services. These systems are fundamentally designed to enhance user engagement by suggesting content aligned with individual preferences. The traditional static models, while effective to an extent, often struggle to adapt to the dynamic nature of user interactions and the continual evolution of content(CHANG et al., 2016). To address these limitations, this project introduces an alternative approach by implementing a hybrid, self-adaptive system that dynamically switches between a neural network model [1](Keras) and a K-nearest neighbors [2](KNN) model based on real-time evaluation of their performance metrics, specifically RMSE (Root Mean Square Error). This blend of machine learning techniques is engineered to optimize the user experience by maintaining high recommendation accuracy in real-world scenarios and scalability - specifically in terms of increasing data volumes and user load (CHENG et al., 2009).

Movie recommendation systems face unique challenges that necessitate advanced

[1]Keras is an open-source library that provides a Python interface for artificial neural networks. It was eventually integrated into TensorFlow library, allowing for seamless and scalable deep learning applications.

[2]K-Nearest Neighbors (KNN) is a non-parametric supervised machine learning algorithm used for both classification and regression. It operates by calculating the distance between data points and predicting a value or class based on the majority of its $'k'$ nearest neighbors, where $'k'$ is a user-defined parameter.

solutions like the one proposed in this study. First among these is the issue of varying user preferences, which can change frequently and are influenced by contemporary trends, personal circumstances, and even seasonal changes. Additionally, the "cold start" problem presents a significant hurdle, particularly when onboarding new users who have limited interaction history, making it difficult for the system to generate accurate predictions. Moreover, the overarching need for high recommendation accuracy is paramount, as it directly impacts user satisfaction and platform retention rates. By leveraging a self-adaptive approach, the proposed system is designed to continuously learn and adjust, thereby providing highly accurate and personalized recommendations to both new and returning users (XIA et al., 2013; BURKE, 2002).

The incorporation of a hybrid model employing both deep learning and neighborhood-based algorithms allows the recommendation system to not only address these challenges effectively but also to do so with an agility that matches the pace of user preference evolution. This ensures that every recommendation made is as effective and engaging as possible, enhancing the overall user experience (BURKE, 2002; CHENG et al., 2009)

1.3 Motivation

The proliferation of digital content has made it increasingly challenging for users to navigate the vast landscape of available movies effectively. Recommendation systems serve as a critical tool in enhancing user experience by filtering and personalizing content suggestions according to individual preferences. The motivation for this project stems from the necessity to refine these systems to keep pace with the dynamic and evolving nature of user interactions and preferences in digital entertainment platforms.

A major motivating factor for the development of a self-adaptive recommendation system is the potential to significantly improve system effectiveness and accuracy. By incorporating a mechanism that allows the system to automatically switch between different algorithms based on performance metrics, the system can maintain high precision and relevance under varying conditions. This adaptability is essential in environments where user preferences are not static but evolve with new content releases and cultural trends (CHENG et al., 2009).

Moreover, the integration of a self-adaptive approach addresses several limitations of traditional recommendation systems, such as the cold start problem and the inability to respond to rapid changes in user behavior. By ensuring the system can adapt in real-time, we anticipate not only an increase in user engagement but also a boost in platform reliability and trust. This shift towards more responsive and personalized user experiences is seen as a key advancement in the field of recommendation systems (LOPS; GEMMIS; SEMERARO, 2011)

Thus, the motivation behind this project is to explore and evaluate an architecture that not only meets the immediate needs of users but also dynamically adapts to their changing preferences over time. This ongoing adaptation is crucial for maintaining the effectiveness and accuracy of the system, ensuring that it continuously enhances the user experience and sustains engagement on movie streaming platforms.

1.4 Problem

The core challenge in modern movie recommendation systems lies in their need for higher adaptability and precision to enhance user satisfaction and engagement. Traditional recommendation algorithms, while robust in many scenarios, often fall short when confronted with the dynamic nature of user preferences and the rapid turnover of content libraries in streaming platforms (CHANG et al., 2016). These systems typically struggle to adapt in real-time and may not effectively handle sparse user data, particularly when dealing with new users or less popular content (BURKE, 2002; LOPS; GEMMIS; SEMERARO, 2011).

A significant limitation of existing recommendation models is their reliance on static data sets and predefined algorithms that do not evolve as user interaction data grows or changes. This static approach can lead to the cold start problem, where insufficient data about new users or new movies hampers the system's ability to deliver accurate and relevant recommendations (CHANG et al., 2016). Furthermore, these models often fail to capture the temporal changes in user preferences, a critical aspect given the ever-evolving trends in movie viewership (XIA et al., 2013).

Moreover, the accuracy of recommendations is a paramount concern, as it directly influences the user's decision to continue using the platform. Inaccurate suggestions can lead to user dissatisfaction and decreased platform engagement, which in turn impacts the commercial success of a streaming service. The proposed self-adaptive system seeks to address these issues by employing a hybrid model that dynamically switches between a neural network and a KNN model based on their real-time performance evaluated through RMSE metrics. This approach will ensure that the system continuously adapts to changes in user behavior and preferences, maintaining its effectiveness over time. This continuous refinement of recommendation precision and adaptability maintains the relevance and effectiveness of the recommendations provided to users (CHENG et al., 2009; SALEHIE; TAHVILDARI, 2009).

1.5 General and Specific Objectives

This section outlines the overarching aim and specific targets of our project. These objectives are designed to comprehensively enhance the functionality and responsiveness of movie recommendation systems through thoughtful architectural developments.

1.5.1 General Objective

The primary goal of this project is to enhance the effectiveness and adaptability of movie recommendation systems through the development of a viable self-adaptive architecture. This architecture will dynamically select the most appropriate recommendation model based on ongoing performance evaluations, thereby ensuring that users receive the most accurate and timely suggestions tailored to their evolving preferences. Specifically, this project aims to: develop a hybrid recommendation system that combines deep learning and K-nearest neighbors models; address the cold start problem to improve recommendations for new users; and and continuously adapt to evolving user preferences. By refining our approach based on real-time user interaction data, the system is designed to maintain its effectiveness and accuracy, ensuring a high-quality user experience that evolves with the changing landscape of viewer preferences.

1.5.2 Specific Objectives

1. **Develop a Hybrid Recommendation System:** Implement a web application to assess the architectural feasibility, which intelligently alternates between a deep learning model (Keras) and a K-nearest neighbors (KNN) model based on real-time Root Mean Square Error (RMSE) performance metrics. This approach leverages the strengths of both algorithms to maximize recommendation accuracy across varying user data scenarios. Specifically, the Keras model is used for regular user recommendations, while the KNN model effectively handles recommendations for new users with limited historical data or for users whose patterns haven't been accurately mapped by Keras yet.

2. **Address the Cold Start Problem:** Enhance the system's ability to provide accurate recommendations for new users by employing the KNN model when the neural network lacks sufficient data. This approach aims to mitigate the impact of sparse user interaction data, which is a common challenge in traditional recommendation systems (LOPS; GEMMIS; SEMERARO, 2011).

3. **Adapt to Evolving User Preferences:** Design the system to continuously learn from user interactions and adapt its recommendation strategies accordingly. This includes refining the model selection process over time to respond to changes in user behavior and preferences, thus maintaining a high level of personalization and user satisfaction.

These objectives are designed to address specific challenges identified in movie recommendation systems and contribute to the broader field of personalized content delivery by providing an adaptive solution that can be applied to various digital media platforms.

1.6 Methodology

The methodology for this project is designed to ensure a rigorous and comprehensive evaluation of the self-adaptive recommendation system. The approach is divided into several key phases, each focusing on a specific aspect of system development and assessment:

1.6.1 Model Selection

Keras was chosen for its ease of use and flexibility, making it suitable for rapid development and experimentation. According to Yapıcı and Topaloğlu (2021), Keras is an excellent choice for implementing complex neural network architectures quickly and effectively, especially when considering larger batch sizes.

In contrast, KNN was selected for its simplicity and effectiveness in scenarios with limited historical data (BAHRANI et al., 2024). KNN is particularly useful for new users or users whose patterns have not yet been well mapped by the Keras model. By comparing user similarities, KNN can provide accurate recommendations without requiring extensive historical data. This complements the deep learning approach of Keras, enhancing the overall adaptability of the recommendation system.

1.6.2 Model Development and Training

1. **Data Collection and Preparation:** Gather a comprehensive dataset from a movie streaming platform, including user ratings, movie metadata, and user interaction logs. This data will be preprocessed to handle missing values, encode categorical variables, and normalize numerical inputs, providing the foundational data necessary for model training and refinement.

2. **Training the Keras Model:** Develop a neural network using the Keras library, specifically designed to forecast user preferences from their historical data. This model will be trained on a selected portion of the overall data, aiming to maximize recommendation accuracy.

3. **Training the KNN Model:** Implement a K-nearest neighbors model to provide recommendations based on the similarity of user interaction patterns. This model will be particularly useful in handling new users or items with limited interaction data, effectively addressing the cold start problem.

1.6.3 Performance Evaluation

1. **Defining Performance Metrics:** Establish RMSE as the primary metric for evaluating the accuracy of both the Keras and KNN models.

2. **Real Time Model Switching Logic:** Develop a decision-making algorithm that dynamically selects which model to use based on their real-time performance. This algorithm will factor in the current RMSE of each model and switch to the model that shows superior performance for the current user session.

1.6.4 Feedback Loop Implementation

1. **Continuous Learning and Adaptation:** Develop a feedback loop that enables the system to learn from new user interactions and continuously update the models' parameters without manual intervention. This approach will ensure that the system continuously adapts to changes in user behavior and preferences, maintaining its effectiveness over time.

This methodology is designed to ensure that the proposed architecture for a self-adaptive recommendation system is both feasible and technically proficient, aiming to significantly enhance user engagement and satisfaction. By focusing on the development and refinement of hybrid models, we address the cold start problem and ensure the system's adaptability to evolving user preferences. This ensures a personalized experience for each user, maintaining precision and effectiveness over time.

In the preliminary testing of the application developed to assess this architecture, we have observed indications of strong adaptability and performance. These initial results suggest that the architecture is viable for this research, enhancing the development of a content recommendation system by meeting key software quality metrics.

The structure of this research is organized as follows: Chapter 2 presents the theoretical foundations, including key concepts and technologies underlying the development of recommendation systems. Chapter 3 discusses related works, highlighting existing approaches and their relevance to this project. Chapter 4 elaborates on the architecture of the proposed system and examines the results derived from its implementation. Chapter 5 provides the conclusion, summarizing key findings and contributions of the research. This structured approach allows for a detailed exploration of both the technical and contextual aspects of adaptive recommendation systems.

2 Theoretical Foundation

In today's digital age, streaming services have transformed how we consume media, making sophisticated recommendation system architectures increasingly crucial (HASAN; JHA; LIU, 2018). The dynamic nature of user preferences and the continuous expansion of content offerings demand architectures that are not only responsive but also adept at adapting to new trends and behaviors.

This chapter delves into advanced methodologies in recommendation systems, with a particular focus on the K-Nearest Neighbors (KNN) algorithm, neural networks, and self-adaptive systems. These methodologies represent core components of modern machine learning techniques implemented in CineMatch, significantly enhancing the responsiveness and effectiveness of these systems. Through these approaches, CineMatch aims to deliver personalized content recommendations that are both timely and relevant, adapting dynamically to the ever-evolving landscape of user interactions and preferences.

2.1 K-Nearest Neighbors (KNN)

The K-nearest neighbors (KNN) algorithm is a simple yet versatile tool widely used in both classification and regression tasks within machine learning. Its fundamental concept involves predicting the category of a data point by examining the most frequent category among its $'k'$ nearest neighbors. To utilize KNN, one first selects $'k'$, the number of neighbors to consider. Then, distances between the data point and others in the dataset are computed, typically using Euclidean distance. The $'k'$ closest data points are identified, and for classification purposes, the category most common among these neighbors is assigned to the new point (PETERSON, 2009).

In recommendation systems, such as those for movies, KNN can forecast a user's interest in a specific item based on the preferences of similar users. Here, each user's tastes are mapped as points in a multidimensional space, where each dimension corresponds to ratings for different movies. To generate recommendations, the algorithm identifies $'k'$

users nearest to the target user in terms of preferences and suggests movies highly rated by these neighbors but not yet by the target user.

Despite its widespread use, the KNN algorithm faces challenges and limitations. One significant issue is determining the optimal number of neighbors, $'k'$, which typically requires experimental tuning to achieve satisfactory results. Moreover, its reliance on Euclidean distance for measuring similarity can adversely impact performance, especially when dealing with categorical data, as the method primarily assumes numerical attributes.

Nevertheless, KNN remains a favored choice for tasks involving classification, pattern recognition, and recommendation systems, owing to its simplicity and interpretability. These characteristics enable an intuitive approach to data analysis, where decision-making is informed by the proximity and similarity of data points. As depicted in Figure 2.1, similar items generally cluster closely, illustrating the algorithm's effectiveness in recognizing patterns within datasets.

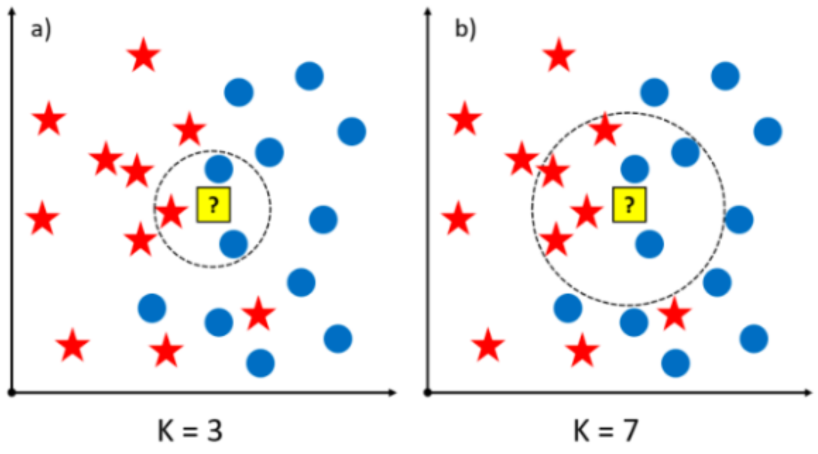


Figure 2.1: Example of KNN Method: a) K equal to 3; and b) K equal to 7 (BRANDÃO; PINTO; SANTOS, 2022)

2.2 Neural Networks

Neural networks are designed to iteratively learn and adjust their internal parameters, such as weights and biases, based on the data they encounter during training. This learning process usually involves techniques that systematically minimize the discrepancies between the predicted and actual outputs. Such models support incremental learning, where they are trained continuously on a stream of incoming data.

Neural networks can also be adapted to new, similar tasks through fine-tuning. This involves retraining a pre-trained model on a new dataset or adjusting a subset of its layers, allowing the model to efficiently adapt to new challenges with relatively little effort (NADER, 2020). The inherent adaptability of neural networks makes them well-suited for dynamic environments, where they can respond to changes over time through what is often referred to as a feedback loop, depicted in Figure 2.2.

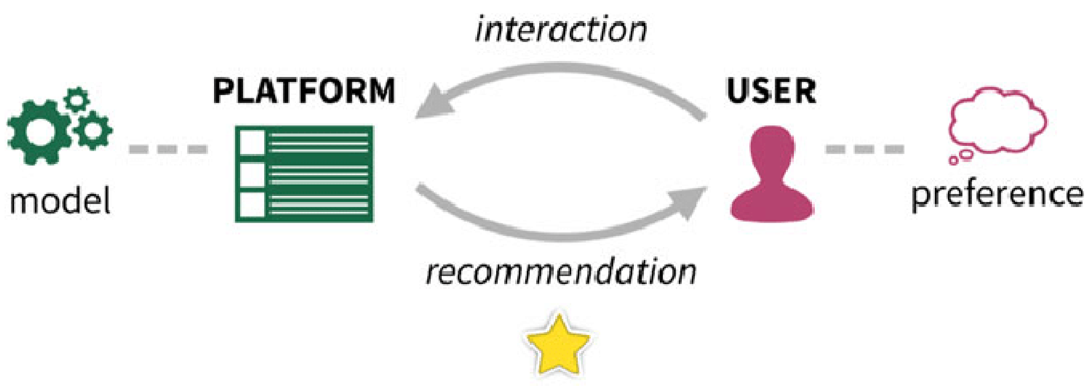


Figure 2.2: Recommendation feedback loop schema (NADER, 2020)

This flexibility is particularly beneficial for personalized recommendation systems, as it enables neural networks to constantly evolve and refine their predictions based on user interactions. This is in stark contrast to static models, which do not update their outputs in response to new information unless they are manually retrained.

2.3 Self-Adaptive System

Self-adaptive systems are engineered to continually assess their performance and the surrounding environment. These systems collect and analyze data to inform decision-making processes about potential adaptations. Decisions are typically driven by predefined rules, machine learning algorithms, or specialized algorithms designed to optimize specific performance metrics. Once a decision is reached, the system executes the necessary modifications. This may involve altering configuration settings, reallocating resources, shifting operational modes, or updating algorithms or policies. A conceptual architecture of a self-adaptive system is depicted in Figure 2.3.

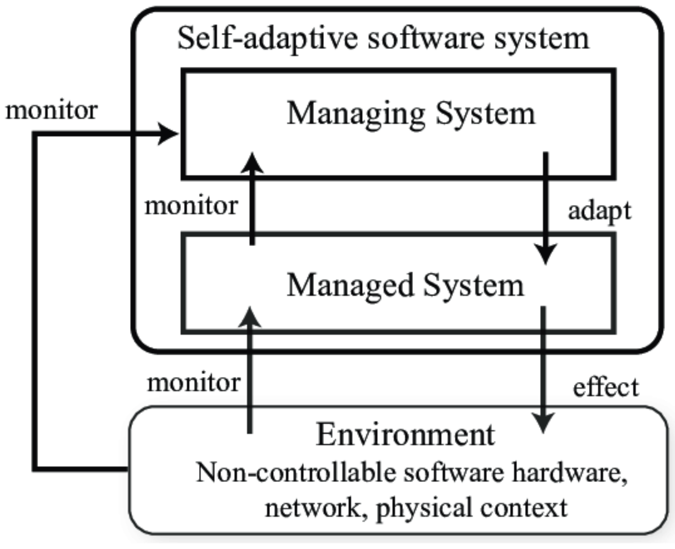

Self-adaptive software system
Managing System
monitor
monitor
adapt
Managed System
monitor
effect
Environment
Non-controllable software hardware, network, physical context

Figure 2.3: Conceptual Architecture of a Self-Adaptive Software System (ABBAS; ANDERSSON, 2018)

Self-adaptive systems, particularly in the realm of real-time recommendation systems, mark a distinct advancement over traditional and hybrid models. This is mainly because they overcome the challenges associated with managing dynamic data and ever-changing user behaviors. Unlike traditional models that rely on batch updates to generate recommendations, self-adaptive systems operate on a continuous input basis, allowing them to instantly respond to shifts in user preferences or activities (GARLAN, 2021).

This feature is crucial in today's digital environment, where user preferences can fluctuate swiftly and unpredictably—a common challenge across various platforms. By employing an ongoing learning and adaptation process, self-adaptive systems can dynamically fine-tune their algorithms. This continuous adaptation enhances the accuracy and relevance of recommendations, which in turn boosts user engagement and satisfaction (SUNNY et al., 2017).

2.4 Chapter Final Remarks

This chapter has thoroughly examined the integration of key methodologies in recommendation systems, focusing on the K-Nearest Neighbors (KNN) algorithm, the use of neural networks, and self-adaptive systems. These technologies form the core of a sophisticated self-adaptive system architecture designed to address the dynamic challenges of modern recommendation systems.

Traditional and hybrid algorithms often encounter difficulties with dynamic user preferences, the cold start problem, or real-time adaptability. In contrast, self-adaptive system architectures excel by continuously learning and evolving with each user interaction, enabling them to seamlessly adjust and refine their algorithms to overcome these hurdles. The adoption of such architectures ensures sustained relevance and accuracy, providing recommendations that are acutely aligned with the users' evolving interests.

In conclusion, the transition towards self-adaptive design marks a significant milestone in the development of recommendation systems. This new approach promises a future where systems anticipate and adapt to user needs in real-time, providing a more personalized and engaging user experience. Building upon this understanding, the next chapter delves into various research studies that focus on different aspects of recommendation systems.

3 Related Works

This chapter delves into various research studies that focus on different aspects of recommendation systems. It synthesizes and compares these works, which employ a range of techniques tailored to specific scenarios, with the ultimate goal of enhancing the effectiveness and personalization of these systems.

3.1 A Recommendation System to Envision Movie Ranking Using Collaborative Filtering

In Aggarwal et al. (2023), the researchers aimed to develop a recommendation system to rank movies using collaborative filtering techniques. They tested both user-based and item-based collaborative filtering methods. The system's core functionality lies in predicting user ratings for movies based on past user behavior and similar user profiles. The model was trained using the Movie Lens small dataset, which includes movie IDs, titles, genres, and user ratings. The dataset was cleaned and organized to ensure accurate analysis.

The front-end interface of the system allows users to log in, rate movies, and receive personalized suggestions. This interface was designed to enhance user experience through its simple and intuitive layout, enabling easy interaction with the recommendation system. The researchers evaluated the system's performance using various metrics, measuring the accuracy of their predictions and comparing the results with existing recommendation systems. They highlighted the importance of continuously updating the datasets to adapt to changing user preferences and improve the system's recommendations.

One of the strengths of this research is its focus on enhancing the precision of movie recommendations through collaborative filtering techniques. The system's ability to adapt and improve based on user feedback is a notable advantage. Additionally, the user-friendly interface ensures that users can easily interact with the system and benefit from its recommendations.

However, there are several limitations. The system's performance can be affected by the sparsity of user ratings, which may lead to less accurate recommendations for movies with fewer ratings. Another limitation is the potential computational complexity associated with large datasets, which can impact the system's scalability and response time. Additionally, the recommendation algorithm may struggle to make accurate predictions for new users or new movies (cold start problem).

Overall, this research presents a robust recommendation system using collaborative filtering techniques, emphasizing user experience and adaptability. The findings underscore the importance of data quality and the need for continuous improvement in recommendation algorithms.

3.2 Implementation Of A Hybrid Recommendation System For Movies

In Prajna et al. (2022), the authors developed a hybrid movie recommendation system that integrates both content-based and collaborative filtering techniques. By using Cosine Similarity for content-based filtering and Singular Value Decomposition (SVD) for collaborative filtering, the system provides precise and relevant movie suggestions.

The researchers sourced data from publicly available datasets such as Kaggle and the TMDB dataset, containing information on 5000 movies. They cleaned and preprocessed the data to ensure a robust foundation for their model. The hybrid system utilized key features like genre, title, cast, crew, and keywords to generate recommendations, while the collaborative filtering component, powered by SVD, predicted user ratings based on patterns observed in similar users' preferences.

A significant aspect of this research is the development of a user-friendly web application that allows users to receive personalized suggestions seamlessly. The application caters to both new and existing users, offering popular movie recommendations to newcomers and tailored suggestions based on historical data for returning users. The hybrid model's performance was evaluated using metrics such as Mean Absolute Error (MAE) and Root Mean Square Error (RMSE), achieving an MAE score of 0.6, which surpasses standalone content-based or collaborative filtering models.

Despite its strengths, the study acknowledges limitations such as the cold start problem, challenges in data collection and processing, and the computational demands of implementing SVD, which affect the system's scalability and response time. In summary, the research presents a sophisticated hybrid recommendation system that significantly improves recommendation accuracy, with ongoing efforts needed to address existing issues.

3.3 Dynamic generation of personalized hybrid recommender system

In their research on recommendation algorithms, Dooms (2013) shed light on the prevailing assumption that users in a system exhibit similar behavior. However, recent studies have highlighted the uniqueness of each user, suggesting that a combination of different algorithms could be more effective for individuals. To address this, the researchers aimed to develop a hybrid recommendation platform capable of seamlessly integrating existing algorithms and dynamically combining them into an optimal hybrid recommender. The system employs multiple algorithms concurrently, treating each as a black box, and combines their predictions using reinforcement learning to adapt to real-time user feedback.

The learning module optimizes the recommendation process by adjusting the weights assigned to different algorithms for each user, minimizing a predefined objective function. This dynamic combination allows the system to continuously evolve based on user interactions. For movie recommendations, the researchers used a large online dataset from 2000, integrating rating data from social media platforms to include new and relevant items, thereby addressing the cold start problem.

One of the significant advantages of this research is its dynamic and adaptive nature. By employing reinforcement learning, the system continuously evolves based on user feedback, leading to highly personalized recommendations. Additionally, treating each algorithm as a black box allows for the easy integration of new algorithms without modifying the existing system. This modularity enhances the system's flexibility and scalability. Moreover, the inclusion of user control over the recommendation process helps prevent filter bubbles, ensuring that users receive diverse and relevant content.

However, the study acknowledges several limitations. The computational complexity of combining multiple algorithms and adjusting weights in real time can be resource-intensive, potentially affecting system performance and scalability. While the system addresses the cold start problem, new users with no interaction history may still receive less accurate recommendations. Additionally, reliance on the quality of external datasets, such as social media data, could affect recommendation accuracy if the data is noisy or biased. Overall, the research presents a robust hybrid recommendation system that leverages reinforcement learning for continuous improvement, though challenges in computational complexity and data quality remain.

3.4 Implementation of a self-adaptive real-time recommendation system using Spark

In Sunny et al. (2017), the researchers focused on building the architecture of a real-time recommendation system for suggesting TV channels to viewers instantly, by using libraries that offer several classic recommendation algorithms.

The developed system consists of five major modules: the streaming process, training process, recommendation and analysis process, database or storage, and user interface.

The streaming process module serves as the initial step, involving the acquisition of a continuous stream of data. This data is then processed and filtered to extract essential information. This filtered data is subsequently stored in the system's memory, which serves multiple purposes. Firstly, it is utilized for training purposes, enabling the system to learn and improve its recommendations over time. Additionally, the stored data is used to identify the currently trending channels and to build a comprehensive profile of the viewers.

In the training process module, they train a machine learning model to create a self-adapting real-time recommendation system that predicts the most suitable channels. It's important to note that for the cold start problem, the system predicts the viewer's profile using the Logistic Regression algorithm. Generally, for new viewers, the system flexibly runs the model to generate more relevant recommendations, while for experienced viewers, it produces more accurate models.

For the recommendation process, it follows the Figure 3.1 below:

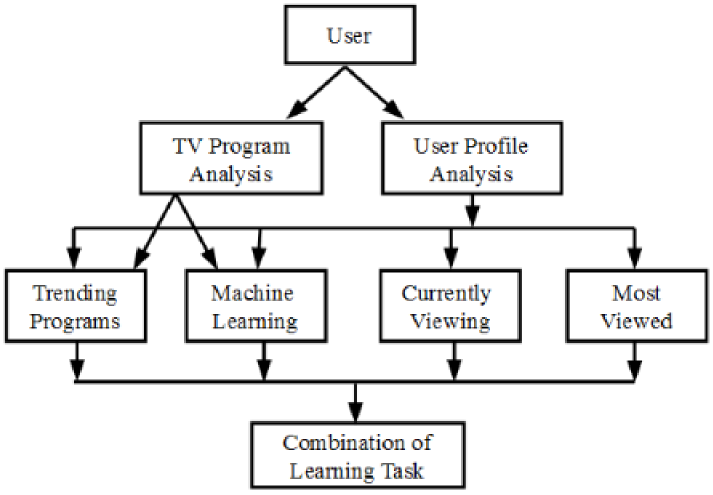


Figure 3.1: Recommentation Process (SUNNY et al., 2017)

The system stores an analysis of the TV programs and the viewer's profiles, combined with trending programs and the user's implicit feedback (viewing patterns and history). Additionally, it considers explicit feedback from social and geographical factors (such as the most popular programs in a specific zip code) to generate recommendations.

For this system to function effectively, it requires a reliable and adaptable database that can handle large amounts of data without errors. The researchers implemented the Lambda architecture, which manages both real-time and batch data processing, ensuring smooth and efficient operations.

Lastly, the user interface plays a crucial role in this system. Since it is designed for TV usage, it is presented to users as a pop-up window. Within this window, users can easily view accurate recommendations, with the most suitable suggestion displayed first. The interface provides simple buttons that allow users to accept a recommendation, move to the next suggestion, or close the pop-up.

This system offers several advantages, including user independence. The recommendations are tailored to align with the user's current viewing behavior and history. Moreover, it self-adapts through the utilization of machine learning algorithms. Additionally, it ensures real-time processing capabilities, enabling accurate recommendations to be generated as new data is incrementally added to the system. The system effectively addresses the cold start problem, allowing it to provide recommendations for new users without any issues. Furthermore, it enhances user profiles by incorporating implicit and explicit feedback, as well as historical data, to deliver precise results. Lastly, the system employs Lambda architecture for independent data processing, enabling seamless real-time and batch data processing without any delays.

However, this architecture employs TV channel recommendations, which typically requires less deep personalization and deals wit less complex preference patterns compared to movie recommendations.

3.5 A Recommendation Engine for Predicting Movie Ratings Using a Big Data Approach

In Awan et al. (2021), the authors investigated the utilization of machine learning, specifically collaborative filtering, in conjunction with the ALS (Alternating Least Squares) algorithm to develop a robust recommender system capable of predicting user ratings based on their behavior. The ALS algorithm, similar to regression analysis, uses matrix factorization to propose strong recommendations, effectively addressing metadata sparsity and the cold start problem.

The proposed system uses both user-level and item-level techniques, employing matrix factorization with ALS. The researchers divided the rating dataset into item-based and user-based matrices along with latent features. By recombining these matrices, a new user matrix with missing entries was obtained, filling gaps based on user and item-level similarities. The system was built using Apache Spark MLlib on a cloud-based platform, processing large datasets efficiently and providing high accuracy in recommendations.

One of the significant advantages of this research is its comprehensive approach to handling common challenges in recommendation systems. The integration of user and item-based techniques, combined with the ALS algorithm, enhances the system's accuracy and scalability. The use of Apache Spark's machine learning libraries further improves efficiency, enabling the system to handle large volumes of data. The system achieved a 97% accuracy rate, demonstrating the effectiveness of the model.

However, the study acknowledges several limitations. The computational complexity of dynamically combining multiple algorithms and adjusting their weights in real time can be resource-intensive, potentially impacting system performance and scalability. While the system addresses the cold start problem to some extent, new users with no interaction history might still receive less accurate recommendations initially. Additionally, the reliance on the quality of external datasets poses a challenge; noisy or biased data can affect the recommendation accuracy.

Overall, this research presents a sophisticated recommendation system that successfully blends collaborative filtering techniques with big data approaches, leveraging reinforcement learning for continuous improvement. The strengths lie in its adaptability, flexibility, and user empowerment, though challenges in computational complexity and data quality remain areas for further enhancement.

3.6 Comparative table

The related works presented here employ various techniques for different scenarios. Therefore, when considering these works, it is crucial to highlight the gaps that this study aims to clarify and explore more comprehensively within the current context. Table 3.1 summarizes the primary features of each system, comparing them to the capabilities of our proposed system, "CineMatch."

Each system is evaluated based on seven key attributes: collaborative filtering, content-based filtering, hybrid approach, real-time processing, addressing the cold start problem, accuracy, and scalability. These attributes are essential for assessing the effectiveness and applicability of recommendation systems in different scenarios.

Collaborative filtering is a technique used by all the systems reviewed, underlining its significance as a cornerstone in recommendation systems. However, content-based fil-

Work	**(AGGARWAL et al., 2023)**	**(PRAJNA et al., 2022)**	**(DOOMS, 2013)**	**(SUNNY et al., 2017)**	**(AWAN et al., 2021)**	**CineMatch**
Collaborative Filtering	x	x	x	x	x	x
Content-Based Filtering		x				x
Hybrid Approach		x	x			x
Real-Time Processing			x	x	x	x
cold start Problem Addressed			x	x	x	x
High Accuracy	x	x	x	x	x	x
Scalability			x	x	x	x

Table 3.1: Comparison between works

tering is only utilized by Prajna et al. (2022) and CineMatch, which points to a specialized approach where recommendations are tailored based on the characteristics of items. A hybrid approach, which enhances both the robustness and relevance of recommendations, is adopted by CineMatch, Prajna et al. (2022), and Dooms (2013).

Real-time processing is crucial for adapting recommendations to dynamically changing data. This feature is integrated into CineMatch and the systems developed by Dooms (2013), Sunny et al. (2017), and Awan et al. (2021) to ensure that suggestions remain timely and relevant. Addressing the cold start problem, which ensures reliable recommendations are available for new users, is a feature incorporated by CineMatch, Dooms (2013), Sunny et al. (2017), and Awan et al. (2021).

All systems strive to achieve high accuracy in their predictions, a critical factor for user satisfaction and the credibility of the system. Scalability is another vital attribute, essential for managing growing data volumes and user bases efficiently. CineMatch, along with the systems by Dooms (2013), Sunny et al. (2017), and Awan et al. (2021), are particularly noted for their ability to perform efficiently at scale.

By addressing gaps identified in the methodologies of Aggarwal et al. (2023), Prajna et al. (2022), Dooms (2013), Sunny et al. (2017), and Awan et al. (2021), CineMatch aims to offer a more nuanced understanding and application of recommendation systems in today's digital landscape.

3.7 Chapter Final Remarks

In summary, this chapter has provided a detailed examination of various research studies on recommendation systems, highlighting multiple techniques used to enhance personalization and user satisfaction. While significant advancements have been made in collaborative filtering, content-based filtering, and hybrid models, challenges remain in real-time processing, scalability, precision, and cold start problems. By identifying these gaps, we gain a deeper understanding of the current state of recommendation systems and establish a solid foundation for the innovative architecture to be introduced shortly.

4 Proposed Architecture

This chapter outlines the detailed architecture of CineMatch, a self-adaptive movie recommendation system developed to enhance user experience through accurate and effective movie suggestions. Building upon the robust foundation provided by the MovieLens dataset, this chapter will delve into the system's architecture, emphasizing the integration of advanced machine learning models and user-centric functionalities.

4.1 Architectural Overview

The foundation of CineMatch is built upon the MovieLens[3] dataset, which comprises 100,000 ratings ranging from 1 to 5, contributed by 943 users across 1682 movies. Each user in the dataset has rated at least 20 movies. This dataset, released in April 1998, serves as the primary data source for our experiments and is freely available online.

The preprocessing of the MovieLens dataset was tailored for two distinct machine learning models: Keras and KNN. This involved transforming the raw data into two separate datasets optimized for each model's requirements, executed within a Python-based environment[4]. The preprocessing steps were crucial for ensuring the data was suitable for training the models, which are designed to recommend new, unseen movies that users are likely to enjoy based on their historical preferences.

Following the data preparation, the refined datasets were utilized to train the recommendation models. These models are integral to the system, enabling it to generate personalized movie suggestions for users. The trained models are then deployed within a web application that allows users to retrieve movie recommendations by entering their IDs. Thanks to the modular and scalable design of this project, it is possible to seamlessly incorporate alternative models into the existing architecture, provided they

[3]MovieLens is a web-based recommender system and virtual community that recommends movies for its users to watch, based on their film preferences using collaborative filtering of members' movie ratings and movie reviews. (HARPER; KONSTAN, 2015). Download: https://grouplens.org/datasets/movielens/100k/

[4]Python is a versatile, high-level programming language known for its easy readability and broad applicability across different technology domains.

interface correctly with the feature engineering and model selector modules. This platform also provides functionality for users to view a list of previously watched movies and rate unseen movies. These user interactions are captured and fed back into the system, enhancing the recommendation process through a continuous learning loop.

A critical component of the system's architecture is the implementation of the Root Mean Square Error (RMSE) metric, which is used to monitor the accuracy of the recommendations. If the RMSE rises above a predefined threshold of 0.3, the system dynamically switches between the Keras and KNN models depending on which is more effective under current conditions. This adaptability ensures that the system remains responsive to changes in user behavior and preferences.

The system is developed using the SciPy ecosystem, which includes prominent open-source packages such as Scikit-learn, Pandas, and Numpy. These tools are well-regarded in the scientific and engineering communities for their robust capabilities in handling complex data manipulations and mathematical operations. The web application, responsible for interacting with users, is built using Flask. This framework manages user requests, serves the recommendation results, and integrates user feedback into the system. For the machine learning component, Keras is utilized to build and train the neural network models that generate these movie recommendations. The choice of Python and its associated libraries for this project is driven by their widespread adoption and proven effectiveness in machine learning applications, as highlighted in (RASCHKA; MIRJALILI, 2017).

The architecture is schematically outlined as follows:

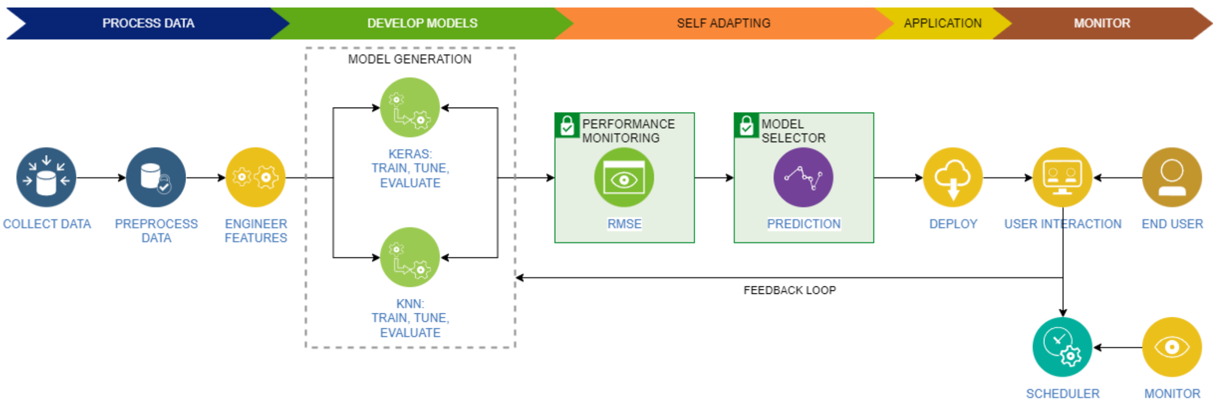


Figure 4.1: Architecture overview of CineMatch.

The components diagram presented in Figure 4.2 offers a more detailed view, incorporating the explanations discussed above. This visualization further elucidates the intricate interactions and functionalities of each system component:

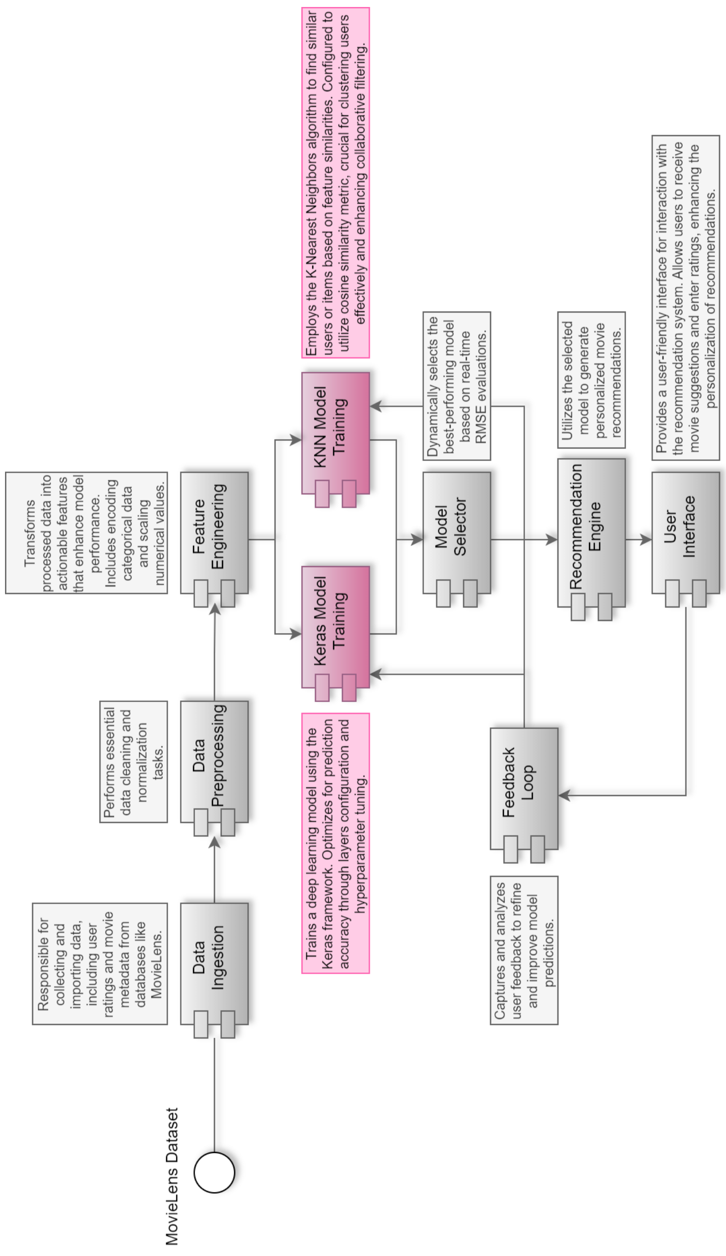

MovieLens Dataset
Data Ingestion
Responsible for collecting and importing data, including user ratings and movie metadata from databases like MovieLens.
Data Preprocessing
Performs essential data cleaning and normalization tasks.
Feature Engineering
Transforms processed data into actionable features that enhance model performance. Includes encoding categorical data and scaling numerical values.
KNN Model Training
Employs the K-Nearest Neighbors algorithm to find similar users or items based on feature similarities. Configured to utilize cosine similarity metric, crucial for clustering users effectively and enhancing collaborative filtering.
Keras Model Training
Trains a deep learning model using the Keras framework. Optimizes for prediction accuracy through layers configuration and hyperparameter tuning.
Model Selector
Dynamically selects the best-performing model based on real-time RMSE evaluations.
Recommendation Engine
Utilizes the selected model to generate personalized movie recommendations.
User Interface
Provides a user-friendly interface for interaction with the recommendation system. Allows users to receive movie suggestions and enter ratings, enhancing the personalization of recommendations.
Feedback Loop
Captures and analyzes user feedback to refine and improve model predictions.

Figure 4.2: Component Diagram of CineMatch.

The diagram emphasizes the interconnectedness of the system's components, from initial data ingestion from the MovieLens dataset through to the final user interaction with the web application. It visually breaks down the roles and relationships between the preprocessing tasks, model training phases, and the dynamic model selection process, facilitated by real-time RMSE evaluations. Each component's function is delineated, showing how data moves through the system to produce personalized recommendations. The feedback loop's integration is also depicted, underlining its role in continuously refining the models based on user interactions. This schematic aids in understanding how the diverse elements of CineMatch coalesce to maintain system efficiency and adaptability, ensuring the architecture can effectively respond to evolving user needs and data inputs.

By integrating these technologies and methodologies, the architecture of the recommendation system is designed not only to fulfill the immediate needs of movie recommendation, but also to adapt and evolve in response to new data and user feedback, ensuring its long-term effectiveness and relevance.

4.2 Database and Grouping

The dataset comprises three primary files: *data*, containing fields such as user ID, item ID, rating, and timestamp; *genre*, which lists all movie genres; and *item*, which provides detailed information about each movie.

For preprocessing tailored to the Keras framework, a crucial initial step involved normalizing the movie IDs. This normalization entailed adjusting the IDs to start at 0 and increment automatically, which was essential for matrix operations and as model input due to the original dataset's non-sequential ID arrangement. These adjusted indices were then appended to the original dataset in a new column. Subsequently, columns deemed unnecessary were dropped to streamline the processing workflow.

The movie genre data was also structured into a binary format to facilitate its use in the KNN model, preparing it for integration with user ratings data. This preprocessing is critical for setting up the data for the next steps in model training.

4.3 Model generation and training

This section delves into the setup and execution of two distinct models: Keras and K-Nearest Neighbors (KNN), each tailored to enhance the accuracy of predicting user preferences uniquely.

4.3.1 Keras

In the Keras model setup, we first extract user and movie identifiers from the dataset. To mitigate sequence bias and ensure the model does not overfit early in the sequence of data, we shuffle the data thoroughly before splitting. The dataset is divided into an 80% training set and a 20% test set, allowing for robust evaluation of the model's predictive accuracy under different conditions.

During the training phase, the Keras model learns to minimize the loss function, adjusting its weights to better predict movie ratings based on user-movie pairs. This phase includes rigorous validation against the test set, crucial for monitoring the model's performance and avoiding overfitting, ensuring that the model generalizes well on new, unseen data.

4.3.2 K-Nearest Neighbors (KNN)

For the K-Nearest Neighbors model, we utilize the structured movie genre data combined with user ratings, which were preprocessed as described in the ”Database and Grouping” section. By transforming movie genres into a binary format and merging them with user ratings, we create a detailed dataset that serves as the input for the NearestNeighbors algorithm, applying the cosine similarity metric with a brute force approach. This method is particularly effective for identifying clusters of users with similar tastes in movies, which is critical for generating personalized movie recommendations.

The efficacy of the KNN model in grouping user preferences and thereby enhancing the accuracy of recommendations is demonstrated through analytical visualizations. For instance, Figure 4.3 showcases how effectively the KNN model clusters movie preferences by user groups, providing a visual validation of the model's capability to cater to individual tastes based on aggregated user data.

4.4 Feedback Loop and Recommendations

With the models previously developed, integrating new data from incoming users is straightforward: the data is input into the models to identify the appropriate clusters,

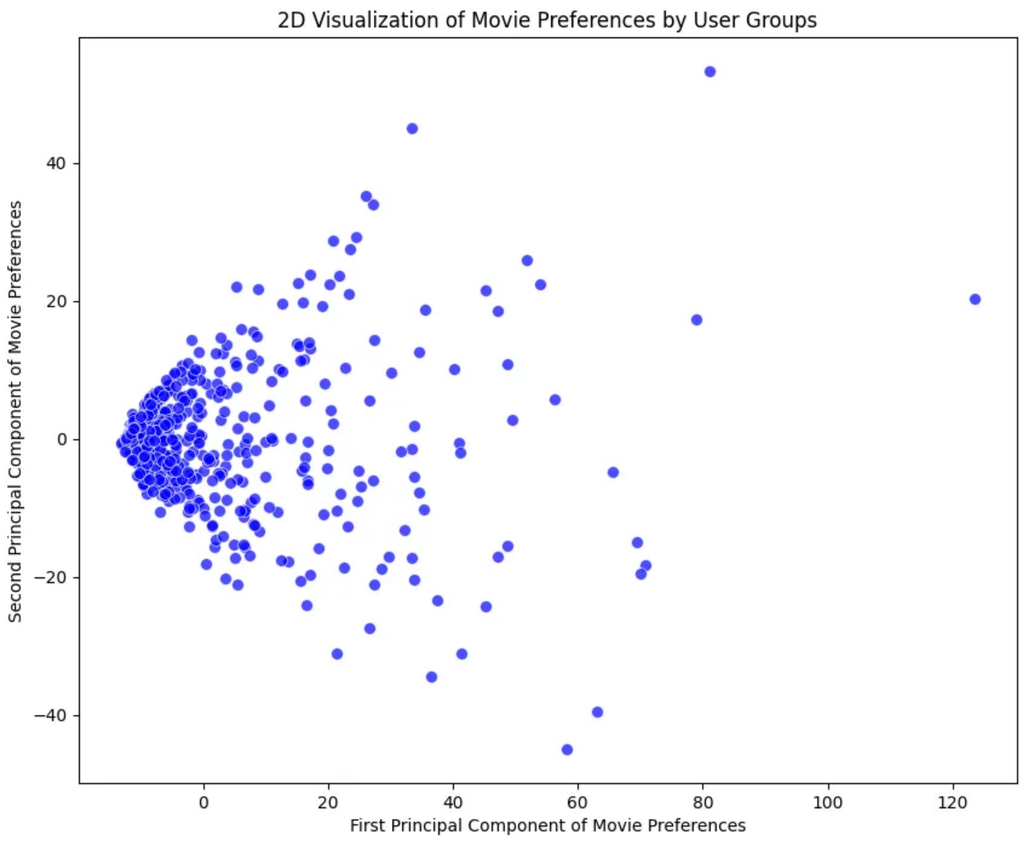


Figure 4.3: 2D visualization of movie preferences by user groups, demonstrating the clustering effect for user ID 104.

which then inform personalized recommendations. To efficiently integrate new users into the system, they are initially presented with a questionnaire designed to gather essential information. This preliminary data is sufficient to place them into a relevant cluster, predefined by the behaviors and preferences of existing users. This strategy ensures that even users with minimal interaction history receive recommendations that are likely to resonate with their tastes.

As users engage with the recommended movies, their interactions are captured as critical inputs for the system's feedback loop. The recommendation models are periodically retrained, incorporating the latest user interactions to refine and adjust the predictive algorithms. Consequently, user clusters may be updated based on this new data, ensuring that the clusters remain reflective of current user behaviors and preferences. This dynamic updating mechanism allows the system to consistently deliver more precise and personalized recommendations, adapting to the evolving tastes and feedback of its user base.

4.5 Self-Adaptation Mechanism

CineMatch employs a self-adaptation mechanism that dynamically selects the most suitable recommendation model based on real-time performance metrics, ensuring optimal accuracy and user satisfaction. The core of this mechanism revolves around the Root Mean Square Error (RMSE) metric, which is calculated for the Keras model's predictions. RMSE is a widely used measure to assess the accuracy of a predictive model, indicating how closely the predicted ratings match the actual ratings.

The self-adaptation process begins when the system receives input from a user. This input is processed and fed into both the Keras and K-nearest neighbors models simultaneously. Each model generates a set of recommendations based on the user's profile and historical data. However, the system only returns the recommendations from one model, selected based on the calculated RMSE for the Keras model.

If the RMSE for the Keras model is below a predefined threshold of 0.3, it indicates that the model's predictions are highly accurate. In such cases, the recommendations generated by the Keras model are presented to the user. Conversely, if the RMSE exceeds the 0.3 threshold, it suggests that the Keras model's predictions are less reliable. Therefore, the system switches to the KNN model, leveraging its collaborative filtering capabilities to provide more accurate recommendations.

This dual-model approach ensures that the system can adapt to different user scenarios effectively. The Keras model is well-suited for users with a substantial amount of historical data, as it can capture complex patterns and preferences through deep learning. On the other hand, the KNN model is advantageous for new users or those with limited historical data, as it relies on similarity-based techniques to make predictions.

4.6 Chapter Final Remarks

This chapter has presented a comprehensive overview of the architecture behind the proposed self-adaptive movie recommendation system. The application of hybrid machine learning techniques contributes to solving real-world problems, enhancing accuracy, personalization, effectiveness, and data volume scalability. This architecture not only addresses current limitations in existing systems but also opens avenues for future innovations in personalized user experiences. The next chapter will provide a comprehensive overview of the evaluation metrics and architectural features that underscore the robustness of the CineMatch system.

5 Evaluation and Metrics

This chapter sets the stage for a detailed discussion of the metrics used for evaluation and the architectural features that support system effectiveness and adaptability.

5.1 Root Mean Square Error (RMSE)

Root Mean Square Error (RMSE) is a standard statistical measure used extensively to gauge the accuracy of predicted values against the actual observed values. In the context of CineMatch, RMSE quantifies the average magnitude of the prediction error, which represents the difference between the values predicted by the model and the actual values. A lower RMSE indicates that the model's predictions are closer to the actual ratings, reflecting the higher accuracy and effectiveness of the model. Conversely, a higher RMSE signals greater prediction errors, suggesting areas where the model may need refinement or adjustment.

5.2 System Evaluation and Self-Adaptation

Our architecture is designed to be self-adaptive, continuously evaluating its effectiveness and autonomously making decisions to optimize recommendation accuracy based on real-time data. This self-adaptation is primarily driven by ongoing assessments of model effectiveness, using RMSE as a key metric.

The system incorporates a self-monitoring mechanism that continuously tracks the RMSE scores of both the Keras and K-Nearest Neighbors (KNN) models. This tracking is automated, and decision-making algorithms use these metrics to determine the optimal model for recommendation at any given time. When the RMSE of the Keras model exceeds a predefined threshold, indicating a decrease in prediction accuracy, the system autonomously switches to the KNN model. This switch is not permanent; the system is designed to re-evaluate the performance of the Keras model after integrating new data and re-training, providing a dynamic and responsive recommendation environment.

Autonomous re-training cycles are a cornerstone of the system's self-adaptive features. Re-training is triggered automatically by the system based on certain criteria such as a significant deviation in RMSE scores or after a fixed period has elapsed. During these cycles, both models are updated with new user interaction data, which might also lead to recalibrations of user clusters in the KNN model or significant model parameter adjustments in the Keras model.

5.3 Feasibility Study

The developed web application[5], using the MovieLens dataset, serves as a platform to test and refine the system's capabilities under a variety of user scenarios. It features a straightforward interface where users can input an ID from the MovieLens dataset range. Based on real-time RMSE metrics, it dynamically chooses between the Keras and KNN models, generating tailored recommendations that reflect the user's profile. This approach not only shows the efficacy of each model under different conditions but also highlights its adaptability.

The interface enables users to discover and rate movies they haven't watched but are likely to enjoy, according to the system's predictions. Users can also update their previous ratings from their list of watched movies. This continuous feedback is crucial for the system to use in subsequent re-training cycles to refine its predictions and enhance accuracy. Although the feasibility study uses a 5-star rating scale, the architecture is versatile and can easily adapt to the 3-rating scale ("disliked", "liked", "loved") recently adopted by many platforms, ensuring broader applicability and relevance.

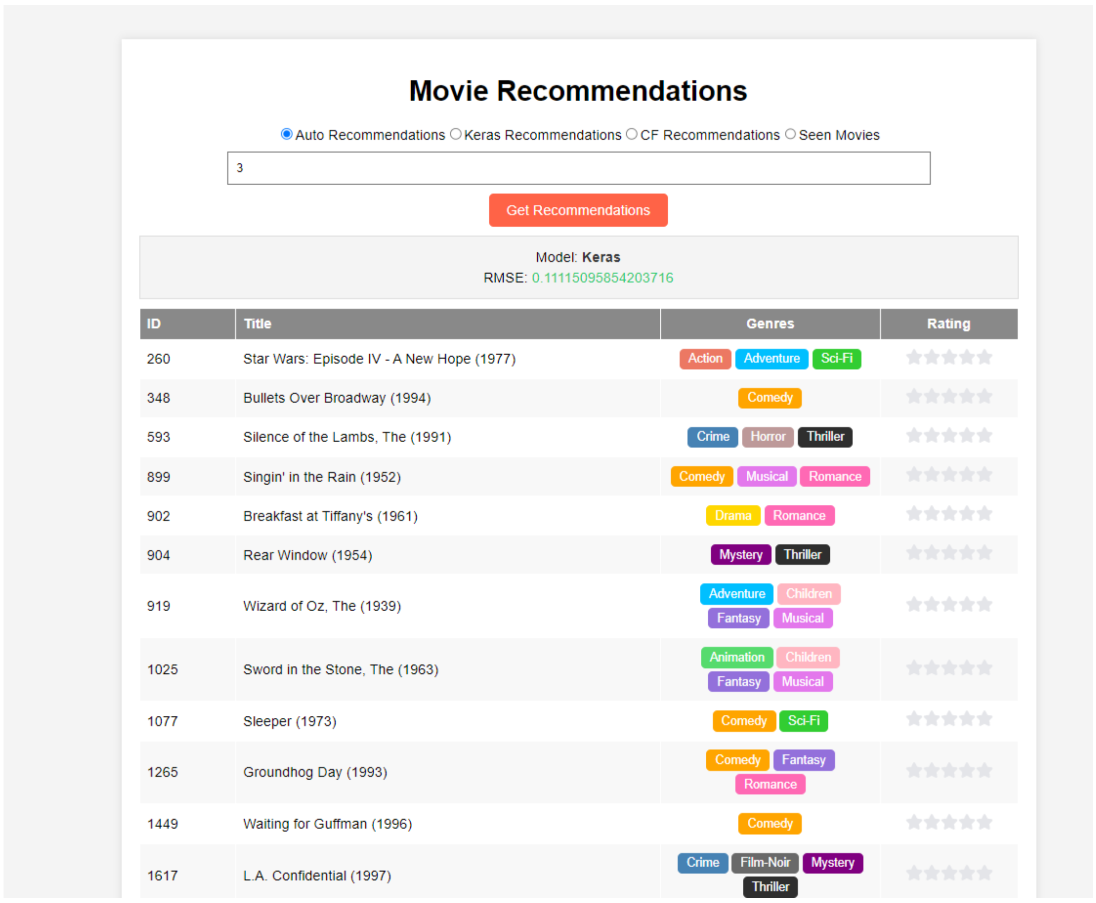
The application not only tests the accuracy of the models in a controlled environment but also simulates real-world scenarios where user satisfaction depends on recommendation efficiency.

Additionally, an initial version of the implementation allows users to manually select between the Keras and KNN models. This capability aids in a direct comparison of the models and also provides deeper insights into their operational effectiveness, ensuring

[5]Code available on GitHub: https://github.com/vitorianatalia/self-adaptive-architecture

it consistently meets the evolving preferences of its users and maintains high predictive accuracy.

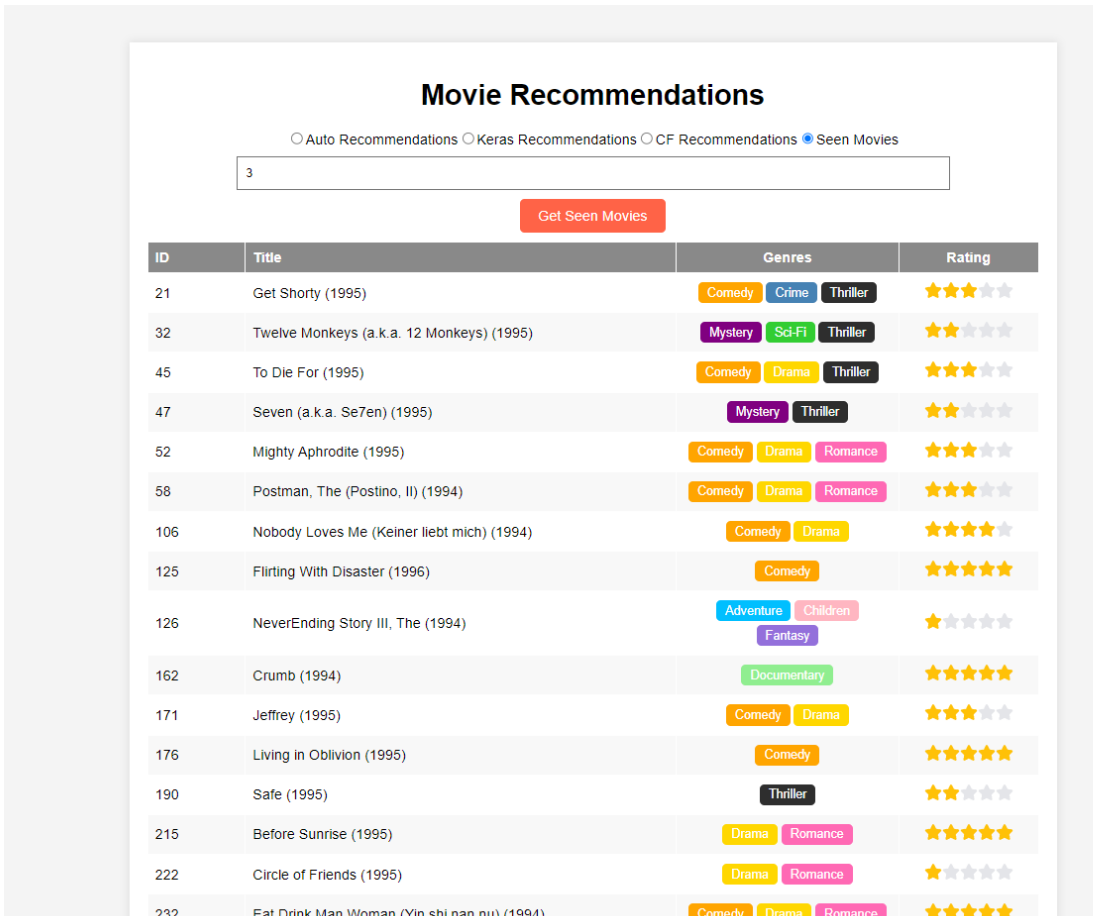
Below are screenshots from the application, illustrating the interface and the type of results it delivers for a few user IDs:


Movie Recommendations
[✓ Auto Recommendations] Keras Recommendations CF Recommendations Seen Movies
3
Get Recommendations

Model: **Keras**
RMSE: 0.11115095854203716

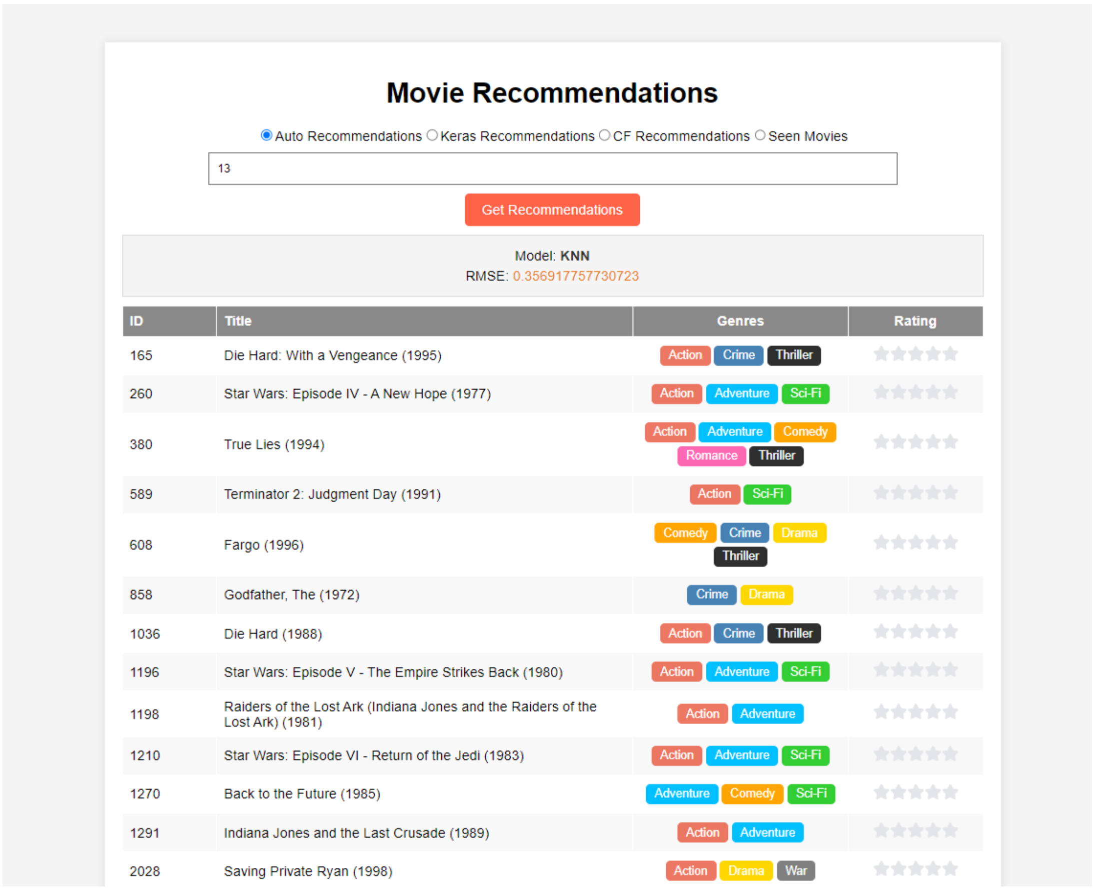
ID	Title	Genres	Rating
260	Star Wars: Episode IV - A New Hope (1977)	Action Adventure Sci-Fi	☆☆☆☆☆
348	Bullets Over Broadway (1994)	Comedy	☆☆☆☆☆
593	Silence of the Lambs, The (1991)	Crime Horror Thriller	☆☆☆☆☆
899	Singin' in the Rain (1952)	Comedy Musical Romance	☆☆☆☆☆
902	Breakfast at Tiffany's (1961)	Drama Romance	☆☆☆☆☆
904	Rear Window (1954)	Mystery Thriller	☆☆☆☆☆
919	Wizard of Oz, The (1939)	Adventure Children Fantasy Musical	☆☆☆☆☆
1025	Sword in the Stone, The (1963)	Animation Children Fantasy Musical	☆☆☆☆☆
1077	Sleeper (1973)	Comedy Sci-Fi	☆☆☆☆☆
1265	Groundhog Day (1993)	Comedy Fantasy Romance	☆☆☆☆☆
1449	Waiting for Guffman (1996)	Comedy	☆☆☆☆☆
1617	L.A. Confidential (1997)	Crime Film-Noir Mystery Thriller	☆☆☆☆☆

Figure 5.1: Preview of recommendations using the Keras model for User ID 3.


Movie Recommendations
Auto Recommendations Keras Recommendations CF Recommendations [✓ Seen Movies]
3
Get Seen Movies

ID	Title	Genres	Rating
21	Get Shorty (1995)	Comedy Crime Thriller	★★★☆☆
32	Twelve Monkeys (a.k.a. 12 Monkeys) (1995)	Mystery Sci-Fi Thriller	★★☆☆☆
45	To Die For (1995)	Comedy Drama Thriller	★★★☆☆
47	Seven (a.k.a. Se7en) (1995)	Mystery Thriller	★★☆☆☆
52	Mighty Aphrodite (1995)	Comedy Drama Romance	★★★☆☆
58	Postman, The (Postino, Il) (1994)	Comedy Drama Romance	★★★☆☆
106	Nobody Loves Me (Keiner liebt mich) (1994)	Comedy Drama	★★★★☆
125	Flirting With Disaster (1996)	Comedy	★★★★★
126	NeverEnding Story III, The (1994)	Adventure Children Fantasy	★☆☆☆☆
162	Crumb (1994)	Documentary	★★★★★
171	Jeffrey (1995)	Comedy Drama	★★★☆☆
176	Living in Oblivion (1995)	Comedy	★★★★★
190	Safe (1995)	Thriller	★★☆☆☆
215	Before Sunrise (1995)	Drama Romance	★★★★★
222	Circle of Friends (1995)	Drama Romance	★☆☆☆☆
232	Eat Drink Man Woman (Yin shi nan nu) (1994)	Comedy Drama Romance	★★★★★

Figure 5.2: Preview of rated movies by User ID 3.



Movie Recommendations

[✓ Auto Recommendations] Keras Recommendations CF Recommendations Seen Movies

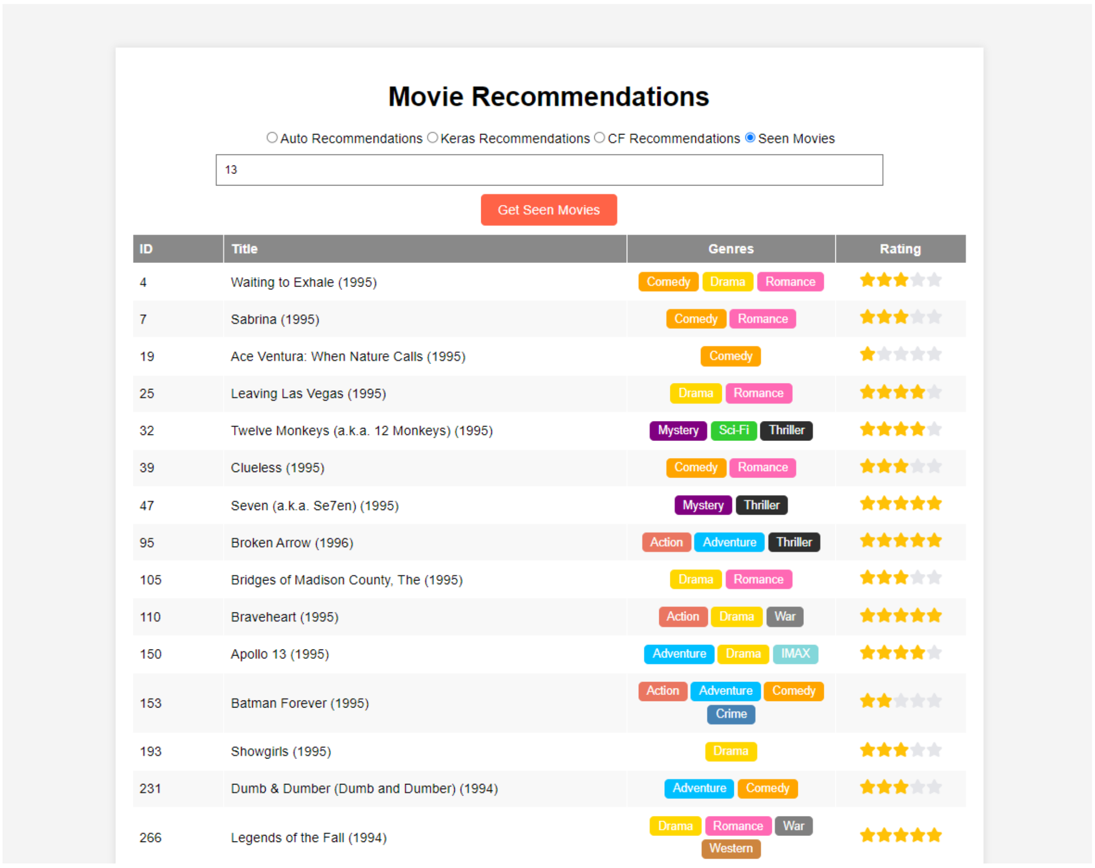
13

Get Recommendations

Model: **KNN**
RMSE: 0.356917757730723

ID	Title	Genres	Rating
165	Die Hard: With a Vengeance (1995)	Action Crime Thriller	☆☆☆☆☆
260	Star Wars: Episode IV - A New Hope (1977)	Action Adventure Sci-Fi	☆☆☆☆☆
380	True Lies (1994)	Action Adventure Comedy Romance Thriller	☆☆☆☆☆
589	Terminator 2: Judgment Day (1991)	Action Sci-Fi	☆☆☆☆☆
608	Fargo (1996)	Comedy Crime Drama Thriller	☆☆☆☆☆
858	Godfather, The (1972)	Crime Drama	☆☆☆☆☆
1036	Die Hard (1988)	Action Crime Thriller	☆☆☆☆☆
1196	Star Wars: Episode V - The Empire Strikes Back (1980)	Action Adventure Sci-Fi	☆☆☆☆☆
1198	Raiders of the Lost Ark (Indiana Jones and the Raiders of the Lost Ark) (1981)	Action Adventure	☆☆☆☆☆
1210	Star Wars: Episode VI - Return of the Jedi (1983)	Action Adventure Sci-Fi	☆☆☆☆☆
1270	Back to the Future (1985)	Adventure Comedy Sci-Fi	☆☆☆☆☆
1291	Indiana Jones and the Last Crusade (1989)	Action Adventure	☆☆☆☆☆
2028	Saving Private Ryan (1998)	Action Drama War	☆☆☆☆☆

Figure 5.3: Preview of recommendations using the KNN model for User ID 13.



Movie Recommendations

Auto Recommendations Keras Recommendations CF Recommendations [✓ Seen Movies]

13

Get Seen Movies

ID	Title	Genres	Rating
4	Waiting to Exhale (1995)	Comedy Drama Romance	★★★☆☆
7	Sabrina (1995)	Comedy Romance	★★★☆☆
19	Ace Ventura: When Nature Calls (1995)	Comedy	★☆☆☆☆
25	Leaving Las Vegas (1995)	Drama Romance	★★★★☆
32	Twelve Monkeys (a.k.a. 12 Monkeys) (1995)	Mystery Sci-Fi Thriller	★★★★☆
39	Clueless (1995)	Comedy Romance	★★★☆☆
47	Seven (a.k.a. Se7en) (1995)	Mystery Thriller	★★★★★
95	Broken Arrow (1996)	Action Adventure Thriller	★★★★★
105	Bridges of Madison County, The (1995)	Drama Romance	★★★☆☆
110	Braveheart (1995)	Action Drama War	★★★★★
150	Apollo 13 (1995)	Adventure Drama IMAX	★★★★☆
153	Batman Forever (1995)	Action Adventure Comedy Crime	★★☆☆☆
193	Showgirls (1995)	Drama	★★★☆☆
231	Dumb & Dumber (Dumb and Dumber) (1994)	Adventure Comedy	★★★☆☆
266	Legends of the Fall (1994)	Drama Romance War Western	★★★★★

Figure 5.4: Preview of rated movies by User ID 13.

As depicted in the example of Figure 5.1, the RMSE for User ID 3 using Keras recommendations was below the threshold of 0.3. This low RMSE indicates that the predictions are close to the actual ratings, showcasing the high accuracy and effectiveness of the Keras model for this user. Consequently, the list of recommended movies, derived from the Keras model, aligns closely with the user's preferences. Figure 5.2 displays the list of movies already watched by User ID 3, along with their ratings, providing insight into the user's preferences and the model's performance. Although it is not possible to identify all preference patterns from a small sample, it is observable that this user tends to favor genres such as comedy, drama, and romance. These preferences were also considered in the generated recommendations, demonstrating the Keras model's ability to identify and suggest content aligned with the user's viewing history. The last column in the interface references the 5-star scale present in the application, where users can add new ratings or edit their old ones from the seen list.

In contrast, Figure 5.3 shows the application of the KNN model for User ID 13, triggered by the Keras model's RMSE surpassing the 0.3 threshold. This higher RMSE suggests notable prediction errors, indicating a need for model adjustments for this particular user. Figure 5.4 presents the movies that User ID 13 has watched, helping to contextualize the user's tastes and the subsequent model recommendations. The varied taste of this user includes a wider range of genres and film styles, which partly explains the Keras model's difficulty in identifying clear patterns, resulting in a higher RMSE. The KNN model, on the other hand, is capable of identifying clusters of users with similar preferences, offering recommendations that may be better received by users with eclectic tastes. Switching to KNN, a method based on collaborative filtering, likely offers more accurate recommendations by comparing user similarities.

5.4 Architectural Features

CineMatch is engineered for scalability, achieved through a modular architecture where components such as data processing, model training, and recommendation engines can independently scale according to demand. This modularity allows for the seamless integration of additional models and techniques into the existing framework as needed, ensuring that the system remains robust and responsive.

With cloud-based services to dynamically allocate resources, the system is optimized for performance during periods of peak load. This dynamic resource management is essential for scaling data volume, maintaining rapid response times, and efficient data handling, irrespective of user traffic.

Adaptability is a fundamental aspect of CineMatch's design. The system is configured to dynamically switch between modeling techniques, based on real-time performance metrics. This flexibility enables CineMatch to quickly adapt to changes in user preferences and market conditions, ensuring that recommendations stay precise and relevant. By continuously monitoring performance and user engagement, the system leverages these insights to fine-tune its algorithms, thereby enhancing the accuracy of its predictions.

5.5 Chapter Final Remarks

In summary, the evaluation strategies and architectural innovations demonstrate that CineMatch effectively harnesses both deep learning and similarity-based techniques, ensuring high recommendation accuracy. Its design, though not yet tested with real users, is well-prepared for real-world applications. Theoretical evaluations suggest it is capable of meeting practical demands effectively, as the foundational technologies employed are aligned with industry best practices, providing a robust and modular framework for recommendations. However, the absence of real-world data could potentially reveal uncertainties about how the system will handle actual user behavior patterns and preferences, and might necessitate further tuning and adjustments to meet market expectations. The following chapter will provide a detailed conclusion of this research, summarizing the key findings and discussing directions for future research.

6 Conclusion

This research has effectively demonstrated the feasibility of implementing CineMatch, a self-adaptive movie recommendation system combining Keras and KNN models. By dynamically adapting to changes in user data and preferences, the system enhances prediction accuracy and effectiveness. Its modular design and employed technologies focus on continuous monitoring and autonomous adjustments based on RMSE metrics, facilitating scalable data handling and demonstrating practical applicability.

CineMatch has risen to meet significant challenges inherent in modern recommendation systems, including adapting to the evolving landscape of user behaviors and addressing the initial hurdles presented by new users. The integration of a feedback loop allows the architecture to continuously evolve, leveraging learned data to refine and optimize its recommendation strategies effectively.

However, since the primary focus of this research was to establish a foundational self-adaptive movie recommendation architecture, the system faces limitations such as handling non-uniform data distribution and the computational intensity of real-time model training. Additionally, the functional results of the recommendation system have not yet been validated with real-user feedback, which is crucial for confirming its success and usability in practical scenarios.

Further research could explore integrating more detailed user feedback mechanisms and applying federated learning approaches to streamline data processing. It's crucial to conduct field research with real users to test the system in live environments. This would offer valuable insights into its performance and user satisfaction in practical settings. Such studies are essential not only for refining the system's algorithms based on direct user interactions but also for uncovering user behaviors and preferences that emerge during real-world use. Additionally, exploring testing the architecture in diverse domains such as health, economy, or audio streaming services, due to their similarities, could provide valuable insights.

Moreover, developing new methods to assess user satisfaction and implementing a more advanced adaptive strategy—like a detailed component-level self-adaptation system where each part of the recommendation architecture can adjust independently based on real-time data—would be beneficial. Investigating the possibility of leveraging initial social media connections to skip initial questionnaires and understand user preferences could also be explored. Furthermore, exploring unused models could reveal more efficient approaches, enhancing the recommendation system's performance and adaptability.

By pushing the boundaries of current recommendation technologies, this research contributes to the fields of machine learning, data science, and software engineering, offering a blueprint for building more accurate and adaptive recommendation systems.

Bibliography

ABBAS, N.; ANDERSSON, J. *ASPLe-A Methodology to Develop Self-Adaptive Software Systems with Reuse*. 2018.

AGGARWAL, M. et al. A recommendation system to envision movie ranking using collaborative filtering. In: *2023 International Conference on Artificial Intelligence and Smart Communication (AISC)*. [S.l.: s.n.], 2023. p. 755–758.

AWAN, M. J. et al. A recommendation engine for predicting movie ratings using a big data approach. *Electronics*, v. 10, n. 10, 2021. ISSN 2079-9292. Disponível em: ⟨https://www.mdpi.com/2079-9292/10/10/1215⟩.

BAHRANI, P. et al. A new improved knn-based recommender system. *The Journal of Supercomputing*, v. 80, n. 1, p. 800–834, 2024. ISSN 1573-0484. Disponível em: ⟨https://doi.org/10.1007/s11227-023-05447-1⟩.

BISWAS, D.; HASAN, K. M. A. Mfid: An improved matrix factorization model for incremental data. In: *2023 26th International Conference on Computer and Information Technology (ICCIT)*. [S.l.: s.n.], 2023. p. 1–6.

BRANDÃO, H.; PINTO, G.; SANTOS, T. Application of k-nearest neighbors (knn) method to undrained shear strength prediction of bauxite tailing. *Tailings*, 2022.

BURKE, R. Hybrid recommender systems: Survey and experiments. *User modeling and user-adapted interaction*, Springer, v. 12, n. 4, p. 331–370, 2002.

CHANG, S. et al. Streaming recommender systems. *CoRR*, abs/1607.06182, 2016. Disponível em: ⟨http://arxiv.org/abs/1607.06182⟩.

CHENG, B. H. et al. Software engineering for self-adaptive systems: A research road map. *Software Engineering for Self-Adaptive Systems*, Springer, p. 1–26, 2009.

DOOMS, S. Dynamic generation of personalized hybrid recommender systems. In: . [S.l.: s.n.], 2013.

GARLAN, D. Foreword. In: WEYNS, D. (Ed.). *An Introduction to Self-Adaptive Systems: A Contemporary Software Engineering Perspective*. New Jersey, USA: John Wiley & Sons Ltd, 2021. Foreword.

HARPER, F. M.; KONSTAN, J. A. The movielens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, v. 5, n. 4, p. Article 19, Dec 2015.

HASAN, M. R.; JHA, A. K.; LIU, Y. Excessive use of online video streaming services: Impact of recommender system use, psychological factors, and motives. *Computers in Human Behavior*, v. 80, p. 220–228, 2018. ISSN 0747-5632. Disponível em: ⟨https://www.sciencedirect.com/science/article/pii/S0747563217306581⟩.

LOPS, P.; GEMMIS, M. de; SEMERARO, G. Content-based recommender systems: State of the art and trends. In: _____. *Recommender Systems Handbook*. Boston, MA: Springer US, 2011. p. 73–105. ISBN 978-0-387-85820-3. Disponível em: ⟨https://doi.org/10.1007/978-0-387-85820-3_3⟩.

NADER, K. Dating through the filters. *Social Philosophy and Policy*, v. 37, p. 237–248, 01 2020.

PETERSON, L. E. K-nearest neighbor. *Scholarpedia*, v. 4, n. 2, p. 1883, 2009. Revision #137311.

PRAJNA, K. B. et al. Implementation of a hybrid recommendation system for movies. In: *2022 IEEE 2nd Mysore Sub Section International Conference (MysuruCon)*. [S.l.: s.n.], 2022. p. 1–7.

RASCHKA, S.; MIRJALILI, V. *Python Machine Learning: Machine Learning and Deep Learning with Python, scikit-learn, and TensorFlow, 2nd Edition*. 2nd. ed. [S.l.]: Packt Publishing, 2017. ISBN 1787125939.

ROY, D.; DUTTA, M. A systematic review and research perspective on recommender systems. *Journal of Big Data*, v. 9, n. 1, p. 59, May 2022. ISSN 2196-1115. Disponível em: ⟨https://doi.org/10.1186/s40537-022-00592-5⟩.

SALEHIE, M.; TAHVILDARI, L. Self-adaptive software: Landscape and research challenges. *ACM Trans. Auton. Adapt. Syst.*, Association for Computing Machinery, New York, NY, USA, v. 4, n. 2, may 2009. ISSN 1556-4665. Disponível em: ⟨https://doi.org/10.1145/1516533.1516538⟩.

SUNNY, B. K. et al. Implementation of a self-adaptive real time recommendation system using spark machine learning libraries. In: *2017 IEEE International Conference on Signal Processing, Informatics, Communication and Energy Systems (SPICES)*. [S.l.: s.n.], 2017. p. 1–7.

XIA, F. et al. Mobile multimedia recommendation in smart communities: A survey. *IEEE Access*, v. 1, p. 606–624, 2013.

YAPıCı, M. M.; TOPALOğLU, N. Performance comparison of deep learning frameworks. *Computers and Informatics*, Adem TEKEREK, v. 1, n. 1, p. 1–11, 2021.