Optimizing Movie Recommendation System: A Comparative Study Using Deep Collaborative Filtering

Zhe Luo

A Thesis Submitted in Partial Fulfilment

of the requirements for the

Degree of

Master of Science in Data Analytics

Diagram

Description automatically generated with medium confidence

Novermber 2024

Supervisor: Dr Kislay Raj

Optimizing Movie Recommendation System: A Comparative Study Using Deep Collaborative Filtering

Zhe Luo

2024

Master(120 Credits)

Master of Science in Data Analytics

**Abstract**

This research focuses on the development and optimization of recommendation systems using advanced deep learning technologies. By comparing traditional collaborative filtering methods (e.g., matrix factorization (Yehuda Koren, 2009)) with modern models such as Hybrid Matrix Factorization with Self-Attention (HMFA), Graph Neural Networks (Franco Scarselli, December 2008) (GNNs), and Transformers (Ashish Vaswani, 2017), the research explores the advantages of these techniques in improving user-item interaction modeling and system performance

The key methodologies include deep collaborative filtering, which enhances traditional approaches by mapping users and items into low-dimensional latent spaces and utilizing non-linear neural networks to capture complex relationships. The Hybrid Matrix Factorization with Self-Attention mechanism improves upon basic matrix factorization by dynamically adjusting feature weights, allowing for better capture of user preferences and movie characteristics. GNNs leverage graph structures to capture multi-level relationships, while Transformers excel in modeling temporal dependencies and long-range interactions, which are particularly useful in recommendation systems.

This study used the PyTorch framework to construct models, integrating Automatic Mixed Precision (AMP) (Paulius Micikevicius, 2018)with learning rate scheduling through CosineAnnealingLR to optimize training efficiency and convergence. Additionally, Optuna (Takuya Akiba, 2019)'s Bayesian optimization was employed to tune hyperparameters, ensuring efficient and precise parameter selection for large-scale models.

Moreover, SHAP (Shapley Additive Explanations) (Scott M. Lundberg, 2017)was used to explain the contribution of individual features to predictions, enhancing model interpretability. This analysis provided clear insights into how features impact model performance, increasing the transparency of complex models. The integration of these advanced techniques demonstrates the potential to build more accurate, robust, and scalable recommendation systems.

This study showcases the ability of these cutting-edge methods to create a Movie recommendation systems that are not only more efficient but also capable of providing personalized and transparent user experiences.

**Key Words**: Deep Collaborative Filtering ,Graph Neural Networks (GNNs)Transformers ,Bayesian Optimization ,SHAP

Table of Contents

[1 Introduction 5](#_Toc178355632)

[1.1 Data Science as a Service at Framed 5](#_Toc178355633)

[1.2 General Project Aim 5](#_Toc178355634)

[1.3 Practical Feasibility 6](#_Toc178355635)

[1.4 Objectives: 7](#_Toc178355636)

[1.5 Research Questions 8](#_Toc178355637)

[2 Literature Review 9](#_Toc178355638)

[2.1 Literature Review Introduction 9](#_Toc178355639)

[2.2 The Development and Classification of Movie Recommendation Systems 10](#_Toc178355640)

[2.3 Common Techniques and Model Architectures in the Field of Recommendation Systems 13](#_Toc178355641)

[2.4 Literature Review Conclusion 19](#_Toc178355642)

[3 Methodology 22](#_Toc178355643)

[3,1 Research Design 22](#_Toc178355644)

[3.1.1 Research Type 22](#_Toc178355645)

[3.1.2 Research Methods 22](#_Toc178355646)

[3.1.3 Research Process Framework 23](#_Toc178355647)

[3.2 Data Collection Methods 24](#_Toc178355648)

[3.2.1 Data Types 24](#_Toc178355649)

[3.2.2 Data Collection Process 25](#_Toc178355650)

[3.3 Ethical Considerations 26](#_Toc178355651)

[3.3.1 Ethical Issues for Expert Interviews 26](#_Toc178355652)

[3.3.2 Ethical Issues for the MovieLens 1M Dataset 26](#_Toc178355653)

[3.4 Data Process 27](#_Toc178355654)

[3.4.1 Data Preparation 27](#_Toc178355655)

[3.4.2 Feature Engineering 27](#_Toc178355656)

[3.5 Model Architecture 27](#_Toc178355657)

[3.5.1 Deep Collaborative Filtering Model Framework Overview 28](#_Toc178355658)

[3.5.2 Overview of Matrix Factorization Models 29](#_Toc178355659)

[3.5.3 The structure of Hybrid Matrix Factorization with Self-Attention (HMFA) 30](#_Toc178355660)

[3.5.4 Mechanism Explanation of Graph Neural Networks (GNNs) 31](#_Toc178355661)

[3.5.5 Mechanism Explanation of Transformer 32](#_Toc178355662)

[3.6 Training and Tuning 33](#_Toc178355663)

[3.6.1 Model Building 33](#_Toc178355664)

[3.6.2 Optimizer Selection 34](#_Toc178355665)

[3.6.3 Acceleration and Resource Management 34](#_Toc178355666)

[3.6.4 Learning Rate Scheduling 34](#_Toc178355667)

[3.6.5 Overfitting Prevention 34](#_Toc178355668)

[3.6.6 Hyperparameter Tuning 35](#_Toc178355669)

[3.6.7 Model Interpretability Analysis 35](#_Toc178355670)

[3.7 Validity 35](#_Toc178355671)

[3.7.1 Validity of Expert Interviews 36](#_Toc178355672)

[3.7.2 Model Validity Evaluation 36](#_Toc178355673)

[3.7.3 Data Validity 36](#_Toc178355674)

[3.7.4 Model Selection and Evaluation 36](#_Toc178355675)

[3.8 Environment and Setup 37](#_Toc178355676)

[3.8.1 Programming Language and Framework 37](#_Toc178355677)

[3.8.2 Hardware Setup 37](#_Toc178355678)

[4 Results & Analysis 37](#_Toc178355679)

[4.1 Phase of Exploratory Data Analysis 38](#_Toc178355680)

[4.2 Phase of Matrix Factorization Model 39](#_Toc178355681)

[4.3 Phase of Hybird Matrix Factorization with Self-Attention Model 44](#_Toc178355682)

[4.4 Phase of Graph Neural Networks (GNN) Model 46](#_Toc178355683)

[4.5 Phase of Transformer 48](#_Toc178355684)

[4.6 The Development and Application of the Movie Recommendation System 58](#_Toc178355685)

[5 Conclusion 61](#_Toc178355686)

[5.1 Response to Experimental Question One 61](#_Toc178355687)

[5.2 Response to Experimental Question Two 62](#_Toc178355688)

[5.3 Limitations and Strengths of the Research 65](#_Toc178355689)

[5.4 Contribution and Future Research 66](#_Toc178355690)

[References 67](#_Toc178355691)

[Appendix 68](#_Toc178355692)

[1 Interview One 68](#_Toc178355693)

[2 Interview Two 73](#_Toc178355694)

[3 Interview Three 78](#_Toc178355695)

[4 Interview Four 82](#_Toc178355696)

# 1 Introduction

## 1.1 Data Science as a Service at Framed

In recent years, with the rapid development of big data and deep learning technologies, recommendation systems have become a key technology for enhancing user experience and enabling personalized services. This is especially true in the media consumption sector, such as movie recommendation systems, where these technologies are gaining increasing attention from both industry and academia. Traditional recommendation methods, such as collaborative filtering and content-based recommendations (Mohammad & Urolagin, 2022), have achieved some success in practice. However, these methods typically only handle surface-level features of users and items, often overlooking the complex user behavior patterns and latent relationships between items. As a result, despite their ability to provide basic recommendations, the performance of these traditional models is often limited by their capacity for data processing and feature analysis.

To overcome these limitations and further improve the quality and accuracy of recommendation systems, researchers and industry professionals have begun exploring the application of deep learning models. Deep learning techniques can leverage complex network structures to mine and learn deep features within the data, allowing for a better understanding of users' subtle preferences and the intricate relationships between items. This approach not only promises to significantly enhance user satisfaction but also increases user engagement and platform retention, bringing greater commercial value to businesses.

## 1.2 General Project Aim

This study aims to explore the application of deep learning models in movie recommendation systems, with a focus on analyzing the performance of these models during their construction and optimization. To achieve this, a mixed-methods approach combining qualitative and quantitative research is employed. The qualitative research involves semi-structured interviews with four experts from academia and industry, including associate professors, PhD researchers, and professionals with extensive practical experience. These interviews provided valuable insights into the current needs, challenges, technological trends, and effective strategies in recommendation systems, offering important guidance for model development. The quantitative research involves conducting experiments to compare various deep collaborative filtering models, using the publicly available MovieLens 1M dataset. This dataset contains 1 million ratings from 6,000 users on 4,000 movies, providing a rich source of data for evaluating the models.

This study explores the performance and effectiveness of traditional matrix factorization methods and modern deep collaborative filtering models, such as self-attention mechanisms and graph neural networks (GNN) (Franco Scarselli, December 2008), in movie recommendation tasks. As a classic and efficient recommendation method, matrix factorization is widely applied in both industry and academia. In this study, matrix factorization is used as the baseline model to evaluate its performance in movie recommendation systems, while also analyzing its potential limitations in complex data scenarios, particularly issues related to overfitting.

To further improve model performance, this study incorporates self-attention mechanisms with the aim of enhancing the model's ability to capture global information and thereby increase recommendation accuracy. Additionally, the study investigates the application of graph neural networks (GNN), which have advantages in modeling non-linear relationships, to further optimize the performance of the recommendation system. Transformer models, known for their exceptional performance in handling sequential data, are also included in the experiments to assess their ability to capture user behavior patterns. Through a comparative analysis of these models, this study aims to identify optimal model choices for recommendation tasks across varying data scales and complexities.

## 1.3 Practical Feasibility

The aim of this study is not only to compare the performance of different models but, more importantly, to explore their feasibility and limitations in practical applications through experimental validation and expert interviews. Future research could focus on further optimizing technical parameters, such as embedding dimensions, to better address overfitting issues and enhance the robustness of models. Additionally, with the rapid development of large language models (LLMs) (Yuyue Zhao, July 2024), researchers may explore their potential applications in recommendation systems, particularly in leveraging LLMs to better understand user preferences and contextual information, thereby improving recommendation accuracy and personalization.

Through the in-depth analysis and comparison conducted in this study, we hope to provide valuable insights for the future development of movie recommendation systems and serve as a bridge for translating theoretical research into practical applications.

## 1.4 Objectives:

1 Develop a traditional matrix factorization model, and utilize its optimized results as a baseline for comparison with subsequent models.

Construct a classic matrix factorization-based model to serve as the baseline for evaluating the performance of more advanced models. This model will provide a foundation for comparison, highlighting the advantages and limitations of traditional collaborative filtering techniques in movie recommendation systems.

2 Design and implement a series of deep collaborative filtering models, including hybrid approaches combining matrix factorization with self-attention mechanisms, graph neural networks (GNN) (Franco Scarselli, December 2008), and Transformer model (Ashish Vaswani, 2017)s  
Develop a range of advanced deep learning models that extend traditional collaborative filtering approaches. This will include hybrid models that incorporate self-attention mechanisms to capture global information, GNNs to model complex user-item interactions, and Transformer models known for their strength in sequence data processing.

3 Conduct a comparative analysis between deep collaborative filtering models (Abhishek Kumar Rai1, 25/04/2024) and the matrix factorization baseline model, evaluating performance metrics and discussing the results  
Perform a rigorous comparative study of the deep collaborative filtering models against the baseline matrix factorization model. Various evaluation metrics, such as MSE, overfitting rate, and execution time, will be used to assess the models. A detailed discussion of the findings will provide insights into the strengths and weaknesses of each approach.

4 Utilize the best-performing model as the core algorithm to develop a movie recommendation system.  
Based on the comparative analysis, select the model with the highest performance and robustness to serve as the core algorithm for the movie recommendation system. This system will be designed to deliver personalized and accurate recommendations, optimizing both user satisfaction and system efficiency.

## 1.5 Research Questions

Expert interviews form a core part of this study, aiming to gather insights from industry experts through qualitative research to explore the practical needs, technical challenges, and future trends in recommendation systems. The opinions of these experts will provide valuable practical guidance for model development and help us better understand the critical issues in recommendation systems. Therefore, the first key research question is: How can expert interviews help us better understand the direction of model development and the evaluation process?

Additionally, based on the background and objectives of this study, we investigate the performance differences between traditional collaborative filtering models (such as matrix factorization) and modern deep collaborative filtering models (such as self-attention mechanisms and graph neural networks) in movie recommendation systems. While traditional collaborative filtering methods are widely applied in recommendation systems, they face challenges such as overfitting when dealing with complex data scenarios. On the other hand, deep learning models can improve performance by capturing deeper features. Thus, the second research question focuses on: How do traditional collaborative filtering models (e.g., matrix factorization) compare to deep collaborative filtering models in terms of accuracy and overfitting control in movie recommendation systems?

# 2 Literature Review

## 2.1 Literature Review Introduction

In the modern era of digital entertainment, movie recommendation systems are crucial in providing users with personalized content. These systems have seen significant advancements through the use of algorithms like content-based filtering, collaborative filtering, and hybrid models. However, despite these improvements, there are still limitations in terms of performance, particularly in achieving the highest levels of recommendation accuracy and relevance. This study aims to explore and compare the effectiveness of various models and techniques currently in use to identify the best-performing approach for generating movie recommendations.

The primary focus of this research is to improve the accuracy and overall performance of recommendation systems, with an emphasis on identifying which modern techniques—such as deep learning models, including Transformers (Ashish Vaswani, 2017), matrix factorization approaches (Yehuda Koren, 2009), and hybrid models—can deliver the most accurate recommendations.

Deep learning techniques, particularly Transformer models, have demonstrated significant potential in handling large datasets and providing better personalization by capturing complex user behaviors (Linhan Xia, Jul 2024) . Likewise, matrix factorization models continue to offer strong performance in collaborative filtering tasks by effectively managing user-item interactions (Jesús Bobadilla, July 2022) . By comparing these advanced techniques, this study seeks to identify the optimal model that maximizes recommendation accuracy and enhances user experience, thereby advancing the current capabilities of movie recommendation systems.

## 2.2 The Development and Classification of Movie Recommendation Systems

Movie recommendation systems play an essential role in helping users discover films that match their preferences by analyzing large collections of movies. These systems use various techniques, including content-based filtering, collaborative filtering, hybrid models, and deep learning.

One of the foundational techniques in recommendation systems is content-based filtering, which suggests items based on the characteristics of movies, such as genre, director, cast, and plot. By analyzing these features, the system attempts to align movies with user preferences.

Avasthi et al (Sandhya Avasthi, July 2024) extended this approach by integrating deep learning to address challenges like the cold-start problem, where there is insufficient data on new users. Their model not only improved the accuracy of predictions but also demonstrated how deep learning can better capture user preferences compared to traditional content-based methods.

Building on this, Nilla & Setiawan (Setiawan, February 12, 2024) advanced content-based filtering by incorporating convolutional neural networks (CNNs) alongside text analysis models like RoBERTa and TF-IDF. Their multimodal system, which analyzed both movie posters and reviews, created a more holistic view of user preferences. This integration of visual and textual data significantly enhanced the accuracy of recommendations.

Mahesha et al (Mahesha, Kumara, & Banujan, 2024) introduced another innovative approach by employing natural language processing (NLP) techniques such as TF-IDF and Latent Dirichlet Allocation (LDA) to analyze movie reviews. This allowed their system to extract key thematic topics, enabling more personalized and contextually relevant recommendations. By focusing on thematic content, their system could better align movie suggestions with a user’s nuanced preferences.

While content-based filtering focuses on the features of individual movies, collaborative filtering looks at patterns of user interactions to predict preferences. It identifies similarities between users or items based on past behaviors, making it one of the most widely used methods in recommendation systems.

Mahesha et al (Priyanka Meel, 2020) enhanced traditional collaborative filtering by integrating semantic and frequency-based filtering, which helped address the issue of data sparsity and cold-start problems. Their hybrid approach improved the system’s ability to predict user preferences by combining multiple techniques, thus overcoming the limitations of traditional models.

On the other hand, Gohzali & Panjaitan (Panjaitan, 2024)explored the combination of bisecting K-Means clustering with collaborative filtering. By clustering users before applying collaborative filtering, they managed to lower the mean absolute error (MAE), particularly in large datasets like MovieLens. This approach showed potential for handling scalability challenges while maintaining accuracy.

Similarly, Wu (Wu, 13-03-2024) experimented with K-Nearest Neighbors (KNN) combined with cosine similarity to enhance collaborative filtering. By using Netflix’s open dataset and representing rating data as sparse matrices, Wu’s model showed notable improvements in recommendation accuracy and user satisfaction. The results highlighted how hybrid approaches like KNN, when paired with collaborative filtering, could offer better personalization.

Mohammad & Urolagin (Mohammad & Urolagin, 2022) took a broader approach, combining content-based and collaborative filtering with data clustering and computational intelligence. By integrating these diverse techniques, their model optimized recommendation quality and improved user satisfaction, illustrating the value of hybrid approaches in improving overall system performance.

Hybrid recommendation systems combine the best of both content-based and collaborative filtering methods to overcome their respective limitations. These systems are particularly effective in addressing issues like cold-start problems and data sparsity.

Meel (Priyanka Meel, 2020) demonstrated how combining semantic and frequency-based filtering with collaborative filtering could improve recommendation quality. Their model was especially effective in online movie markets, where data sparsity can be a significant challenge for traditional filtering methods.

Further refining the hybrid approach, Naskar & Joseph (Naskar & Joseph, February 2024) incorporated sentiment analysis into their recommendation system. By analyzing the emotional tone of movie reviews, they were able to offer more personalized and emotionally resonant recommendations. This integration of sentiment with user preferences helped to create a more nuanced and engaging recommendation experience for users.

Rai et al (Abhishek Kumar Rai1, 25/04/2024)also explored hybrid models by combining content-based and collaborative filtering within a Python-based recommendation engine. Their work showed how using multiple similarity measures—such as Pearson correlation, Spearman rank correlation, and cosine similarity—could enhance recommendation precision, offering deeper insights into user preferences.

Deep learning techniques have become increasingly influential in movie recommendation systems, especially in handling large and complex datasets.Zhan et al (Zhan, Xie, Huang, & Chen, February 2024) developed a deep learning-based recommendation model that simulates human neural networks to extract features from movie data. Through backpropagation and parameter optimization, their model achieved a 1.4% improvement in accuracy compared to traditional algorithms, highlighting the growing importance of deep learning in enhancing recommendation systems.

Taking a multimodal approach, Xia et al (Linhan Xia, Jul 2024) introduced a recommendation system that leverages Transformer-based architectures like BERT for textual data and Vision Transformer (ViT) for visual data. By integrating these modalities, their model provided more comprehensive and accurate recommendations, demonstrating the potential of deep learning in combining different data sources to create better user experiences.

Aliberti et al (Aliberti, D'Aniello, Gaeta, & Marzolo, 2024) tackled the issue of data sparsity by introducing a fuzzy signature-based method. Using fuzzy logic to represent user preferences, their model performed particularly well in cold-start scenarios, providing an innovative solution to one of the most common challenges in recommendation systems.

In addition to established methods, recent studies have introduced new approaches to further improve movie recommendation systems.

Nilla & Setiawan (Setiawan, February 12, 2024) demonstrated how combining CNNs and multimodal analysis could significantly enhance content-based filtering. By analyzing both text and images, their system offered a deeper understanding of user preferences, providing more accurate and relevant recommendations.

Similarly, Naskar & Joseph (Naskar & Joseph, February 2024) showed that integrating sentiment analysis into hybrid systems could improve the emotional resonance of recommendations. This approach added an extra layer of personalization, addressing limitations in traditional systems that focus solely on behavioral data.

Finally, Goyani & Chaurasiya (Chaurasiya, 26th of august 2020) emphasized the importance of combining collaborative filtering with content-based filtering to address limitations in both methods. Their review highlighted how using different similarity measures, such as cosine similarity and Pearson correlation, can improve recommendation accuracy, offering valuable insights for enhancing system performance.

Wu (Wu, 13-03-2024) also illustrated how combining KNN with cosine similarity could help mitigate cold-start and data sparsity issues. Applied to Netflix’s dataset, this hybrid approach delivered better accuracy and more relevant recommendations, showcasing the strength of combining multiple techniques.

## 2.3 Common Techniques and Model Architectures in the Field of Recommendation Systems

Matrix factorization (MF) techniques have become central to collaborative filtering recommender systems due to their ability to extract latent factors that reflect user preferences and item features. Koren, Bell, and Volinsky (Yehuda Koren, 2009) emphasize the effectiveness of MF models like Singular Value Decomposition (SVD), demonstrating that MF outperforms traditional neighborhood-based methods by incorporating implicit feedback, temporal dynamics, and regularization techniques. MF models offer accurate predictions and personalized recommendations by capturing complex user-item relationships. Extensive experimentation on real-world datasets reveals that MF models are scalable and superior in various recommendation tasks compared to traditional methods.

Bobadilla et al (Jesús Bobadilla, July 2022) evaluate six MF models using multiple collaborative filtering datasets, analyzing both accuracy and beyond-accuracy measures, such as novelty, diversity, and ordered and unordered list recommendations. They highlight how each MF model can address specific goals, including improving novelty and transparency. The study also examines the importance of assigning semantic interpretations to hidden factors, enhancing the explainability of recommendations. Additionally, the authors explore the role of MF models in group recommendations and the generation of reliability values, demonstrating the versatility of MF techniques in addressing different recommendation needs.

Venkatesan et al (León, 2023) examine MF models' scalability and robustness, especially when dealing with large and sparse datasets like Netflix. They demonstrate that MF models excel in handling imbalanced data with limited user-item interactions. The authors propose a variant of the MF model that introduces a flexible prior to manage model complexity, ensuring reliable recommendations even for users with sparse historical data. This scalability allows MF models to perform effectively in large-scale environments, making them a reliable choice for recommendation tasks.

Bobadilla et al (Bobadilla, 2024)further evaluate six representative MF models across multiple datasets, considering both prediction accuracy and beyond-accuracy measures such as novelty and diversity. Their findings confirm that MF models can be adapted to achieve specific recommendation goals, such as improving the transparency of latent factors or enhancing recommendation novelty. The authors also emphasize reproducibility in their experiments by providing an open framework and implementation code, ensuring that their research can be replicated and extended by others.

Matrix factorization (MF) combined with attention mechanisms has emerged as a robust hybrid approach in recommender systems, significantly improving recommendation accuracy by addressing the limitations of traditional MF models. Mao et al (Chengzhi Mao, 2024)emphasize that while MF is widely used for its simplicity and efficiency, the traditional dot-product method cannot effectively capture the nonlinear relationships between user and item latent features. They propose the Attention Interaction Matrix Factorization (AIMF) model, which incorporates a multi-layer perceptron (MLP) structure and an improved "slide-attention" algorithm. This design extracts nonlinear features from both user and item latent spaces, solving the issue of feature interaction when dimensions differ and reducing computation time while enhancing recommendation accuracy.

Hanafi et al (Hanafi, July27, 2024)further explore the hybridization of MF with advanced neural network models like BERT and attention mechanisms to overcome the sparsity problem inherent in collaborative filtering. Their approach integrates Probabilistic Matrix Factorization (PMF), BERT, and an attention module to deepen the understanding of contextual information in user-item interactions. The authors demonstrate that the hybrid model surpasses traditional MF models, showing an average improvement of over 16%, with a particular focus on reducing Root Mean Squared Error (RMSE). Their findings highlight the advantage of hybrid models in capturing both contextual and latent patterns in user-item data, offering a substantial performance gain over standard MF approaches.

Nguyen et al (Vuong Quoc Nguyen, 2024) explore a hybrid model combining sentiment analysis with matrix factorization. By incorporating additional dimensions such as sentiment and temporality into the MF framework, the authors address the limitations of two-dimensional product-user matrices. The integration of sentiment analysis allows the model to predict user rating tendencies more accurately, with empirical results showing superior predictive performance on a dataset of mobile app reviews. This combination of sentiment analysis with MF highlights the growing trend of enriching collaborative filtering models with external information to improve recommendation precision.

Anand et al (Prakhar Anand, June 2024) further explore the hybrid approach by integrating content-based filtering and collaborative filtering models with matrix factorization to enhance movie recommendation systems. Their hybrid model incorporates user ratings, watch history, and movie metadata, combining classical machine learning methods with MF to improve recommendation quality. The authors argue that hybrid models provide a more comprehensive solution by leveraging the strengths of multiple recommendation techniques, thus increasing user satisfaction and, ultimately, platform revenue. Their results show that hybrid models outperform classical filtering techniques, underscoring the growing importance of blending different approaches in dynamic and data-driven environments.

Graph Neural Networks (GNNs) have emerged as a powerful method to process graph-structured data, expanding the applicability of neural networks to domains such as computer vision, molecular chemistry, and pattern recognition. Scarselli et al (Franco Scarselli, December 2008) introduce the GNN model as an extension of traditional neural networks designed to process data represented in graph domains. Their model maps a graph and its nodes into an m-dimensional Euclidean space, allowing for the direct processing of various types of graphs, such as acyclic, cyclic, directed, and undirected ones. GNNs effectively capture the relationships within graph structures, which is critical for tasks involving interconnected data. They also propose a supervised learning algorithm to estimate the parameters of the GNN model while considering computational efficiency, thus validating its generalization capabilities through experiments.

Wang et al.(Zeyu Wang, 2024) demonstrate the potential of GNNs in the sports domain, specifically in football formation strategies. They propose a GNN-based framework to assist coaches in making tactical decisions during matches. By formulating the passing ball paths and player positions as a network, the GNN framework helps identify core players based on clustering coefficients and passing relationships. Additionally, they implement a reinforcement learning-based graph-to-graph framework, which optimizes the team's structure by modifying player positions step by step. The model evaluates the team's performance against different levels of opponents, showcasing how GNNs can provide valuable insights in real-time decision-making processes.

Mendes da Silva et al (Angelo Cesar Mendes da Silva, August 2024) apply GNNs to the music industry, focusing on artist similarity in music recommendation systems. They propose a novel heterogeneous graph-based artist representation that integrates multimodal data, including audio, lyrics features, and artist relationships. This multimodal representation is further enhanced through a network regularization process and GNN aggregation, which produces a unified and robust artist representation. The model addresses artist similarity as a link prediction problem, introducing a new importance matrix to emphasize artist relationships in the multimodal space. This hybrid approach significantly improves the accuracy of artist similarity predictions, outperforming baseline models by leveraging the multimodal representation and the importance matrix, which proves essential for enhancing GNN performance.

Gao et al (Chen Gao, February 2022) explore the application of GNNs in recommender systems, a field that has benefited significantly from their ability to process structured data and capture high-order relationships. GNNs have become the new state-of-the-art in various recommendation tasks due to their capacity to explore complex interactions within the data. However, Gao et al. emphasize the challenges in designing GNN methods for specific recommendation problems. They categorize the challenges into four critical areas: graph construction, network design, optimization, and computation efficiency. Addressing these challenges is crucial for the successful implementation of GNN-based recommender systems. Recent advances in GNN models are organized into four key perspectives: stages, scenarios, objectives, and applications. By systematically addressing each challenge, GNNs can be better utilized in various real-world scenarios, such as e-commerce, social media, and content streaming platforms. The tutorial highlights the importance of customization in GNN architecture for different recommendation tasks, ensuring that GNN-based models are both efficient and effective.

The Transformer model, introduced by Vaswani et al. (Ashish Vaswani, 2017), brought a fundamental shift in sequence processing by replacing traditional recurrent and convolutional neural networks with a self-attention mechanism. This mechanism allows the model to capture dependencies across the entire input sequence regardless of distance, improving parallelization and scalability. Transformers rely solely on self-attention to dynamically focus on the most relevant parts of the sequence, making them more efficient and effective compared to previous models. The model’s success has extended beyond natural language processing (NLP) tasks to other sequential data problems such as machine translation and time series analysis. The Transformer’s ability to handle long-range dependencies efficiently and scale to large datasets has made it a dominant architecture in modern deep learning.

In the context of recommendation systems, Kang and McAuley (Wang Cheng Kang, 2018) proposed SASRec, a self-attentive sequential recommendation model that leverages the self-attention mechanism to model user behavior over time. The model focuses on capturing long-range dependencies in users' interaction histories to predict future actions. Unlike Recurrent Neural Networks (RNNs), which are traditionally used for sequential data, SASRec does not suffer from the same limitations in terms of training time and scalability. By eliminating the need for recurrence, SASRec achieves better performance and reduces computational costs. Extensive experiments demonstrate that the self-attention mechanism in SASRec significantly improves the accuracy of recommendations, outperforming RNN-based models on benchmark datasets.

Yao et al (Zhiyu Yao, 2023) further extend the Transformer model for recommendation systems with their Recommender Transformer (RETR). They introduce a novel Pathway Attention mechanism that selectively focuses on pivotal user behaviors while filtering out less relevant actions. This mechanism differs from traditional transformer-based models that apply attention uniformly across all past behaviors. RETR tailors attention to user-specific behavior pathways, which improves the accuracy of predictions in sequential recommendation tasks. The model demonstrates state-of-the-art performance across both intra-domain and cross-domain benchmarks, effectively handling dynamic and evolving user behavior patterns. The selective attention mechanism in RETR enhances the model's ability to capture the most relevant aspects of user interaction histories, resulting in more precise recommendations.

In a different application of Transformers,Kumar et al (Anup Kumar, November 2023) applied the model in the context of tool recommendation systems on the Galaxy platform, a web-based platform for scientific analysis. Their Transformer-based tool recommender significantly outperforms previous systems built using RNNs, CNNs, and dense neural networks (DNNs). In their experiments, the Transformer model exhibited faster convergence times, lower model usage costs, and improved generalization to workflows beyond the training data. It also achieved higher precision in tool recommendations compared to DNNs, reaching approximately 0.98 precision@k versus 0.9 for DNN. The superior performance of the Transformer in this context underscores its versatility and efficiency across a variety of domains.

Zhao et al (Yuyue Zhao, July 2024) introduce ToolRec, a novel framework aimed at enhancing recommendation systems by leveraging large language models (LLMs) to address the challenge of capturing fine-grained user preferences. ToolRec uses LLMs as surrogate users, guiding the recommendation process through the use of external tools designed to explore user interests at a more granular attribute level. The framework bridges the existing gap between item semantics and user behavior, incorporating both rank and retrieval tools to refine recommendations. By integrating LLMs, ToolRec enables recommendation systems to operate with a natural language interface, making the recommendation process more intuitive and adaptable. The experimental results validate the effectiveness of ToolRec, particularly in contexts rich with semantic content, positioning the framework as a potential future direction for recommendation models.

Loshchilov and Hutter (Ilya Loshchilov, 2016) introduced Stochastic Gradient Descent with Warm Restarts (SGDR), which improves training performance by periodically resetting the learning rate through cosine annealing. This technique prevents the model from stagnating in suboptimal solutions and enhances convergence on datasets like CIFAR-10 and CIFAR-100, showcasing its efficiency in deep neural network training.

Shaikh et al (Shamal Shaikh, 2023) explored data augmentation and refinement techniques in recommender systems, particularly using Maximum Margin Matrix Factorization (MMMF). Their semi-supervised approach augments training data by incorporating high-confidence predictions and removing low-confidence entries. This iterative refinement process addresses data sparsity in collaborative filtering models, improving prediction accuracy across various recommendation tasks.

Du et al (Xiaoyu Du, 2019) demonstrated the effectiveness of Recurrent Neural Networks (RNNs) in session-based recommendations by capturing temporal dependencies in user interactions. Their approach models user behavior within sessions as sequential data, allowing the model to predict future interactions based on past behavior. The model significantly outperforms traditional methods in session-based recommendation systems by adapting to evolving user preferences.

Lundberg and Lee (Scott M. Lundberg, 2017) proposed SHapley Additive exPlanations (SHAP) as a unified framework for interpreting model predictions. SHAP assigns importance values to features in complex models like deep learning and ensemble methods, addressing the challenge of model interpretability. By unifying various existing methods, SHAP provides a consistent and theoretically sound approach to understanding model behavior and decision-making processes.

Akiba et al (Takuya Akiba, 2019)introduced Optuna, a hyperparameter optimization framework built on the principles of Bayesian optimization, specifically the Tree-structured Parzen Estimator (TPE). Optuna’s define-by-run API allows dynamic construction of parameter search spaces, offering flexibility and efficiency in hyperparameter optimization across various scales, from distributed computing environments to lightweight experiments.

Micikevicius et al (Paulius Micikevicius, 2018) presented Mixed Precision Training, a technique that uses half-precision floating-point numbers for training deep neural networks. This approach reduces memory requirements and increases computational speed on GPUs while maintaining accuracy. The method includes safeguards like loss scaling and single-precision accumulation to prevent the loss of critical information, making it effective for large-scale models.

## 2.4 Literature Review Conclusion

Movie recommendation systems have significantly advanced, effectively meeting the growing demand for personalized and highly accurate film suggestions by leveraging cutting-edge techniques. These include content-based filtering, collaborative filtering, hybrid models, deep learning, matrix factorization (MF), attention mechanisms, graph neural networks (GNNs), and Transformer-based architectures. These technologies have greatly enhanced recommendation accuracy and personalization, empowering systems to provide more relevant and tailored experiences. With the continuous evolution of these methods, there are promising opportunities for even greater improvements in recommendation performance and system efficiency

Matrix Factorization (MF) (Yehuda Koren, 2009)is a fundamental technique in the field of recommendation systems, widely utilized for its ability to capture user preferences and item features through latent factors. As a classic representative of traditional recommendation algorithms, MF is often used as a benchmark to evaluate the performance of emerging models. This benchmark not only provides a reliable framework for assessing the effectiveness of modern models but also highlights the advantages of new approaches in handling complex user behaviors and large-scale data. By leveraging MF’s well-established metrics, we aim to conduct a comprehensive comparison and analysis of the performance improvements offered by contemporary models.

Hybrid models leverage the strengths of different recommendation techniques, combining them to enhance overall system performance. As described by Anand et al (Prakhar Anand, June 2024), integrating matrix factorization with content-based and collaborative filtering enriches the data analysis, allowing systems to deliver more personalized recommendations. This approach not only increases user satisfaction but also drives platform revenue by providing more accurate predictions. The synergy of these methods in hybrid models addresses the complexities of user preferences more effectively than any single technique alone, demonstrating a significant advancement in recommendation technology.

Graph Neural Networks (GNNs) play a pivotal role in recommendation systems due to their ability to handle complex, structured data such as user-item interactions. As noted by Scarselli et al (Franco Scarselli, December 2008), GNNs excel at capturing intricate relational patterns, which enhances prediction accuracy. Mendes da Silva et al (Angelo Cesar Mendes da Silva, August 2024) demonstrated the versatility of GNNs by applying them to analyze artist similarities using multimodal data. Despite the computational efficiency challenges highlighted by Gao et al (Chen Gao, February 2022), the unique strength of GNNs in capturing deep user-item relationships makes them indispensable for improving recommendation systems.

The Transformer model, exemplified by Vaswani et al (Ashish Vaswani, 2017), revolutionizes recommendation systems by efficiently handling long-range dependencies and large datasets through its self-attention mechanism. This capability significantly improves the accuracy and scalability of these systems compared to traditional methods. By focusing dynamically on the most relevant parts of user interaction sequences, Transformers like SASRec and RETR not only enhance recommendation precision but also adapt to evolving user behavior, demonstrating their effectiveness across various application domains.

Advanced techniques like Stochastic Gradient Descent with Warm Restarts (SGDR) and Mixed Precision Training significantly enhance the training and computational efficiency of deep neural networks, as demonstrated by Loshchilov and Hutter (Ilya Loshchilov, 2016) and Micikevicius et al (Paulius Micikevicius, 2018). These methods prevent stagnation and accelerate convergence, making them indispensable for handling large datasets efficiently. Additionally, frameworks like Optuna facilitate optimal hyperparameter tuning, boosting model performance across diverse scenarios. Incorporating SHAP for model interpretability ensures that complex models remain understandable and trustworthy, vital for deploying AI in sensitive or impactful domains.

The core strategy in refining recommendation systems focuses on leveraging advanced models such as Graph Neural Networks, Transformers, and hybrid models to transcend the capabilities of traditional matrix factorization. These innovative approaches are adept at handling structured data, processing sequential dependencies, and integrating multiple recommendation techniques to enhance the accuracy and personalization of outputs. Supporting these advancements are optimization technologies like Stochastic Gradient Descent with Warm Restarts (SGDR) and Mixed Precision Training, along with tools like Optuna for hyperparameter tuning and SHAP for model interpretability. These auxiliary technologies enhance model efficiency and ensure their applicability in dynamic environments, making the systems more effective and trustworthy.

The evolution of movie recommendation systems, while promising, faces significant challenges such as data sparsity, the cold-start problem, and ensuring the real-time performance of increasingly complex models. Data sparsity and the cold-start problem inhibit the system's ability to deliver accurate recommendations for new users or items with limited interaction data. Real-time processing demands are exacerbated by the sophisticated computations required by advanced models like GNNs and Transformers, which may limit their deployment in latency-sensitive environments.

Looking ahead, the future of recommendation systems lies in tackling these challenges through the integration of emerging technologies and methodologies. One promising direction is the exploitation of real-time data streams and richer contextual information, such as current events or social media trends, to enhance the models' responsiveness and accuracy. Large Language Models (LLMs), as demonstrated by Zhao et al (Yuyue Zhao, July 2024) in ToolRec, offer significant potential by enabling systems to understand fine-grained user preferences through natural language processing. LLMs can draw from a vast array of data sources like user reviews and social media, enriching the user profiles and improving personalization, especially in addressing cold-start problems. Additionally, advancements in unsupervised learning could offer new ways to address data sparsity by extracting more meaningful features from unlabelled data. The development of more efficient algorithms and hardware optimizations will also be crucial in enabling the practical deployment of sophisticated models like LLMs, GNNs, and Transformers in real-time scenarios. Together, these efforts will push the boundaries of what recommendation systems can achieve, making them even more personalized and accurate .

# 3 Methodology

## 3,1 Research Design

### 3.1.1 Research Type

This study adopts a mixed-methods approach, combining qualitative and quantitative research. The qualitative part involves interviewing industry experts to gather insights on movie recommendation systems, current challenges, and trends. The quantitative part involves conducting experiments and comparisons on deep learning models using the MovieLens 1M dataset to build and optimize a movie recommendation system.

The reason for adopting a mixed-methods approach is to combine expert practical experience with data-driven model analysis, ensuring the recommendation system design reflects industry needs while achieving optimal algorithm performance.

### 3.1.2 Research Methods

Qualitative Research: Semi-structured expert interviews were conducted with four experts in the field of movie recommendation systems. These interviews aimed to gain a deeper understanding of their views on movie recommendation technologies, user behavior, and the main challenges faced by movie recommendation systems. The insights gathered provide practical context and direction for model selection and optimization.

Quantitative Research: Using the MovieLens 1M dataset, several mainstream deep learning models (such as matrix factorization-based models, graph neural networks, and Transformer models) were experimented with and compared. Each model's performance was evaluated using multiple metrics (Mean Squared Error and strict overfitting standards, 5%), with the best-performing model being selected as the core algorithm for the movie recommendation system.

### 3.1.3 Research Process Framework

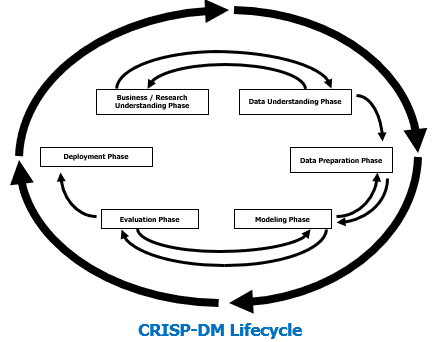


Figure 1 : CRISP-DM Process Diagram

As illustrated in Figure 1, the CRISP-DM (Cross-Industry Standard Process for Data Mining) framework was employed to maintain a structured and systematic research process. In the Business Understanding phase, qualitative interviews were conducted with industry experts to gather insights into the industry demands, user behavior patterns, and the challenges faced by movie recommendation systems. During the Data Understanding phase, the MovieLens 1M dataset was used to analyze user rating behaviors and identify potential patterns, establishing a solid foundation for subsequent modeling.

In the Data Preparation phase, data cleaning, feature engineering, and transformation were performed to ensure the models could effectively utilize the various data features. To address the complexity and diversity inherent in movie recommendation systems, several mainstream deep learning models were explored during the Modeling phase, including matrix factorization, hybrid matrix factorization with self-attention mechanisms, graph neural networks (GNN), and Transformer models. These models were designed to capture more intricate interactions between users and movies.

In the Evaluation phase, model performance was rigorously assessed using metrics such as Mean Squared Error (MSE), while measures to control overfitting, such as a 5% overfitting threshold, were implemented to ensure optimal results. Lastly, the Deployment phase involves integrating the best-performing model into the recommendation system to provide more personalized and accurate recommendations.

By adopting the CRISP-DM framework, this study followed an orderly progression through each phase, aligning closely with industry needs and data analysis objectives. This approach offers a robust theoretical and practical foundation for model optimization and real-world application.

## 3.2 Data Collection Methods

### 3.2.1 Data Types

**Primary Data**

In this research, four experts in recommendation systems were interviewed: a PhD student, an associate professor from a prestigious university, and two industry veterans with extensive experience in data science. The judgment sampling method was used for participant selection, allowing for a focused approach by targeting individuals with deep domain knowledge. While this method provides high-quality insights, it may introduce bias and limit generalizability due to the non-random nature of the sampling.

The experts' contributions were invaluable, offering diverse perspectives on technical challenges, industry trends, and the practical applications of recommendation systems. Their insights significantly influenced the design of the experiment, helping refine the research objectives and the methodology.

**Secondary Data**

The experimental data comes from the comes from the MovieLens 1M public dataset, a pre-processed, publicly available dataset that includes 1 million ratings from approximately 6,000 users on over 3,900 movies. The dataset contains key information such as user IDs, movie IDs, ratings, and timestamps. Widely used in recommendation system research, MovieLens 1M serves as a robust benchmark for evaluating and comparing different recommendation models. Its structured format provides a solid foundation for experimental analysis in this study, ensuring consistency and comparability across various models.

### 3.2.2 Data Collection Process

**Expert Interview Process**

All interviews were conducted via online video conferencing (Zoom), with each interview lasting approximately 60 minutes. The experts who participated came from various institutions in the field of movie recommendation systems, ensuring diversity and broad representation across the industry。

Prior to the interviews, the experts received the interview objectives and guide, and confirmed in advance through communication that they understood the purpose of the interviews and how the data would be used.

During the interviews, the researcher followed the interview guide but allowed the experts to freely express their views and adjusted the questions flexibly based on the experts' responses.

**The use of the MovieLens 1M Dataset**

The dataset was obtained from the official MovieLens website, a publicly available dataset commonly used for recommendation system research. The original structure of the dataset includes time-series data of user ratings, as well as user demographic information (such as age, gender, occupation) and metadata of the movies (such as title and genre).

The dataset was used directly for model training and evaluation, without further collection or processing. Detailed data cleaning and pre-processing methods are described in the subsequent data analysis section.

## 3.3 Ethical Considerations

### 3.3.1 Ethical Issues for Expert Interviews

Prior to each interview, the researcher ensured that informed consent was obtained from the experts. All participants were fully informed about the purpose, content, and use of the interview data, and agreed to the recording of the interview process.

Privacy Protection: In the analysis and reporting, the identities of all experts were anonymized, and any information that could identify individuals was removed to ensure privacy protection. The recordings and interview transcripts were accessible only to the research team and were stored on encrypted devices.

Voluntary Participation: All experts participated voluntarily, with the right to withdraw from the interview at any time or refuse to answer specific questions.

### 3.3.2 Ethical Issues for the MovieLens 1M Dataset

This dataset is a publicly available secondary dataset that was anonymized during its release, so the researcher did not need to obtain additional personal or privacy information from users. Therefore, no direct ethical issues are involved.

Nonetheless, the study follows best practices in data usage, ensuring that the handling and analysis of the data comply with relevant ethical standards and strictly adhere to the usage terms provided by the dataset publisher.

**License**: [MovieLens 1M Dataset License](https://files.grouplens.org/datasets/movielens/ml-1m-README.txt)

## 3.4 Data Process

### 3.4.1 Data Preparation

The MovieLens 1M dataset was divided into three parts. The first step was conducting an exploratory data analysis (EDA) to review the basic information, statistical descriptions, and identify any outliers or missing values in each part. Important features (such as movie features) were then individually examined, and the distributions of all features were visualized. It was verified that the distribution of user characteristics was in line with general population demographics, ensuring the credibility of the dataset.

### 3.4.2 Feature Engineering

In this experiment, various models were used, and accordingly, different data processing techniques were applied. To enhance code readability and modularity, feature engineering was integrated with model construction. Key tasks included embedding user and movie IDs, as most models require vectorized inputs. Demographic features were excluded for two reasons: first, stricter privacy regulations (e.g., GDPR) limit the use of user data; second, context-based methods, which use initial interactions and short-term behaviors, perform better in addressing the cold-start problem without relying on demographic information.

Movie genres were encoded using one-hot encoding to avoid misinterpretation as ordinal features. Timestamps were used to sort input data and split it into training and test sets. In the Transformer model, relative time differences were used as a substitute for position encoding, significantly improving performance.

## 3.5 Model Architecture

### 3.5.1 Deep Collaborative Filtering Model Framework Overview

Deep Collaborative Filtering (Abhishek Kumar Rai1, 25/04/2024) enhances the traditional user-item matrix of collaborative filtering by leveraging deep learning to create a non-linear mapping. While traditional collaborative filtering predicts user preferences based on historical behaviors, deep collaborative filtering uses the powerful representation capabilities of neural networks to capture more complex interactions and relationships between users and items.

Key Points:

1. Embedding Learning: Users and items are mapped into a low-dimensional latent space, where embedding vectors capture their latent features.
2. Non-linear Modeling: Deep neural networks introduce non-linear representations, allowing the model to learn the complex relationships between user preferences and item characteristics.
3. Enhanced Generalization: Unlike traditional linear models, deep collaborative filtering better handles sparse data and improves generalization through its complex network structures.

The core of deep collaborative filtering lies in its ability to leverage deep learning's non-linear feature extraction capabilities, combining user and item embeddings to more accurately predict user interests. Compared to traditional collaborative filtering, it offers stronger representation power and provides more accurate predictions for recommendation systems.

To overcome the limitations of traditional recommendation models, variants such as matrix factorization, hybrid networks combining self-attention mechanisms with matrix factorization, graph neural networks (GNNs), and Transformers have been introduced. Matrix factorization effectively handles sparse data and captures latent features of users and items. The combination of self-attention mechanisms with matrix factorization enables better modeling of complex user-item interactions. GNNs utilize multi-level relational information in graph structures, further enhancing recommendation performance. Meanwhile, Transformers, with their strength in sequence modeling, capture temporal patterns and long-term dependencies in user behavior. By integrating these technologies, recommendation systems are expected to more accurately predict user preferences and interests.

### 3.5.2 Overview of Matrix Factorization Models

Matrix Factorization is a widely used collaborative filtering technique in recommendation systems, primarily aimed at predicting user ratings for items. In recommendation systems, users and items are typically represented in matrix form, where each user-item pair's rating is an element of the matrix. The goal of matrix factorization is to decompose this sparse rating matrix into two smaller matrices: the user latent factor matrix and the item latent factor matrix. The product of these two matrices can approximate the original rating matrix and predict unrated items.



Formula 1: Objective Function for Matrix Factorization in Collaborative Filtering

Formula 1 (Yehuda Koren, 2009) represents the objective function of the matrix factorization model, used to predict user ratings for items. Its main goal is to minimize the error between the user's actual ratings and the model's predicted ratings while incorporating a regularization term to prevent overfitting.

• is the actual rating of item i by user u.

• is the predicted rating of item i by user u, calculated as the inner product of the user latent factor pu​ and the item latent factor qi​.

• represents the squared difference between the actual rating and the predicted rating, indicating the prediction error.

•The regularization term λ is included to prevent overfitting by penalizing overly large parameters, ensuring that the model's complexity does not become excessive.

Thus, the core mechanism of matrix factorization lies in representing users and items as low-dimensional latent factor vectors to make predictions. These latent factors capture the underlying relationships between user preferences and item characteristics. By optimizing the objective function, the model minimizes the error between actual and predicted ratings. The inclusion of a regularization term helps prevent overfitting while capturing the user-item interactions, thereby enhancing the model’s generalization capability.

### 3.5.3 The structure of Hybrid Matrix Factorization with Self-Attention (HMFA)

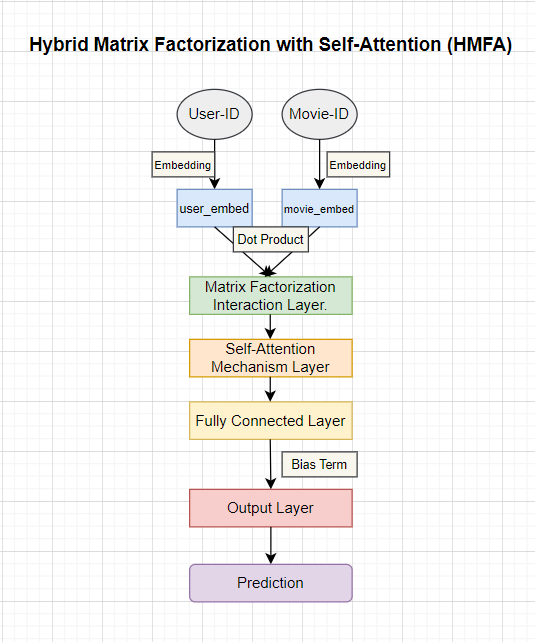


Figure 1 : Hybrid Matrix Factorization with Self-Attention Model Architecture

As Figure 1 illustrates the structure of the Hybrid Matrix Factorization with Self-Attention (HMFA) model. The model takes User-ID and Movie-ID as inputs and follows these key steps:

1. Embedding Layer: Maps user IDs and movie IDs to low-dimensional embedding vectors, representing the features of users and movies, respectively.
2. Matrix Factorization Interaction Layer: The user and movie embedding vectors are combined via a dot product to generate the initial user-item interaction information.
3. Self-Attention Mechanism Layer: The interaction result is passed through a self-attention layer to capture more complex user-movie relationships and enhance the representation of the interaction.
4. Fully Connected Layer: The output of the self-attention mechanism is processed through a fully connected layer to generate a scalar value.
5. Bias Terms: User bias, movie bias, and global bias are added to the final prediction.
6. Output Layer: This layer generates the predicted rating of the user for the movie.

The uniqueness of this model lies in its combination of traditional matrix factorization and the self-attention mechanism. Matrix factorization effectively captures the basic interactions between users and items, while the self-attention mechanism further enhances these interactions by dynamically adjusting the weights of different features, enabling the model to capture more complex and nuanced relationships between user preferences and item characteristics. This design is expected to help the model better understand user behavior, resulting in more personalized and accurate recommendations.

### 3.5.4 Mechanism Explanation of Graph Neural Networks (GNNs)

Graph Neural Networks (GNNs) are a powerful deep learning technique used to handle data that is structured as graphs. In recommendation systems, GNNs can be used to capture complex relationships between users and items. Each user and item is represented as a node in a graph, and the interactions between them (such as ratings or clicks) are represented as edges. The primary goal of GNNs is to learn a representation (embedding) for each node by aggregating information from neighboring nodes.



Formula 2: Formal Representation of a Graph-Based Dataset with Node Features

••Represents a specific graph structure, consisting of nodes

and edges . The graph set G contains multiple graphs

•Node Represents j-th node in graph and the node set includes all nodes in the graph.

•Node feature: Each node is associated with a feature vector , which belongs to R^m space. This vector is used to describe the attributes or state of the node, such as user or item features in a recommendation system.

•Range: The formula specifies that 1≤i≤p , represents the total number of graphs, with p raphs. Each graph contains , nodes, and the node index j ranges from 1≤j≤

.

Formula 2 (Franco Scarselli, December 2008)defines the input format for Graph Neural Networks, explaining how multiple graphs, along with their nodes and features, are structured in a unified way. Using this structure, GNNs can aggregate information from each node and its neighbors, allowing the network to learn the features of nodes in the graph and perform tasks such as node classification, link prediction, or recommendations.

### 3.5.5 Mechanism Explanation of Transformer

The Transformer model, proposed by Vaswani et al (Ashish Vaswani, 2017), revolutionized natural language processing by replacing traditional recurrent models like RNNs and LSTMs. Its key innovation is the self-attention mechanism, which enables the model to capture relationships between all tokens in a sequence, regardless of length, allowing for faster parallel processing and improved handling of long-range dependencies.



Formula 3 : Scaled Dot-Product Attention Mechanism

Formula 3 (Ashish Vaswani, 2017) represents the self-attention mechanism, a core concept in the Transformer model. Here's a simple breakdown:

Q (Query), K (Key), and V (Value): These vectors come from the input.

•Q represents the query, the element we are focusing on

•K represents the key, which helps determine the relevance between different elements in the sequence.

•V represents the value, which holds the actual information we want to combine based on the attention scores.

: This is the dot product of the query and the key vectors, which gives a similarity score between each query and key pair in the sequence.

1/: This is a scaling factor (where is the dimension of the key vector) used to prevent the resulting dot product from becoming too large as the sequence grows.

softmax: This normalizes the scores into a probability distribution, so that the sum of all attention scores for a given query is 1, emphasizing the most relevant information.

V: Finally, these normalized attention scores are multiplied by the value vector V, effectively selecting the most relevant parts of the input sequence for further processing.

In short, this formula calculates how much focus (attention) each element in a sequence should give to the other elements, allowing the model to better understand contextual relationships.

The Transformer model combines an encoder and decoder, utilizing self-attention and feed-forward networks to efficiently process sequential data. The encoder captures relationships within the input sequence through multi-head self-attention, while the decoder generates new predictions based on prior outputs and encoder information. Self-attention computes the relevance of elements in a sequence, improving the handling of long-range dependencies. This architecture excels not only in natural language processing but also finds applications in fields like image processing and recommendation systems

## 3.6 Training and Tuning

### 3.6.1 Model Building

All models were constructed using the standard design patterns of the PyTorch framework. The constructor (\_\_init\_\_()) was used to initialize the model's structure and parameters, while the forward() function handled the data flow through the network and computed the output. This approach ensured that the code was both clear and maintainable, while also offering flexibility. Reproducibility was achieved by setting random seeds and controlling CuDNN behavior.

### 3.6.2 Optimizer Selection

The Adam optimizer was used as the optimization algorithm due to its adaptive learning rate capabilities, which make it efficient for handling sparse and large-scale data.

### 3.6.3 Acceleration and Resource Management

AMP (Automatic Mixed Precision) (Paulius Micikevicius, 2018)was employed due to its ability to enhance training performance by reducing memory usage and computation time. It achieves this by using 16-bit precision for most operations while retaining 32-bit precision where needed to maintain accuracy, allowing for faster training without compromising model performance. Additionally, AMP optimizes resource consumption, making it ideal for large-scale model training on GPUs.

### 3.6.4 Learning Rate Scheduling

CosineAnnealingLR (Ilya Loshchilov, 2016)was used due to its ability to dynamically adjust the learning rate, improving model convergence. This scheduler gradually decreases the learning rate following a cosine curve, starting high and annealing it to a lower value. This helps the model avoid overshooting minima and prevents it from getting stuck in local optima. By periodically lowering the learning rate, CosineAnnealingLR promotes smoother convergence, especially in large-scale training, ensuring more stable and efficient optimization over time.

### 3.6.5 Overfitting Prevention

Dropout, L2 regularization (L2 weight decay), and early stopping were used due to their effectiveness in preventing overfitting. Dropout randomly deactivates neurons during training, promoting generalization.L2 regularization applies a penalty to the loss function, specifically on the model's weight parameters, encouraging smaller weights to reduce model complexity.

Early stopping halts training when the validation loss starts to increase, preventing the model from learning noise in the training data. These techniques ensure better model performance on unseen data by limiting overfitting.

### 3.6.6 Hyperparameter Tuning

Optuna (Takuya Akiba, 2019)was selected due to its advanced Bayesian optimization, which dynamically adapts the search process based on prior trial results, ensuring faster convergence to optimal hyperparameters compared to traditional methods like grid or random search. A key feature of Optuna is its ability to analyze the importance of each hyperparameter, providing insights into which ones significantly impact model performance. This allows for more efficient tuning. Additionally, Optuna's pruning functionality halts underperforming trials early, saving computational resources, making it highly suitable for large-scale and complex models with extensive hyperparameter spaces.

### 3.6.7 Model Interpretability Analysis

SHAP (Shapley Additive Explanations) (Scott M. Lundberg, 2017) is used to analyze the contribution of each feature to the model’s predictions, enhancing interpretability. By computing Shapley values for each feature, SHAP reveals the impact of individual inputs on the model's decisions. It provides useful visualizations, such as feature importance and local explanation plots, to offer insights into how the model makes predictions. SHAP is particularly valuable in complex models, allowing for both high performance and transparency in understanding feature interactions and their effects on outcomes.

## 3.7 Validity

### 3.7.1 Validity of Expert Interviews

The validity of expert interviews in this research is supported by the reliability and practical relevance of the data. Experts with extensive experience in recommendation systems provided deep insights beyond standard literature, ensuring the experimental design aligned with industry standards and addressed real-world challenges. Their suggestions were crucial in refining and optimizing the research methodology, significantly enhancing the accuracy, reliability, and applicability of the findings.

### 3.7.2 Model Validity Evaluation

All models in this experiment were based on user movie ratings, and the Mean Squared Error (MSE) was employed as the primary evaluation metric, commonly used in prediction tasks. This evaluation approach not only effectively reflects the models' predictive performance but also allows for performance comparisons across models. To ensure reproducibility, model training and evaluation were conducted by setting random seeds and controlling CuDNN behavior.

### 3.7.3 Data Validity

The MovieLens 1M public dataset was used, ensuring data authenticity and minimizing bias. The dataset contains 1,000,209 rating records from 6,040 users and 3,883 movies, providing sufficient sample size to represent a wide range of cases. All models used an 85/15 split for the training and validation sets, with the split based on timestamps to ensure consistent data distribution and enhance the generalization ability of the models.

### 3.7.4 Model Selection and Evaluation

The models selected for this research are deep learning architectures based on collaborative filtering, utilizing an embedding approach for input data representation. By embedding user and item interactions into vector space, the models effectively capture latent relationships. To ensure a fair comparison, all models share a similar level of complexity, minimizing the risk of bias due to varying model depths. This consistent structure helps focus the comparison on performance differences arising from model architecture, rather than depth-related factors.

## 3.8 Environment and Setup

### 3.8.1 Programming Language and Framework

The system was developed using Python, with the GPU-enabled version of PyTorch utilized for building and training deep learning models. PyTorch is favored for its dynamic computational graph, which allows for greater flexibility and ease of debugging during model development. Additionally, PyTorch's strong GPU acceleration capabilities make it highly efficient for large-scale models, significantly reducing training time. Its extensive library support and active community also contribute to rapid prototyping and experimentation, making it an excellent choice for deep learning tasks.

### 3.8.2 Hardware Setup

Model training was performed on a server equipped with an NVIDIA GPU. The system had 16GB of RAM, and the GPU type was GTX1660Ti.

# 4 Results & Analysis

## 4.1 Phase of Exploratory Data Analysis

Before the experiment began, a comprehensive Exploratory Data Analysis (EDA) was conducted on the entire dataset. This process was crucial, with the primary purpose of assessing the data quality. The analysis indicated that the overall quality of the dataset was satisfactory, as highlighted by the following points:

1 No significant missing or anomalous values were found. For instance, both user IDs and movie IDs were fully recorded, and the ratings fell within the reasonable range of 1 to 5, with no invalid or missing data.

2 Duplicate records: No substantial number of duplicate entries was detected. The combination of timestamp, user ID, and movie ID in the rating data was unique, ensuring the dataset's consistency and integrity.

This version presents a polished and professional translation suitable for academic contexts.

As a result, the overall distribution of user and movie data was further examined to determine its alignment with general demographic patterns. Specifically, an analysis was conducted on users' demographic characteristics such as age, gender, and occupation, alongside movie-related attributes like genre, rating distribution, and release date. This analysis enabled verification of whether the data reasonably reflects the diversity of real-world users and the broad appeal of the movies, ensuring that the distribution does not display significant biases or anomalies.

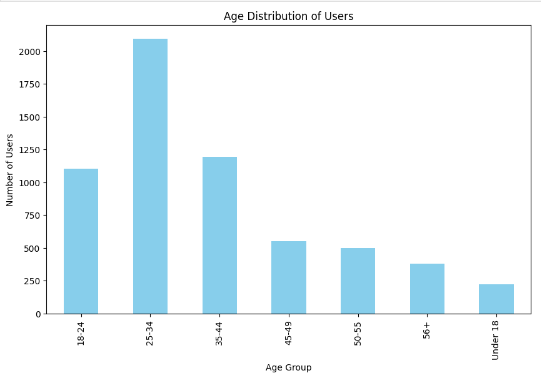


Figure 1 : Age Distribution of Users

Figure 1 reflects the typical user distribution for internet services. Generally, the user base on online platforms is primarily composed of young and middle-aged individuals who tend to rely heavily on the internet, especially for entertainment, social interaction, and information retrieval. This distribution trend has been validated across many internet platforms. Therefore, from the perspective of user distribution, this data aligns with expectations. It also provides an important reference for the subsequent model building, ensuring that the model can offer more targeted recommendations for the primary user groups.

## 4.2 Phase of Matrix Factorization Model

During the experiment, a classic matrix factorization model was initially constructed, a technique widely applied in recommendation systems, particularly in collaborative filtering tasks. User ID and movie ID embedding vectors were utilized as input features, mapping these discrete identifiers into a continuous vector space to capture the latent relationships between users and items. Additionally, bias terms were introduced for both users and items to account for global effects, such as a user's overall rating tendencies or the general popularity of certain movies, thereby enhancing the model's robustness.

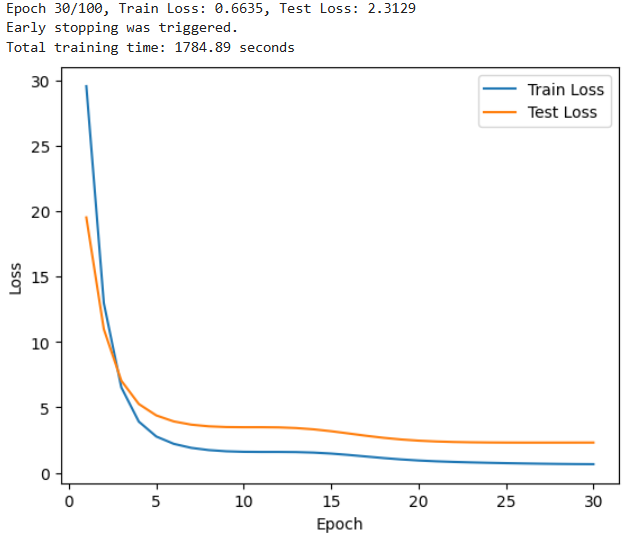


Figure 1 : Initial Matrix Factorization Model Training Results

As shown in Figure1, the initial training outcomes were less than ideal. The model exhibited poor predictive performance on the validation set, with a significant gap between the training error and the test error, indicating severe overfitting. The model appeared to overfit the training data and failed to generalize effectively to unseen data. This issue may have arisen due to the model's excessive complexity or improper hyperparameter settings.

To address these problems, an effective method for optimizing the model's hyperparameters was required. Inspired by a conversation with an interviewed expert—an experienced professor with deep expertise in machine learning and optimization techniques—the decision was made to incorporate the Optuna hyperparameter optimization framework into the workflow. The professor highlighted the efficiency of Bayesian optimization methods in hyperparameter tuning, emphasizing how such frameworks intelligently explore the hyperparameter space by learning from previous evaluations, selecting potentially higher-performing combinations, accelerating convergence to optimal solutions, and saving time and computational resources.

Optuna, utilizing advanced algorithms like Bayesian optimization with Tree-structured Parzen Estimator (TPE), was integrated to optimize hyperparameters. The study explored learning rates, embedding dimensions, regularization (L2 weight decay), and dropout rates. By defining the search space and iteratively exploring, Optuna identified configurations that enhanced model performance.

As a result of this optimization process, a reproducible and better-performing matrix factorization model was successfully developed. Furthermore, Optuna provided valuable insights into the influence of different hyperparameters on model performance, offering a better understanding of which factors were most critical to the model's success.

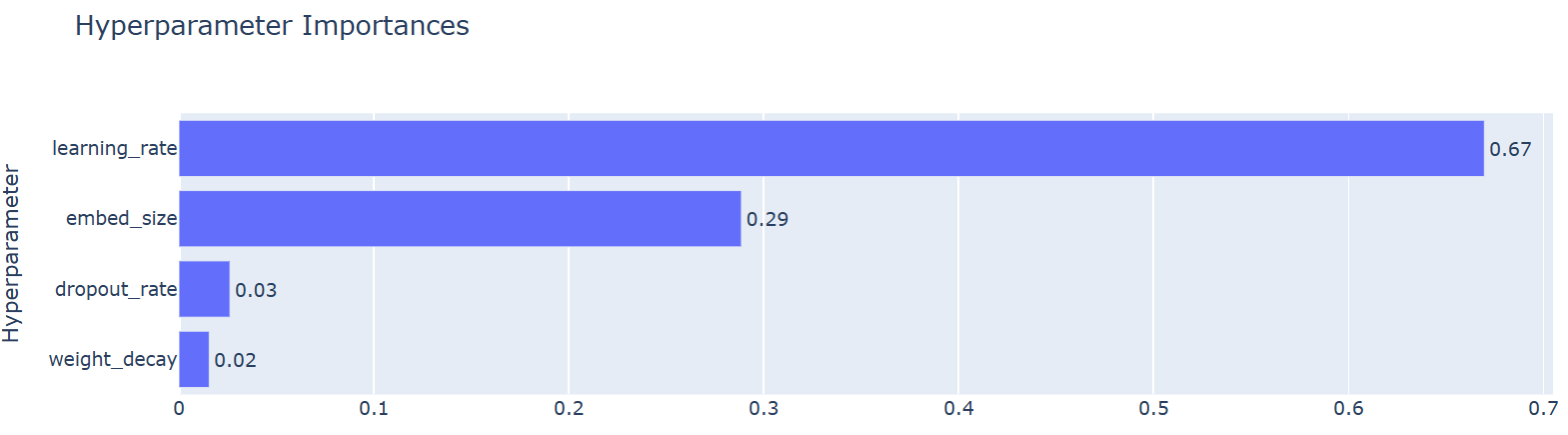


Figure 2 : Hyperparameter Importance Shown During Optuna Tuning of the Matrix Factorization Model

Figure 2 indicated that the learning rate had the greatest impact on model accuracy, accounting for more than half of the total influence. During the model training process, a Cosine Annealing mechanism was employed to dynamically adjust the learning rate, and this strategy played a crucial role in the model's training. The embedding dimension also had a significant impact, which was expected, as it is closely related to the overall complexity of the model and directly affects its representation capability and generalization performance. In contrast, the dropout rate and L2 regularization penalty had a smaller effect on model accuracy, suggesting that in the current experimental setup, their contribution to preventing overfitting was not particularly significant.

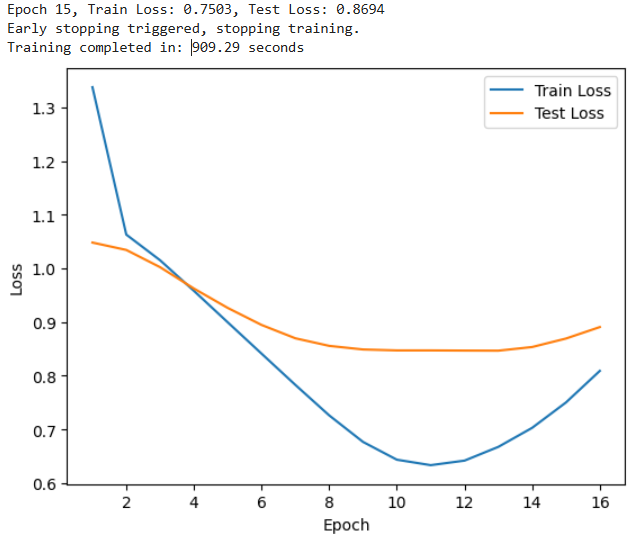


Figure 3 : Matrix Factorization Model Training Results with the Best Parameters from Tuning

As shown in Figure 3 above, the model tuned with Optuna still exhibited overfitting, necessitating further fine-tuning. The experimental phase indicated that the embedding dimension, dropout rate, and L2 regularization penalty were the most effective hyperparameters for addressing overfitting. In addition, Early Stopping was introduced, with the patience parameter set to 3, to prevent the model from continuing to train once its performance on the validation set no longer improves. Based on these measures, a series of tuning experiments were conducted focusing on these three hyperparameters, aiming to enhance the model's generalization ability, reduce the risk of overfitting, and improve its performance on unseen data.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Tuning Iteration | Embed Size | Dropout Rate | Weight Decay | Train Loss | Test Loss | Overfitting Rate（%） |
| 1 | 64 | 0.3258 | 1e-5 | 0.7503 | 0.8684 | 15.9 |
| 2 | 32 | 0.3258 | 1e-5 | 0.7308 | 0.8519 | 16.6 |
| 3 | 32 | 0.3258 | 5e-5 | 0.8124 | 0.8818 | 8.6 |
| 4 | 16 | 0.3258 | 5e-5 | 0.8148 | 0.8832 | 8.4 |
| 5 | 16 | 0.5 | 5e-5 | 0.8234 | 0.8867 | 7.7 |
| 6 | 12 | 0.5 | 1e-4 | 0.8730 | 0.9119 | 4.5 |
| 7 | 32 | 0.3258 | 1e-4 | 0.8575 | 0.9034 | 5.4 |

Table 1 : Display of Matrix Factorization Model Fine-Tuning Results over 7 Rounds

As observed in Table 1 above, different embedding dimensions, dropout rates, and weight decay values have a significant impact on the model's performance. A larger embedding dimension (such as 64) exacerbates the overfitting issue, while a smaller embedding dimension (such as 12) effectively reduces the overfitting rate but results in increased training and test losses, negatively affecting the model's learning capacity. Increasing the dropout rate (for example, setting it to 0.5) helps the model reduce overfitting, especially when combined with a higher weight decay value (1e-4), which significantly improves the model's generalization ability. Notably, in the 6th and 7th tuning rounds, the combination of dropout rate and weight decay effectively reduced the overfitting rate.

Based on these observations, the 7th tuning result is considered optimal. In this round, the embedding dimension was set to 32, the dropout rate to 0.3258, and the weight decay to 1e-4, with the overfitting rate rounded to within 5% (specifically 5.4%), as shown in the Figure 4 below.

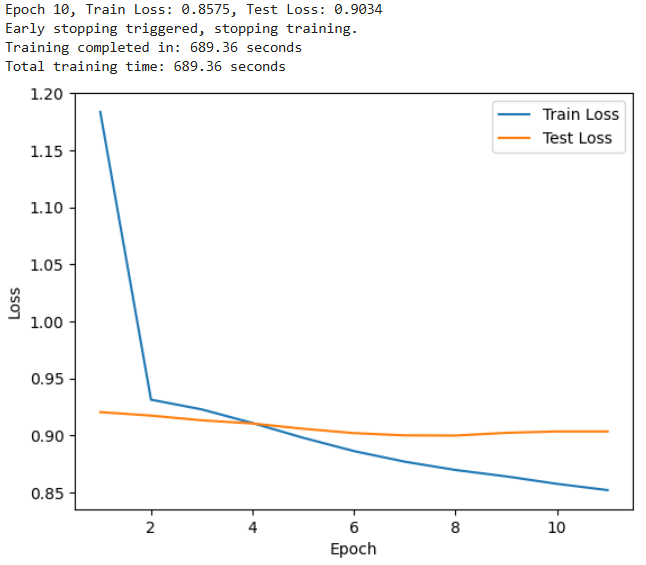


Figure 4 : Best Tuning Results of the Matrix Factorization Model

From the series of tuning processes, it is evident that the matrix factorization model exhibits significant overfitting issues, indicating that the model's performance lacks stability and has poor generalization ability. When confronted with new datasets, the model's performance may not remain consistent. This issue is particularly critical for recommendation systems, which require robust performance across different users and scenarios. Therefore, a strict overfitting standard was set for the model, with the allowable overfitting rate capped at 5%.

However, as the regularization penalties were progressively increased during the tuning process, a decline in model accuracy was observed—an unavoidable trade-off in the optimization process. To mitigate overfitting, some degree of accuracy had to be sacrificed.

Through repeated experimentation, a balance was eventually found. Although there was a slight decrease in accuracy, the overfitting issue was effectively controlled. At this balance point, As Figure 4 shown,the mean squared error (MSE) value (0.9034) of the matrix factorization model was established as the benchmark for the baseline model. In future research, further exploration of more complex model architectures and advanced techniques will continue, aiming to enhance model performance while maintaining good generalization ability.

## 4.3 Phase of Hybird Matrix Factorization with Self-Attention Model

Building on previous experimental results, a new research question was proposed: Can the performance of the matrix factorization model be improved by embedding a self-attention mechanism layer? This idea stems from observations of the successful application of self-attention mechanisms in other fields, such as natural language processing and computer vision, and has been a recurring topic in expert interviews. Although most experts did not mention the widespread use of this method, one data scientist with extensive practical experience in recommendation systems did note their prior attempts to use self-attention mechanisms, though they had not been applied on a large scale. This presents a unique opportunity to explore the feasibility of combining the self-attention mechanism with the matrix factorization model.

To investigate this, the model structure was adjusted. First, the user ID vector and movie ID vector were combined through an inner product operation to generate the interaction representation between users and movies. On this basis, a self-attention mechanism layer was introduced, aiming to enable the model to more effectively capture the complex interactions between users and movies. By calculating the relationships between different parts of the input sequence, the self-attention mechanism can efficiently extract global information, potentially improving the accuracy and depth of recommendations. This mechanism is expected to address the limitations of traditional matrix factorization models, which rely only on local interactions, and to enhance the overall performance of the model through deeper feature extraction.

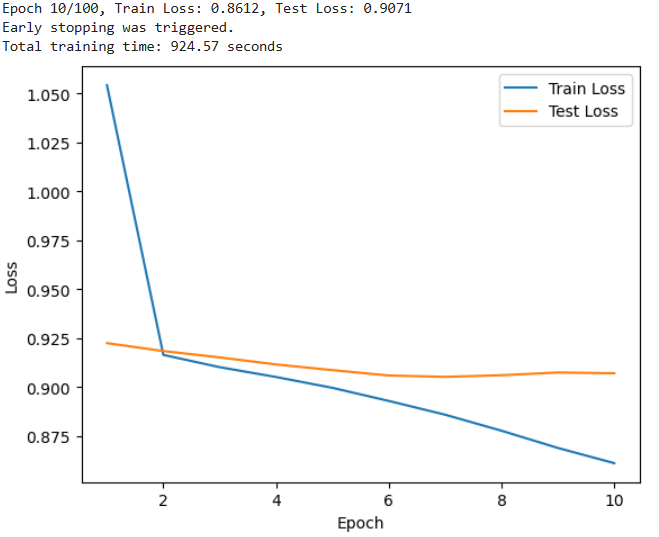


Figure 1 : Training Results of the Matrix Factorization Model with Self-Attention Mechanism Introduced

The final experimental results are shown in the Figure 1 above,after introducing the self-attention mechanism to the previously best-tuned matrix factorization model, the data indicates that, despite applying an appropriate dropout rate (0.33) and L2 regularization penalty (1e-4) to the optimally tuned matrix factorization model, the introduction of the self-attention mechanism not only failed to improve model performance but also led to a decline. Additionally, overfitting became more pronounced. This suggests that the self-attention mechanism did not have the anticipated positive impact on the matrix factorization model's performance and, in some cases, even had negative effects.

From these results, it can be inferred that the self-attention mechanism may not be suitable for direct integration with the matrix factorization model. This could be because the matrix factorization model is designed to extract latent features between users and movies, while the self-attention mechanism, in this context, fails to effectively leverage its ability to capture global relationships. On the contrary, it might have increased the model's complexity, making it more difficult to train effectively with limited data, thus exacerbating the overfitting issue. In light of this, the idea of combining these models was abandoned, and further attempts to embed the self-attention mechanism into the matrix factorization model were discontinued. Instead, the focus will shift to exploring other, more suitable model structures and methods to improve the performance of the recommendation system.

## 4.4 Phase of Graph Neural Networks (GNN) Model

In recent years, Graph Neural Networks (GNN) have seen significant development across various fields and have been widely applied in recommendation systems. The strength of GNN lies in its ability to capture nonlinear relationships between users and items through their historical interactions. This capability allows GNN to perform exceptionally well in handling complex recommendation tasks, making it a model worth further exploration in this research.

To test the potential of GNN, a simple two-layer Graph Neural Network model was first constructed. In this baseline model, only user ID and item ID were used as input features, with the GNN layers employed to build an interaction graph between users and items. The output layer of this model consisted of a simple linear layer responsible for generating the final predictions. The preliminary experimental results are shown in the figure below.

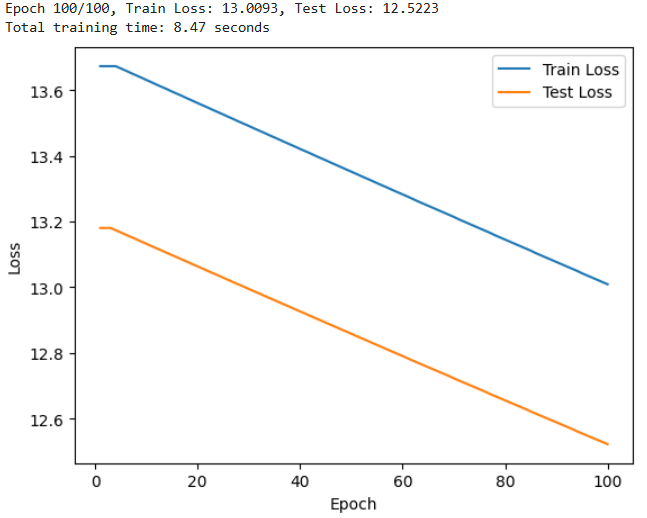


Figure 1 : Training Results of the Graph Neural Network Using Only User ID and Movie ID

As shown in Figure 3 above ,the training speed of the Graph Neural Network (GNN) was remarkably fast, with the initial experiments completing the training in less than 9 seconds. However, despite the short training time, both the training loss and test loss showed minimal improvement, indicating a clear underfitting problem. This suggests that the model failed to effectively capture the latent features between users and movies. The likely cause of this issue was the overly simplistic graph structure, resulting from the limited input features, which prevented the model from learning meaningful relationships and patterns.

To address this issue, the GNN model was restructured. In addition to retaining user ID and movie ID as input features, movie genre information was introduced using One-Hot encoding to increase the dimensionality of the input features. Additionally, the release year of the movies was extracted from the movie titles and incorporated as a separate feature input. The purpose of these improvements was to enrich the model's input features, enhancing its ability to capture complex interaction relationships and thereby improve its learning outcomes.

In the new model architecture, Dropout layers and L2 regularization techniques were also introduced to further reduce the risk of overfitting. Dropout helps by randomly dropping neurons, preventing the model from becoming overly reliant on specific features, while L2 regularization constrains the growth of weights, thereby reducing model complexity. Following these adjustments and fine-tuning, the model successfully converged, with significant improvements in both training loss and test loss, demonstrating the model's enhanced ability to capture the complex relationships between users and movies.

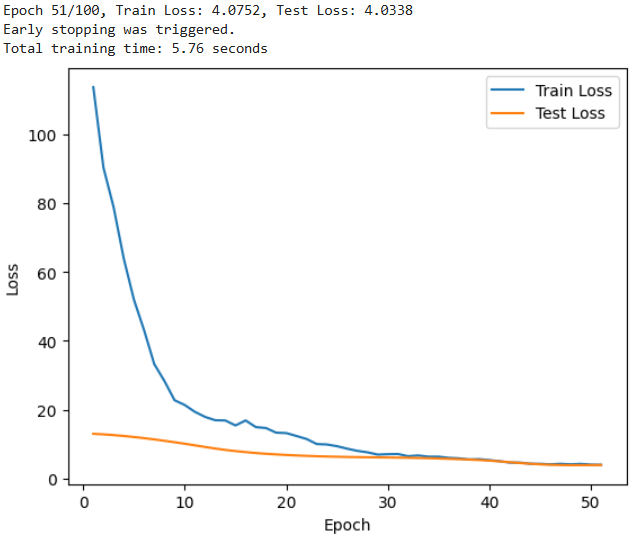


Figure 2 : Tuning Results of the Graph Neural Network After Introducing More Features, L2 Regularization, and Dropout

As shown in Figure 2 above, the experimental results demonstrated a significant improvement compared to the previous results. Both the training and test losses showed a steady downward trend, ultimately achieving good convergence. The entire training process was also highly efficient, taking only 6 seconds. However, despite the impressive training performance, the final mean squared error (MSE) was lower than expected, reaching only 4.03, which still lagged significantly behind the performance of the matrix factorization model. This suggests that, although enriching the input with more diverse features improved the performance of the Graph Neural Network (GNN), its predictive capability remained limited in this experiment.

This result indicates that, while the diversity of features positively contributed to the GNN's performance, its overall predictive capacity was still constrained. This could be attributed to the relatively small dataset and insufficiently complex feature interactions, which prevented the GNN from fully leveraging its advantages. In recommendation systems, GNNs are known to excel in handling complex relationships and diverse feature interactions, especially in the context of large-scale datasets and rich feature sets. Therefore, although this experiment did not surpass the performance of the matrix factorization model, the findings further highlight the importance of feature diversity and data volume for GNNs.

With a larger dataset and more complex feature interactions, the performance of GNNs may significantly improve.

## 4.5 Phase of Transformer

In the final phase, a Transformer-based model centered around the self-attention mechanism was introduced. In recent years, the application of self-attention mechanisms in recommendation systems, particularly in the domain of movie recommendations, has made significant progress. Transformer models have garnered considerable attention due to their powerful capability in handling sequential data. The self-attention mechanism effectively captures complex relationships between users and items, especially in recommendation systems where it can process information from different sources to improve the accuracy of recommendations. For instance, by combining information such as movie genres and user reviews, the self-attention mechanism can generate more personalized and precise recommendations.

During interviews with industry experts, positive feedback regarding the practical application of self-attention mechanisms was received. Two experts in recommendation systems expressed high regard for the self-attention mechanism. One expert noted that, in real-world scenarios, self-attention mechanisms generally outperform other models and handle complex recommendation tasks more effectively. Another expert added that not only does the self-attention mechanism excel in complex application scenarios, but its computational speed is relatively fast, especially when processing large amounts of data, significantly improving efficiency. These insights further enhanced the expectation for the self-attention mechanism, and it was hoped that the experimental results would validate the experts' perspectives.

In earlier matrix factorization model experiments, an attempt was made to integrate the self-attention mechanism into the matrix factorization model, but the results were unsatisfactory. To further explore the potential of the self-attention mechanism, it was decided to construct a complete Transformer model from scratch to fully leverage its features.

The core of the Transformer model lies in the self-attention mechanism layer, which processes all elements of the input data in parallel and captures the global dependencies between users and items. However, based on previous experimental experience, overfitting has remained a persistent issue. Therefore, a more simplified approach was adopted when constructing the new Transformer model. First, only the basic self-attention mechanism layer was used, avoiding overly complex model structures. Additionally, only user ID and item ID were used as input features, resulting in a relatively simple Transformer model. This approach allows a focus on testing the fundamental performance of the self-attention mechanism in handling user-item interactions, without too many factors interfering with the interpretation of the initial results.

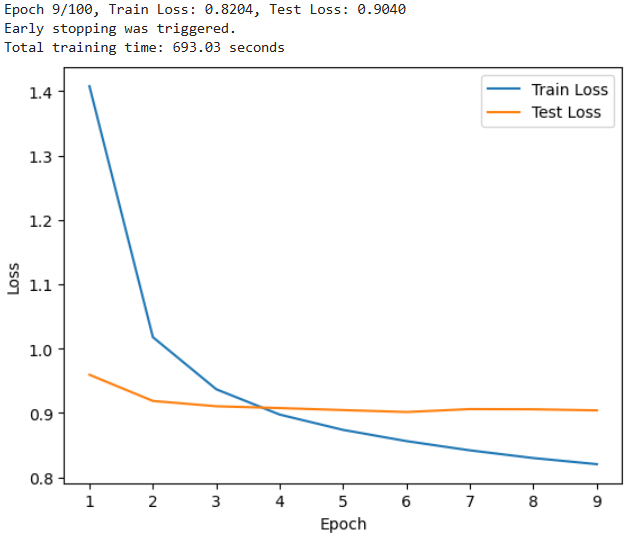


Figure 1 : Initial Transformer Training Results

Figure 1 indicate that, although the model exhibited some degree of overfitting, the overall performance on the test set remained relatively stable. Both the training and test losses decreased to low levels, suggesting that the self-attention mechanism has robust performance and strong learning capability. Based on this, it was decided to further increase the model's complexity to enhance overall performance.

First, a positional encoding module was introduced to help the model process sequential information. Additionally, multiple layers of self-attention and linear layers were added to increase the model's depth and feature representation capability. Furthermore, a multi-head self-attention mechanism was incorporated, enabling the model to capture complex interactions between users and items from multiple perspectives. With these improvements, it is expected that the model will better understand user behavior patterns and further improve recommendation accuracy.

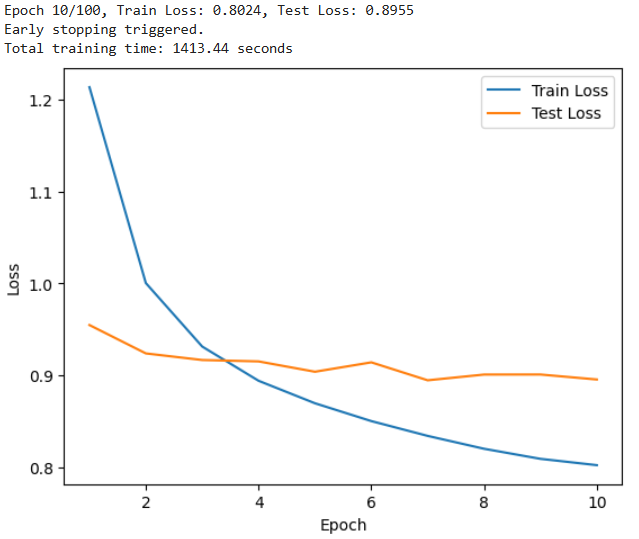


Figure 2 : Training Results of the Complete Transformer Model

However, as Figure 2 shown that, with the significant increase in model complexity, the model's performance did not improve accordingly. Instead, a notable overfitting issue emerged. While the model performed well on the training set, its generalization ability on the test set significantly declined. This suggests that the introduction of overly complex structures did not lead to the expected improvements but rather exacerbated the model's reliance on the training data.

In light of this, the experimental strategy was adjusted, and the pace of experimentation was slowed. By gradually introducing modules or altering the model structure, each module's specific contribution will be analyzed and evaluated individually. This approach will allow for a deeper understanding of the role each component plays within the model, thereby avoiding the negative effects of excessive model complexity. Incremental adjustments and optimizations of the model structure will help identify the optimal configuration of each module, ensuring that as complexity increases, the model's performance and generalization ability are genuinely enhanced.

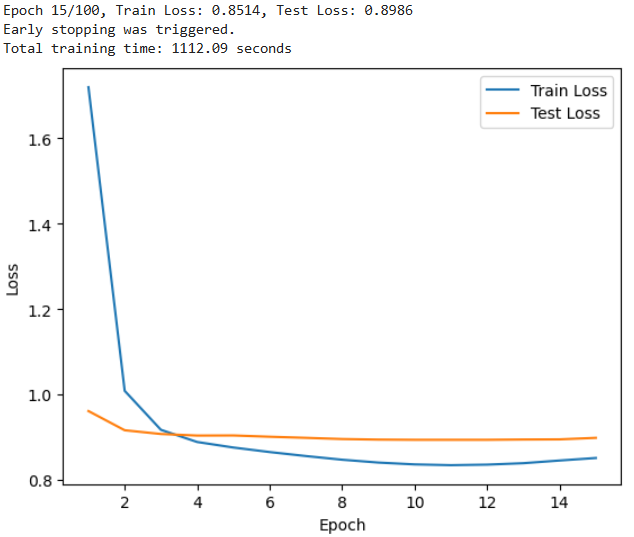


Figure 3 : Results of the Transformer Model with Only Self-Attention Layers After Simple Tuning

As shown in Figure 3 above, a simple tuning of the Transformer model with a single-layer self-attention mechanism resulted in an MSE of 0.8986 and an overfitting rate of 5.5%. This model was established as the baseline for this phase, and an Ablation Study was conducted to perform a series of tests.

First, only the basic self-attention mechanism and positional encoding module were retained. This strategy was adopted to explore the specific role of the positional encoding module within the model and assess its impact on performance. By gradually removing or simplifying different components of the model, a more precise understanding of the contribution of positional encoding in capturing sequence information could be obtained.

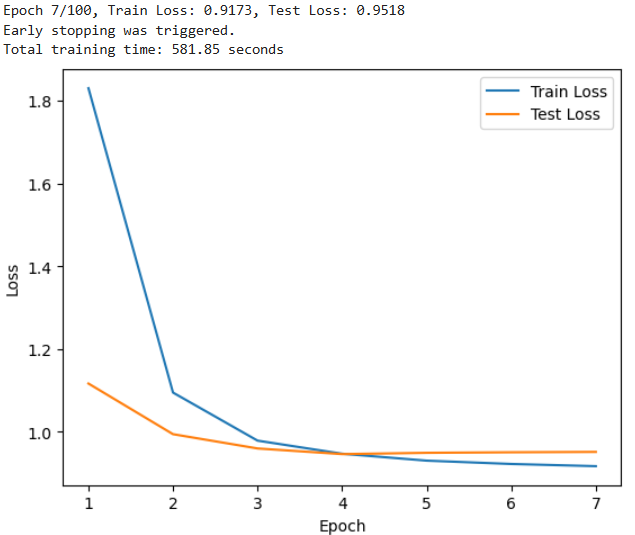


Figure 4 : Training Results of the Transformer with Only Self-Attention Layers After Adding the Positional Encoding Module

The Figure 4 shown that the positional encoding module reduced overfitting to some extent, with the overfitting rate decreasing from 5.5% to 3.8%. However, despite controlling overfitting, the model's accuracy declined significantly, with the MSE rising from 0.8986 to 0.9518. This substantial drop in accuracy is unacceptable, indicating that while positional encoding helps improve generalization, it negatively impacts overall prediction accuracy in this context. Therefore, relying on the positional encoding module alone is insufficient to enhance the model's overall performance, and further exploration of other potential optimization methods is needed.

To further improve model performance and more effectively address the overfitting issue, two data augmentation methods were attempted: adding noise and sequence modeling.

First, noise was introduced to a certain proportion of the ratings to simulate uncertainty and noise interference in the rating data. This approach enhances the model's robustness, enabling it to better generalize when handling noisy real-world data and thus reducing the likelihood of overfitting.

Second, sequence modeling techniques were applied. The basic principle involves segmenting the user's interaction history (such as movie viewing and rating) into multiple subsequences based on a specified window size and step size. Each subsequence represents the user's behavior patterns and trends within a specific time period. By structuring a user's behavior history into multiple time segments, the model is better equipped to capture the changes in user preferences over time, leading to more accurate predictions of future behavior.

The combination of these two methods aims to improve the model's ability to perceive temporal changes and handle noisy data, thereby enhancing the overall performance of the recommendation system.

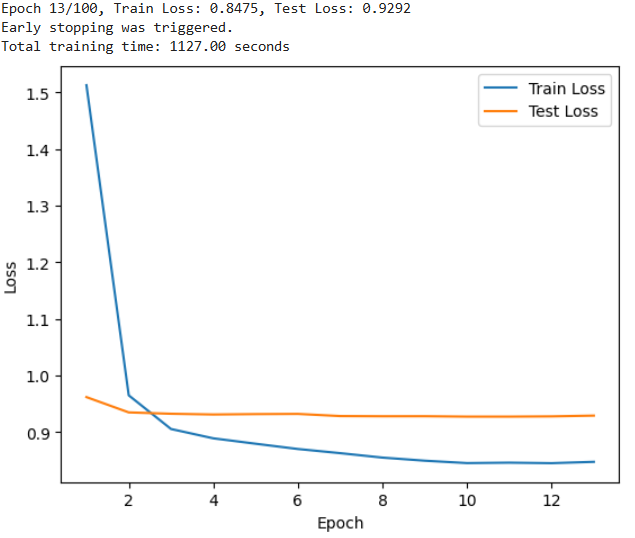


Figure 5 : Training Results of the Transformer with Only Self-Attention Layers After Introducing Noise Perturbation

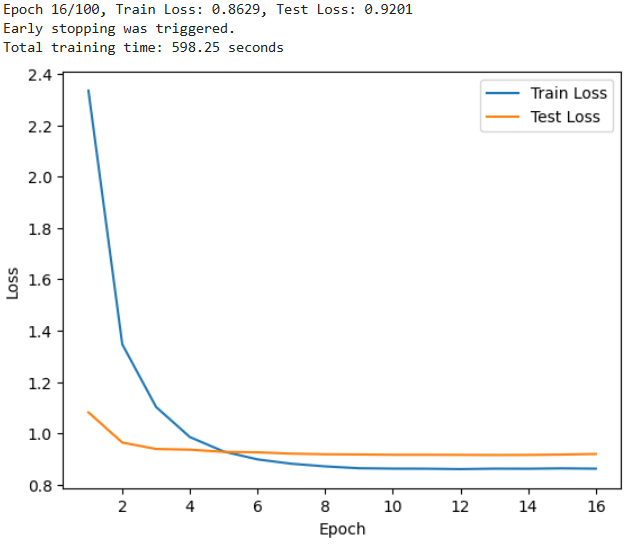


Figure 6 : Training Results of the Transformer with Only Self-Attention Layers After Introducing Sequence Modeling Techniques

Figure 5 and Figure 6 indicate that the data augmentation did not produce the expected improvements; instead, it led to a decline in model performance. This may be due to the excessive addition of noise, which confused the model and hindered its generalization ability. Additionally, suboptimal choices in sequence modeling, such as window size and step length, may have caused important long-term patterns to be overlooked. The combination of these two methods also increased complexity, potentially raising the risk of overfitting.

Movie genre features have always been regarded as a critical factor in recommendation systems, significantly influencing users' viewing experiences and preferences. To fully leverage this feature, three different approaches were employed to incorporate movie genre information in an effort to improve model performance and recommendation accuracy.

|  |  |  |  |
| --- | --- | --- | --- |
| Model Attempt | Train Loss | Test Loss | Overfitting Rate(%) |
| Movie feature embedding added to other features | 0.8510 | 0.9050 | 6.3 |
| Only the first movie feature concatenated with other features | 0.8511 | 0.9037 | 6.2 |
| Linear fusion of movie features with other features | 0.8510 | 0.9050 | 6.3 |

Table 1 : Training Results of the Transformer with Only Self-Attention Layers After Introducing Movie Features in Three Different Ways

From the Table 1, three different methods were tested for incorporating movie genre features into the model, and their effects were evaluated:

1. Movie Feature Embedding Combined with Other Features: In this method, the movie genre features were converted into low-dimensional vectors using an embedding layer, which were then combined with other features for model training. The training loss was 0.8510, and the test loss was 0.9050. Although the test loss was slightly lower, this approach did not significantly improve the model's overall performance.
2. Concatenation of Only the First Movie Genre Feature with Other Features: This approach selected only the first genre of each movie and concatenated it with other features. The training loss was 0.8511, and the test loss was 0.9037. While this method performed slightly better on the test set compared to the others, the improvement was quite limited.
3. Linear Fusion of Movie Features with Other Features: In this method, the movie genre features were fused with other features through a linear layer. The training loss was 0.8510, and the test loss was 0.9050. Although this approach was similar to embedding features, it also failed to significantly enhance the model's test performance.

The results show that despite incorporating movie genre features, there was no significant improvement in the model's performance. To explain this phenomenon and explore the underlying reasons, SHAP (SHapley Additive exPlanations) was employed to analyze the model. SHAP is a tool based on the Shapley value from game theory, designed to assign importance scores to each feature and quantify its contribution to the model's prediction results. This method allows for a better understanding of the model's decision-making process and clarifies the influence of each feature on the final prediction.

In this experiment, SHAP analysis was performed on the model with incorporated movie genre features. By evaluating SHAP values, the goal was to uncover the actual contribution of movie genre features within the model and explain why they did not effectively enhance model performance. This approach helps identify poorly performing feature handling methods within the model and provides a basis for subsequent optimization. SHAP analysis will assist in transparently understanding how the model assigns feature weights, and further exploration can focus on how to more effectively utilize movie features to improve recommendation accuracy.

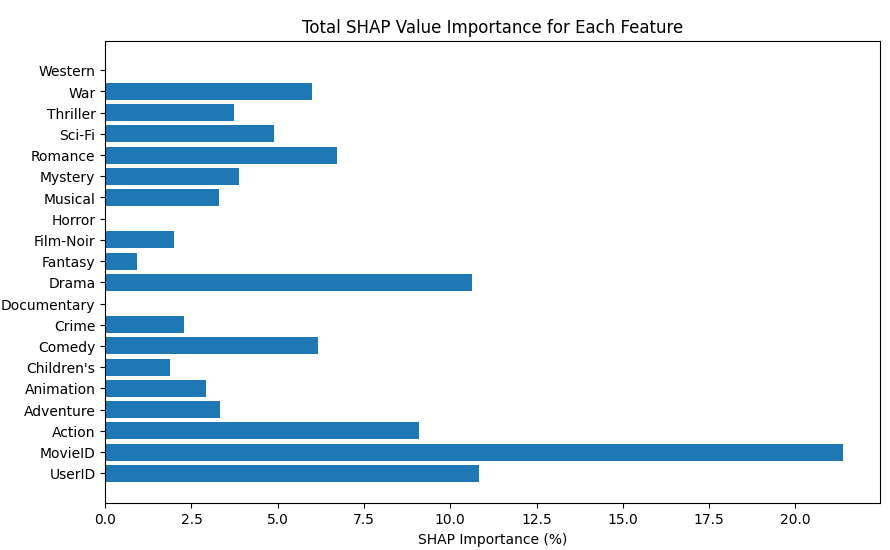


Figure 7 : Total SHAP Value Importance for Each Feature

Figure 7 indicate that user ID and movie ID are the two most important features in the model, with movie ID in particular having a decisive impact on prediction accuracy. This helps explain why the introduction of movie genre features led to a decline in model performance. It is likely that the high correlation between movie genre and movie ID features introduced redundant information, increasing the model's complexity and exacerbating the overfitting issue. This phenomenon highlights that adding more features does not always benefit the model, especially when the original features already sufficiently capture the information needed for the task. In such cases, introducing new features can actually degrade model performance.

Through this series of experiments, it became clear that the core challenge the model faced was overfitting. Although the self-attention mechanism initially showed robustness in early experiments, subsequent attempts to enhance performance by introducing data augmentation and additional features did not yield improvements; in many cases, they further worsened the overfitting problem. This realization led to the understanding that increasing the complexity of the model structure is not necessarily the solution. In some cases, adding complexity can increase computational burden and model sensitivity, especially when the existing data features are already sufficient. Based on this, it is hypothesized that the single-layer self-attention mechanism may already be the most appropriate structure for the current task, and further complexity may be unnecessary.

With this hypothesis, the focus shifted to optimizing the existing single-layer self-attention mechanism model through more systematic tuning. To achieve this, the Optuna framework was employed to conduct 100 rounds of hyperparameter tuning, systematically searching for the optimal combination of key hyperparameters, including learning rate, embedding dimension, regularization coefficient, and dropout rate.

Following these 100 rounds of tuning, a relatively stable model was obtained. Based on this, further fine-tuning was carried out.

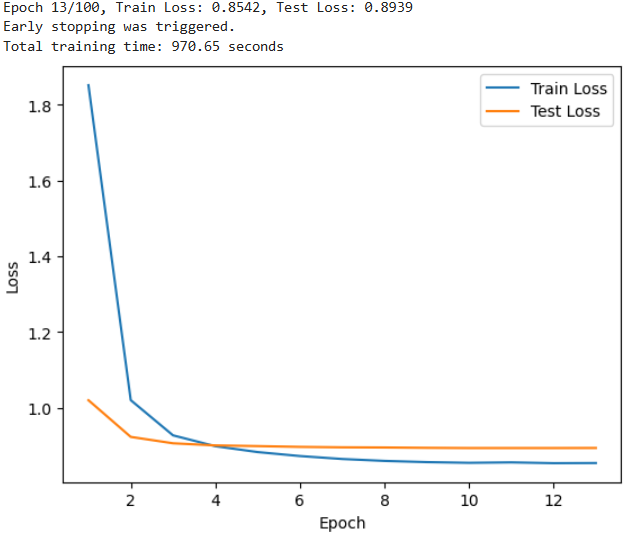


Figure 8 : The Best Transformer Model Training Results

The final results as Figure 8 show that the self-attention mechanism outperformed the matrix factorization model in this experiment, particularly in terms of the Mean Squared Error (MSE) of 0.8939, where the self-attention mechanism achieved a lower value. Additionally, its overfitting rate was only 4.6%, nearly 1% lower than the best-performing matrix factorization model. Although this difference is small, it is sufficient to demonstrate the performance advantage of the self-attention mechanism.

From this study, it can be confidently concluded that while the overall performance of the self-attention mechanism and matrix factorization techniques is comparable, the self-attention mechanism slightly surpasses in the context of recommendation systems, especially in terms of model robustness and generalization ability. It is better suited to handling complex data scenarios, exhibiting greater robustness and a lower risk of overfitting. This suggests that the self-attention mechanism, as a modern technique, holds more potential than traditional matrix factorization in deep learning models for recommendation tasks.

## 4.6 The Development and Application of the Movie Recommendation System

Finally, the project moved into the application phase, where a complete recommendation system was built using the optimal model algorithm. The system consists of two main components.

In the first component, the optimal model was trained on the full dataset, and the model's weight parameters were saved. By storing the trained model weights, the system ensures that the model does not need to be retrained for subsequent operations, significantly improving the system's efficiency and response time.

The second component involves the core structure and evaluation process of the recommendation system. When a user inputs their user ID through the GUI, the system automatically combines this user ID with all movies the user has not yet watched, feeding them into the model for prediction and evaluation. The model generates a predicted rating for each movie based on the movie's features and the user's historical behavior. The system then selects the top K movies with the highest predicted ratings and generates a personalized recommendation list for the user. Through this design, the recommendation system can quickly provide personalized movie recommendations, ensuring that the model does not need to be retrained, while dynamically generating recommendation results based on the user's latest input, thus achieving both efficiency and accuracy improvements.

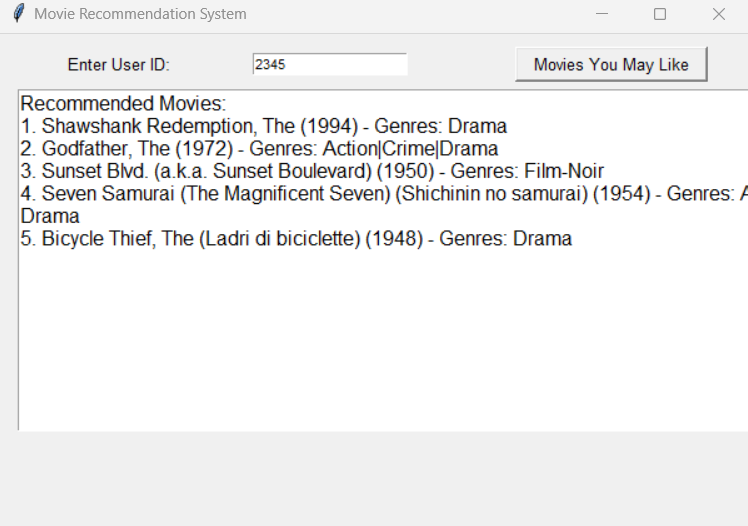


Figure 1 : Movie Recommendation System User Interface

As shown in Figure 1 above, the interface design effectively reflects Edward Tufte’s data visualization principles. The interface is simple and intuitive, adhering to the high data-ink ratio concept. All displayed content is related to the recommended movies, avoiding unnecessary visual elements or decorations. There is no "chart junk" in the interface; movie names, years, and genres are clearly presented in a simple text format, allowing users to easily access key information.

In practical applications, it may be necessary to increase the model’s complexity and add more visual elements and decorations to the user interface based on specific requirements. However, the core trained model and recommendation system algorithm have been fully demonstrated, providing significant reference value for future academic research and the development of a complete commercial movie recommendation system.

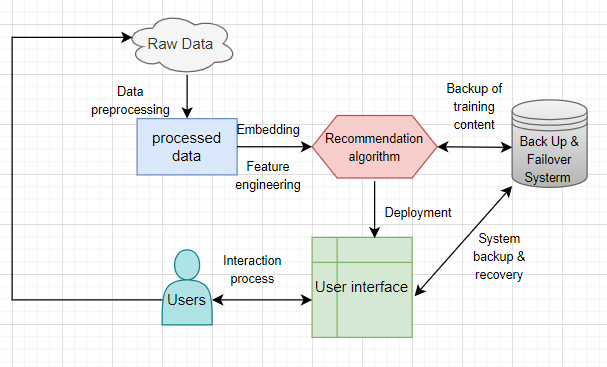


Figure 2 : Deployment of the Movie Recommendation System in a Production Environment

From a practical application perspective, as illustrated in the Figure 2, a movie recommendation system in a production environment typically consists of multiple steps to ensure both efficiency and fault tolerance. First, the system collects user behavior data (such as viewing history and ratings) along with movie feature data (such as genre and cast), which undergoes data preprocessing to remove noise and generate embedding vectors. Subsequently, the system utilizes algorithms like deep collaborative filtering to train a model that learns the relationships between users and movies. Once training is complete, the model is deployed into the production environment to provide users with personalized recommendations in real time.

Thus, the exploration from traditional matrix factorization models to modern self-attention mechanism models has been successfully completed. During the experimental process, despite facing overfitting challenges, through the tuning of hyperparameters such as embedding dimensions, dropout rate, and L2 regularization, as well as the introduction of early stopping, the overfitting rate was effectively controlled within a reasonable range. Ultimately, the self-attention mechanism model demonstrated greater robustness and generalization ability compared to the matrix factorization model, particularly excelling in complex data scenarios. With this, the experimental phase was successfully concluded, and the implementation of the recommendation system was also completed. The system can now generate personalized movie recommendations based on user input, with high efficiency and accuracy.

# 5 Conclusion

## 5.1 Response to Experimental Question One

How can expert interviews help us better understand the direction of model development and the evaluation process?

During this research, the interviews with experts served as primary data and played a crucial role in guiding the project's development. Many key conclusions they provided were validated through experiments, such as the observation that self-attention mechanisms often outperform other models in real-world production environments. Additionally, the experts introduced several methods, including Bayesian optimization, data selection, and model evaluation techniques, which were indispensable to the smooth progress of this study.

For instance, in the case of Bayesian optimization, the complexity of the model structure and the large number of hyperparameters used in this research made traditional tuning methods like grid search highly inefficient, almost akin to searching for a needle in a haystack. The Optuna framework, based on Bayesian optimization, offered a novel solution by intelligently narrowing the parameter space, significantly enhancing the effectiveness of the experiment.

Furthermore, insights from the expert interviews revealed a substantial gap between theoretical research and practical application in recommendation systems. Bridging this gap and applying theoretical findings more effectively in practice remains a critical area for further exploration and research.

This demonstrates how expert interviews helped shape the direction of model development and provided practical solutions that aligned theoretical research with real-world application.

## 5.2 Response to Experimental Question Two

How do traditional collaborative filtering models (e.g., matrix factorization) compare to deep collaborative filtering models in terms of accuracy and overfitting control in movie recommendation systems?

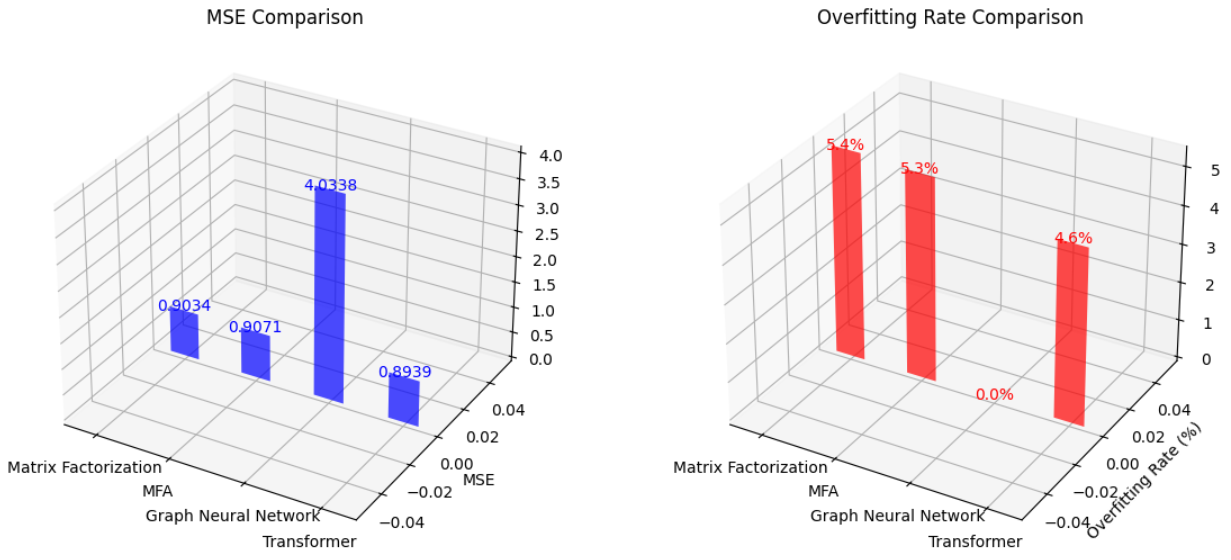


Figure 1 : Performance Comparison of Four Types of Models

Figure1 is evident that matrix factorization, as a bridge between traditional collaborative filtering and deep learning, can trace its core ideas back to the last century, yet its performance in practical applications remains outstanding, with an MSE of 0.9034 but a slightly higher overfitting rate of 5.4%. This also explains why, in an interview with an associate professor, it was mentioned several times that many companies still use similar matrix factorization models in production environments, especially in situations where the data size is relatively small or computational resources are limited. These traditional models have a wealth of practical experience to draw upon and can effectively meet the demands of many real-world scenarios.

In this experiment, if the overfitting tolerance were relaxed to 10%, matrix factorization would likely outperform all other models. However, with the continuous development of movie recommendation systems and the emergence of newer, more powerful technologies, the pursuit of more efficient recommendation systems remains urgent. This is why matrix factorization was chosen as the baseline model, aiming to compare it with current popular technologies and find models that could surpass it.

The initial idea was to improve matrix factorization by embedding different mechanisms. A hybrid model combining matrix factorization and the self-attention mechanism was created, with the hope that self-attention could further capture the interactions between users and movies on top of matrix factorization. However, as the chart shows, the MSE of matrix factorization actually worsened after introducing the self-attention mechanism. This could be because the hybrid model incorporated new technologies, increasing the complexity of the model and the number of hyperparameters, leading to overfitting or underfitting on certain datasets, especially sparse datasets like MovieLens 1M. The additional parameters did not effectively improve the model’s performance, which is why, in interviews with two industry experts, they mentioned that although the concept of hybrid models is attractive, there are not many cases of such models being applied in the industry. This is due to the high difficulty in integrating different technologies, which often results in negative outcomes.

Since 2017, Graph Neural Networks (GNNs) have developed rapidly. Compared to traditional recommendation algorithms, GNNs can capture higher-order relationships by considering not only user-item interactions but also the influence of neighboring nodes (such as similar users and similar items). Therefore, choosing GNN as a comparison model is well justified. Although in the initial version, which only included user and item features, GNN failed to effectively capture the latent interactions between users and movies, after the introduction of additional features such as movie genres and release years, the prediction accuracy of GNN improved significantly. This shows that additional feature inputs greatly contribute to enhancing GNN’s performance. Nonetheless, the experimental results show that although GNN eventually achieved perfect convergence with an overfitting rate of 0%, its accuracy was only 4.03, indicating that the model’s performance in practical applications still leaves much to be desired. GNN excels at handling complex user-item interactions, which is especially valuable in today’s information-rich era. As recommendation systems scale and diversify, GNN is expected to perform even better. However, experts in interviews also pointed out that despite experimenting with GNN, the performance of self-attention mechanisms often surpasses that of GNN, which is why introducing the Transformer model for comparison is of significant importance.

From the Figure 1, it is clear that the Transformer model, centered around the self-attention mechanism, was undoubtedly the best-performing model in this study. From the outset, it demonstrated strong stability and significant potential for improvement. To better understand the superiority of the model, an Ablation Study was conducted, gradually dismantling the Transformer model and removing ineffective parts. At the same time, data augmentation techniques such as noise addition and sequence modeling were introduced to control overfitting. In addition, structural modifications were made for horizontal comparisons of movie features. The final results confirmed that the self-attention mechanism layer was the key factor driving the model’s performance. Whether in terms of final accuracy (0.8939) or overfitting rate (4.6%), the Transformer model outperformed the traditional matrix factorization model. These findings also validate the insights of industry experts. The performance of self-attention mechanisms is often superior to other models in most scenarios, likely because they can simultaneously attend to each element in a sequence, capturing global dependencies in user-item interactions. In contrast, matrix factorization can only handle local user-item interactions. Additionally, self-attention mechanisms can capture user behavior sequences based on their viewing history, helping to understand the dynamic changes in user preferences. By comparison, while matrix factorization can uncover latent relationships between users and items, it cannot capture temporal information and typically only deals with static user preferences.

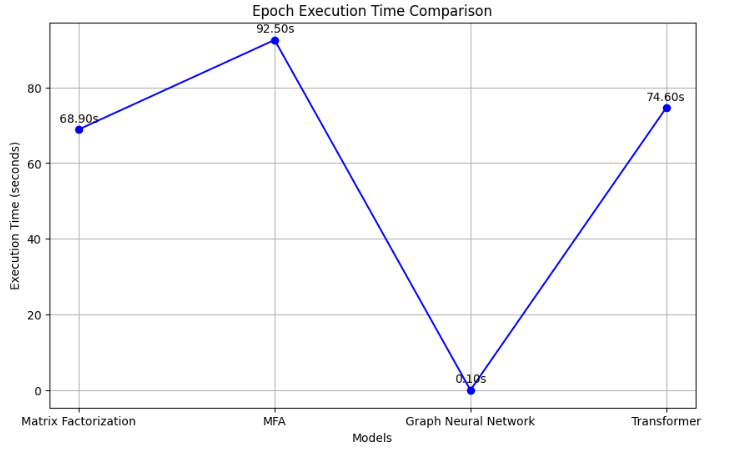


Figure 2 : Training Speed Comparison of Four Types of Models

In terms of training time, as shown in Figure 2, GNN had the fastest training speed, likely due to the sparse nature of the dataset and the relatively small number of features in the MovieLens 1M dataset. Next was the matrix factorization model, with an average training time of 68.9 seconds per epoch, 5.7 seconds less than the Transformer model’s 74.6 seconds per epoch. This indicates that matrix factorization has certain advantages in terms of model size and computational speed. In specific application scenarios, where recommendation accuracy requirements are not high but fast response times are critical, matrix factorization still has potential for use. For the hybrid model of matrix factorization and self-attention, the longest training time was 92.5 seconds per epoch, which aligns with its higher model complexity and is therefore unsurprising.

## 5.3 Limitations and Strengths of the Research

This study presents several notable limitations. First, in terms of primary data collection, expert interviews were conducted using the judgment sampling method. Although this approach effectively improved sampling efficiency and reduced the occurrence of invalid samples, it inevitably increased the risk of sample bias. Second, there was an imbalance in research efforts dedicated to different types of models. For instance, after introducing the self-attention mechanism, despite the decline in model performance, its potential was not thoroughly explored, and the focus shifted to other models. In fact, each model offers significant research potential, but in order to improve research efficiency, more attention had to be devoted to models that demonstrated stronger performance, with the goal of achieving better experimental results.Lastly, the dataset used in this experiment was not large, which also points to future research directions. By incorporating more complex and diverse features, model performance could be further enhanced.

The strengths of this study are equally noteworthy. First, it scientifically and effectively compared the performance of four deep learning models based on the collaborative filtering framework, with particular attention given to key results, such as the matrix factorization baseline model and the Transformer model. These models underwent multiple rounds of fine-tuning to ensure the objectivity and accuracy of the experimental results. Second, several key technologies were introduced in the study, significantly contributing to the smooth progress of the experiments. For example, mixed precision techniques significantly reduced memory usage during training, preventing memory overflow issues. The dynamic adjustment of learning rates using cosine annealing also played an important role. Based on Optuna's tuning results, the learning rate was confirmed as a key hyperparameter affecting model accuracy, and the introduction of dynamic learning rates greatly facilitated model training. Furthermore, the application of the SHAP framework significantly enhanced the interpretability of the Transformer model, which was initially treated as a black-box model. It clearly demonstrated the impact of each feature on model performance, providing strong guidance for the direction of the experiments.Lastly, selecting the already high-performing matrix factorization model as the benchmark presented its own challenges. Throughout the experiment, various explorations were conducted, including modifications to the model architecture, the application of different data processing methods, attempts at various feature introduction techniques, and the objective presentation of the entire experimental process through vertical and horizontal comparisons, as well as ablation studies. This study provides valuable references for future research on movie recommendation systems.

## 5.4 Contribution and Future Research

This study contributes to the growing body of research on recommendation systems by systematically comparing traditional collaborative filtering methods with modern deep learning approaches, particularly self-attention mechanisms and GNNs. It provides valuable insights into the strengths of these models in areas such as overfitting control, generalization, and model interpretability, specifically regarding the implementation mechanisms of movie recommendation systems and executable deployment strategies in real-world scenarios. Additionally, by integrating expert interviews, the study bridges the gap between theoretical advancements and practical applications, offering useful guidance for future model development in recommendation systems.

For future research, expanding the feature set by incorporating multimodal data—such as movie posters, user reviews, and background music—through large language models (LLMs) could provide a more comprehensive understanding of user preferences. Promoting diverse recommendations and ensuring user privacy through advanced data security measures will also be essential for refining recommendation systems. These advancements can help build more intelligent, robust, and user-centric systems that adapt to evolving preferences while maintaining high standards in data management.

Word count : 15850 words

# References

Abhishek Kumar Rai1Khanna2, Pawan SinghPooja. (25/04/2024). Movie Recommender System: Collaborative Filtering Methods.

AlibertiLuca, D'AnielloGiuseppe, GaetaMatteo, & MarzoloAlice. (2024). A Fuzzy Signature-Based Approach for Recommendation Systems. IEEE.

Angelo Cesar Mendes da SilvaFurtado Silva, Ricardo Marcondes MarcaciniDiego. (August 2024). Artist Similarity Based on Heterogeneous Graph Neural Networks. IEEE.

Anup KumarGrüning and Rolf BackofenBjörn. (November 2023). Transformer-based tool recommendation system in Galaxy.

Ashish VaswaniShaoamzeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, Illia PolosukhinN. (2017). Attention Is All You Need. 04 December 2017 Publication History.

BobadillaJorge，Ortega, Fernando，Gutiérrez, AbrahamJesús，Dueñas-Lerín,. (2024). Comprehensive Evaluation of Matrix Factorization Models for Collaborative Filtering Recommender Systems.

ChaurasiyaGoyani and NehaMahesh. (26th of august 2020). A Review of Movie Recommendation System: Limitations, Survey and Challenges.

Chen GaoWang, Xiangnan He, Yong LiXiang. (February 2022). Graph Neural Networks for Recommender System. WSDM '22: Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining.

Chengzhi MaoWu, Yingjie Liu and Zhiwei ShiZhifeng. (2024). Matrix Factorization Recommendation Algorithm Based on Attention Interaction.

Franco ScarselliGori, Ah Chung Tsoi, Markus Hagenbuchner, Gabriele MonfardiniMarco. (December 2008). The Graph Neural Network Model. IEEE.

HanafiSri Widowati, Ika Nur Fajri, Norhikmah,Arief Nur Rohman, Nanang Fitria Kurniawan ,Muhammad Akbar MaulanaAnik. (July27, 2024). Improvement of E-commerce Recommender System Using Hybridization of Bert Matrix Factorization and Attention Mechanism. International Journal of Intelligent Engineering & Systerms.

Ilya LoshchilovHutterFrank. (2016). SGDR: Stochastic Gradient Descent with Warm Restarts. Conference: International Conference on Learning Representations (ICLR).

Jesús BobadillaDueñas-Lerín, Fernando Ortega, Abraham GutierrezJorge. (July 2022). Comprehensive Evaluation of Matrix Factorization Models for Collaborative Filtering Recommender Systems.

LeónVenkatesan，Sandeep Bhatnagar，José Luis TinajeroSrinath. (2023). A Recommender System Based on Matrix Factorization Techniques Using Collaborative Filtering Algorithm.

Linhan XiaYang, Ziou Chen, Zheng Yang, Shengxin ZhuYicheng. (Jul 2024). Movie Recommendation with Poster Attention via Multi-modal Transformer Feature Fusion. IEEE.

MaheshaK., KumaraB.T.G.S., & BanujanKuhaneswaran. (2024). Movie Recommendation System Based on User Ratings and Critique.

MohammadFirdawsJurreyyah, & UrolaginSiddhaling. (2022). Movie Recommender System Using Content-based and Collaborative Filtering.

NaskarIshika, & JosephP.Niju. (February 2024). Implementation of Movie Recommendation System Using Hybrid Filtering Methods and Sentiment Analysis of Movie Reviews. IEEE.

PanjaitanGohzali，Dita MariaHernawati. (2024). Movie Recommendation System Model using Bisecting K-Means Technique and Collaborative Filtering. Vol. 3 No. 2 (2024): Journal of Multimedia Trend and Technology.

Paulius MicikeviciusNarang, Jonah Alben, Gregory Diamos, Erich Elsen, David García, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, and Hao WuSharan. (2018). Mixed Precision Training. International Conference on Learning Representations (ICLR) 2018.

Prakhar AnandAzam,Shrddha SagarSajid. (June 2024). A Hybrid Approach Towards a Movie Recommendation System. IEEE.

Priyanka MeelBano, Agniva Goswami & Saloni GuptaFarhin. ( 2020). Movie Recommendation Using Content-Based and Collaborative Filtering.

Sandhya AvasthiLata Tripathi, Namrata Dhanda, Satya Bhushan VermaSuman. (July 2024). Content Filtering‐Based Movie Recommendation System Using Deep Learning.

Scott M. LundbergLeeSu-In. (2017). A Unified Approach to Interpreting Model Predictions. Advances in Neural Information Processing Systems (NeurIPS).

SetiawanNilla，Erwin BudiArliyanna. (February 12, 2024). Film Recommendation System Using Content-Based Filtering and the Convolutional Neural Network (CNN) ClassificationMethods.

Shamal ShaikhRao Kagita, Vikas Kumar, Arun K PujariVenkateswara. (2023). Data augmentation and refinement for recommender system: A semi-supervised approach using maximum margin matrix factorization. arXiv.

Takuya AkibaSano, Toshihiko Yanase, Takeru Ohta, Masanori KoyamaShotaro. (2019). Optuna: A Next-generation Hyperparameter Optimization Framework.

Vuong Quoc NguyenNhan Tran,Thin Si NguyenKiet. (2024). A study on hybrid recommend system combined sentiment analysis with matrix factorization.

Wang Cheng KangMcAuleyJulian. (2018). Self-Attentive Sequential Recommendation. IEEE.

WuYuetong. (13-03-2024). Movie Recommendation System Using KNN, Cosine Similarity and Collaborative Filtering.

Xiaoyu DuWang, et al.Xiang. (2019). Sequential Recommendation with User Memory Networks. International Conference on Web Search and Data Mining (WSDM).

Yehuda KorenBell, Chris VolinskyRobert. (2009). Matrix Factorization Techniques for Recommender Systems. IEEE Computer.

Yuyue ZhaoWu, Xiang Wang, Wei Tang, Dingxian Wang, Maarten de RijkeJiancan. (July 2024). Let Me Do It For You: Towards LLM Empowered Recommendation via Tool Learning. SIGIR 2024.

Zeyu WangZhu,Zichao Li,Zhuoyue Wang,Hao Qin and Xinqi LiuYue. (2024). Graph Neural Network Recommendation System for Football Formation.

ZhanYangfei, XieHaozhi, HuangHuabin, & ChenSumei. (February 2024). Research on Movie Recommendation Algorithm based on Deep Learning. IEEE.

Zhiyu YaoChen, Sinan Wang, Qinyan Dai, Yumeng Li, Tanchao Zhu, Mingsheng LongXinyang. (2023). Recommender Transformers with Behavior Pathways. International Conference on Learning Representations (ICLR) 2023.

# Appendix

## 1 Interview One

Me: I'm very glad to have you here today.

Anonymous: Thank you.

Me: Currently, I'm doing my final project on a movie recommendation system. I'm trying to use some deep learning models to enhance the system's performance. I’ve tried self-attention, collaborative filtering, graph neural networks, and graph attention neural networks.

Anonymous: That sounds interesting.

Me: So today, I'd like to have a brief interview with you to gain some valuable insights from your experience. First of all, could you briefly introduce your current role and your experience with recommendation systems?

Anonymous: Sure. I work at a computer science school and also at a data analytics research center. I’ve been involved in recommender systems research for about 10 years now. During this time, I’ve worked on various research projects and also on a large industry collaboration with Samsung, where we built a recommender system platform for them and designed distributed recommender systems.

Me: That sounds like very extensive experience in the computer science industry.

Anonymous: Yes, I've worked with many different types of models, including deep learning, matrix factorization models, natural language processing for review-based and opinion-based recommender systems, and more recently, graph-based neural networks for recommendation systems. One of my students is currently finishing up a project on synthetic data generation for training recommender systems, using diffusion modeling and other techniques.

Me: Wow, I really appreciate you sharing your experience today. I’ve already prepared several questions for you and sent them via email. If you’d like, you can open them now. I'll ask you a few of these questions next.

Anonymous: Sure.

Me: The first question is: What types of models do you typically use in recommendation system problems?

Anonymous: I was looking at some of your questions earlier, and I noticed that the way they’re framed assumes that there’s one "right" or "best" model. This is a common assumption among students, but it’s not often the case. When building a recommender system, especially in a company, the focus is often on user modeling—understanding what kind of users they have—and tailoring the model to the specific domain, like what they are trying to sell or recommend to the user.

Me: I see. So, what kinds of models do companies tend to use in real-world recommendation systems?

Anonymous: Surprisingly, many commercial operators use very simple recommendation models. They prefer simple models like matrix factorization or collaborative filtering because they're easier to develop, train, and deploy. The more complex models—such as deep learning models—are harder to train, debug, and maintain, which is crucial for companies relying on recommender systems for revenue. A small bug can be challenging to fix if the system is too complicated.

Me: That makes sense. So, in your experience, which models are often used in real-world systems?

Anonymous: In many cases, companies use very simple models like matrix factorization. The choice of model also depends on the context—some domains might benefit from collaborative filtering, while others might require content-based models. For example, if you're building a recommendation system for a platform like TripAdvisor or Booking.com, content-based models might be more suitable because users are often more interested in specific hotels or trips rather than what other users have chosen.

Me: That’s interesting. Have you researched the application of deep collaborative filtering, self-attention mechanisms, graph neural networks, or adversarial neural networks in recommendation systems? Could you briefly discuss their performance?

Anonymous: Yes, I’ve worked with various models, including graph neural networks and a graph signal processing approach, which offers a different way of interpreting graph neural networks. The most commonly used model in recommender systems is still the Graph Convolutional Network (GCN), which opened up the use of graph neural networks in the field. Many newer models are based on this.

Me: What can you tell me about the performance of these models?

Anonymous: Well, when you look at papers, many claim that their graph neural network models are state-of-the-art, but it’s hard to gauge what that really means, as it depends on how they’ve benchmarked them. Companies tend to evaluate models on their own internal data. One major issue with deep learning and graph neural network models is the high training cost, which can be orders of magnitude higher than that of simpler models like matrix factorization. Often, companies don’t find the small performance improvement worth the extra cost or complexity.

Me: What about in academic settings—how do these models perform there?

Anonymous: In academic research, we’ve found that graph neural networks can be useful in specific cases, such as solving cold start problems or when we don’t have much user information but do have item information. We also did some theoretical work reinterpreting GCNs using a graph signal processing approach. While we did see small performance improvements, the major benefit was the simplification of training.

Me: That's really insightful. Do you think there’s a performance hierarchy among these models? Which model do you find the most optimal, and why?

Anonymous: That’s not really the right question to ask. Researchers and practitioners don’t typically think in terms of a "best" model. It’s more about what metric you’re optimizing for and the context of your problem. For example, some businesses might prioritize diversity in recommendations, while others care more about accuracy or user engagement. So, there’s no single optimal model—it really depends on the use case and what you're trying to achieve.

Me: I understand. Have you ever combined different models to leverage their strengths for improved performance?

Anonymous: Yes, what you're referring to is an ensemble approach. In academia, we don’t usually focus on ensembles because we’re more interested in evaluating specific models. But in industry, I imagine some companies use ensemble methods to improve performance. However, ensembles add a lot of complexity, and most companies prefer simpler models because they are easier to train and debug.

Me: In your experience, what was the biggest challenge you faced while using deep learning to enhance recommendation system performance, and how did you overcome it?

Anonymous: The biggest challenge, in my opinion, is usually related to data representation. Preparing the data in such a way that it can be utilized by deep learning models or graph neural networks is tricky. In academia, we often use standard datasets like MovieLens or Amazon reviews, but these datasets are formatted in ways that may not always be optimal for deep learning or graph-based approaches. Formulating the problem correctly is key. Another challenge, especially with reinforcement learning, is generating the right kind of data to train the models effectively.

Me: So, data processing and representation are the main hurdles?

Anonymous: Yes, for me, most issues come from the data. Once the data is correctly formatted and processed, training the model becomes much easier. The real difficulty is ensuring that the data captures the necessary features and aligns with the metrics or goals of the model.

Me: That’s very helpful, thank you for sharing your insights!

Anonymous: My pleasure.

Me: We can skip this question. What are the limitations of current evaluation metrics for joint models? Or how could they be improved? When you build models, you definitely need to evaluate their performance. What limitations have you faced with current evaluation metrics? And which types of metrics do you often use for evaluation?

Anonymous: Evaluation is a big issue in the field of recommender systems. There are many papers that discuss different evaluation metrics and why we would use one over another. I don’t think I can cover the entire field here, but in summary, this is one of the most important areas of recommender systems. When people ask for a recommender system, the hard part is not building the model itself, but understanding the appropriate metric to use. Often, things like precision or accuracy are not enough. We frequently use ranking metrics like NDCG (Normalized Discounted Cumulative Gain), which emphasizes that better items should appear higher on the recommendation list.

Me: For example, I'm building a movie recommendation system based on the MovieLens dataset.

Anonymous: Yes, but it depends on how you implement it. For example, Netflix uses very simple recommender systems, but for various purposes. If you’re familiar with Netflix, they show strips of movies—maybe 3 to 5 movies in a row. Those movies are the output of a recommender system. The system decides which movies to show in a specific order. They even use recommender systems to decide the thumbnails of movies because different people respond differently to various images.

Me: So, the metrics they use depend on the purpose?

Anonymous: Exactly. And they also consider context. For instance, Netflix might change recommendations depending on the time of day, or whether it’s a weekday or weekend. They even factor in whether you’re watching with family or alone. Your recommendations are influenced by what you’ve watched previously and by what’s popular among other users, adding a social aspect.

Me: In my case, I’m using the MovieLens dataset, and I’m comparing several deep learning models like self-attention and graph neural networks. I’ve been using MSE (Mean Squared Error) as my evaluation metric. What do you think about that? Is there any limitation to using MSE?

Anonymous: MSE is perfectly fine if you’re doing rating prediction. If you’re predicting ratings, then MSE or MAE (Mean Absolute Error) are good choices. You could also consider producing ranked lists and use ranking metrics like NDCG. But that’s a different task—ranking instead of rating prediction. You’d need to check how high a recommended item appears on the list for the user. For example, Netflix is more concerned with ranking, and they’re happy if one of the movies in the top 5 that the user sees immediately gets picked.

Me: I see. It’s a ranking problem, not just a rating prediction one.

Anonymous: Exactly. It's about how close you can push the right item to the top of the list.

Me: In your experience, have you ever encountered issues with insufficient computational power? What improvements have you made to reduce dependency on hardware and enhance training speed?

Anonymous: Yes, this is a common issue. I don’t know anyone in academia who says they have enough computing power. We are always looking for more. When we get more computing power, we just try bigger models, which continues the cycle. Some people work specifically on scalability, creating models that achieve similar accuracy but are faster and more efficient to train. But that’s a specific research area and not a primary focus for me. In academia, we mostly care about testing the model architecture, not necessarily optimizing for hardware constraints.

Me: That makes sense. I’m currently facing some computational limits with my own project. Do you think using a smaller dataset for training deep learning models to compare performance is feasible for academic research? What issues might arise from this practice?

Anonymous: This is actually very common. I always advise my students to start with a subset of their data. When you’re developing a model, you often retrain it multiple times due to modifications in the architecture, data representation, or metrics. Training on the full dataset each time would be computationally expensive. So, you need an efficient training loop, with a subset of the data that’s representative of the entire dataset. The challenge is ensuring that this subset is indeed representative, which is tricky.

Me: Yes, sampling seems to be the key.

Anonymous: Correct. There are papers on how to sample recommendation datasets effectively. Most people either sample by users or items, depending on the context. For example, sampling by users might be a good approach in your case.

Me: I’ve tried using the MovieLens 20M dataset, but my laptop couldn’t handle it, so I switched to MovieLens 1M, which works fine. But now I’m worried about hyperparameter tuning because that stage requires a lot of computational power. I was thinking of using 10% of the dataset for hyperparameter tuning. Do you think that’s possible?

Anonymous: Yes, that’s generally what we do—use something like a 70/10/20 split, with 10% for validation, 10% for testing, and the rest for training. Another option is to use Bayesian hyperparameter optimization instead of grid or random search, which is very inefficient. Bayesian optimization is much faster and just as effective.

Me: Can you repeat the name of that method?

Anonymous: Bayesian hyperparameter optimization. It uses prior sampling of the probability of finding good hyperparameters to adjust the search space intelligently.

Me: I’ll look into that. Thank you for the suggestion!

Anonymous: You’re welcome! You’ll find Bayesian optimization in libraries like skopt and hyperopt. They come with built-in Bayesian samplers.

Me: That’s great to know. Thanks again for your help!

Me: In the actual production process, what are your criteria for selecting a model for deployment? Could you provide the top three most important factors you consider, ranked in order of priority, such as model accuracy, hardware requirements, computational time, or other metrics?

Anonymous: It’s hard to say definitively because every company I've seen has a different approach, set of metrics, and systems. Some outsource their recommendation services, using an external API, which reduces infrastructure needs. Others, with skilled data science teams, might use something like Spark for distributed training due to large data requirements. If you're in the business of building your own recommender systems, you likely need a big data science team with senior researchers, machine learning engineers, and software engineers. It's a big operation that requires database experts and cloud engineers for deployment. Different companies may use different programming languages—some in Java, others in Python or even C++. And once you add GPU-based models, it gets more complicated.

One big difference in commercial implementations is that the business side is heavily involved. If a slight tweak to the recommendation model reduces revenue, you're in trouble. Business teams are cautious, and any changes need to undergo rigorous A/B testing to ensure they won’t negatively impact revenue. That’s a challenge we don’t really have in academic settings.

Me: I see. So, you’re saying the business department plays a major role?

Anonymous: Yes, they do. In commercial recommenders, the academic perspective of accuracy metrics like MSE or NDCG isn’t the only focus. They also optimize for business goals. For example, Netflix is selling movies, Amazon is selling books, etc. The recommendation system isn’t just serving the user, but also business partners. For instance, Amazon might push sponsored items higher up in the list due to paid promotions, even though they may not be the most relevant for the user.

Me: So, revenue generation becomes a crucial factor?

Anonymous: Yes, exactly. On Amazon, for example, when you search for something, the items at the top might not necessarily be the best for you. They could be promoted by companies paying to bump up their items. So, there’s always a tension between what’s best for the user and what’s best for the company.

Me: I understand. In real-world cases, what would you say are the top three factors you consider when evaluating a model?

Anonymous: In commercial settings, accuracy is important, but speed might be even more crucial, depending on the context. For example, Spotify uses recommenders to create playlists, and they can do this overnight without time pressure. However, when recommending the next song, it has to be almost instant, within milliseconds. Latency is a critical factor there.

On Amazon, they’ve noticed that if page response times increase from 100 milliseconds to 200 milliseconds, people get frustrated and buy less. Speed makes a difference to conversion rates, so it’s really important to optimize for that.

Me: So, it’s highly context-dependent?

Anonymous: Yes. If the recommendation has to be delivered in under 100 milliseconds, the model has to be very simple. Complex deep learning models, unless backed by massive infrastructure, just won’t work with such tight latency constraints. Companies often opt for simpler models that are easier to scale and troubleshoot.

Me: I get it now. Simpler models are faster and more reliable in critical scenarios.

Anonymous: Exactly. If you’re dealing with millions or billions of requests, simplicity matters. Complex models may break more easily, and fixing them could be a big challenge. With simpler models, it’s easier to troubleshoot and maintain low latency.

Me: That makes sense. I actually interviewed a developer from a major Chinese company, and they handle billions of requests for their recommendation system. They need powerful infrastructure to manage this.

Anonymous: Yes, I can imagine. Companies with that kind of scale need to optimize for simplicity. They also pay a lot for compute resources and data storage to keep these systems running efficiently with low latency.

Me: That brings up a good point. In your opinion, what are some potential future research directions for movie recommendation systems and the current challenges they face?

Anonymous: It’s hard to say. I go to conferences like RecSys every year, and while Netflix is present, there isn’t a huge focus on movies specifically. Researchers often use MovieLens to evaluate other recommender systems. The real challenges are more around user modeling, context understanding, contrastive learning, and graph neural networks. Reinforcement learning is also a big area. However, I’m not sure how movie recommendation alone could be a major research challenge—it’s just not a problem that affects a large portion of the research community.

Me: I see. Academics don’t have access to specific movie recommendation data, which limits what they can work on.

Anonymous: Exactly. It’s more about using these datasets to validate broader recommender systems research.

Me: Well, I think we’ve covered all the questions I had. I really appreciate your time and insights today. This has been very helpful for my project.

Anonymous: Glad I could help. Best of luck with your project!

Me: Thank you! I hope we can talk again in the future. Take care!

Anonymous: You too!

## 2 Interview Two

Me: Hello!

Anonymous: Hello!

Me: Can you hear me?

Anonymous: Yeah, of course.

Me: Glad to see you here. Good evening.

Anonymous: Yeah, good evening.

Me: My name is John , and I'm happy to have you here today for this academic research interview. Could you please briefly introduce yourself and your experience with recommendation systems?

Anonymous: Sure. It's a great honor for me to be here for this interview. My name is Anonymous, and I'm currently in Ireland, but before that, I was a data scientist in China. I worked at several companies, mostly focusing on recommendation systems. The last company I worked for is a large company in China, similar to Quora, where I was part of the recommendation team.

There were several business parts in my team, one related to education and another to paid articles. We focused on recommending articles to users. My role involved building models and creating strategies for these recommendations. Before that, I worked at a financial company, but the algorithms there were different. However, many of the fundamentals are similar to recommendation systems. My first job was also related to recommendation systems, specifically building a video recommendation application. In that role, I had to consider algorithms, strategies, and backend workloads, which are more complex in industry compared to academic research.

Me: It sounds like you have a lot of practical experience with recommendation systems. You're definitely an expert in this area.

Anonymous: I guess so. Thank you!

Me: For the next part of the interview, I’d like to ask you a few questions about your experience with recommendation systems. First, let me briefly introduce my project. I’m building a movie recommendation system using the MovieLens public dataset. It's a popular dataset, and I'm trying to enhance the recommendation system using several deep learning methods like self-attention, graph neural networks, and adversarial neural networks. My goal is to create a hybrid deep learning collaborative filtering system.

So, for the first question, what type of models do you typically use in recommendation system problems in the industry?

Anonymous: In the industry, we usually deal with massive amounts of data—sometimes over a billion data points a day. So, the models we use are mostly large-scale neural networks. These aren't small models; they often have hundreds or thousands of parameters.

Me: So, they’re very deep with many layers, right?

Anonymous: Yes, exactly. Some models have many layers, but in other cases, we use fewer layers with a large number of neurons per layer. For example, we might have 1,024 neurons in a single layer, but only three to six layers. So, the network isn’t necessarily deep, but each layer processes a lot of data.

Me: Is there a specific name for this kind of neural network?

Anonymous: Well, before I left the company, we were using two main types of neural networks. One was developed by Google, known as the YouTube DNN, and the other was created by Alibaba, called ESSM.

Me: So, this neural network has multiple layers with a large number of neurons. Is it a typical neural network, or does it have a special structure like LSTM or GRU?

Anonymous: No, it doesn’t have a special structure like LSTM or GRU. To give you an example, in recommendation systems, the challenge is to quickly recommend a small number of items—say 10 videos—from a large pool of millions. This has to be done very quickly, within 100 to 200 milliseconds. We typically use models like YouTube DNN, where we input user data into the network. The network has several layers with different numbers of neurons, and after training, we get a lot of parameters.

We focus on the parameters rather than just the output. For example, we might train the network and then extract one layer of neurons to use for other tasks. This is a common approach in the industry.

Me: I see. Let’s move on to the next question. Have you ever used or researched the application of deep collaborative filtering, self-attention mechanisms, graph neural networks, or adversarial neural networks in recommendation systems? If so, could you discuss their performance?

Anonymous: Yes, I’ve worked with several of those models. I’ve used graph neural networks and self-attention mechanisms, and in our context, most deep learning models for recommendation systems are essentially forms of deep collaborative filtering. The data we use, whether it's user-item interactions, is inherently collaborative.

Me: So, it could be either user-based or item-based collaborative filtering, correct?

Anonymous: Exactly. Models like YouTube DNN and ESSM are all based on deep collaborative filtering. I’ve also tried self-attention models and graph neural networks. However, while graph neural networks may sound fancy in academic research, they don’t perform as well as self-attention models in real-world applications. Self-attention models are more precise and straightforward, and they scale better.

Me: So, self-attention is faster than graph neural networks?

Anonymous: Yes, I would say it works better. If you use this algorithm, you get higher accuracy, and if you use typical metrics like AUC or something similar, self-attention tends to perform better.

Me: So, in terms of performance, self-attention is superior?

Anonymous: Yes, it's currently the best model in our industry, especially in China. Most companies use self-attention models.

Me: Is it the multi-head self-attention mechanism, or just a single head?

Anonymous: In our company, we use a single head, but I can't say for sure whether the multi-head would be better. We would need to conduct experiments to determine that.

Me: Yes, testing is crucial.

Anonymous: Exactly. There are explanations suggesting that multi-head could be better, but in practice, it depends. In our industry, the difference may not always be significant. We would need to experiment.

Me: I understand. What about adversarial neural networks? Have you had any experience with them?

Anonymous: No, we haven't used adversarial networks in our recommendation system. I haven’t seen any examples in my team either.

Me: So, you think adversarial networks are more suitable for other industries, like image or video generation?

Anonymous: Yes, exactly. They might work better in those fields, but in recommendation systems, they’re too complicated and not very common.

Me: Is it because adversarial networks are slow and computationally expensive?

Anonymous: Yes, speed is crucial for recommendation systems, and adversarial networks are too slow and complex. That’s why we don't use them.

Me: Got it. Do you think there is a performance hierarchy among these models? Based on what you've mentioned, it seems that self-attention is the most optimal, followed by graph neural networks, and adversarial networks are not used. Am I right?

Anonymous: Generally, yes. Self-attention is often the best model, but I wouldn't say that it’s always the case. In some situations, a different model might perform better. So, it depends on the specific case.

Me: But in most cases, self-attention tends to give the best performance?

Anonymous: Yes, generally speaking.

Me: And the most common model you’ve used is the YouTube DNN?

Anonymous: Yes.

Me: Have you ever tried combining these models or others to leverage their individual strengths for improved performance?

Anonymous: No, we haven't tried that. It sounds fancy, but in practice, we prioritize speed. Making the model heavier by combining different architectures could slow things down.

Me: But it might give higher accuracy. What do you think?

Anonymous: Yes, that’s possible, but we haven’t tried it.

Me: I see. It sounds promising but might be challenging in terms of efficiency.

Anonymous: Yes, exactly. Combining models with different principles, like self-attention and graph neural networks, could be beneficial because their strengths might complement each other. However, combining similar models might just make things more complicated and could even worsen the performance.

Me: I agree. What was the biggest challenge you faced when using deep learning to enhance recommendation system performance, and how did you overcome it?

Anonymous: That’s a big question. One of the biggest challenges was building a pipeline that could handle quick responses. We had to ensure that our system could process a large amount of data and respond to users in less than 300 milliseconds, which was difficult given the size of the input data.

Me: How did you overcome that challenge?

Anonymous: We solved it by separating the pipeline into different parts. Instead of trying to do everything at once, we broke the process into smaller tasks. For example, we divided the recommendation process into three parts: recall, ranking, and re-ranking. Each part focuses on a specific task, making it easier to handle and quicker to process.

Me: So, you divided the pipeline, but did you run the different parts at the same time?

Anonymous: No, we didn’t run them simultaneously. We had to finish one part before moving on to the next. The key was to make each part as efficient as possible.

Me: What was the advantage of dividing the process like that?

Anonymous: Dividing the process allowed us to reduce response time because each part focused on a specific task. By keeping each step clear and precise, we could make the system faster and easier to analyze.

Me: So the goal was to reduce response time by breaking down the tasks and making each step more manageable?

Anonymous: Yes, exactly. It’s a major challenge in recommendation systems, and most data scientists in the industry spend a lot of time optimizing these pipelines to reduce response time while maintaining accuracy.

Me: Thank you for sharing your insights!

Interviewee: Reduce the time and effort because recommendation is a very complex task. So, we need to do it step by step.

Me: If you put them together, what will be the problem?

Interviewee: It will be too heavy. For example, when we get a request from the user, there may be millions of candidates. But we don’t want to recommend all of them to the user, maybe just 1%. If we try to do everything in one go, we would need to build a very large model, which would bring a lot of burden on our system. By separating the process, it becomes clearer and easier to analyze.

Me: So, you do some preprocessing as the first stage, and based on that, you proceed to the next step, right?

Interviewee: It’s not exactly preprocessing. It’s more about separating the pipeline. For every part, we build several different models. Each model focuses on different problems. If we don’t separate them, we’d need a massive model. But if we break it into parts, the models will be smaller and more precise.

Me: Okay, so you build different models for different tasks to solve separate problems. If you mix them together, the model becomes too large and heavy. Is that correct?

Interviewee: Yes. Also, a large model is harder to analyze. One big model to handle everything would become very complicated.

Me: I understand. How do you combine these separate models in the end?

Interviewee: For each part, we focus on a different question. First, we might start with millions of candidates, but we want to reduce that number. The first model reduces it to maybe a few hundred candidates. Then the second model reduces it further, down to 20 or 30. The last model will select the top 10. Every part has its own model, and each one narrows down the number of candidates.

Me: And you use self-attention for this?

Interviewee: Yes, but we only use self-attention in the last part to make things really precise.

Me: What about at the beginning?

Interviewee: In the beginning, we have millions of candidates, so the focus isn’t on precision, just on narrowing down the pool. After the first stage, we still have hundreds of candidates, and we’re just looking for some close matches. At this stage, we don’t have very high demands for the model.

Me: Thank you, I understand. Now, what are the limitations of current evaluation metrics for joint models, and how could they be improved?

Interviewee: There are many strategies and methods, but I’ll answer from a model perspective. For a model, we want it to be as precise as possible. There are three key areas we focus on:

1. Data: We want to feed as much data into the model as we can. We create lots of records and do data sampling and cleaning.
2. Model Complexity: We want to make the model more complex, adding layers beyond just the input and output layers. For example, we might try inserting self-attention or other structures into the neural network.
3. Metrics: We need to design appropriate metrics to evaluate the system’s performance. These metrics help us understand whether the model is doing well or not.

Me: Which kind of metrics have you used before?

Interviewee: Actually, AUC.

Me: Yes, I’m familiar with that.

Interviewee: It's the most common metric we use, and sometimes we use GAUC, but AUC is the most important one. In fact, it's often the only one we use.

Me: Okay.

Interviewee: For other tasks that aren’t related to recommendations, people might use metrics like accuracy. But for recommendations, AUC is better.

Me: I’m using MSE in my case.

Interviewee: MSE isn’t as precise in recommendation systems. AUC is better because it’s a ranking-based metric. In recommendations, it’s all about ranking. If your ranking is good, the AUC will be higher.

Me: Yes, I see.

Interviewee: AUC is a really good metric for us to evaluate performance.

Me: Thank you for your answer. We can move to the next question. This is the one I’m most interested in because it’s about your real-world experience in the industry. Could you discuss the structure of recommendation systems in the industry, along with the models you use most frequently and their performance? What is the reasoning behind your choice of these models?

Interviewee: Which part do you want me to focus on?

Me: You already explained part of it by saying that you build a pipeline and divide the process into different parts. You also mentioned using self-attention models in the last part for precision.

Interviewee: Yes.

Me: So, what about the first and second parts? Which models have you used for those, and how well do they perform?

Interviewee: For the first part, we often use models like YouTube DNN. The key point is that we don’t focus on the output in the first stage.

Me: Output, okay.

Interviewee: Instead, we focus on what happens after we train the model. We use the layers to do something, not just focus on the output. After training the network, we collapse the layers and use them to conduct some search tasks. The key is that the model should create precise vectors for us after training.

Me: So, it’s a kind of transfer learning?

Interviewee: No, it’s more like learning representations. We want to learn the representation of the item or the user. In the beginning, we just have raw data for the items or users, but after training, we transform each item into a vector that represents it. This vector can be considered a representation of the candidate or the user.

Me: So, you can collect information from both the users and the candidates?

Interviewee: Yes, we do this for both users and candidates. YouTube DNN is really good for that.

Me: I see. For the first part, you’re not focusing on the output but rather on learning the behavior of the candidates. Is that right?

Interviewee: Yes.

Me: What about the next part?

Interviewee: In the next part, we already have the candidates and users. We continue using the vectors from the first part and add more information. Then, we build another network. Here, we focus on the output because now we want to score the candidates. If a user likes a certain candidate, that candidate’s score should be higher. So in this part, we focus on the output and use different metrics to evaluate the results.

Me: So, what kind of models do you use for this ranking?

Interviewee: Usually, we use models like LightGBM or any kind of ranking models. These models are good for ranking candidates because we have many candidates, and some should have higher scores than others. It's a ranking problem, so we often use LightGBM or other similar models.

Me: Okay.

Interviewee: You can also use self-attention in this part, but it’s a choice. Sometimes, we put it in the last part. We also want to avoid making the model too complex because it would consume a lot of time. We want it to be precise, but not too precise.

Me: I see.

Interviewee: There are technical challenges behind this, but they’re more backend-related and not model-related. We need to use backend technology to make the response quicker.

Me: So, in this part, you focus on making the model as precise as possible?

Interviewee: Yes, especially in the last part where we need higher accuracy.

Me: Thank you, I think I understand now.

Interviewee: From our perspective, we design things differently for each part because the problems are different. Sometimes the model is very complex, and other times, we just need it to be quick and precise enough without being too precise. It’s a trade-off.

Me: That’s very interesting. In academic research, we often focus only on the final results and aim for the highest accuracy. It’s good to know about these different approaches.

Me: Since you're studying in this field, can you discuss the potential future research directions for movie recommendation systems, the current challenges, and possible solutions?

Interviewee: Right now, the state-of-the-art research in recommendation systems is going in two directions. One is to make the models more precise, often by using self-attention. We might change the structure, the layers, or the timing of when to use self-attention to make things more complicated.

The second direction is what a team from Facebook is doing—they rebuilt the whole pipeline for recommendation systems from scratch. They call it TT Structure, and they use large language models, like a ChatGPT-like model. Their approach is totally different from the traditional methods.

Me: Okay, I understand.

Me: I think we’ve covered most of the questions. Thank you for your valuable answers and your time. It’s getting late, so we can wrap up here.

Interviewee: Yes, thank you.

Me: Thank you again for sharing your experience. I hope we get another chance to discuss this further.

Interviewee: Yes, absolutely.

Me: Have a good night.

Interviewee: You too, good night.

## 3 Interview Three

John: Good evening. I’m very glad to have you here. My name is John, and I’m doing a master’s degree at CCT College, Dublin, in Data Analytics. For my final project, I’m working on a movie recommendation system.

Today, I’m very happy to talk with you, Mr. [Anonymous]. First, let me briefly introduce my project. My final project is about a movie recommendation system, and I’m trying to use self-attention on graph neural networks, along with adversarial neural networks, to enhance the performance of the recommendation system. For the dataset, I’m using MovieLens 1M.

Could you briefly introduce your current role and your experience with recommendation systems, please?

Interviewee: Yes, recommendation systems are used to match the similarity between items. For example, if you go to McDonald’s to buy a burger, you might see recommendations suggesting that you buy a drink or fries along with it. That’s the basic idea behind recommendation systems.

While my current research is not directly related to recommendation systems, I have worked on them in the past. These days, there are many models developed using deep learning techniques for recommendation systems.

John: Could you tell me more about your current role and your experience with recommendation systems?

Interviewee: Sure. My name is [Anonymous], and I’m a PhD student atXXX, currently working as a dissertation supervisor. My research area is data augmentation for images. I mostly work with computer vision problems.

In my master’s degree, I worked on recommendation systems, although not extensively. I do understand the basic concepts behind them.

John: Thank you. So for the next part, we’ll discuss several questions. First, in your experience, what type of models do you typically use in recommendation system problems?

Interviewee: In recommendation systems, I usually work with machine learning models like Naive Bayes, which is based on probabilistic reasoning. I also use collaborative filtering and sometimes k-nearest neighbors (k-NN), which is a similarity-based method.

John: k-NN is a machine learning method, right?

Interviewee: Yes, that’s correct. It's a machine learning method.

John: Have you ever used or researched the application of deep collaborative filtering, self-attention mechanisms, graph neural networks, or adversarial neural networks in recommendation systems? Could you discuss their performance?

Interviewee: I’ve used collaborative filtering in some projects, like one involving a store to recommend items, and another in the hospitality sector for hotel items. However, I haven’t personally worked with the other techniques like self-attention, graph neural networks, or adversarial neural networks.

In terms of performance, I saw better results with machine learning ensembles, like combining Naive Bayes with k-NN and decision trees.

John: I see. So in your collaborative filtering models, were they user-based or item-based?

Interviewee: They were mostly user-based. For example, I would try to find patterns in user behavior. If several users are buying bread, I’d look at what other items they’re purchasing—perhaps some also buy milk, while others buy eggs. Based on these patterns, the system would recommend eggs if most people who buy bread also buy eggs. So, it was based on user preferences.

John: I understand. Thank you very much for your insights!

John: In which recommendation problems have you worked?

Interviewee: It wasn't a movie recommendation system. It was for a shopping system. We had to recommend items in a shop. I worked on it during my master’s studies, about two or three years ago.

John: Okay. Have you ever tried combining models or using other techniques to leverage the individual strengths of different models for improved recommendation system performance? What were the results?

Interviewee: Yes, I have tried using ensemble modeling. I believe ensembles can definitely improve performance. For example, if you have two models performing well and one performing poorly, combining them can push the overall performance higher. It definitely works to improve the results.

John: Do you have any specific results you can share? How much improvement can we expect?

Interviewee: Yes, it improves performance. From a research perspective, even a 0.1% improvement is considered significant. In my case, the improvement was around 1–1.7% in absolute terms, which was quite a substantial improvement.

John: Thank you. Now, in your experience, what was the biggest challenge you faced while using deep learning to enhance recommendation system performance, and how did you overcome it?

Interviewee: The biggest challenge is usually related to the dataset. If the dataset is large, it can be difficult to load. Also, handling missing or null values is a challenge. If you fill null values with zeros or some constant, it can make the data sparse, and this can negatively affect the model's performance. Additionally, if you fill the nulls with different values, it can change the model’s decisions.

John: I see. So how did you deal with this?

Interviewee: What I did was count the occurrences of different values in the column and fill the null value with the most frequent value. For example, if one value appeared twice, and others appeared only once, I filled the nulls with the most frequent one. Alternatively, if the column contained floats, I would use the average to fill the missing values.

John: I see, that makes sense. Can you describe any specific architecture considerations or optimizations that are necessary for integrating models?

Interviewee: The structure of the model plays a huge role, and optimization is very important. When training large models, it can take a lot of time. I’ve used Bayesian optimization, which helps find the optimal parameters in a shorter time compared to other methods, like grid search. This process, called hyperparameter tuning, can significantly improve performance.

John: So you used Bayesian optimization for hyperparameter fine-tuning, correct?

Interviewee: Yes, exactly.

John: Which method have you used for hyperparameter fine-tuning?

Interviewee: I’ve used Bayesian optimization to optimize the parameters of decision trees. For instance, you have to tune parameters like the depth, which refers to the number of branches or subtrees in the decision tree.

John: So you used Bayesian optimization to fine-tune the hyperparameters of decision trees?

Interviewee: Yes, exactly. I used Bayesian optimization to optimize the parameters of decision trees.

John: Did you use any other methods like grid search?

Interviewee: Yes, besides Bayesian optimization, you can also use grid search or random search. There are many methods available.

John: What are the limitations of current evaluation metrics for joint models, and how could they be improved?

Interviewee: There are several limitations. For example, if you're only using accuracy as a metric, the accuracy could be high, but other metrics like recall or precision could be low. Even metrics like the F1 score could suffer. To address this, you shouldn't rely on a single metric. Instead, use multiple metrics and analyze the performance from different angles. For instance, you can use a confusion matrix to see which classes the model is predicting incorrectly.

John: Could you clarify what you mean by a confusion matrix?

Interviewee: A confusion matrix shows the actual labels versus the predicted labels. For example, in a 2x2 matrix for binary classification, you can see how many true positives, false positives, true negatives, and false negatives your model predicts. If your model is consistently predicting more positive labels than negative ones, it might be biased towards the positive class. The confusion matrix helps identify these kinds of issues.

John: Oh, I see! That’s interesting.

Interviewee: Yes, from the confusion matrix, you can calculate accuracy, precision, recall, and other metrics. It gives you a more comprehensive view of the model’s performance rather than just relying on accuracy alone, which might not tell the full story.

John: What about the regularization problem?

Interviewee: Regularization is important, especially when using models like support vector machines (SVMs). Proper regularization helps prevent overfitting by controlling the complexity of the model. There are different techniques, like L1 and L2 regularization, which add penalties to the model's complexity and help in balancing performance and generalization.

Interviewee: There are two types of regularization, L1 and L2. Regularization is like adjusting the model's flexibility. Think of it like adjusting the volume on a mobile phone—you control how much flexibility you want to give to the model. It’s a hyperparameter that you should tune based on optimization techniques.

John: I see. In the case of regression problems, for example, in a movie recommendation system where I want to predict ratings, we are focusing on metrics like Mean Squared Error (MSE) to evaluate the model. Do you think there are any limitations to using MSE?

Interviewee: Well, in regression, you usually have parameters like the bias and variance, represented by terms like "a" and "b" in a simple equation like Y = aX + b. These parameters are heavily dependent on the dataset, and while training the model, the evaluation metrics you use, such as MSE, influence these parameters. The limitations are often tied to the dataset itself, so it’s hard to point out an exact limitation without considering the dataset.

John: That makes sense. It’s more about the dataset and how it affects those parameters.

Interviewee: Exactly.

John: Have you ever faced issues with insufficient computational power? What improvements have you made to reduce the dependency of your models on hardware and to enhance training speed?

Interviewee: Yes, I’ve encountered that problem. A simple technique I use is pruning and knowledge distillation. Pruning helps reduce the model size by cutting away unnecessary parts, and knowledge distillation involves transferring knowledge from a large model to a smaller one. Also, I use data augmentation to artificially create more data during training. This helps a small model learn more effectively without needing excessive computational power.

John: So you mean we can take a sample and create multiple variations of it?

Interviewee: Yes, exactly. Let’s say you have one image of a dog. You can apply different transformations like flipping, rotating, or adjusting the brightness to create 10 different versions of that single image. By passing those 10 variations through the model, it learns more compared to just training on one image.

John: I see. That’s a clever approach to maximize learning with fewer resources.

Interviewee: Yes, and if you apply pruning and knowledge distillation, you can further reduce the model size and still maintain good performance. But for this, you would still need sufficient computational power for the initial stages.

John: I’ve encountered a similar issue when working with the MovieLens 20M dataset. I can only run the model on 30% of the data due to resource limitations.

Interviewee: You can process 100% of the data by splitting it into chunks. Load 20% of the data at a time and train the model in iterations. After processing one chunk, you can release the memory and load the next chunk. This way, you can train on the entire dataset in smaller parts.

John: But how do you handle the training and testing datasets in this approach?

Interviewee: You can fix a small percentage, say 5%, of the dataset as the test set and keep it aside. Then, you split the remaining 95% into smaller chunks for training. After each chunk is processed, you update the model parameters.

John: That sounds like a good approach! I’ve never tried it this way, but it seems very feasible.

Interviewee: Yes, it works. I used this method while working at Samsung during a competition in Korea. We won $20,000 in that competition using this approach.

John: Wow, that’s impressive! I’ll definitely give it a try.

John: So you mean we can split the dataset, keep maybe 10% as a test dataset, and then use a loop to train on 20% of the training set at a time. After the loop, we use the test dataset for evaluation, right?

Interviewee: Yes, but in the loop, you should evaluate the test dataset after each chunk to see how the performance is improving. This way, you can observe how each chunk affects the overall model performance.

John: So we use the same test dataset every time in the loop for fairness, right?

Interviewee: Exactly. We need to keep the test dataset fixed. If you change the test set each time, you won't be able to evaluate the model performance consistently.

John: Got it. We separate the test set and load the training data in chunks. And for the best-performing chunk, what should we do?

Interviewee: For the best-performing chunk, you should save the results and weights, and then you can load that chunk for final training.

John: Ah, I understand now. Save the weights and treat it as the final trained model.

Interviewee: Yes, exactly.

John: Good idea! I think this will work.

Interviewee: Yes, it should work.

John: So, do you always use this kind of strategy for dealing with insufficient computational power, or do you use other methods?

Interviewee: I used these techniques a few years ago when I was doing competition work. Nowadays, there might be more advanced techniques, but what I’ve shared are traditional and effective methods.

John: I see. Another question: Do you think using smaller datasets due to computational limitations to train deep learning models for performance comparison is a feasible approach in academic research? What issues might arise from this?

Interviewee: Using a small dataset is okay, but the dataset must be balanced. If it's not, the model might become biased toward one class, especially if there are unequal class distributions. You should check for balance or use techniques like oversampling, undersampling, or SMOTE to handle the imbalance.

John: So, you think small datasets are fine as long as they are balanced?

Interviewee: Yes, otherwise the model could be biased.

John: What are your criteria for selecting a model for deployment? Can you provide three important factors you consider, such as model accuracy, hardware requirements, or computational time?

Interviewee: It depends on the requirements. If accuracy is the priority, you might have to compromise on hardware or time. If the hardware is limited, you’ll have to trade off accuracy. It really depends on the specific use case and user requirements.

John: So, it’s all about balancing different factors based on the situation?

Interviewee: Exactly. You have to consider all the stakeholders and situations to make the right trade-offs. If the user has lots of hardware but demands accuracy, that’s one approach. If resources are limited, then you’ll have to optimize and perhaps sacrifice some accuracy.

John: Can you share an example of how you’ve done these trade-offs in your practical experience?

Interviewee: Sure, let’s take your current situation as an example. You don’t have enough memory to load the entire dataset at once. We came up with the loop-based solution to deal with it. But if your situation required higher accuracy and more resources, that would be a different challenge. Ultimately, it depends on the constraints you're working within.

John: Yes, that makes sense. Google, for example, has lots of resources, and it’s hard to compete with their scale.

Interviewee: Exactly, with all the servers and resources they have, it’s hard to match that. But that’s why we make trade-offs depending on what’s available.

John: Could you discuss potential future research directions for movie recommendation systems? What are the current challenges and possible solutions?

Interviewee: One area that hasn’t been fully explored is capturing user emotions in recommendations. For example, it’s hard to account for how a user is feeling when recommending a movie. If we could incorporate user emotions, it could greatly improve recommendations, but it’s difficult to capture this data, especially with GDPR and other regulations. However, finding a way to factor in user mood would be a big step forward.

John: That’s true, capturing emotions would be a challenge but could add a lot of value.

Interviewee: Exactly. It’s something worth exploring, but we need to account for privacy concerns. There are many other potential directions, but that’s one that stands out.

John: Yes, I agree. I think we’ve covered most of the questions. I’m very glad to have had you here tonight, and it’s getting quite late. Thank you for your time and patience.

Interviewee: It was my pleasure. I enjoyed discussing these topics, and I’m happy I could help, especially with your data-loading problem.

John: Yes, the loop idea is something I’ll definitely try out. It’s the first time I’ve heard of it, but I think it will work.

Interviewee: Yes, it should. Just make sure to free up the memory after each chunk to avoid RAM issues.

John: Yes, I’ll keep that in mind. Thanks again. Have a good night and a great weekend!

Interviewee: You too! Take care!

## 4 Interview Four

Interviewee: So if I take ancilla...

Me: Okay.

Interviewee: Can you hear me?

Me: Yeah, yeah.

Interviewee: Can you hear me?

Me: Yeah, loud and clear.

Interviewee: I was at the gym. Until we do the introduction, I think I’ll be back home. Give me a few minutes, and I'll call you from there.

Me: Okay.

Interviewee: In the meantime, you can tell me a bit about how you’ve been if you want.

Me: Yeah, sure. If possible, could you sit somewhere more comfortable?

Interviewee: I get it, I get it. Give me 5 minutes, I'll be home, and I'll call you from there.

Me: Okay, see you later.

Interviewee: I was just coming from the gym, so I thought maybe...

Me: Yeah, thank you, I really appreciate your time and patience.

Interviewee: No problem, it was nice to see your message about catching up regarding your project. You sent me a well-crafted list of questions, about 14 to 15 of them. I had the chance to prepare answers for 10, but I can also answer impromptu if needed.

Me: First of all, could you briefly introduce your current role and your experience with recommendation systems?

Interviewee: Sorry, could you repeat that?

Me: Could you briefly introduce your current role and your experience with recommendation systems?

Interviewee: So, is it my interview or your interview?

Me: It’s my interview, but I have to provide evidence of who I’m speaking with, right?

Interviewee: Okay, so this conversation is recorded for your project?

Me: Yes, it’s for my final project. We won’t publish your personal information; we just need to know your role and experience.

Interviewee: Okay, no problem. My name is [anonymized]. I’ve been in Ireland for the last 5 years. I did my master’s in data science...

Me: Just your current role and experience with recommendation systems is enough, thank you.

Interviewee: Oh, okay. I work in [anonymized], and I was working as a data analyst. For a brief time, I was involved in recommendation systems where customer service agents rated calls and prioritized customer complaints. The system used recommendation algorithms to help them manage their workload based on relevance and priority.

Me: Oh, nice, that sounds good. Next, I’ll ask a few questions, and I’ll start with the most important ones. For the rest, we can discuss them as we go along. Is that okay?

Interviewee: Sure, go ahead.

Me: First, what type of models do you typically use in recommendation system problems?

Interviewee: Could you hold on for one second?

Me: Sure, no worries.

Interviewee: Sorry, I was just looking for my phone, it was in the other room. Okay, I’m ready. What was your question again?

Me: What type of models do you typically use in recommendation system problems?

Interviewee: I used deep learning techniques like neural collaborative filtering to capture nonlinear user-item interactions. We also used collaborative filtering techniques combined with matrix factorization.

Me: Yes, I know that. It’s a kind of collaborative filtering, right?

Interviewee: Yes, collaborative filtering combined with deep learning, exactly.

Me: Got it, continue.

Interviewee: I also used matrix factorization techniques, deep learning models like neural collaborative filtering, and sometimes I combined these techniques to handle larger datasets. Other techniques included graph neural networks and adversarial neural networks.

Me: That sounds great. Could you discuss the structure of recommendation systems in the industry, along with the models you most frequently use and their performance?

Interviewee: Could you repeat the question?

Me: Sure, I’ll split it up. First, could you discuss the structure of the recommendation system in the industry, along with the models you most frequently use and their performance?

Interviewee: So basically, the models we use are based on a hierarchy. If we have a bigger dataset, we choose the model accordingly. In my company, we mostly use self-attention mechanisms to capture priority patterns, which are very efficient for this purpose.

Me: Yeah.

Interviewee: We also use deep collaborative filtering, as I mentioned before. If you’re going for hierarchy, the simpler ones are matrix factorization or basic collaborative filtering. These work well with smaller datasets. But when the datasets get more complex, we use self-attention mechanisms, graph neural networks (GNNs), or deep collaborative filtering. These are the main techniques used.

Me: What about the reasoning behind your choice of these models?

Interviewee: It depends on the complexity of the dataset. When data is simpler, we can use traditional models like matrix factorization, but when the data gets more complex, we need models like GNNs or deep collaborative filtering because they can capture intricate patterns in the data. So, the most optimal model depends on the specific problem and the structure of the data.

Me: So, it depends on the problem and the dataset?

Interviewee: Exactly.

Me: Have you ever tried combining these models or others to leverage their strengths for better performance?

Interviewee: Yes, combining models, or hybrid models, is quite common. Each model has its strengths, and combining them can enhance performance. One successful combination I’ve used is a graph neural network with a self-attention mechanism.

Me: Oh, so GAT, right? I’ve tried that too.

Interviewee: Yes, exactly. Self-attention helps prioritize, while GNNs capture long, complex user-item relationships. When we combined these, there was a significant improvement in recommendation accuracy. However, tuning and regularization are key when combining models to avoid overfitting.

Me: Yes, I use techniques like dropout and batch normalization to avoid overfitting. It’s worked well so far.

Interviewee: Yes, those methods help a lot.

Me: Could you share some examples of problems where you’ve used these models?

Interviewee: In my company, we provide customer service for utilities. We use self-attention mechanisms to prioritize calls based on the complaints recorded by the system. The model helps customer service agents prioritize which calls to handle first, based on the relevance and urgency of the complaint.

Me: That’s amazing. Thank you. So, what was the biggest challenge you faced while using deep learning to enhance recommendation system performance, and how did you overcome it?

Interviewee: The biggest challenge is often the complexity of the data. When the data is complex, it requires a lot of computational resources, and data preprocessing becomes more complicated. After preprocessing, you need to ensure that the models don’t overfit. So, the challenge is twofold: handling large datasets and making sure the models generalize well. It’s also computationally expensive and time-consuming.

Me: Have you ever used big data technologies to manage large datasets?

Interviewee: My company doesn’t use Hadoop or similar technologies. We rely on tools like Alteryx for cleaning and preprocessing data, which has made things easier than traditional methods.

Me: I see. Could you describe some architectural considerations or optimizations needed when integrating different models?

Interviewee: The key is balancing complexity with performance. When you integrate models, you need to ensure that the performance meets the requirements without making the system too complex. Regularization is essential to avoid overfitting. Techniques like data parallelism and model parallelism can help optimize the system and prevent slow data processing pipelines, especially with larger models.

Me: What are the limitations of current evaluation metrics for joint models, and how can they be improved? When you build a recommendation system, how do you evaluate its performance?

Interviewee: There are some limitations with current evaluation metrics. Most evaluation metrics for recommendation systems focus on accuracy, but they might not fully capture other important aspects like diversity or novelty in recommendations. It’s important to look at a combination of metrics to get a fuller picture of model performance. More research is needed to improve how we evaluate these systems.

Interviewee: So what are the limitations?

Me: For evaluating? Yeah, what kind of method do you use to evaluate your model's performance?

Interviewee: We use methods like recall, precision, F1 score...

Me: Oh, yeah, recall, precision, and confusion matrix, right?

Interviewee: Yes, those are useful, but they don’t capture the full performance. They typically focus on accuracy.

Me: Accuracy, yes.

Interviewee: But these metrics may overlook aspects like diversity or novelty.

Me: So it’s a classification problem, and you’re focused on accuracy?

Interviewee: Yes, but improvements could involve multi-objective evaluation metrics that balance accuracy with other factors. Additionally, user satisfaction or engagement scores could provide a more holistic view of model performance.

Me: So you mean using user-centric metrics?

Interviewee: Yes, exactly. User-centric metrics like satisfaction or engagement scores could give a better perspective on how the model is performing.

Me: That’s good to know, thanks.

Me: Have you ever encountered issues with insufficient computational power? What improvements have you made to reduce your models' dependency on hardware and enhance training speeds?

Interviewee: Yes, sometimes I feel like throwing my PC out the window because it doesn’t work as I want! But computational limitations are very common, especially with deep learning models. To address this, we can use quantization, which reduces the precision of model parameters, like from 32-bit down to 16-bit floating-point numbers. This reduces the computational load.

Me: Okay.

Interviewee: You can also reduce the number of features. Quantization is a very effective method for reducing computational load.

Me: I haven’t heard of that before. Could you explain it briefly?

Interviewee: Of course. When you have model parameters in 32-bit precision, you reduce them to 16-bit floating-point numbers to lower the computational demand.

Me: I understand, thanks. What else can be done?

Interviewee: Another technique is model pruning, which eliminates redundant parameters from the model.

Me: Pruning? I’ve heard that term used in decision trees.

Interviewee: Yes, it’s similar. Pruning reduces unnecessary parts of the model to improve efficiency.

Me: Interesting. Please go on.

Interviewee: Besides pruning and quantization, you can also use efficient architectures like MobileNet, which is designed for accuracy and efficiency.

Me: MobileNet? How does that help?

Interviewee: MobileNet is an architecture that helps reduce computational requirements. Once you import your data, MobileNet handles computations in a way that reduces the need for high computational power.

Me: Is MobileNet an online service or software?

Interviewee: You can find it on platforms like GitHub. It's an interesting tool, though I haven’t used it in a while. It’s a good option for certain datasets.

Me: That sounds very interesting. Thanks for sharing!

Interviewee: You can also use cloud computing resources.

Me: Yes, that could be a final solution, but for academic research like mine, using cloud computing is expensive. I checked the pricing, and it’s around $800 per month, which is costly for me.

Interviewee: It can definitely get expensive. Alternatively, you could use multiple GPUs, which can also be costly. However, if your institute provides access to certain services, that might help you manage the costs. Otherwise, techniques like pruning and quantization can help reduce computational needs.

Me: Thank you, that's helpful.

Me: Do you think using smaller datasets due to computational limitations for deep learning model training and hyperparameter tuning is a feasible approach in academic research?

Interviewee: It’s possible, but it comes with trade-offs. Training on smaller datasets might not generalize well to larger datasets, which can lead to incorrect conclusions and inaccurate model performance.

Me: So, the results might not be accurate?

Interviewee: Yes, your model's performance could suffer as a result.

Me: What strategies do you employ when faced with large datasets that exceed computational capabilities?

Interviewee: You need to acknowledge the limitations of your computational power. Where possible, you should validate your findings on larger datasets. You can use techniques like data augmentation and transfer learning to mitigate these issues.

Me: So, transfer learning allows you to use pre-trained models, right?

Interviewee: Exactly. Once you’ve trained a model, you can leverage it for new tasks. Augmentation can also help by introducing small variations to your dataset.

Me: Augmentation is like creating "fake" data, right? Adding noise or making small changes?

Interviewee: Yes, exactly. It can help improve model performance without the need for large datasets.

Me: In real-world production, what are your criteria for selecting a model for deployment? Could you provide the three most important factors you consider, ranked in order of priority?

Interviewee: The top priority is model accuracy. The model must meet accuracy requirements and improve user experience. Second is scalability and efficiency—considering computation time and resources. Lastly, maintainability. The model should be easy to maintain, update, and integrate with existing systems.

Me: So, accuracy is the most important?

Interviewee: Yes, it’s crucial that the model is accurate enough for the application. Then, scalability and efficiency come into play, followed by maintainability.

Me: Thank you for clarifying that.

Me: Considering the integration of deep learning methods in real recommendation systems, how do you assess the trade-off between system complexity and user experience?

Interviewee: Good question. It's very critical because sometimes you have to make a choice. A highly complex model might offer better accuracy, but it could require more computational resources and introduce latency. This could degrade the user experience.

Me: Yeah, the more complex it is, the more it costs and needs more training time.

Interviewee: Exactly. Personally, I would prioritize models that offer a balance between sufficient accuracy and acceptable computational efficiency and response time.

Me: So in your job experience, do companies have strict requirements, or do you have some flexibility to decide on the trade-off?

Interviewee: In my company, the goal is always to optimize the user experience. The person on the other side is the priority. Sometimes, we choose a slightly less complex model that can deliver fast and reliable recommendations to ensure user satisfaction.

Me: In your experience, between models like self-attention, multiple-head self-attention, graph neural networks, or adversarial neural networks, which one is the fastest?

Interviewee: Self-attention is generally faster because it prioritizes the relevant data more efficiently.

Me: Why do you think self-attention is faster from a technical point of view?

Interviewee: Self-attention works by giving priority to the most relevant parts of the input. For example, in our case, when we have different reasons for a power outage, self-attention can prioritize the most relevant cause and respond faster. Once it's prioritized, the response is smoother and faster.

Me: Have you tried multiple-head self-attention? Does it perform better than single-head self-attention in your experience?

Interviewee: Performance definitely varies. We use multiple heads for complex datasets, and we see performance improvements. But it depends on the dataset—more complex data benefits more from multiple heads.

Me: So in your opinion, multiple-head self-attention is generally better for more complex data?

Interviewee: Yes, but again, it depends on the dataset. For more complex data, multiple-head self-attention tends to perform better.

Me: I see. Thank you. So we already discussed model evaluation methods. One last question: Could you discuss potential future research directions for movie recommendation systems, the current challenges they face, and possible solutions for improvement?

Interviewee: One potential future direction could be incorporating more contextual data—like user mood or social context—to improve personalization. We can also make recommendations more interpretable to users, which builds trust. For example, ensuring that recommendations are understandable across different languages can increase user satisfaction.

Me: That makes sense.

Interviewee: Another challenge is fairness and bias reduction. We need to ensure that recommendations are diverse and fair. This involves mitigating biases in the system to provide diverse, fair, and personalized recommendations.

Me: Can you briefly explain bias reduction?

Interviewee: Bias reduction ensures the system doesn't disproportionately favor certain users or content based on irrelevant factors. It focuses on making recommendations more relevant to each individual, which helps build trust and improve the user experience.

Me: I see. That sounds very interesting.

Interviewee: Glad you think so!

Me: I think we've covered everything. It's been great talking to you about recommendation systems. Thank you very much!

Interviewee: Likewise! Good luck with your project.

Me: Thanks! Have a good day.