PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA

MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH

Yahia Fares University of Médéa



Faculty of Science
Department of Mathematics and Computer Science

Final Year Project Thesis

Specialization: Applied Artificial Intelligence

Dynamic Hybrid Model for Sequential and Explainable Recommendation Using GCN, Memory, and Attention

Prepared by:
TEBBAL Walid
BELMEBARKI Ayoub

Supervised by:
Ms. F. SADOUKI

Academic Year: 2024–2026

إهداء

أولاً وقبل كل شيء، نحمد المولى عز وجل على توفيقه لنا في هذا العمل، فلم نطلب العلم إلا لوجهه الكريم، وهو القائل: ﴿ عَلَّمَ الْإِنسَانَ مَا لَمْ يَعْلَمْ ﴾.

أهدي هذا العمل إلى والديّ العزيزين، أبي وأمي، اللذين كانا النور الذي أضاء طريقي، والسند الذي لم يملّ يومًا من دعمي، والقلبين اللذين حملا همّي قبل فرحي. إلى أبي، رمز الكفاح والعطاء، الذي علّمني أن النجاح لا يُمنح بل يُنتزع بالصبر والعمل. وإلى أمي، نبع الحنان والدعاء، التي كانت كلماتها قوةً في لحظات الضعف، وابتسامتها أملًا في أوقات التعب. أسأل الله أن يحفظهما، ويبارك في عمرهما، وأن يجعلني دائمًا مصدر فخرٍ لهما كما كانا أعظم نعمةٍ في حياتي.

وإلى شريكة العمر القادمة، أهديك هذا العمل المتواضع.

فأنتِ أجملُ ما تخبئه الأقدار، وأبهى فصول العمر التي لم تبدأ بعد، وأروعُ حكايةٍ تنتظر أن تكتبها الأيام.

كما أهدي هذا العمل إلى جميع أحبتي وأصدقائي، الذين كانوا دائمًا مصدر دعمٍ وتشجيعٍ لي خلال هذه المسيرة.

وأخيرًا، أهدي هذا العمل إلى نفسي، كتذكير بان المثابرة و الانضباط و العمل الجاد يقود دائما الى النجاح, مهما كانت صعوبة الطريق.

وليد طبال

إهداء

ما كان لهذا الإنجاز أن يكتمل لولا أشخاصٌ جعلوا الطريق أكثر إشراقًا، وكان حضورهم سندًا في لحظات التعب قبل الفرح.

إلى عائلتي الكريمة، التي منحتني من الدعم والمحبة ما جعلني أؤمن دائمًا بقدرتي على المضي قدمًا، فكان فضلها حاضرًا في كل خطوة من هذه الرحلة.

وإلى أصدقائي الأوفياء، الذين رافقوني خلال هذه السنوات، فخففوا مشقة الطريق بحسن صحبتهم وصدق مشاعرهم.

إليكم أهدي ثمرة هذا الجهد، عربون محبة وامتنان، وتقديرًا لكل أثر جميل تركتموه في حياتي، ولكل لحظة دعمٍ كانت سببًا في الوصول إلى هذا اليوم.

أيوب بلمباركي

Acknowledgments

First, we thank Allah Almighty for giving us the strength, patience, and success to complete this work.

We also extend our sincere thanks to our supervisor, **Mrs. Sadouki Fatiha**, for her invaluable guidance and unwavering support throughout this project. Her expertise and encouragement were instrumental in the successful completion of this thesis.

We also extend our thanks to the members of the judging panel for evaluating this work and to all our teachers for the knowledge and guidance they have provided throughout the years.

Finally, we extend our deepest gratitude to our families, loved ones, and friends for their unwavering encouragement and support. Their encouragement has been a source of motivation throughout our studies.

To everyone who contributed, directly or indirectly, to the realization of this dissertation, we offer our sincere thanks.

Walid and Ayoub

Abstract

Recommendation systems have become essential components of modern digital platforms, helping users discover relevant items within increasingly large collections of content. Despite significant progress in recommendation research, traditional approaches still face several challenges, including data sparsity, cold-start issues, limited modeling of long-term user preferences, and insufficient explainability of generated recommendations.

This dissertation proposes **DynamicHybridRec**, a dynamic hybrid recommendation architecture that combines graph-based collaborative learning, sequential behavior modeling, external memory mechanisms, and explainable artificial intelligence techniques within a unified framework. The proposed model integrates LightGCN to capture collaborative user-item relationships, a Transformer-based sequential encoder to model short-term behavioral dynamics, and an External Memory Module to preserve long-term preference patterns that extend beyond the fixed sequence window. These heterogeneous representations are fused through a Cross-Attention Fusion layer to produce a comprehensive user representation for recommendation generation.

To improve transparency and user trust, a hybrid explainability framework is incorporated by combining attention weight analysis, GradientSHAP attribution, and component contribution estimation.

We implemented and tested the proposed architecture on rank standard metrics (HR@10, NDCG@10, MRR@10) in the MovieLens 1M Dataset. Experiments show that the hybrid model obtains reliably good recommendation performance as well as interpretability while successfully discovering proximate user preferences and long term user preferences.

Keywords: Recommender Systems, Hybrid Recommendation, LightGCN, Transformer, External Memory, Cross-Attention Fusion, Explainable Artificial Intelligence, Sequential Recommendation, Context-Aware Recommendation, Deep Learning.

Résumé

Les systèmes de recommandation sont devenus des composants essentiels des plateformes numériques modernes, aidant les utilisateurs à découvrir des éléments pertinents au sein de collections de contenu toujours plus volumineuses. Malgré les progrès importants réalisés dans ce domaine, les approches traditionnelles continuent de présenter plusieurs limitations, notamment la rareté des données, le problème du démarrage à froid, la difficulté de modéliser les préférences à long terme des utilisateurs et le manque d'explicabilité des recommandations générées.

Ce mémoire propose **DynamicHybridRec**, une architecture hybride dynamique de recommandation qui combine l'apprentissage collaboratif basé sur les graphes, la modélisation séquentielle du comportement utilisateur, des mécanismes de mémoire externe et des techniques d'intelligence artificielle explicable au sein d'un cadre unifié. Le modèle proposé intègre LightGCN pour capturer les relations collaboratives utilisateur–élément, un encodeur séquentiel basé sur Transformer pour modéliser les dynamiques comportementales à court terme, ainsi qu'un module de mémoire externe destiné à conserver les préférences à long terme dépassant la fenêtre séquentielle fixe.

Afin d'améliorer la transparence et la confiance des utilisateurs, un cadre hybride d'explicabilité est intégré en combinant l'analyse des poids d'attention, l'attribution GradientSHAP et l'estimation de la contribution des différents composants du modèle.

Nous avons implémenté et testé l'architecture proposée sur des métriques standard de classement (HR@10, NDCG@10, MRR@10) dans l'ensemble de données MovieLens 1M. Les expériences montrent que le modèle hybride obtient des performances de recommandation fiables et une bonne interprétabilité, tout en découvrant avec succès les préférences immédiates et à long terme des utilisateurs.

Mots-clés : Systèmes de recommandation, Recommandation hybride, LightGCN, Transformer, Mémoire externe, Cross-Attention Fusion, Intelligence artificielle explicable, Recommandation séquentielle, Recommandation contextuelle, Apprentissage profond.

ملخص

أصبحت أنظمة التوصية مكوّناً أساسياً في المنصات الرقمية الحديثة، حيث تساعد المستخدمين على اكتشاف العناصر ذات الصلة ضمن مجموعات محتوى متزايدة الحجم. وعلى الرغم من التقدم الكبير في أبحاث أنظمة التوصية، لا تزال الأساليب التقليدية تواجه عدة تحديات، من بينها ندرة البيانات، ومشكلة المستخدمين أو العناصر الجدد ، وضعف نمذجة التفضيلات طويلة المدى للمستخدم، إضافة إلى محدودية قابلية تفسير التوصيات الناتجة.

تقترح هذه المذكرة نموذجًا هجينًا ديناميكيًا للتوصية يجمع بين التعلم التعاوني المعتمد على الرسوم البيانية، ونمذجة السلوك التسلسلي، وآليات الذاكرة الخارجية، وتقنيات الذكاء الاصطناعي القابل للتفسير ضمن إطار موحد. يدمج النموذج المقترح خوارزمية **LightGCN** لالتقاط العلاقات التعاونية بين المستخدمين والعناصر، ومشفّرًا تسلسليًا يعتمد على **Transformer** لنمذجة الديناميكيات السلوكية قصيرة المدى، بالإضافة إلى وحدة ذاكرة خارجية للحفاظ على أنماط التفضيلات طويلة المدى التي تتجاوز نافذة التسلسل المحددة. يتم دمج هذه التمثيلات غير المتجانسة عبر طبقة **Fusion Cross-Attention** لإنتاج تمثيل شامل للمستخدم من أجل توليد التوصيات.

ولتحسين الشفافية وتعزيز ثقة المستخدم، تم دمج إطار تفسير هجين يجمع بين تحليل أوزان الانتباه (Attention Weights)، وتقنية **GradientSHAP** للإسناد (Attribution)، وتقدير مساهمة كل مكوّن في القرار النهائي.

تم تنفيذ واختبار البنية المقترحة على مجموعة بيانات **1M MovieLens** باستخدام مقاييس التقييم القياسية في الترتيب مثل **HR@10** و **NDCG@10** و **MRR@10**. أظهرت النتائج التجريبية أن النموذج الهجين يحقق أداءً جيدًا وموثوقًا في التوصية، مع قدرة على توفير قابلية تفسير، إضافة إلى نجاحه في اكتشاف التفضيلات القريبة وطويلة المدى للمستخدمين.

الكلمات المفتاحية: أنظمة التوصية، التوصية الهجينة، LightGCN، Transformer، الذاكرة الخارجية، دمج الانتباه المتقاطع، الذكاء الاصطناعي القابل للتفسير، التوصية التسلسلية، التوصية المراعية للسياق، التعلم العميق.

Table of Contents

Abstract **4**

List of Figures **11**

List of Tables **12**

General Introduction **13**

1 Recommandation Systems **17**

1.1 Introduction 17

1.2 History 17

1.3 Definition of a recommendation system 18

1.4 Types of Recommender Systems 19

1.4.1 Collaborative filtering 19

1.4.2 Content-based recommendation 20

1.4.3 Hybrid filtering 21

1.5 Extending Capabilities of Recommender System 23

1.6 Real-World Applications of Recommendation Systems 24

1.6.1 Education and E-Learning 25

1.6.2 Medicine and Healthcare 25

1.6.3 E-commerce and Business 25

1.6.4 Media and Entertainment 25

1.6.5 Financial Services 25

1.6.6 Travel and Tourism 25

1.7 Challenges and Limitations of Recommender Systems 26

1.7.1 Data Sparsity 26

1.7.2 Scalability 26

1.7.3 Cold Start Problem 26

1.8 Evaluation techniques and metrics 26

1.8.1 Offline Evaluation 26

1.8.2 Online Evaluation 28

1.9 Conclusion 28

2 Theoretical Background on Modern Recommender Systems 31
2.1 Introduction 31
2.2 The Evolution of Recommender Systems: From Traditional Models to Deep Learning 31
2.2.1 Traditional Recommender Approaches 32
2.2.2 Classical Machine Learning Models in Recommenders 32
2.2.3 Emergence of Neural and Deep Learning Models 33
2.2.4 Classical vs. Deep Learning-Based Approaches 35
2.2.5 Common Deep Learning Architectures for Recommendation 36
2.3 Graph Convolutional Networks 37
2.3.1 Graph Representation 37
2.3.2 Core Concept of GCN 39
2.3.3 GCN Layer Propagation Rule 39
2.3.4 Architecture of GCN 40
2.3.5 LightGCN 40
2.3.6 Representative Graph-Based Recommendation Models 41
2.4 Sequential Modeling with Transformer 43
2.4.1 Limitations of RNN-Based Models 43
2.4.2 Transformer History 43
2.4.3 Transformers in Recommender Systems 44
2.4.4 Transformer Architecture 44
2.4.5 Representative Sequential Recommendation Models 45
2.4.6 Introduction to Attention 46
2.4.7 Role of Attention in Recommender Systems 47
2.4.8 Self-Attention Mechanism 47
2.4.9 Multi-Head Attention 48
2.4.10 Memory-Augmented Recommendation Models 48
2.4.11 External Memory Mechanisms 49
2.4.12 Explainable AI in Recommender Systems 50
2.4.13 Explainability Limitations of Existing Models 51
2.4.14 SHAP for Explainable Recommendation 52
2.4.15 Attention-Based Explanations in Recommender Systems 53
2.4.16 Other Common Explainability Methods 54
2.5 Conclusion 55
3 Development of the Proposed Hybrid Approach 57
3.1 Introduction 57
3.2 Dataset and Data Collection 58

3.3 Data Preprocessing . . . 58
3.3.1 User and Item Filtering . . . 58
3.3.2 Stratified User Sampling . . . 59
3.3.3 ID Remapping (user2idx / item2idx) . . . 59
3.3.4 Chronological Sequential Splitting (Train / Validation / Test) . . . 59
3.4 Model Architecture . . . 60
3.4.1 LightGCN Collaborative Filtering Backbone . . . 60
3.4.2 Transformer-Based Sequential Encoder . . . 61
3.4.3 External Memory Module . . . 62
3.4.4 Cross-Attention Fusion Layer . . . 64
3.5 Design Choices . . . 65
3.5.1 Why Four Components? . . . 65
3.5.2 Hyperparameter Selection Strategy . . . 65
3.6 Training Strategy . . . 66
3.6.1 Dataset Construction with Hard Negative Mining . . . 66
3.6.2 Composite Loss Function . . . 67
3.6.3 Optimiser and Learning-Rate Schedule . . . 68
3.6.4 Training Configuration Summary . . . 69
3.7 Evaluation Methodology . . . 69
3.8 Explainability Framework . . . 70
3.8.1 Explainability Methods . . . 71
3.8.2 Hybrid Cross-Validation of Attribution Signals . . . 74
3.8.3 Visualization . . . 74
3.8.4 Backend Serving with FastAPI . . . 76
3.9 Conclusion . . . 77
4 Expermentation and Evaluation . . . 79
4.1 Introduction . . . 79
4.2 Development Environment and Tools . . . 79
4.2.1 Programming Language . . . 79
4.2.2 Libraries and Frameworks . . . 80
4.2.3 Development Platform . . . 80
4.2.4 Hardware . . . 80
4.3 Training Dynamics . . . 81
4.3.1 Loss Convergence . . . 81
4.3.2 Validation Metric Progression . . . 81
4.4 Main Results . . . 82

4.4.1 Test Set Performance . . . 83
4.4.2 Comparison with Baselines . . . 83
4.4.3 Ablation Study . . . 85
4.4.4 Generalization to Sparse Datasets . . . 87
4.5 Explainability Evaluation . . . 89
4.5.1 Faithfulness . . . 89
4.5.2 Explanation Robustness . . . 90
4.5.3 Population-level Coverage . . . 91
4.6 Strengths and Limitations . . . 92
4.6.1 Strengths . . . 92
4.6.2 Limitations . . . 92
4.7 CinéRec Interface Implementation . . . 93
4.7.1 Custom History mode . . . 93
4.7.2 User Explorer mode . . . 94
4.8 Conclusion . . . 98
General Conclusion . . . **99**
Bibliography . . . **101**

List of Figures

1.1 Types of Recommender Systems 19
1.2 Recommendation based on collaborative filtering 20
1.3 Recommendation based on content-based 21
1.4 Recommendation based on hybrid filtering 21
1.5 Hybridization methods [1]. 22
1.6 Classification of recommender systems research [2]. 24

2.1 The difference between Euclidean space and non-Euclidean space 37
2.2 Message Passing in Graph Convolutional Networks 39
2.3 Transformer architecture [3] 45
2.4 Classic encoder-decoder network with attention mechanism in Seq2Seq structure [4] 47

3.1 DynamicHybridRec architecture parallel components and fusion 60
3.2 Hybrid explainability pipeline attention, SHAP, and gate contributions . . 71
3.3 Attention and SHAP bar chart 75
3.4 Attention–SHAP scatter plot 76
3.5 Component contribution pie chart 76

4.1 Train vs Val Loss Convergence 81
4.2 Train vs Val For HR@10 & NDCG@10 82
4.3 Comparison with Baseline Methods on MovieLens 1M 84
4.4 Component Ablation Study 86
4.5 User interaction distribution in the Yelp Reviews dataset. 87
4.6 Comparison of Dataset Sparsity: Yelp (Preprocessed) vs. MovieLens 1M. . 88
4.7 Component-wise contribution analysis. 89
4.8 AOPC Faithfulness Curves 90
4.9 Comparison with Baseline Methods on MovieLens 1M 93
4.10 Profile Card 94
4.11 History Timeline and Recommendations Grid 95
4.12 AI Explanation Card 95
4.13 Model Analytics Charts 97

List of Tables

2.1 Classical vs. Deep Learning-based Recommender Systems 35

3.1 Dataset Statistics 58
3.2 Main Hyperparameter Configuration 66
3.3 Model Hyperparameters 66
3.4 Hybrid Loss Components and Their Roles 68
3.5 Training Strategy Parameters 69
3.6 Interpretation Categories Based on Attention and SHAP Scores 74

4.1 Training and Validation Performance Across Epochs 82
4.2 Final Test Set Performance at Different Cutoff Values 83
4.3 Ablation Study of the Proposed Recommendation Model 85
4.4 Performance comparison across datasets 88

General Introduction

Context and Motivation

In recent years, recommendation systems have become one of the most influential technologies in modern digital platforms. Their rapid proliferation across e-commerce, entertainment, education, healthcare, and social media has transformed the way users interact with online services. Instead of manually searching through vast amounts of information, users now rely on intelligent recommendation mechanisms capable of filtering relevant content based on their interests and behavioral patterns. Major companies such as Amazon, Netflix, YouTube, and Spotify are increasingly relying on recommendation algorithms to improve user engagement, personalize services, and increase customer retention.

Traditional recommendation approaches, particularly collaborative filtering and content-based methods, achieved great success during the early stages of recommendation system development. However, as user interactions have become more dynamic and the size and complexity of datasets have increased, these classical approaches have begun to reveal several significant limitations. Scattered interaction matrices, cold-start scenarios, the inability to model long-term dependencies, and poor context understanding lead to a significant decline in the quality of recommendations in real-world environments. Furthermore, modern users generate sequential behavioral data in which preferences continuously evolve over time, rendering static recommendation strategies inadequate for capturing complex interaction dynamics.

To address these challenges, recent research has increasingly focused on deep learning architectures capable of learning richer representations from sequential and graph-structured data. Transformer-based sequential models demonstrated strong ability in modeling temporal dependencies through self-attention mechanisms, while Graph Convolutional Networks (GCNs) became effective for extracting collaborative relationships from user-item interaction graphs. Nevertheless, relying on a single modeling paradigm often remains insufficient to fully capture the diversity and complexity of user behavior. In addition, many highly accurate recommendation models suffer from limited interpretability, making it difficult to explain why a particular recommendation was generated.

Research Gap

Although several recent recommendation architectures combine sequential modeling and graph learning, many existing systems still suffer from important limitations. Most sequential recommendation models focus mainly on recent interactions while ignoring older but still relevant long-term preferences. On the other hand, graph-based approaches generally overlook the temporal evolution of user behavior. Moreover, many hybrid architectures prioritize predictive accuracy without providing interpretable explanations capable of improving transparency and user trust. Existing explainability techniques are also frequently limited to a single attribution mechanism, which may provide incomplete or misleading interpretations of recommendation decisions.

Research Problem

How can a hybrid recommendation architecture effectively combine collaborative graph learning, sequential modeling, long-term memory representation, and explainability mechanisms in order to improve recommendation accuracy while maintaining interpretability and generalization capability?

Contributions

The main contributions of this work can be summarized as follows:

1. We propose a hybrid recommendation architecture named DynamicHybridRec that combines LightGCN, Transformer-based sequential encoding, external memory representation, and cross-attention fusion within a unified recommendation framework.

2. We design an external memory module capable of preserving compressed long-term user preference patterns beyond the fixed sequence length limitation commonly found in sequential recommendation models.

3. We develop a multi-level explainability framework combining attention analysis, GradientSHAP attribution, and component contribution estimation in order to improve recommendation transparency and interpretability.

4. We evaluate the proposed architecture on the MovieLens 1M dataset through extensive experiments including training analysis, baseline comparisons, and ablation studies to assess the effectiveness of each architectural component.

Document Organization

This dissertation is organized into four main chapters.

Chapter 1 presents the theoretical foundations of recommender systems, their categories, applications, challenges, and evaluation methodologies.

Chapter 2 introduces the machine learning and deep learning concepts related to recommendation systems, including graph neural networks and Transformer-based sequential recommendation models.

Chapter 3 describes the proposed DynamicHybridRec architecture, including the LightGCN module, Transformer encoder, external memory mechanism, fusion strategy, and explainability framework.

Finally, Chapter 4 presents the experimental setup, training process, evaluation metrics, empirical results, ablation studies, and discussion of the strengths and limitations of the proposed system.

Chapter 1

Recommandation Systems

Chapter 1

Recommandation Systems

1.1 Introduction

The ever-increasing variety of products and services available online makes it difficult for users to find relevant information. This often leads to confusion when selecting items that align with their interests. To address this problem, recommendation systems have been developed to provide personalized suggestions and help users discover content they might not have otherwise considered. Although significant progress has been made in this field, several challenges remain, including data scarcity, scalability, time efficiency, and prediction accuracy. In this chapter, we provide an overview of recommendation systems, covering classical methods such as content-based filtering, collaborative filtering, knowledge-based approaches, and hybrid models. In addition, we analyze evaluation metrics that consider not only prediction accuracy but also diversity and interpretability, laying the groundwork for understanding the challenges addressed in this study [5].

1.2 History

Initial developments in the 1920s and 1960s introduced methods for indexing and retrieving documents using mathematical representations such as vector space models.In the 1970s,Systems like Usenet have allowed users to share and access content.Although they did not provide personal recommendations.The first true personalized recommender system appeared in 1979 with the Grundy system,Which collected user preferences and categorized them into patterned groups to suggest items.However,this approach had a problem because users often fail to describe their preferences well.As a result, researchers have shifted towards analyzing user behavior rather than relying on explicit input.Over time, two main approaches to recommendation emerged. Collaborative filtering focuses on identifying similarities between users and recommending items liked by similar individual.and content-based filtering relies on analyzing the attributes of items and building

user profiles based on previously liked content.In the 2000s,recommendation systems became a key element in large-scale online platforms.Systems like Netflix's CineMatch and Amazon's recommendation engine demonstrated the commercial value of personalization.Today, recommendation systems are widely used in e-commerce, streaming services, and social media, providing personalized experiences that enhance user engagement and drive business success[6].

1.3 Definition of a recommendation system

A recommendation system is an intelligent framework designed to generate effective recommendations based on a user's past behavior, suggesting relevant items. The design of recommendation engines depends on the domain and specific characteristics of the available data. In addition, the system may have access to user profile attributes and product attributes, such as demographics and product descriptions, respectively. Recommendation systems differ in how they analyze these data sources to develop concepts of compatibility between users and products, which can be used to identify matching pairs.

Collaborative filtering systems analyze only past interactions, while content-based filtering systems rely on profile attributes; hybrid technologies attempt to combine both approaches. The design and evaluation of recommendation systems to solve real-world problems is an active research area [7, 5].

1.4 Types of Recommender Systems

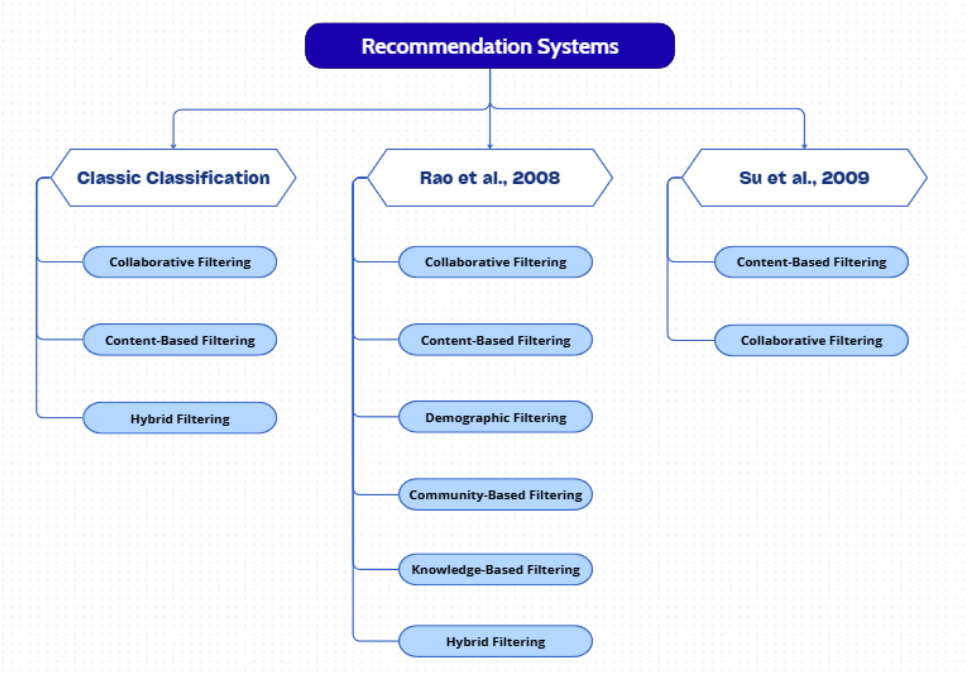


Figure 1.1: Types of Recommender Systems

1.4.1 Collaborative filtering

Collaborative filtering (CF) [3] is a recommendation technique based on the idea that users with similar preferences and behaviors tend to prefer similar items [8, 9]. Although the term ”collaborative filtering” (CF) has only been around for a little over a decade, its essence is rooted in an age-old human practice, the exchange of experiences and perspectives. In the past, people discussed the books they had read, the restaurants they had tried, and the movies they had watched [9] . In our modern era, with the advent of the internet, this is no longer limited to verbal communication. Rather than being limited to small social circles, the internet allows us to access and process the opinions of thousands of people,analyze the preferences of a much larger population about a particular product, and even form a truly personal opinion of that product using the opinions most relevant to a specific user or group of users.

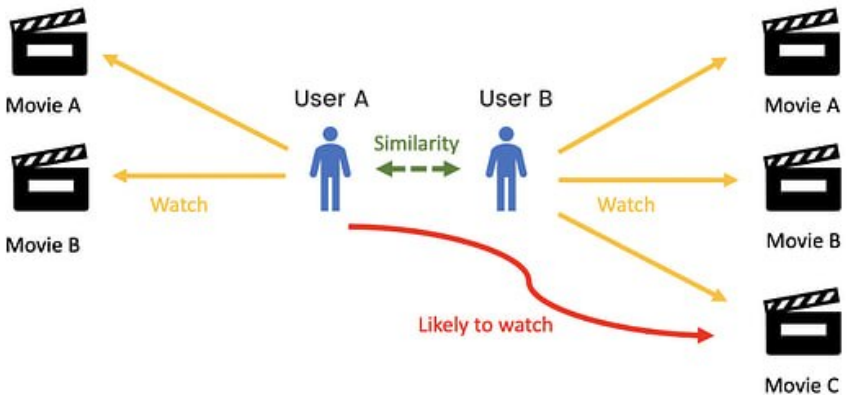


Figure 1.2: Recommendation based on collaborative filtering

1.4.2 Content-based recommendation

Content-based recommendation systems [10, 11] suggest items to users based on the similarity between item representations and a user profile derived from historical interactions. The user profile can be explicitly defined or implicitly learned from feedback signals such as ratings, clicks, or browsing history.

Various machine learning techniques have been proposed to model user preferences, and the choice of algorithm largely depends on the representation of item content and user profiles [11]. A common approach is to represent both users and items in a shared feature space, where similarity measures (such as cosine similarity) are used to identify relevant recommendations.

Content-based methods rely heavily on feature extraction techniques to transform raw item data into structured representations that enable effective matching between users and items. This makes them particularly suitable for scenarios with sparse interaction data, such as cold-start users or newly introduced items. However, these methods may suffer from limited diversity and over-specialization, as they tend to recommend items similar to those already consumed by the user.

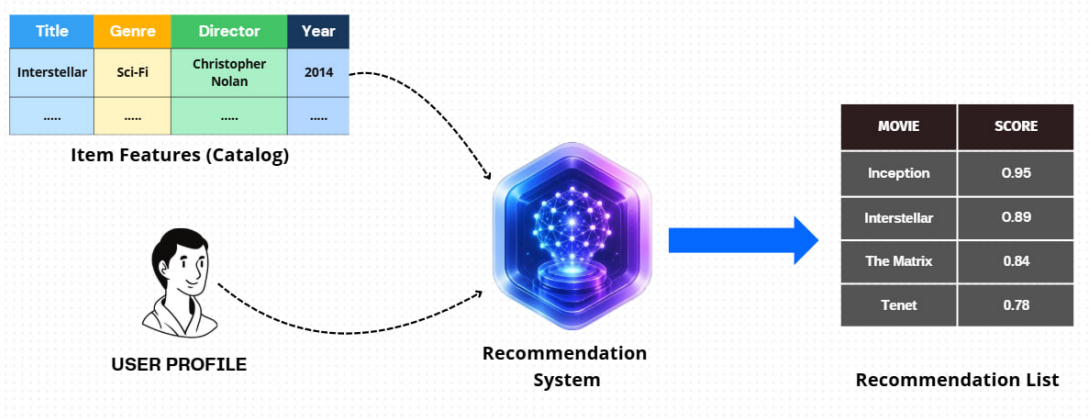


Figure 1.3: Recommendation based on content-based

1.4.3 Hybrid filtering

Hybrid recommendation systems combine two or more recommendation technologies to achieve better performance while minimizing the drawbacks of any single technology.

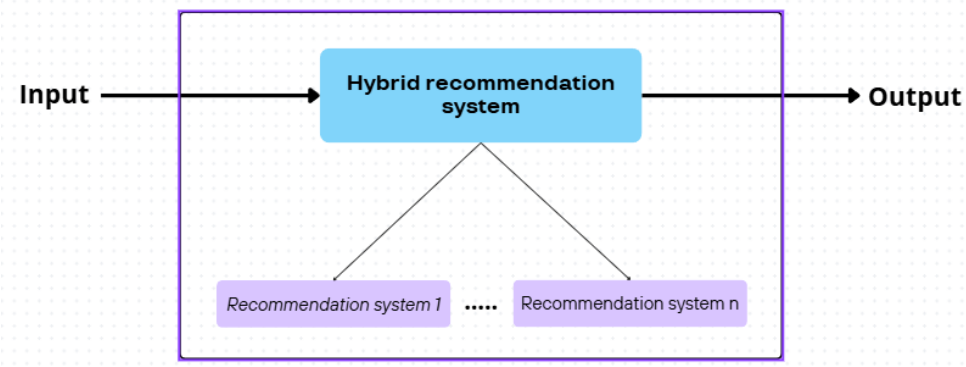


Figure 1.4: Recommendation based on hybrid filtering

Several hybridization strategies have been proposed in the literature to combine recommendation techniques in different ways depending on the target application. Figure 1.5 shows some of the most common hybridization methods.

Hybridization method	Description
Weighted	The scores (or votes) of several recommendation techniques are combined together to produce a single recommendation.
Switching	The system switches between recommendation techniques depending on the current situation.
Mixed	Recommendations from several different recommenders are presented at the same time
Feature combination	Features from different recommendation data sources are thrown together into a single recommendation algorithm.
Cascade	One recommender refines the recommendations given by another.
Feature augmentation	Output from one technique is used as an input feature to another.
Meta-level	The model learned by one recommender is used as input to another.

Figure 1.5: Hybridization methods [1].

Weighted : A weighted hybrid recommendation system combines multiple techniques, like collaborative filtering and content-based systems, by assigning different weights to each recommendation technique. For example, the P-Tango system starts with equal weights and adjusts them as user rating predictions are confirmed or refuted [1].

Switching : A switching hybrid recommendation system dynamically selects the most suitable technique based on past performance, adapting to each recommender's strengths and weaknesses for more accurate recommendations. Although effective, this strategy is generally more complex, requiring proper criteria and exchange factors [9, 1].

Mixed : A hybrid recommendation system combines multiple techniques, like content-based and collaborative filtering, to generate more diverse recommendation results [9, 11, 1]. For example, a PTV system integrates program descriptions and user preferences into a final recommendation list [1]. It can recommend new items without prior ratings but still faces challenges for new users when technologies conflict [1].

Feature Combination : Content-based and collaborative filtering can be integrated by treating collaborative data as additional item features. For example, Basu, Hirsch, and Cohen (1998) used the Ripper algorithm with both user ratings and content features to improve recommendation accuracy. This approach reduces dependence on the number of user ratings while leveraging item similarity information.

Cascade : A sequential hybrid recommendation system applies techniques in phases: one technique generates an initial ranking, and a second refines it [1]. For example, the EntreeC system combines specialized knowledge with collaborative filtering to rank restaurants efficiently. This approach avoids processing already well- or poorly-rated items and is more tolerant of noise, focusing refinement only where needed [9, 1].

Feature Augmentation : In feature augmentation hybrids, one technique generates ratings or assessments that are used as additional features for a second technique [1]. For example, the Libra system uses a Naive Bayes classifier with collaborative data to improve book recommendations, and GroupLens used "filterbots" to enhance Usenet news filtering [9, 1]. This approach boosts performance without altering the core system and differs from

cascade hybrids in that the second recommender depends on the first's outputs [1].

Meta-Level : In cascade hybrid systems, the model from the first recommendation technique serves directly as input for the second, unlike feature-augmentation hybrids where outputs are only used as additional features. Examples include the Fab system using the Rocchio method [12], example-based user profiles [13], and instance-based or meta-level learning [14]. This approach provides a compact representation of user interests, improving accuracy and efficiency [1].

1.5 Extending Capabilities of Recommender System

As recommendation systems evolved, researchers began introducing additional mechanisms and extensions intended to improve personalization quality, flexibility, and contextual understanding. including improving user and product understanding, integrating contextual information into the recommendation process, supporting multi-criteria assessments, and providing more flexible and less intrusive recommendation types. These more comprehensive models of recommendation systems can deliver enhanced recommendation capabilities.

Recommendation Approach	Recommendation Technique	
	Heuristic-based	Model-based
Content-based	Commonly used techniques: • TF-IDF (information retrieval) • Clustering Representative research examples: • Lang 1995 • Balabanovic & Shoham 1997 • Pazzani & Billsus 1997	Commonly used techniques: • Bayesian classifiers • Clustering • Decision trees • Artificial neural networks Representative research examples: • Pazzani & Billsus 1997 • Mooney et al. 1998 • Mooney & Roy 1999 • Billsus & Pazzani 1999, 2000 • Zhang et al. 2002
Collaborative	Commonly used techniques: • Nearest neighbor (cosine, correlation) • Clustering • Graph theory Representative research examples: • Resnick et al. 1994 • Hill et al. 1995 • Shardanand & Maes 1995 • Breese et al. 1998 • Nakamura & Abe 1998 • Aggarwal et al. 1999 • Delgado & Ishii 1999 • Pennock & Horwitz 1999 • Sarwar et al. 2001	Commonly used techniques: • Bayesian networks • Clustering • Artificial neural networks • Linear regression • Probablistic models Representative research examples: • Billsus & Pazzani 1998 • Breese et al. 1998 • Ungar & Foster 1998 • Chien & George 1999 • Getoor & Sahami 1999 • Pennock & Horwitz 1999 • Goldberg et al. 2001 • Kumar et al. 2001 • Pavlov & Pennock 2002 • Shani et al. 2002 • Yu et al. 2002, 2004 • Hofmann 2003, 2004 • Marlin 2003 • Si & Jin 2003
Hybrid	Combining content-based and collaborative components using: • Linear combination of predicted ratings • Various voting schemes • Incorporating one component as a part of the heuristic for the other Representative research examples: • Balabanovic & Shoham 1997 • Claypool et al. 1999 • Good et al. 1999 • Pazzani 1999 • Billsus & Pazzani 2000 • Tran & Cohen 2000 • Melville et al. 2002	Combining content-based and collaborative components by: • Incorporating one component as a part of the model for the other • Building one unifying model Representative research examples: • Basu et al. 1998 • Condliff et al. 1999 • Soboroff & Nicholas 1999 • Ansari et al. 2000 • Popescul et al. 2001 • Schein et al. 2002

Figure 1.6: Classification of recommender systems research [2].

1.6 Real-World Applications of Recommendation Systems

Beyond their theoretical foundations, recommendation systems have become widely integrated into many real-world domains and industrial applications.Recommendation systems are used today in many aspects of life and industry to improve user experience, increase efficiency, and facilitate decision-making. These systems are not limited to e-commerce; they extend to multiple fields serving both individuals and organizations.

1.6.1 Education and E-Learning

In education, recommendation systems suggest courses, learning resources, and personalized learning activities based on students' prior knowledge. This adaptive learning approach helps improve outcomes and engagement, especially in distance learning environments. [15].

1.6.2 Medicine and Healthcare

In the medical sector, recommendation systems assist physicians in suggesting personalized treatments based on patient data and medical history. They can guide treatment, rehabilitation, and medication decisions, as well as recommend preventive programs, improving outcomes and reducing errors [16, 17].

1.6.3 E-commerce and Business

The e-commerce sector is one of the most prominent areas for the application of recommendation systems, with major companies like Amazon and Netflix using sophisticated recommendation systems to analyze user preferences and past behaviors in order to suggest products, films, or services that suit their interests. This type of recommendation not only increases user satisfaction but also boosts purchase and customer retention rates, as it makes the shopping experience more personalized and relevant [18].

1.6.4 Media and Entertainment

Well-known examples of recommendation systems in entertainment include Netflix, which suggests movies or shows based on viewing history, and music platforms like Spotify and Last.fm, which provide personalized playlists based on listening patterns. These systems enhance user engagement and reduce churn [19, 20] .

1.6.5 Financial Services

In banking, insurance, and financial services, recommendation systems help customers select suitable products, such as savings plans, insurance, or loans. They combine customer preferences with market data to provide personalized suggestions, improving service quality and reducing decision-making errors [21].

1.6.6 Travel and Tourism

In the tourism industry, recommendation systems assist users in selecting destinations, hotels, tours, and entertainment based on past preferences and budget. This personalization enhances the travel experience and saves time compared to manual searches [22].

1.7 Challenges and Limitations of Recommender Systems

Recommender systems face challenges and limitations that impact their accuracy, scalability, and user experience.

1.7.1 Data Sparsity

the user–item interaction matrix is often extremely sparse because most users interact with only a small subset of available items, which brings about the challenges in the performances of the recommendation. One of the problems caused by the data sparsity is the cold start problem. As collaborative filtering methods recommend items based on users past preferences, new users must interact with a sufficient number of items to enable the system to capture their preferences accurately and thus provides reliable recommendations [23].

1.7.2 Scalability

As the number of users and items increases, the computational cost of recommendation algorithms grows significantly,These systems require rapid response, but the massive computing demands lead to time-related challenges [23].

1.7.3 Cold Start Problem

New users (with no interaction history) or new items (with few or no ratings) represent a major challenge,especially for collaborative filtering, which relies on past preferences [23].

1.8 Evaluation techniques and metrics

Recommender systems are evaluated using two techniques:

1.8.1 Offline Evaluation

The main purpose of offline testing is to estimate the model's performance using historical data and compare it against alternative recommendation models.The metrics used in offline evaluation include the following:

Mean Absolute Error (MAE):

measures the average absolute difference between predicted and actual ratings [7].

$$MAE = \frac{1}{N} \sum_{i=1}^{N} |r_i - \hat{r}_i| \tag{1.1}$$

Root Mean Square Error (RMSE):

RMSE penalizes larger prediction errors more heavily than MAE because the errors are squared before averaging [7].

$$RMSE = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (r_i - \hat{r}_i)^2} \tag{1.2}$$

- Precision@K:

Precision@K is the proportion of relevant items among the top-K recommended items [24].

$$Precision@K = \frac{\text{Number of relevant items in top-}K}{K} \tag{1.3}$$

- Recall@K:

Recall@K measures the proportion of relevant items that are successfully retrieved within the top-K recommendations [24].

$$Recall@K = \frac{\text{Number of relevant items in top-}K}{\text{Total number of relevant items}} \tag{1.4}$$

- F1-Score:

F1-score combines Precision and Recall using their harmonic mean [24].

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{1.5}$$

- Hit Rate@k:

Proportion of users for whom at least one relevant item appears in the top-K list (

$$Hit@K = \frac{\text{Number of users with at least one hit in top-}K}{\text{Total number of users}} \tag{1.6}$$

Hit Rate@K measures the percentage of users for whom at least one relevant item appears in the top-K recommendation list [24].

- Normalized Discounted Cumulative Gain (NDCG@K)

Normalized Discounted Cumulative Gain (NDCG) evaluates the quality of a ranked recommendation list by comparing it with the ideal ranking, where all relevant items appear at the top positions. Unlike metrics such as Hit Rate, NDCG considers both the relevance of recommended items and their positions in the ranking. It can be applied to both binary and graded relevance scores [25].

$$NDCG@K = \frac{DCG@K}{IDCG@K} \tag{1.7}$$

$$DCG@K = \sum_{i=1}^{K} \frac{2^{rel_i} - 1}{\log_2(i+1)} \tag{1.8}$$

where:

- rel_i is the relevance score of the item at position i.
- $IDCG@K$ is the Ideal Discounted Cumulative Gain, corresponding to the best possible ranking.

The NDCG metric produces values between 0 and 1, where higher values indicate better ranking quality and a recommendation list that is closer to the ideal ordering [24].

1.8.2 Online Evaluation

Online testing for recommender systems is the process of evaluating the system and it's performance in a live environment.The goal of online testing is to ensure that the system works as expected in a real-world environment with respect to the objective function and metrics.

Deploying recommendation models to subsets of users through A/B testing allows researchers and practitioners to measure real-world metrics such as click-through rate (CTR), conversion rate, and revenue per user. Although online evaluation provides a more realistic assessment of business impact, it is generally more expensive and time-consuming than offline evaluation [24].

1.9 Conclusion

In this chapter, we presented the fundamentals of recommendation systems and explored their importance in modern applications.different recommendation paradigms were also discussed, including collaborative, content-based, and hybrid recommendation approaches.along with examples of their practical applications across multiple domains in

real applications.also,we examined the key challenges faced by recommender systems, such as data sparsity, cold-start problems and scalability issues which motivate the need for more adaptive solutions.The next chapter will focus on machine learning,Introducing its core concepts and technologies that form the basis for building intelligent recommendation systems.

Chapter 2

Theoretical Background on Modern Recommendation Systems

Chapter 2

Theoretical Background on Modern Recommender Systems

2.1 Introduction

Although traditional methods have been and remain a cornerstone of recommender system development, they have faced a number of challenges and difficulties. While effective in some cases, they often suffer from limitations related to data scarcity and scalability. These limitations hinder their ability to provide accurate recommendations in diverse environments. To overcome these challenges, new techniques based on deep learning have been introduced. However, deep learning-based recommender systems initially received little attention. With the rapid advancement of deep learning technologies in areas such as computer vision and natural language processing, their adoption in recommender systems has increased, offering enhanced capabilities for modeling complex user–item interactions [26].

These developments have led to the emergence of more advanced recommendation models that surpass traditional techniques, including graph-based methods, sequential modeling, and attentional mechanisms. These models enable a deeper understanding of user behavior and item relationships, paving the way for more accurate, dynamic, and interpretable recommender systems.

2.2 The Evolution of Recommender Systems: From Traditional Models to Deep Learning

Recommender systems often face limitations due to model ineffectiveness with new users or items and struggle to adapt to dynamic shifts in user preferences. With the advent of deep learning, particularly through graph neural networks (GNNs) and knowledge graphs (KGs), new avenues for optimizing recommender systems have unfolded. These

cutting-edge techniques harness complex data relationships and rich semantic information, addressing challenges such as data sparsity and the novelty of users while enhancing personalization and accuracy.

2.2.1 Traditional Recommender Approaches

Traditional recommender approaches are the classical, non-deep-learning methods that form the foundation of modern recommender systems. They are usually classified into three main families: collaborative filtering, content-based filtering, and knowledge-based or hybrid systems, as discussed in Chapter 1. These classical techniques rely on explicit user–item interactions, item features, or domain knowledge to generate recommendations, often using surface models and hand-crafted features rather than deep neural architectures. While they remain relatively explainable and lightweight, they can struggle with data scarcity, cold-start scenarios, and capturing complex, nonlinear user–item relationships, thus driving the shift towards deep learning approaches in modern systems.

2.2.2 Classical Machine Learning Models in Recommenders

Linear Models and Logistic Regression

Linear models [27] and logistic regression are among the simplest and most widely used classical machine learning techniques in recommender systems. These models estimate recommendation scores as a weighted linear combination of input features, such as user and item attributes, offering efficiency and interpretability.

In particular, SLIM (Sparse Linear Methods for Top-N Recommendation) [28] formulates recommendation as a sparse linear combination of item-item similarities learned directly from user interaction data, making it highly effective for top-N recommendation tasks.

Logistic regression is usually used in binary or click-through rate (CTR) prediction: it estimates the probability of a user liking or clicking on an item based on an attribute vector (user attributes, item attributes, and context). It is widely used in large-scale recommender systems because it is easy to train, performs well online, and can be combined with attribute engineering techniques (such as inclusions, polynomial attributes, or FM-like interactions) [29].

Decision Trees and Gradient Boosting

Decision trees [30] and gradient boosting [31] are powerful classical machine learning models widely used in recommender systems. Decision trees partition the feature space using simple decision rules, making them interpretable and easy to deploy.

Gradient boosting learns to fix the mistakes of earlier decision trees by treating them as weak learners and gradually combining them into a strong ensemble. This method is frequently used in recommender pipelines for ranking and CTR prediction, where candidate items are re-ranked to increase relevance and personalization using GBDT rankers (such as XGBoost, LightGBM, and CatBoost with rank-oriented losses).

Although these tree-based approaches perform better than simpler linear models, scale to large datasets, and handle mixed numerical and categorical features well, they still need careful feature engineering and may not be as expressive as deep neural architectures when modeling extremely complex interaction patterns [32].

Factorization Machines and Similar Extensions

In order to capture pairwise feature interactions in high-dimensional, sparse data like user–item interaction matrices mixed with side features (user demographics, item attributes, context), Factorization Machines (FM) [29] are traditional machine learning methods. FMs are frequently used in recommenders for ranking, rating prediction, and CTR prediction, where they combine collaborative signals with context and content into a single, comprehensible model. FMs frequently act as a bridge between classical models and deep learning, avoiding extensive feature engineering while remaining relatively lightweight because raw user–item IDs and category features may be encoded as fields [32].

2.2.3 Emergence of Neural and Deep Learning Models

In recent years, deep learning has received considerable attention in many research fields and has proven its effectiveness when applied to information retrieval and recommender systems research. Cheng et al. created an app recommendation system for the Google Play Store utilizing a broad and deep model, whereas Covington et al. presented a deep neural network-based recommendation method for YouTube videos. For Yahoo News, Okura et al. introduced a recurrent neural network-based news recommendation system. All these models have proven their worth and shown significant improvement compared to traditional models [33].

Recent research conducted on major e-commerce platforms indicates that deep neural network architectures are capable of automatically extracting and learning hierarchical features from user interaction data with items, reducing manual feature engineering efforts by approximately 71.3%, and have shown remarkable capability in handling sparse datasets, maintaining recommendation accuracy above 82.6% even when dealing with datasets containing up to 65% missing values. According to implementation studies, deep neural collaborative filtering techniques outperform conventional collaborative filtering techniques by 42.8% in recommendation precision and 37.9% in recall rates.

Integration of Natural Language Processing (NLP) has become an essential part of

contemporary recommender systems, allowing for improved semantic comprehension of user preferences and item attributes. Transformer-based models have shown a 58.7% improvement in comprehending and processing user evaluations and item descriptions, according to recent state-of-the-art assessments. The relevancy of recommendations has increased by 45.3% as a result of the incorporation of sophisticated NLP approaches, which improve comprehension of contextual subtleties and user sentiment [34].

2.2.4 Classical vs. Deep Learning-Based Approaches

Aspect	Classical	Deep Learning-based
General approach	such as • Collaborative filtering (CF) [9] • Matrix Factorization (MF) [35] • Ensemble Methods (EM)	such as • NCF [36] • BERT4Rec [37] • AutoRec
Model type [38]	These methods are primarily linear approaches, though some variants such as kernel-based methods introduce limited non-linearity	Neural networks are inherently non-linear models, capable of learning complex representations through stacked layers and activation functions
Capability on patterns [38]	Linear models cannot properly map out the complex pattern in user–item interaction	Ability to model non-linearity in data using the nonlinear activation functions (ReLU, tanh, and sigmoid)
Feature handling [38]	Feature vectors are used by matrix factorization for relational data modelling	Deep neural networks perform as automatic feature extractors from varieties of input data (image, video, text, etc.)
Effectiveness with big data	Performance degrades with very large and sparse datasets, scalability is limited by matrix operations and memory constraints	DL techniques scale more effectively with large volumes of data, as model capacity can be increased to capture richer patterns in big data scenarios
Limitations [38]	The exploration of the data becomes difficult due to the number of choices at hand	Very large data requirements of deep learning models to learn, expensive and iterative hyperparameter tuning to increase model accuracy, and limited interpretability of the models
Application scope [39]	Conventional applications such as e-commerce sites, movies, music, etc.	Extends recommender systems to complex and emerging domains such as healthcare and drug discovery, by leveraging diverse real-world data sources
Cold-start problem	Heavily affected by cold-start, struggles to recommend for new users or items with no interaction history	Can partially mitigate cold-start by incorporating side information such as text, images, or metadata through multi-modal learning
Scalability	Efficient and computationally inexpensive for small to medium-sized datasets	Scales well to large datasets but requires significant computational resources, including GPUs
Interpretability	Generally more interpretable, matrix factorization latent factors can be inspected and analyzed	Black-box nature due to hidden layers and learned weights makes it difficult to explain individual recommendations

Table 2.1: Classical vs. Deep Learning-based Recommender Systems

2.2.5 Common Deep Learning Architectures for Recommendation

Modern recommender systems increasingly rely on a small set of deep-learning architectures that can capture complex user–item patterns, sequences, and side information.

MLPs / Neural Collaborative Filtering

Multi-Layer Perceptron (MLP) [40] is an artificial neural network that simulates the connectivity of neurons in the human brain, enabling it to learn and extract complex features from massive datasets. By comparing the similarities between user preferences and item features, MLP may be used to build user profiles and provide accurate lists of recommendations. It can learn and model these features, allowing for more precise predictions of user preferences.

The Neural Collaborative Filtering (Neural CF) model is an improvement over conventional collaborative filtering techniques (Matrix Factorization, MF) in that it makes use of a multi-layer neural network instead of using the dot product operation between the user and item vectors. This allows for more comprehensive interactions between the two vectors, yielding richer combinations of valuable feature information. To enable the neural network to fully perform collaborative filtering, a multi-layer perceptron is employed to model user–item interactions [41].

Autoencoders for Collaborative Filtering

An autoencoder [42] is a neural network model whose core idea is to use an encoder to convert input data into a low-dimensional feature representation, and a decoder to either produce new data or decode the old data back. Theoretically, an autoencoder can be considered a generative model, as it learns the data distribution and may produce fresh data samples. In recommender systems, autoencoders are a useful tool for learning users' latent features to generate content that aligns with their preferences, thereby enabling more accurate personalized recommendations [41].

CNNs and RNNs for Sequential and Contextual Data

Convolutional Neural Networks (CNNs) are applied mainly for content-based recommendations, especially when items are associated with visual or textual information. CNNs are effective in extracting features from images or text, improving recommendations by utilizing content features.

Recurrent Neural Networks (RNNs) are suitable for sequential recommendation tasks where the order of interactions matters, such as predicting the next item a user might

be interested in. RNNs and their variants (like LSTM and GRU) are adept at modeling time-dependent data [41].

2.3 Graph Convolutional Networks

Traditional neural networks are limited to Euclidean spatial data. however, Graph Convolutional Networks (GCNs) have emerged as an innovative technique in artificial intelligence [43] designed to handle non-Euclidean structures such as graphs. In recommender systems, GCNs are widely used to model user interactions with items as graphs. They have proven superior modeling capabilities and have been successfully applied in natural language processing, image recognition, and other fields. GCNs have also been applied in diverse fields such as image processing, intelligent recommender systems, knowledge-based graphs, and more, thanks to their excellent spatial data processing capabilities [44].

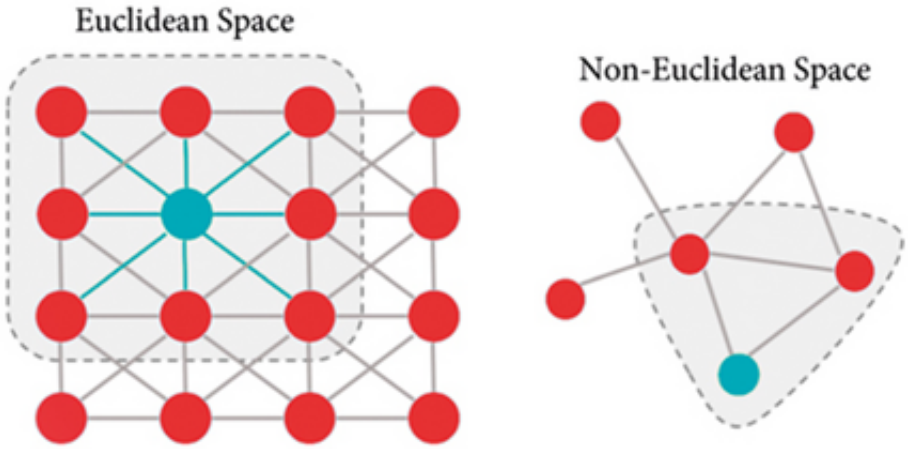


Figure 2.1: The difference between Euclidean space and non-Euclidean space

2.3.1 Graph Representation

A Graph Convolutional Network (GCN) is a class of neural network architectures designed for machine learning on graph-structured data, particularly in the context of node classification, semi-supervised learning, and graph representation learning.

The graph is represented as:

$$G = (V, E) \tag{2.1}$$

where:

- V is the set of nodes (users, items, or both)
- E is the set of edges (links) between nodes (interactions, friendships, etc.)

Graphs can be of various types, for example, in many recommender systems, the graph is a **bipartite graph**, where nodes are divided into two disjoint sets (e.g., users and items), and edges only connect nodes from different sets.

To apply machine learning models to graphs, we first need to convert their abstract structure of nodes and edges into a numerical format that algorithms can process. This is accomplished primarily through two matrices:

- **Adjacency matrix**: captures the graph's topology
- **Feature matrix**: holds the attributes of each node

The Adjacency Matrix

The adjacency matrix A is a square matrix of size $N \times N$. The rule for populating this matrix is simple: the element A_{ij} at the i-th row and j-th column is defined as:

$$A_{ij} = \begin{cases} 1, & \text{if there is an edge between node } i \text{ and node } j \\ 0, & \text{otherwise} \end{cases}$$

By convention, a node is not connected to itself, so the diagonal elements A_{ii} are usually set to 0.

Example. Consider a graph with 4 nodes and edges between nodes 1–2, 1–3, and 3–4. The adjacency matrix A is:

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

For weighted graphs, where edges have different strengths (e.g., interaction frequency), the matrix entries A_{ij} would contain the edge weight instead of just 1.

The Feature Matrix

The feature matrix X is a matrix of size $N \times F$, where each row is the feature vector of a node. For example, we assign two attributes to each person in a group of people: their age and the group they belong to (coded as 0 or 1) [45].

$$X = \begin{bmatrix} 25 & 1 \\ 30 & 0 \\ 22 & 1 \\ 28 & 0 \end{bmatrix}$$

2.3.2 Core Concept of GCN

The idea behind Graph Convolutional Networks (GCNs) is the message passing mechanism, which allows nodes to exchange and aggregate information from their neighboring nodes [43]. The message-passing paradigm consists of three steps:

- **Message**: every node sends its current representation to its neighbors.
- **Aggregate**: each node collects messages from its neighbors and combines them (e.g., sum, mean, or weighted average).
- **Update**: each node transforms the aggregated message to get a new representation [46].

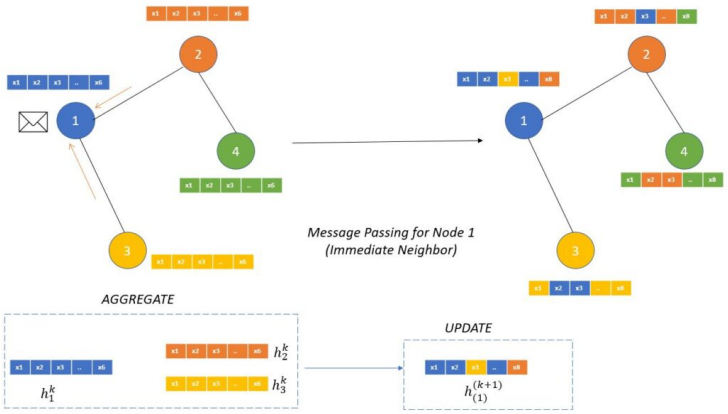


Figure 2.2: Message Passing in Graph Convolutional Networks

The process of information propagation occurs across multiple layers of the network. At each layer, nodes pass their updated representations to their neighbors. As more layers are stacked, information spreads further across the graph, allowing each node to access knowledge from multi-hop neighborhoods and update its state accordingly.

2.3.3 GCN Layer Propagation Rule

The layer propagation rule defines how node embeddings are updated from one layer to the next using message passing and neighborhood aggregation. It is given as follows:

$$H^{(l+1)} = \sigma\left(\tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}H^{(l)}W^{(l)}\right) \tag{2.2}$$

$H^{(l)}$ represents the feature matrix at layer l and $H^{(l+1)}$ represents the updated feature matrix at the next layer. The matrix $W^{(l)}$ is a trainable weight matrix for layer l.

The term $\tilde{A} = A + I$ represents the adjacency matrix with added self-loops, ensuring that each node retains its own features during the aggregation. $\tilde{D}$ is the diagonal degree matrix computed from $\tilde{A}$, and $\sigma(\cdot)$ is a non-linear activation function.

First, self-loops are added to the graph structure, allowing each node to retain its original information. Second, the adjacency matrix is symmetrically normalized using $\tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}$, which distributes information evenly across neighboring nodes. Third, the node features are linearly transformed through multiplication with the learnable weight matrix $W^{(l)}$. By stacking multiple GCN layers, nodes are able to capture information from higher-order neighborhoods and learn more comprehensive representations of the graph structure [47].

2.3.4 Architecture of GCN

GCNs typically consist of multiple layers, each layer is responsible for improving node representations.

1. **Input Layer**: The input layer initializes the node features, usually from raw data or pre-trained embeddings.
2. **Hidden Layers**: Hidden layers perform the graph convolution operations, progressively aggregating and transforming node features.
    - **A. Graph Convolutional Layers**: These layers perform the convolution operation on the graph. Each layer updates the feature representation of a node by aggregating the features of its neighbors.
    - **B. Activation Functions**: Non-linear functions such as ReLU are applied to the output of each convolutional layer to introduce non-linearity into the model.
    - **C. Pooling Layers**: These layers reduce the dimensionality of the graph by merging nodes, which helps in capturing hierarchical structures.
3. **Output Layer**: The output layer produces the final node embeddings or predictions, depending on the task (e.g., node classification, link prediction).
4. **Fully Connected Layers**: These layers are used at the end of the network to perform tasks such as classification or regression [48].

2.3.5 LightGCN

LightGCN is a simplified graph convolutional network specifically designed for collaborative filtering recommendation tasks. Unlike traditional GCN models, which employ feature transformation matrices and nonlinear activation functions at each graph convolution layer, LightGCN removes these components and retains only the neighborhood

aggregation operation. This simplification is based on the observation that, in recommendation scenarios, repeatedly propagating and aggregating user and item embeddings across the interaction graph is sufficient to capture collaborative signals effectively.

The model represents users and items as nodes in a bipartite graph and learns their embeddings by propagating information through multiple graph layers. At each layer, a node updates its representation by aggregating the embeddings of its neighboring nodes. The final embedding of each user and item is obtained by combining the representations learned from all propagation layers, allowing the model to capture both direct and higher-order user–item relationships.

Due to its lightweight architecture, LightGCN reduces computational complexity and training overhead while achieving strong recommendation performance. These advantages have made it one of the most widely adopted graph-based recommendation models and a strong baseline for modern recommender systems. In this work, LightGCN serves as the graph-learning backbone, providing collaborative representations of users and items that are subsequently integrated with sequential and memory-based components to enhance recommendation accuracy and explainability [49].

2.3.6 Representative Graph-Based Recommendation Models

Following the introduction of LightGCN as the graph-learning backbone of this work, it is useful to survey the broader landscape of graph-based recommendation models that have emerged from the same line of research.

NGCF — Neural Graph Collaborative Filtering

Neural Graph Collaborative Filtering (NGCF) is a graph neural network (GNN)–based collaborative filtering model that explicitly integrates the bipartite user–item interaction graph into the embedding learning process. Instead of learning user and item embeddings from IDs only, NGCF propagates embeddings over the user–item graph via learned graph convolution layers. At each layer, a user (or item) embedding is updated by aggregating embeddings of its neighbors (items the user interacted with, or users who interacted with the item), while also preserving the original embedding. This allows the model to capture high-order connectivity (multi-hop relationships) in the graph, injecting the collaborative signal directly into the embedding process[50].

DGCF — Disentangled Graph Collaborative Filtering

Disentangled Graph Collaborative Filtering (DGCF) is a graph-based collaborative filtering model that extends NGCF by learning disentangled latent factors (i.e., separate representation channels) for users and items. Instead of a single monolithic embedding,

DGCF decomposes each user and item embedding into multiple latent factors, each capturing a different "reason" or aspect of preference (e.g., different interests, styles, or functional categories). The model then uses a graph convolution mechanism to propagate these disentangled factors over the user–item graph, and learns to align user factors with item factors that are relevant to them[51].

SGL — Self-Supervised Graph Learning

Self-Supervised Graph Learning (SGL) is a self-supervised graph neural network framework for collaborative filtering that combines:

A graph neural network (typically based on LightGCN-style propagation) over the user–item interaction graph, and

Self-supervised learning via graph augmentations and contrastive objectives.

SGL constructs multiple augmented views of the user–item graph (e.g., by randomly dropping edges, dropping nodes, or dropping features) and then uses a contrastive loss to enforce that the embeddings of the same user/item are consistent across different augmented views. This self-supervised objective supplements the standard recommendation loss, improving the robustness and quality of learned embeddings, especially when data is sparse or noisy[52].

SimGCL — Simple Graph Contrastive Learning

Simple Graph Contrastive Learning (SimGCL) is a graph contrastive learning framework for collaborative filtering that simplifies the design of graph-based contrastive recommendation models. Unlike earlier graph contrastive methods that rely on complex graph augmentations (edge dropping, node dropping, subgraph sampling), SimGCL:

Avoids explicit graph augmentations, and instead

Adds uniform noise directly to the learned user/item representations, and

Uses a contrastive loss (e.g., InfoNCE) to regularize the representation distribution toward uniformity and improve discriminability.

SimGCL argues that the main performance gain comes from optimizing the contrastive loss and obtaining a uniform representation distribution, rather than from sophisticated graph augmentations. This makes the model simpler, more efficient, and easier to train, while still achieving strong recommendation performance[53].

2.4 Sequential Modeling with Transformer

2.4.1 Limitations of RNN-Based Models

In recommender systems, Transformers are commonly used to model user behavior as sequential interactions over time [54]. RNNs and their variants, such as LSTMs and GRUs, represented a significant step forward for sequence modeling in recommender systems [54]. However, they face a number of challenges when handling complex and long sequences.

The Challenge of Long-Range Dependencies: RNNs have difficulty capturing long-term dependencies because the hidden state is updated step by step, and gradients vanish between elements that are far apart in a sequence.

Sequential Computation and Weak Parallelization: RNNs process sequences sequentially (element by element), which becomes a significant bottleneck for long sequences.

These limitations motivated the development of newer architectures, like the Transformer [55].

2.4.2 Transformer History

The Transformer architecture was introduced in June 2017 [3]. This was followed by the introduction of several influential models, including:

- June 2018: GPT, the first pretrained Transformer model.
- October 2018: BERT, designed for bidirectional contextual language understanding.
- February 2019: GPT-2, an improved version of GPT.
- October 2019: T5, a multi-task focused implementation of the sequence-to-sequence Transformer architecture.
- May 2020: GPT-3, a larger version of GPT-2 that can perform a variety of tasks without fine-tuning.
- January 2023: LLaMA, a large language model capable of generating text in multiple languages.
- March 2023: Mistral, a 7-billion-parameter language model that outperforms LLaMA 2 13B across all evaluated benchmarks.
- May 2024: Gemma 2, a family of lightweight, state-of-the-art open models ranging from 2B to 27B parameters that incorporate interleaved local-global attentions and group-query attention.

- November 2024: SmolLM2, a state-of-the-art small language model (135 million to 1.7 billion parameters) that achieves impressive performance despite its compact size. [56]

2.4.3 Transformers in Recommender Systems

Sequential deep learning models have proven benefits in the recommendation industry for generating more relevant and dynamic recommendations based on users' past sequences of actions [57]. Transformers are a type of model that uses self-attention mechanisms to enhance the model's ability to capture long-range dependencies in user interactions [57, 3]. Transformers can consider the importance of different items in a user's history when making recommendations.

For example, when a user prefers item A, but their preferences change to item B after a while, a traditional recommender system might continue to recommend A based on the user's history, but a Transformer-based system could recognize the recent shift in preference and start recommending B. It can also understand the context of a user's actions. for example, if a user often watches horror movies on Friday nights, a Transformer-based system could pick up on this pattern and recommend horror movies on Fridays. Their ability to understand context and handle sequential data can lead to more accurate and personalized recommendations [58].

2.4.4 Transformer Architecture

The first Transformer architecture was introduced in the paper "Attention Is All You Need" [3]. It was designed to translate text from one language to another and is illustrated as follows:

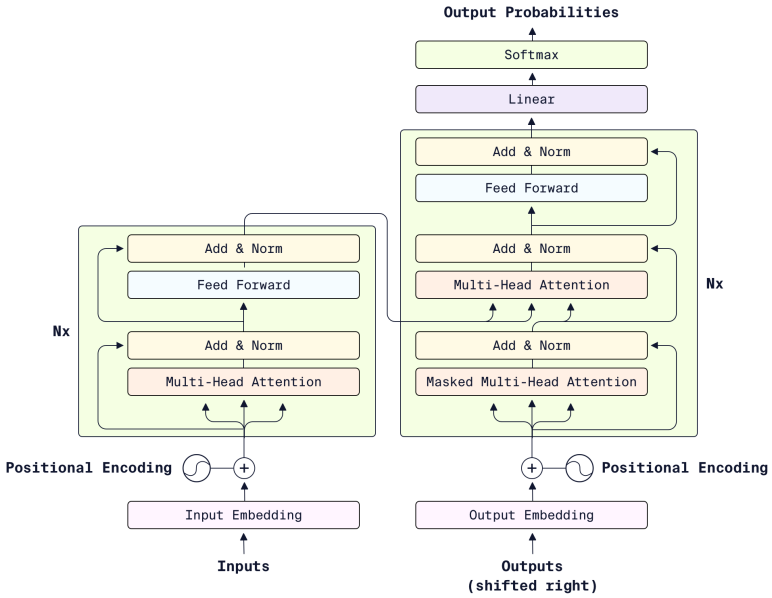


Figure 2.3: Transformer architecture [3]

A Transformer architecture [59] is composed of stacked encoder and decoder layers built upon self-attention mechanisms. Each layer includes multi-head self-attention, feed-forward neural networks, residual connections, and layer normalization.

The encoder processes the entire input sequence in parallel using self-attention, allowing each token to attend to all other tokens. In contrast, the decoder generates outputs sequentially while applying masked self-attention to prevent access to future tokens.

In many modern applications such as recommender systems and representation learning, only the encoder component is used to extract meaningful embeddings from input data.

2.4.5 Representative Sequential Recommendation Models

GRU4Rec — Session-Based Recommendations with RNNs

GRU4Rec (Gated Recurrent Unit for Recommendations) is a sequential / session-based recommendation model that uses Recurrent Neural Networks (RNNs) with Gated Recurrent Unit (GRU) cells to model user interaction sequences. GRU4Rec processes items in a session one by one, updating a hidden state at each step that captures the temporal dynamics of the session. The final hidden state (or a learned combination of hidden states) is used to predict the next item in the session. Later variants (e.g., GRU4RecF, GRU4RecT) introduce enhancements such as re-editing the final hidden state and item-level sampling to improve performance, but the core idea remains: GRU-based RNNs to learn sequential patterns in short interaction sessions[60].

SASRec — Self-Attentive Sequential Recommendation

Self-Attentive Sequential Recommendation (SASRec) is a Transformer-based sequential recommendation model that uses a single-directional self-attention mechanism to model user interaction sequences. At each time step, SASRec adaptively assigns attention weights to all previous items in the user's history, allowing the model to capture both short-term and long-range dependencies without relying on RNNs or Markov chains. The model predicts the next item by computing a representation of the current sequence via self-attention layers and then ranking all items based on this representation[61].

BERT4Rec — Sequential Recommendation with Bidirectional Encoders

BERT4Rec (Bidirectional Encoder Representations from Transformers for Recommendation) is a sequential recommendation model that adapts the BERT architecture from natural language processing to the recommendation domain. Instead of a unidirectional RNN or left-to-right Transformer, BERT4Rec uses a deep bidirectional self-attention network to model user behavior sequences. It is trained with a Cloze (masking) objective: random items in the sequence are masked, and the model learns to predict them by jointly conditioning on both their left and right context. This bidirectional modeling allows each item in the history to fuse information from both past and future items (in the sequence), leading to richer representations and more accurate next-item predictions[37].

2.4.6 Introduction to Attention

Although humans tend to think they perceive the entire scene at once, this is not entirely accurate [62]. The eye focuses on a small area called the fovea, where details are clear, while other elements are not seen with the same level of detail. The brain gradually understands what the scene contains. This is called visual attention simply focusing on the important parts and ignoring the unimportant.

Based on this concept, this behavior has been simulated in neural networks. Instead of assigning equal importance to all input data, attention allows the model to focus more on the relevant parts. This has become extremely useful in deep learning, especially when dealing with long sequences. In 2015, attention was introduced into neural machine translation models. Before that, encoding-decoding models attempted to compress the entire sentence into a single vector, which was not very efficient with long sentences. The new approach allowed the model to consider different parts of the input sentence while producing the output, thus improving the results [62, 3].

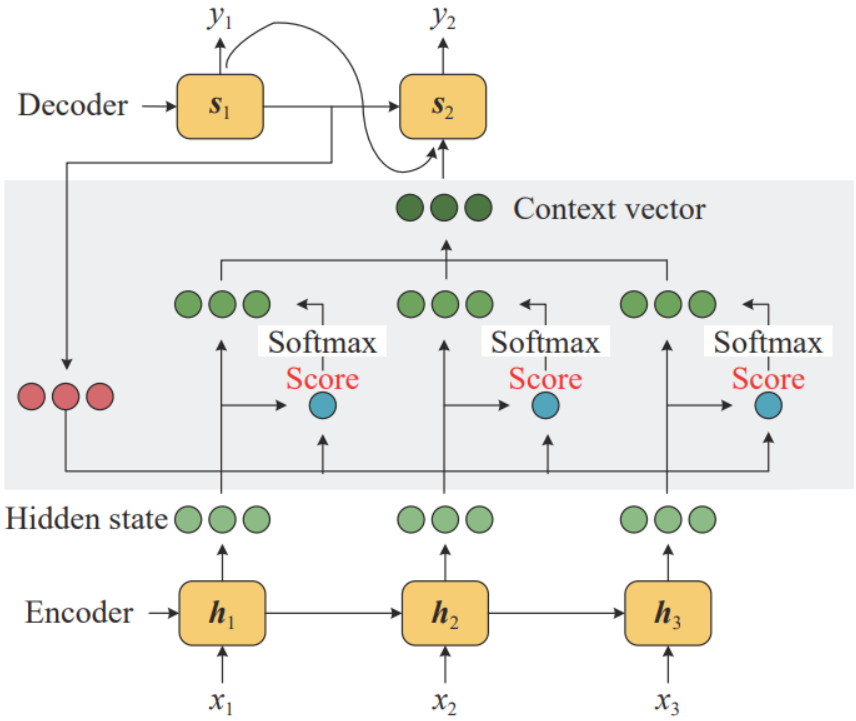


Figure 2.4: Classic encoder-decoder network with attention mechanism in Seq2Seq structure [4]

2.4.7 Role of Attention in Recommender Systems

Attention mechanisms play an important role in modern recommender systems by improving the model's ability to focus on the most relevant parts of user behavior. Instead of treating all past interactions equally, attention assigns different importance weights to different items in the user history. In recommendation scenarios, users often interact with many items, but only a subset of these interactions is truly relevant for predicting future preferences. Attention helps the model identify these important interactions and reduce the influence of less relevant ones.

Moreover, attention improves the interpretability of recommendation models by showing which past actions contributed most to the final recommendation. This is particularly useful in deep learning-based recommender systems where interpretability is often limited. Finally, attention mechanisms are widely used in sequential recommendation models and Transformer-based architectures, where they significantly enhance the ability to capture long-term dependencies and dynamic user preferences [57].

2.4.8 Self-Attention Mechanism

Self-attention is a mechanism in which each token in the input attends to all other tokens, including itself, to generate its contextual embedding. Self-attention computes relation-

ships between all elements in parallel by projecting the input into three vectors: Query (Q), Key (K), and Value (V). The relevance between elements is then calculated using the following formula [3]:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.3}$$

2.4.9 Multi-Head Attention

A multi-head attention mechanism combines multiple attention heads [3] to capture different types of relationships within the input. The model runs several attention processes in parallel, rather than performing a single attention function, allowing it to focus on multiple pieces of information simultaneously. A multi-head attention mechanism is defined as follows [3]:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \cdots, \text{head}_h)\, W^O \tag{2.4}$$

Each attention head is computed by:

$$\text{head}_i = \text{Attention}(QW_i^Q,\, KW_i^K,\, VW_i^V) \tag{2.5}$$

and the attention function is given by:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.6}$$

2.4.10 Memory-Augmented Recommendation Models

Standard sequential recommendation models typically focus on a limited number of recent user interactions. Although this approach is effective for capturing short-term preferences, it may overlook important information contained in older interactions. To address this limitation, researchers have introduced memory-augmented recommendation models that incorporate external memory structures capable of storing and retrieving long-term user preferences.

These approaches extend traditional recommendation architectures by providing an additional storage mechanism that complements the sequential representation. As a result, information that falls outside the fixed sequence window can still contribute to the recommendation process.

One representative example is the User Memory Network proposed by Chen et al. [63], which combines sequential recommendation with an external memory structure. The memory component stores long-term behavioral patterns and allows the model to retrieve relevant information when generating recommendations, improving the representation of user preferences over extended interaction histories.

Memory Networks

Memory Networks were introduced by Weston et al. [64] as neural architectures that combine inference mechanisms with an external memory storage. Information is stored in memory slots and accessed through an attention-based addressing mechanism.

Given a query vector, the model computes similarity scores between the query and memory entries. These scores are then used to retrieve the most relevant information from memory. This mechanism enables the network to incorporate long-term knowledge while maintaining efficient computation.

Memory Networks have been successfully applied in several domains, including natural language processing, question answering, and recommender systems, where long-term information is important for decision making.

Neural Turing Machines

Neural Turing Machines (NTMs) were proposed by Graves et al. [65]. They extend neural networks with an external differentiable memory matrix that supports both read and write operations.

Unlike Memory Networks, which primarily focus on memory retrieval, NTMs can dynamically modify memory contents during learning. This capability enables the model to solve tasks requiring long-term storage, reasoning, and sequential processing.

However, the increased flexibility of NTMs comes at the cost of higher computational complexity and more challenging optimization procedures. Consequently, simpler memory architectures are often preferred in recommender systems where scalability and training efficiency are important considerations.

2.4.11 External Memory Mechanisms

External memory mechanisms represent a lightweight alternative to more complex memory architectures. Instead of relying on iterative read-write operations, they typically maintain a set of learnable memory slots that store compressed representations of user interests.

During recommendation, a query vector is compared with the memory contents through an attention-based retrieval mechanism. The retrieved memory representation is then combined with other model components to enrich the final user representation.

External memory approaches are conceptually related to Memory Networks [64] and Neural Turing Machines [65], as they all extend neural models with differentiable memory components. However, external memory modules used in recommender systems are generally simpler and computationally more efficient.

Compared with Memory Networks and Neural Turing Machines, lightweight external memory mechanisms rely on a single attention-based retrieval step rather than iterative

memory updates. This design reduces computational cost while still preserving useful long-term preference information. For recommendation tasks, such simplified memory structures often provide a practical balance between representational capacity and computational efficiency.

2.4.12 Explainable AI in Recommender Systems

Traditional recommender systems primarily focus on improving prediction accuracy and ranking performance. However, modern recommendation models, especially those based on deep learning, often behave like black boxes, making their decisions difficult to understand and interpret. Explainable recommendations aim to develop models that not only produce high-quality recommendations but also provide clear and intuitive explanations to the user. These explanations may be generated subsequently or derived directly from an explainable model [66]. For this reason, explainable artificial intelligence (XAI) has become an important research direction in recommender systems.

Explainability in recommender systems aims to provide understandable reasons behind the recommendation process. Instead of only suggesting an item to the user, the system also explains why this item was recommended. These explanations help users better trust the system and improve transparency and user satisfaction.

Explainable recommender systems are useful in many practical applications because users generally prefer recommendations that can be justified. For example, a movie may be recommended because the user previously watched similar movies, interacted with the same genre, or showed behavior similar to other users with comparable interests.

Explainability methods in recommender systems can be divided into several categories. Some methods rely on neighborhood-based explanations, where recommendations are explained through similar users or similar items. Other methods use content information such as genres, tags, or item attributes. More recent deep learning approaches employ attention mechanisms, feature attribution techniques, and interpretable neural networks to identify the most influential interactions behind the recommendation process.

In sequential recommender systems, explainability becomes even more important because user preferences evolve over time. In this case, explanations may depend on recent interactions, temporal patterns, contextual information, or sequential transitions between consumed items.

The integration of explainability into recommender systems offers several important advantages:

- Improving user trust and transparency.
- Helping users understand recommendation decisions.
- Allowing developers to analyze model behavior.

- Detecting possible bias or abnormal predictions.
- Increasing user satisfaction and system reliability.

Despite these advantages, explainability remains challenging in deep recommendation models because their architectures are often highly complex and involve large numbers of parameters and latent representations. Therefore, several explainability approaches have been proposed in recent years, including SHAP, LIME, attention-based explanations, surrogate models, and gradient-based interpretation methods [66].

2.4.13 Explainability Limitations of Existing Models

Matrix Factorization (MF / SVD++)

Traditional collaborative filtering models, such as SVD++, decompose user interactions with items into latent factor vectors. These latent dimensions are entirely opaque, carry no semantic meaning, and cannot be linked to any observable behavior or item attribute. Any interpretation requires an external tool that does not reflect the model's internal processes.

NCF

NCF replaces the dot product in MF with a multi-layered neural network, increasing the model's opacity. External methods like LIME or SHAP can be applied, but these tools alone are insufficient to understand how the model processes information internally.

Sequential Models: BERT4Rec and SASRec

Although attention mechanisms improve the ability of recommendation models to capture important user–item interactions, attention weights should not be interpreted as definitive explanations of the model's decisions. Attention weights do not always reflect the true importance of a feature. While they provide useful insights, they are insufficient to be considered entirely reliable explanations [67].

Graph-Based Models: LightGCN

The LightGCN model projects embedded representations onto a graph of user interaction with an item, allowing for partial interpretation through neighborhood structure: a recommended item can be attributed to collaborative signals from similar users. However, this interpretation only covers the graph component and ignores any sequential dynamics.

2.4.14 SHAP for Explainable Recommendation

SHAP (SHapley Additive exPlanations) is one of the most widely used interpretation methods in machine learning. It is based on Shapley values from cooperative game theory, whereby each feature is assigned a contribution value according to its effect on the final prediction [68].

The core idea of SHAP is to explain the prediction by calculating the contribution of each feature individually. Instead of simply giving a final recommendation score, SHAP shows which features increased or decreased that score.

For a prediction function $f(x)$, the explanation model is represented as:

$$g(z') = \phi_0 + \sum_{i=1}^{M} \phi_i z'_i$$

where:

- ϕ_0 represents the base prediction value,
- ϕ_i represents the contribution of feature i,
- z'_i indicates whether the feature is present or absent.

The Shapley value is calculated by measuring the average contribution of each feature to the final prediction within the various possible subsets of other features:

$$\phi_i = \sum_{S \subseteq F \setminus \{i\}} \frac{|S|!(|F| - |S| - 1)!}{|F|!} \left[f(S \cup \{i\}) - f(S) \right]$$

where:

- F is the set of all features,
- S is a subset of features excluding feature i,
- $f(S)$ is the prediction using subset S.

The SHAP framework is used in recommender systems to explain why a particular item is suggested to a user, by illustrating the influence of factors such as recent interactions, preference types, user similarity, and past behavior patterns on the final recommendation.

One of the main advantages of SHAP is that it provides both local and global explanations. Local explanations help interpret a single recommendation, while global explanations help understand the overall behavior of the recommendation model.

However, computing exact Shapley values is computationally expensive because the number of feature combinations grows exponentially with the number of inputs. Therefore, practical implementations usually rely on approximation methods such as KernelSHAP or DeepSHAP [68].

2.4.15 Attention-Based Explanations in Recommender Systems

Most modern recommender systems rely heavily on user interactions and historical behavior data. In recent years, attention mechanisms have become an important component in deep learning recommendation models because they improve the model's ability to focus on the most relevant interactions within the user sequence.

Attention mechanisms are mainly designed to assign different importance weights to the elements of a sequence. Instead of treating all previous interactions equally, the model learns which items are more important for predicting the next interaction. In sequential recommender systems, this allows the model to capture both short-term and long-term user interests more effectively.

Mathematically, attention can be represented as a weighted combination of input vectors [3]:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

where:

- Q represents the query matrix,
- K represents the key matrix,
- V represents the value matrix,
- d_k is the dimensionality scaling factor.

The resulting attention weights determine how strongly the model focuses on different elements of the sequence during prediction.

Because attention weights indicate which interactions receive higher importance, many researchers consider attention mechanisms as a possible source of explanation in recommender systems. The idea is that if the model strongly attends to certain items or interactions, these interactions may help explain why a recommendation was generated.

However, the relationship between attention and explainability remains controversial in the literature. Several studies suggest that attention weights do not always provide reliable explanations of model behavior, while other works argue that attention can become explanatory when the mechanism is properly adapted to the prediction task [69].

The main argument is that changing the attention distribution may alter the prediction results, which indicates that attention contributes directly to the decision-making process. This makes attention mechanisms potentially useful for understanding user preference patterns and analyzing model behavior.

One important advantage of attention-based explanations is that they are naturally integrated inside the neural model itself. Unlike post-hoc explainability methods, atten-

tion weights are generated directly during the forward propagation process, which makes explanation generation computationally efficient.

Attention mechanisms are also useful for detecting potential bias in recommendation models. For example, abnormal attention distributions may reveal that the model focuses too strongly on specific interaction patterns or user groups.

In this work, attention mechanisms are used not only to improve recommendation accuracy but also to provide interpretable information about which previous interactions contributed most to the final recommendation result [69].

2.4.16 Other Common Explainability Methods

Besides SHAP and attention-based explanations, several other explainability methods have been proposed in recommender systems and machine learning. These methods aim to improve transparency and help understand how recommendation models generate their predictions.

- **LIME and Local Explanation Methods**

  LIME (Local Interpretable Model-agnostic Explanations) is one of the most popular local explanation techniques. It explains individual predictions by approximating the behavior of a complex model using a simpler interpretable model around a specific prediction instance [70].

  The main idea of LIME is to slightly perturb the input data and observe how the prediction changes. A simple linear model is then trained locally to estimate the contribution of each feature to the prediction.

  In recommender systems, LIME can help explain why a specific item was recommended to a user by identifying the most influential interactions or behavioral features affecting the recommendation score.

- **Surrogate Models and Rule-Based Explanations**

  Surrogate explanation methods rely on building an interpretable model that approximates the behavior of a more complex black-box model. The surrogate model may take the form of a decision tree, a linear regression model, or a set of logical rules.

  Rule-based explanations are particularly useful because they provide human-readable interpretations. For example, a recommendation may be explained using rules such as:

  > "Users who watched science-fiction movies and highly rated action films are likely to watch this movie."

These methods improve interpretability and help developers better analyze the decision process of recommender systems.

- **Feature-Importance and Gradient-Based Methods**

  Feature-importance methods aim to measure how much each input feature contributes to the final prediction. Some approaches evaluate feature importance by observing the prediction change after removing or modifying specific features.

  Gradient-based explanation methods are mainly used in deep learning models. They analyze gradients computed during backpropagation in order to determine which input features most strongly influence the prediction output.

  In recommender systems, these methods can help identify which previous interactions, genres, contextual information, or sequential patterns contributed most to the generated recommendation.

Although these methods improve transparency and interpretability, each approach has its own limitations regarding computational complexity, stability, and explanation quality. Therefore, modern recommender systems often combine multiple explainability techniques to obtain more reliable and meaningful interpretations [70].

2.5 Conclusion

In this chapter, we presented the theoretical foundations of modern recommender systems, from traditional recommendation approaches to deep learning-based architectures. We also discussed the limitations of classical methods, which were a weakness especially when dealing with sparse data, dynamic user preferences, and complex interaction patterns. We introduced several important deep learning architectures used in recommender systems, including neural collaborative filtering models, autoencoders, and convolutional and recurrent neural networks, as well as graph-based approaches. We focused on Graph Convolutional Networks, which enable us to model user–item relationships through graph structures and message passing mechanisms. In this chapter, we also explored key sequential recommendation models based on the Transformer architecture, highlighting the role of self-attention mechanisms in capturing temporal dependencies between user interactions. Finally, we discussed key interpretation methods such as SHAP, attention-based interpretations, and LIME, with the aim of improving the interpretability and transparency of recommender systems.

Chapter 3

Development of the Proposed Hybrid Approach

Chapter 3

Development of the Proposed Hybrid Approach

3.1 Introduction

Despite recent advancements in recommender systems, relying on a single model remains insufficient to fully capture the complexity of user preferences and their dynamic evolution over time. For this reason, hybrid architectures have emerged as a promising direction, aiming to combine the strengths of different modeling paradigms to improve recommendation accuracy and robustness.

Currently, users are exposed to an extremely large volume of content, particularly in domains such as movie recommendation, where they are confronted with an overwhelming number of choices. This makes the decision-making process increasingly difficult and highlights the need for intelligent systems capable of providing personalized and relevant suggestions.

In this context, this work addresses the design and implementation of a hybrid recommendation model for movie recommendation tasks. The proposed approach integrates graph-based collaborative filtering, Transformer-based sequential modeling, an external memory module, and a cross-attention fusion mechanism in order to exploit user–item interaction structures at multiple levels. This combination enables a richer and more expressive representation of user preferences, leading to more accurate and adaptive recommendations.

This chapter is organized as follows. Section 3.2 presents the dataset and justifies its selection. Section 3.3 describes the preprocessing pipeline. Section 3.4 details each architectural component and its propagation rules. Section 3.5 covers the training strategy and loss design. Section 3.6 defines the evaluation protocol. Section 3.7 discusses the design choices and hyperparameter selection rationale. Section 3.8 presents the three-level explainability framework.

3.2 Dataset and Data Collection

To evaluate the proposed system, a real-world dataset was used to analyse user behavior and preferences during their interaction with films. The MovieLens 1M dataset was selected for the following reasons. First, it is one of the most widely adopted benchmarks in recommender system research, enabling direct and fair comparison with existing baselines such as LightGCN [49] and SASRec [61]. Second, it provides a sufficient volume of interactions (approximately one million) to train the four-component hybrid architecture without severe data sparsity. Third, every interaction carries a Unix timestamp, which is essential for the chronological splitting strategy and for the sequential modeling component. Alternative datasets were considered but found less suitable for this work: MovieLens 20M is larger but would make training the hybrid model computationally prohibitive given the available hardware, Amazon product datasets contain sparser interactions and fewer timestamps per user.

The dataset contains user ratings of films, with each rating representing an interaction from which viewers' preferences can be inferred. Rather than using rating values as regression targets, the focus was placed on the interaction signals themselves treating any recorded rating as evidence of engagement. Most large-scale platforms such as Netflix rely on the same implicit feedback principle, as behavioral signals are a more reliable indicator of user interest than explicit numerical scores. The MovieLens 1M dataset therefore provides a solid foundation combining interaction patterns and timestamp information the two inputs required by the proposed hybrid architecture.

Table 3.1: Dataset Statistics

Statistic	**Value**
Total Users	6,040
Total Items	3,706
Total Interactions	1,000,209
Min. Interactions / User	5
Median Interactions / User	165
Timestamp Span	1995–2003

3.3 Data Preprocessing

3.3.1 User and Item Filtering

In the raw data, a large number of users showed few interactions, and many items were rarely evaluated. This negatively impacted the model's results. A filtering process was therefore applied to the raw data by setting a minimum interaction threshold of five interactions. Only users and items with at least five interactions were retained, which

mitigates the data sparsity problem that is a known limitation of recommender systems. This filtering improved the quality of the model input and led to more accurate results.

3.3.2 Stratified User Sampling

Users were divided into categories based on the number of interactions, and a proportional number of users were selected from each category. This approach prevents the model from being biased towards only the most active users and ensures a balanced representation of all behavioral patterns. This procedure improves the model's generalizability and reduces the risk of bias.

3.3.3 ID Remapping (user2idx / item2idx)

After data cleaning, user and item IDs were converted into sequential numeric indices. Two dictionaries were created:

- `user2idx`: to convert the user ID to a numeric index.
- `item2idx`: to convert the item ID to a numeric index.

This conversion is essential because deep learning models, especially those built with PyTorch, rely on numeric representations rather than textual identifiers. This procedure also contributes to improved storage efficiency and execution speed within the model.

3.3.4 Chronological Sequential Splitting (Train / Validation / Test)

The data for each user was split chronologically, preserving the order of interactions as they actually occurred. The following strategy was adopted:

- Training data: all interactions except the last two.
- Validation data: the second-to-last interaction.
- Test data: the last interaction.

This splitting strategy reflects the real-world scenario of recommender systems, where the next interaction is predicted based on past behavior. This approach helps the model learn temporal patterns and changes in user preferences.

3.4 Model Architecture

The proposed system combines three complementary paradigms: graph-based collaborative filtering (LightGCN), Transformer-based sequential modeling, and an external memory module, unified through a cross-attention fusion mechanism. The architecture is designed to capture user–item collaboration signals, sequential behavior patterns, and persistent user preference representations. This results in a richer and more personalized user representation than any single component could provide.

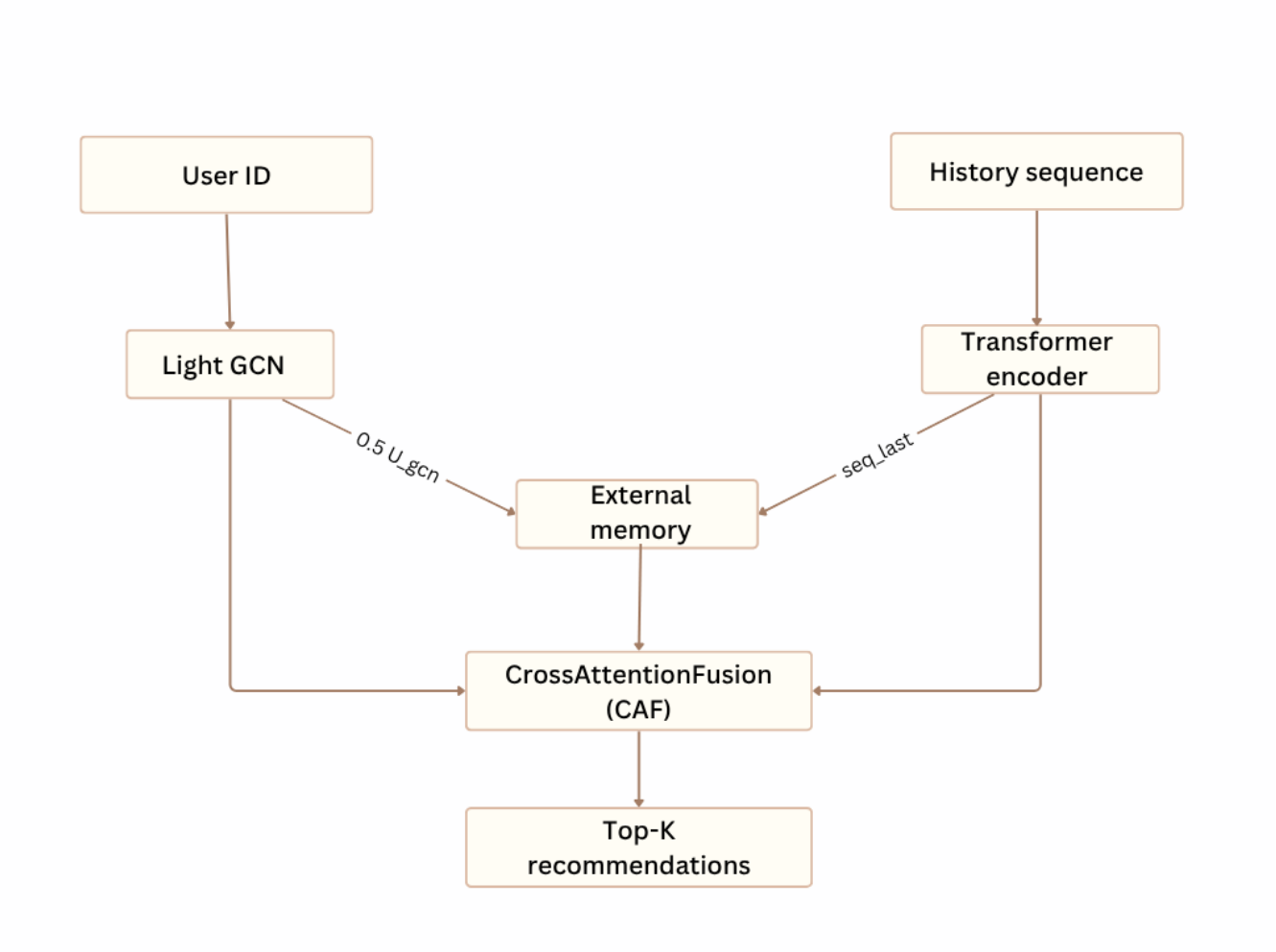


Figure 3.1: DynamicHybridRec architecture parallel components and fusion

3.4.1 LightGCN Collaborative Filtering Backbone

LightGCN propagates information across the interaction graph, allowing each user's embedding to absorb signals from the items they consumed and, transitively, from users who consumed the same items [49]. The model uses a user–item bipartite graph, meaning that users and items are nodes and the interactions between them are edges. The total number of nodes is:

$$N = n_{\text{users}} + n_{\text{items}}$$

The graph is constructed from the training interaction sequences, where each user is connected to all items with which they have interacted. The graph is modeled as an undirected graph by creating bidirectional edges between users and items:

```
all_r = torch.cat([row_t, col_t + n_users])
all_c = torch.cat([col_t + n_users, row_t])
```

Self-loops are additionally included to preserve the original node information during embedding propagation.

3.4.2 Transformer-Based Sequential Encoder

A Transformer-based sequential encoder was incorporated because collaborative filtering alone is insufficient to model the dynamic changes in user interests over time, as recent interactions are often more significant than older ones. The last 80 interactions for each user are extracted and converted into embedding vectors before being processed by the Transformer encoder. The Transformer takes the user's interaction log as input. Each element within the log is represented using embeddings generated by the GCN component, which already contain collaborative information extracted from user–item interactions within the graph. The encoder consists of three main components:

- **Positional Encoding**

  The Transformer model does not inherently understand sequential order, so a learnable positional embedding of size (L_{seq}, D) is added to the item embeddings. This enables the model to determine the relative position of each film within the interaction sequence. Layer Normalization and Dropout are then applied to improve training stability and reduce overfitting.

- **Transformer Blocks**

  The sequential encoder contains two Transformer blocks. Each block consists of several sublayers that work together to model sequential dependencies.

  - **Multi-Head Causal Self-Attention**

    The model employs a multi-head self-attention mechanism with four attention heads. This allows each position in the sequence to attend to previous interactions and learn the relationships between previously watched films. A causal attention mask is applied so that each position can only attend to previous positions and not future ones. This is crucial because recommender systems must predict future interactions based solely on past behavior. A padding

mask is additionally used to ignore positions added to sequences shorter than the specified length.

- **Feed-Forward Network (FFN)**

  The output of the attention layer passes through a feed-forward network consisting of two fully connected layers with a GELU activation function. The hidden dimension is expanded from D to $D_{\text{ffn}} = 256$ and then projected back to its original dimension. Dropout is applied to improve regularization.

- **Residual Connections and Layer Normalization**

  Residual connections are added around both the attention layer and the FFN layer to preserve information flow and facilitate gradient propagation during training. Pre-layer normalization is applied before each sublayer to stabilize training and improve convergence.

- **LayerDrop Regularization**

  A random layer-dropping technique is employed during training, whereby some residual branches are occasionally skipped with a small probability that gradually increases with layer depth. This technique helps reduce overfitting and improves the model's ability to generalize, while preserving the full network structure during inference.

- **Final Sequential Representation**

  After passing through all Transformer blocks, a final layer normalization is applied. The last hidden state is then extracted as the final sequential representation of the user:

$$\mathbf{h}_{\text{last}} \in \mathbb{R}^D$$

  This vector summarizes the recent behavioral sequence of the user and is later combined with the collaborative representation generated by the GCN component.

The Transformer module used in this work is based on a self-attentive sequential recommendation architecture similar to SASRec [61].

3.4.3 External Memory Module

Standard sequence encoders only process information within the last T tokens. For users whose interaction history exceeds T, information from earlier interactions is permanently discarded. The external memory module addresses this limitation by maintaining a set of $M = 128$ learnable memory slots, each of dimension $d_{\text{mem}} = 64$, that store compressed long-term preference patterns.

Soft Read Mechanism

At each forward pass, a query embedding derived from seq_last $+ 0.5 \times u_{\text{gcn}}$ is projected into the memory key space and normalized. Attention weights over the memory cells are computed using a softmax function with a sharpness factor of 8.0:

$$\alpha = \text{softmax}\left(8.0 \cdot \text{normalize}(W_q \cdot q) \cdot \text{normalize}(K)^T\right)$$

$$\text{mem_out} = W_{\text{out}} \cdot (\alpha \cdot V)$$

The combination of L2 normalization and a high sharpness factor encourages sparse, focused retrieval from the memory bank, similar to a nearest-neighbor search in a compressed preference space. Both the key matrix K and the value matrix V are learnable parameters updated via gradient descent.

Design Motivation and Architectural Choice

The external memory module was introduced to preserve long-term user preferences that may be lost when only the most recent interactions are processed by the sequential encoder. Although Transformer-based recommendation models are effective at capturing short-term sequential dynamics, they remain constrained by the fixed sequence window length T. Consequently, interactions located far back in the user history may no longer contribute to the final recommendation representation.

Instead of employing more complex memory architectures such as Memory Networks or Neural Turing Machines, the proposed system adopts a lightweight differentiable memory mechanism based on a single soft-attention retrieval step. Advanced memory architectures generally require iterative read/write operations, recurrent controllers, and multi-hop reasoning mechanisms, which substantially increase computational complexity and training instability in recommendation tasks.

The proposed memory design was therefore selected as a compromise between representational capacity and computational efficiency. The learnable memory bank acts as a compressed latent storage space that captures global long-term behavioral patterns while remaining fully differentiable and easy to optimize jointly with the remaining system components.

Compared to simple pooling operations over historical embeddings, the memory module provides adaptive retrieval rather than uniform aggregation. Global average pooling treats all historical interactions equally and may dilute important preference signals. In contrast, the proposed softmax-based retrieval mechanism dynamically emphasizes more relevant latent memory patterns through attention weighting and sharpness-controlled selection. This simplified formulation allows the model to preserve long-term preference

information without introducing the heavy optimization overhead commonly associated with fully dynamic memory architectures.

3.4.4 Cross-Attention Fusion Layer

At this stage of processing, three user representations are available: u_{gcn} (collaborative), seq_last (sequential), and mem_out (memory-based). A naive concatenation or weighted average loses the rich interactions between signals. The Cross-Attention Fusion (CAF) layer instead computes a query from the combined global context and attends to the full sequence output, enabling fine-grained selection of relevant history positions.

Gated Global Context

Two sigmoid gates modulate the contribution of the GCN and memory signals:

$$g_{\text{gcn}} = \text{sigmoid}(W_{\text{gcn}} \cdot u_{\text{gcn}})$$

$$g_{\text{mem}} = \text{sigmoid}(W_{\text{mem}} \cdot \text{mem_out})$$

$$\text{global_ctx} = g_{\text{gcn}} \odot u_{\text{gcn}} + g_{\text{mem}} \odot \text{mem_out}$$

The gates are input-dependent and allow the model to dynamically balance collaborative signals against memory-based signals on a per-user basis.

Cross-Attention

The global context acts as a query (after scaling by $1/\sqrt{D}$), while the full seq_out sequence is used as keys and values:

$$a = \text{CrossAttn}\left(\text{query} = \frac{\text{global_ctx}}{\sqrt{D}},\ \text{key} = \text{seq_out},\ \text{value} = \text{seq_out}\right)$$

The residually processed output is combined with the global context and passed through a second feed-forward network (FFN) with a hidden dimension of $2D$, resulting in the final fused user embedding $\in \mathbb{R}^{B \times D}$.

3.5 Design Choices

3.5.1 Why Four Components?

Each component of the proposed architecture was designed to complement the limitations of the remaining modules. LightGCN is better at detecting users with similar general tastes but overlooks the chronological order of interactions. The Transformer encoder captures sequential dynamics but can only model the last T interactions. The external memory module provides a condensed summary of interactions beyond the sequence window T. The CAF layer integrates these complementary signals into a single coherent representation.

3.5.2 Hyperparameter Selection Strategy

Several important hyperparameters of the proposed architecture were selected through a series of preliminary validation experiments conducted on the MovieLens 1M dataset. Instead of choosing fixed values arbitrarily, different configurations were evaluated in order to balance recommendation quality, convergence stability, and computational efficiency.

For the Transformer sequential encoder, multiple sequence lengths were tested, including 30, 50, 80, and 100 interactions. Shorter sequences were unable to capture sufficient sequential dependencies in long user histories, while larger sequence lengths significantly increased GPU memory consumption and training time without producing noticeable improvements in HR@10 and NDCG@10. The value `seq_len = 80` achieved the best compromise between performance and computational efficiency.

For the external memory module, several memory capacities were evaluated, including 32, 64, 128, and 256 memory slots. Smaller memory configurations reduced the ability of the model to preserve long-term behavioral information, whereas larger memory sizes increased model complexity and occasionally produced unstable convergence during training. The configuration `memory_slots = 128` consistently produced the most stable validation results.

The weighting coefficients of the hybrid loss function were also adjusted experimentally. Different combinations were evaluated during preliminary experiments, including 0.4/0.4/0.2, 0.6/0.2/0.2, and 0.5/0.3/0.2. The final weighting configuration `0.5 / 0.3 / 0.2` achieved the best balance between collaborative graph learning, sequential modeling quality, and explainability regularization.

Due to computational limitations and the relatively high training cost of the proposed hybrid architecture, a manual empirical search strategy was adopted instead of exhaustive grid search or Bayesian optimization. The final hyperparameter configuration was selected based on validation HR@10, validation NDCG@10, convergence behavior, and training stability.

Table 3.2: Main Hyperparameter Configuration

Hyperparameter	Value
Sequence Length	80
Embedding Dimension	128
Memory Slots	128
Batch Size	256
Learning Rate	0.001
Loss Weights	0.5 / 0.3 / 0.2
Dropout Rate	0.2
Temperature	0.07

3.6 Training Strategy

The principal architectural and training hyperparameters used in the proposed hybrid recommendation model are summarized in Table 3.3.

Table 3.3: Model Hyperparameters

Hyperparameter	Value
embed_dim	128
gcn_layers	3
transformer_layers	2
transformer_heads	4
transformer_ffn	256
memory_slots	128
memory_dim	64
attn_heads	4
seq_len	80
dropout	0.4
layer_drop_rate	0.1
temp	0.15

3.6.1 Dataset Construction with Hard Negative Mining

A critical factor in the quality of learning-to-rank models is the difficulty of the negative training examples. Simple negative examples provide a weak gradient signal, whereas hard negatives provide the most informative training updates. The training set consists of $n_{\text{neg}} = 16$ items, divided as follows:

- **Easy negatives (50%)**: random items chosen from the full catalogue, excluding the user's previously seen items.
- **Hard negatives (50%)**: items chosen proportionally to their popularity (`pop_prob`), excluding seen items. Popular items are strong negatives because they are frequently interacted with by other users but not by this particular user precisely the type of ambiguity the model must learn to resolve.

The popularity distribution is smoothed with a small constant (10^{-6}) before normalization. This mixed-difficulty method has consistently outperformed purely random or purely popularity-based negative sampling in preliminary experiments.

3.6.2 Composite Loss Function

The training is based on a weighted combination of three complementary loss objectives.

Sampled Softmax Loss (SSM)

SSM provides calibrated probability estimates over a set of candidates and is well suited for scenarios with a large number of negative items:

$$L_{\text{SSM}} = -\log\left(\frac{\exp(s_{\text{pos}})}{\exp(s_{\text{pos}}) + \sum_j \exp(s_{\text{neg},j})}\right)$$

Bayesian Personalised Ranking Loss (BPR)

BPR imposes a pairwise ordering constraint between the positive item and each negative item:

$$L_{\text{BPR}} = -\sum_j \log\left(\sigma(s_{\text{pos}} - s_{\text{neg},j})\right)$$

BPR is not affected by the absolute value of scores and focuses only on the ranking relationship a useful inductive bias for recommendation metrics such as NDCG and HR.

Binary Cross-Entropy Loss (BCE)

BCE is applied with label smoothing ($\varepsilon = 0.05$), treating each (user, item) pair as an independent binary classification problem:

$$L_{\text{BCE}} = \text{BCE}(s_{\text{pos}},\, 1-\varepsilon) + \text{mean}_j\left(\text{BCE}(s_{\text{neg},j},\, \varepsilon)\right)$$

Label smoothing prevents overconfident predictions and acts as a regularizer. The smoothed positive label value is 0.95 and the smoothed negative label value is 0.05.

Combined Loss

The three losses are combined with empirically tuned weights:

$$L = 0.5 \cdot L_{\text{SSM}} + 0.3 \cdot L_{\text{BPR}} + 0.2 \cdot L_{\text{BCE}} + \lambda \cdot L_{\text{reg}}$$

L_{reg} represents the L2 regularization penalty applied exclusively to the embedding and bias parameters ($\lambda = 1 \times 10^{-5}$). The SSM component carries the greatest weight because it directly improves ranking performance on the candidate set used during evaluation.

Table 3.4: Hybrid Loss Components and Their Roles

Loss	**Weight**	**Primary Role**
Sampled Softmax (SSM)	0.50	Calibrated ranking over candidate set
Bayesian Personalized Ranking (BPR)	0.30	Pairwise ordering consistency
Binary Cross-Entropy (BCE)	0.20	Per-pair calibration with label smoothing
L2 Regularization	$\lambda = 1 \times 10^{-5}$	Embedding magnitude control

3.6.3 Optimiser and Learning-Rate Schedule

The model parameters are divided into three groups, each with an independent learning rate to reflect the different convergence dynamics of its components:

- GCN parameters (embedding tables, graph convolution): learning rate $= 5 \times 10^{-4}$
- Transformer parameters (positional encoding, attention, FFN): learning rate $= 5 \times 10^{-4}$
- Remaining parameters (memory, fusion, biases): learning rate $= 3 \times 10^{-4}$

All parameter groups share a weight decay of $\lambda_{\text{wd}} = 1 \times 10^{-3}$ applied via the AdamW optimizer. The decoupled weight decay in AdamW correctly regularizes the parameters regardless of the adaptive gradient scale, unlike the standard L2-regularized Adam optimizer.

3.6.4 Training Configuration Summary

Table 3.5: Training Strategy Parameters

Parameter	**Value**
Epochs (total)	50
Warmup epochs	5
Flat epochs	25
Batch size (micro)	256
Gradient accumulation steps	4
Effective batch size	1,024
lr_gcn	5×10^{-4}
lr_transformer	5×10^{-4}
lr_rest	3×10^{-4}
Weight decay	5×10^{-4}
Negative samples / step	32
Hard negative ratio	0.30
Early stopping patience	10 epochs
Evaluation frequency	Every epoch
Top-K for evaluation	10
Eval negative samples	99

3.7 Evaluation Methodology

Model performance is measured using two standard top-K ranking metrics, both computed with $K = 10$.

Hit Rate@K (HR@K)

The proportion of users for whom at least one relevant item appears in the top-K ranked list:

$$\text{HR@}K = \frac{\text{Number of users with at least one hit in top-}K}{\text{Total number of users}} \tag{3.1}$$

Normalized Discounted Cumulative Gain (NDCG@K)

NDCG reflects ranking quality by comparing the predicted ranking to an ideal ordering in which all relevant items are placed at the top. Unlike some ranking metrics, NDCG can accommodate both binary and graded relevance scores [71]:

$$\text{NDCG@}K = \frac{\text{DCG@}K}{\text{IDCG@}K} \tag{3.2}$$

$$\text{DCG@}K = \sum_{i=1}^{K} \frac{2^{rel_i} - 1}{\log_2(i+1)} \tag{3.3}$$

3.8 Explainability Framework

The DynamicHybridRec system combines four heterogeneous components LightGCN, the Transformer sequential encoder, an external memory module, and a cross-attention fusion layer into a single recommendation score. This architectural complexity introduces an inherent ambiguity: the final score for any (user, item) pair is the product of nonlinear interactions between the four components, making it unclear which part of the model is responsible for the recommendation.

In recommender systems, explainability serves several purposes:

- **Model debugging**: identifying when a component is contributing noise rather than a useful signal.
- **User trust**: surfacing the reasoning behind a recommendation in a human-interpretable form.
- **Bias auditing**: revealing whether the model systematically relies on popularity indicators or temporal factors.

One approach to explainability in attention-based models is to report raw attention weights as importance scores. However, attention weights are not explanations in the causal sense: an item that receives high attention may have little effect on the final score, and an item that receives low attention may still exert a meaningful influence. For these reasons, DynamicHybridRec implements a three-level explainability framework that cross-validates multiple attribution signals and provides both token-level and component-level explanations.

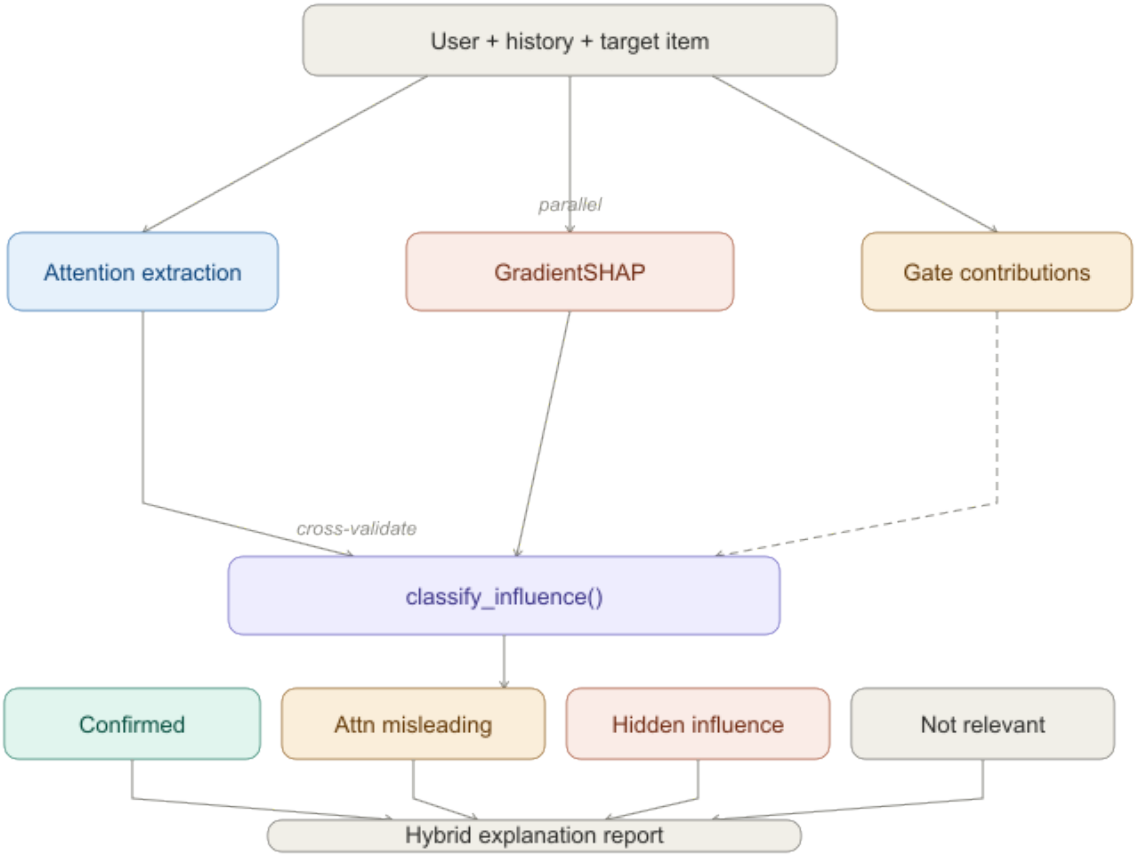


Figure 3.2: Hybrid explainability pipeline attention, SHAP, and gate contributions

3.8.1 Explainability Methods

Attention Weight Extraction

The self-attention mechanism within the Transformer encoder assigns a weight to each position in the user interaction log when producing the final sequential representation. A high attention weight at position i indicates that the model focused on item i when determining the next recommendation.

In each `TransformerBlock`, the multi-head attention layer typically discards its weight matrix after the forward pass. The encoder is therefore re-run with `need_weights=True` and `average_attn_weights=True`, returning an array of shape (seq_len, seq_len) for each layer:

```
a, w = block.attn(
    x_n, x_n, x_n,
    attn_mask=model.seq_enc.causal_mask,
    key_padding_mask=pad_mask,
    need_weights=True,
    average_attn_weights=True
)
```

```
8
9  layer_attns.append(
10     w.squeeze(0).cpu().numpy()
11 )
```

Attention matrices are collected from each Transformer layer. The last row of each matrix which corresponds to the final sequence position attending over its entire history is extracted and averaged across layers to produce a single mean attention vector.

GradientSHAP Attribution

A wrapper module, `SequenceScoreWrapper`, was created to expose the Transformer-to-score computation as a differentiable function of the raw sequence embeddings:

```
1  class SequenceScoreWrapper(torch.nn.Module):
2      def forward(self, seq_embeds):
3          seq_out, seq_last = self.model.seq_enc(seq_embeds, pad_mask)
4          mem_out  = self.model.memory(seq_last + 0.5 * u_gcn)
5          fused    = self.model.fusion(u_gcn, seq_out, mem_out,
               pad_mask)
6          fused    = self.model.output_norm(fused)
7          score = (
8              F.normalize(fused, dim=-1) * F.normalize(target_emb, dim
                   =-1)
9          ).sum(-1) / self.model.temp
10         return score
```

Captum's `GradientShap` is then applied against a zero-vector baseline, averaging over 25 stochastic perturbation samples with a standard deviation of 0.01:

```
1  gs    = GradientShap(wrapper)
2  attrs = gs.attribute(seq_in, baselines=baseline, n_samples=25, stdevs
       =0.01)
```

The attribution tensor `attrs` has shape (seq_len, embed_dim). The embedding dimension is collapsed by computing the L2 norm at each position, yielding a scalar importance score per history item:

```
1  shap_scores = attrs.squeeze(0).norm(dim=-1).detach().cpu().numpy()
```

The resulting vector is normalized to $[0, 1]$ over the non-padded history positions, producing a scale-matched counterpart to the attention vector.

Component Gate Contribution Analysis

DynamicHybridRec derives its user representation from three architectural sources. The `CrossAttentionFusion` layer combines these sources using learned sigmoid gates. Quantifying the relative contribution of each component for a given user provides architectural-level interpretability that neither attention nor gradient methods capture.

A naive reading of the sigmoid gate magnitudes is insufficient, because each gate operates independently and its values can simultaneously approach 1.0 across all components, rendering their sum meaningless as a proportion. Instead, contribution is measured via cosine similarity between each gated component vector and the final fused user representation:

```
g_gcn      = torch.sigmoid(model.fusion.gate_gcn(u_gcn))
g_mem      = torch.sigmoid(model.fusion.gate_mem(mem_out))
gcn_part   = g_gcn * u_gcn
mem_part   = g_mem * mem_out
global_ctx = gcn_part + mem_part

# Cross-attention produces the sequence component
seq_part, _ = model.fusion.attn(query, seq_out, seq_out, ...)

# Rebuild the full fused output
fused_pre  = model.fusion.norm1(seq_part + global_ctx)
fused_post = model.fusion.norm2(fused_pre + model.fusion.ffn(
    fused_pre))

# Cosine similarity as a scale-invariant contribution measure
fused_n = F.normalize(fused_post, dim=-1)
e_gcn   = F.cosine_similarity(fused_n, F.normalize(gcn_part,  dim=-1)
    ).abs()
e_mem   = F.cosine_similarity(fused_n, F.normalize(mem_part,  dim=-1)
    ).abs()
e_seq   = F.cosine_similarity(fused_n, F.normalize(seq_part,  dim=-1)
    ).abs()

total   = e_gcn + e_mem + e_seq
contributions = {
    'GCN (collaborative)' : e_gcn / total,
    'Memory (long-term)'  : e_mem / total,
    'Sequence (recent)'   : e_seq / total,
}
```

3.8.2 Hybrid Cross-Validation of Attribution Signals

Instead of reporting each attribution signal separately, they are combined to produce a four-category classification for each item in the interaction history:

Table 3.6: Interpretation Categories Based on Attention and SHAP Scores

Category	**Attention**	**SHAP**	**Interpretation**
Confirmed influential	≥ 0.1	≥ 0.1	The model attended to the item and it causally influenced the score, representing the most reliable explanation
Attention misleading	≥ 0.1	< 0.1	High attention with low causal impact, indicating a spurious attention pattern
Hidden influence	< 0.1	≥ 0.1	Low attention weight despite significant causal impact; the item influenced the output through a non-attentive pathway
Not relevant	< 0.1	< 0.1	The item played no meaningful role in the recommendation

This classification is applied in the `classify_influence()` function, element by element, to the calibrated attention and SHAP vectors. The "confirmed influential" and "hidden influence" categories are of particular diagnostic value: the former provides highly reliable interpretations, while the latter reveals items that bypass the attention mechanism yet still affect the output a phenomenon that purely attention-based interpretation methods would miss.

3.8.3 Visualization

The annotation output is presented as a four-part figure saved to disk. Each part serves a distinct analytical purpose:

- **Attention & SHAP bar chart**

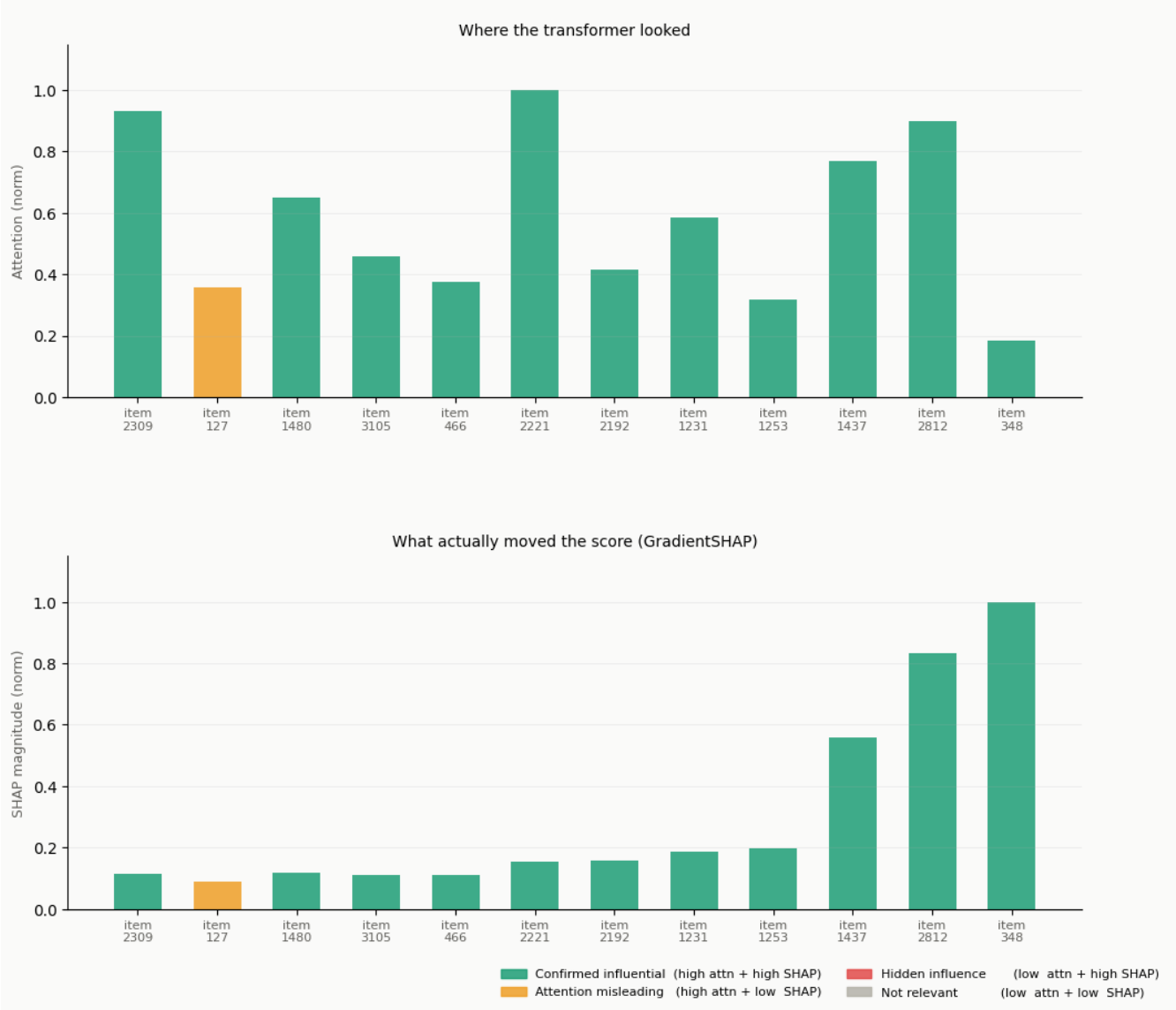


Figure 3.3: Attention and SHAP bar chart

- **Attention bar chart**: Normalized attention scores per history item, color-coded by the four-category label (green = confirmed influential, orange = attention misleading, red = hidden influence, grey = not relevant).
- **SHAP bar chart**: Normalized GradientSHAP magnitude per history item, using the same color scheme to enable direct visual comparison with the attention panel.

- **Attention–SHAP scatter plot and component contribution pie chart**

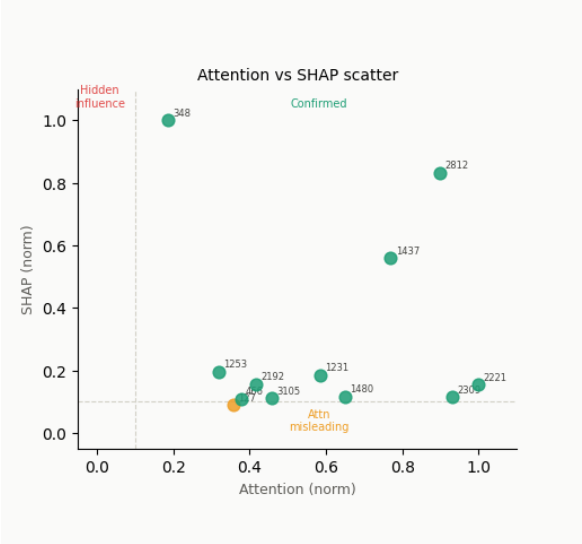


Figure 3.4: Attention–SHAP scatter plot

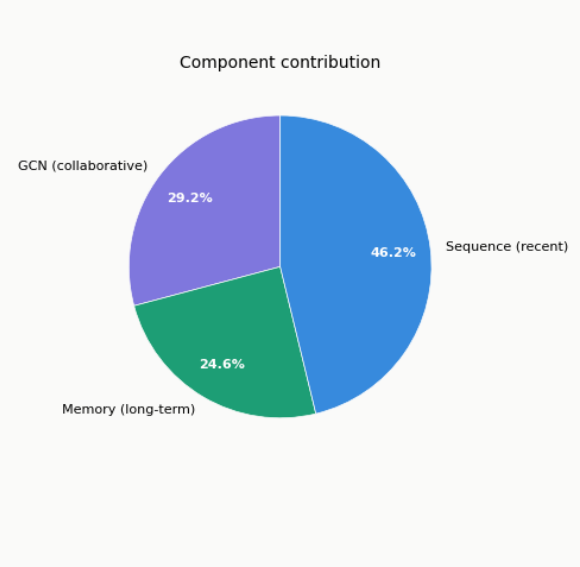


Figure 3.5: Component contribution pie chart

- **Attention–SHAP scatter plot**: Each history item is plotted at coordinates (attention, SHAP). The four quadrants correspond to the four label categories. Items in the top-right quadrant represent the most reliable explanation instances.
- **Component contribution pie chart**: The GCN / Memory / Sequence percentage breakdown for this specific user, providing architectural-level interpretability alongside the item-level signals.

3.8.4 Backend Serving with FastAPI

The trained *DynamicHybridRec* model is deployed as a RESTful API using *FastAPI*. A global state object stores all runtime components required for inference, including the trained model parameters, graph structure, user and item mappings, and interaction histories. During server startup, the graph adjacency matrix is reconstructed from the training interactions, converted into a sparse representation, and loaded onto the available computation device. This graph remains fixed throughout the server's execution.

The API exposes several endpoints, including `/recommend` for generating top-K recommendations, `/explain` for attention-based recommendation explanations, and `/movies` and `/metrics` for accessing dataset information and evaluation results. For registered users, recommendations are generated using the complete model inference pipeline through the `predict_all` method. For new or anonymous users who only provide an interaction history, recommendations are produced using a hybrid strategy that combines three complementary signals: a Transformer-based sequential preference signal, an item-neighborhood similarity signal, and a recency-aware neighborhood signal. These signals

are normalized and averaged to produce the final ranking, compensating for the absence of graph-based user embeddings. To improve efficiency and reduce memory consumption, all inference operations are executed in evaluation mode without gradient computation.

3.9 Conclusion

This chapter has presented the complete design, implementation, and training strategy of DynamicHybridRec, a hybrid recommendation architecture developed and evaluated on the MovieLens 1M benchmark. The system is built on a core architectural principle: that no single model is sufficient to fully capture the complexity of user preferences. By decomposing user representation into four integrated signals structural collaborative patterns via LightGCN, short-term sequential dynamics via the Transformer encoder, long-term preferences via the external memory module, and a unified representation via the cross-attention fusion layer the proposed system achieves richer user modeling than any single-component approach.

The hybrid explainability framework, which combines attention weight extraction, GradientSHAP attribution, and gate contribution analysis, provides multi-layered interpretations at the component, item, and output levels revealing not only which historical interactions influenced the recommendation, but also which architectural branch was responsible and whether the attention weights accurately reflect causal significance. The design decisions documented in this chapter collectively reflect a commitment to building a recommender system that is simultaneously accurate, interpretable, and computationally efficient. The following chapter presents the empirical results of these choices through systematic experiments and ablation studies on the MovieLens 1M benchmark.

Chapter 4

Exprementation and Evaluation

Chapter 4

Expermentation and Evaluation

4.1 Introduction

After presenting the theoretical foundations and architecture of the proposed hybrid recommendation model in previous chapters, the objective of this chapter is to evaluate the effectiveness of the proposed system in real-world recommendation scenarios using the MovieLens 1M dataset. It focuses on the experimental aspects of the hybrid recommendation model architecture proposed in previous chapters.

This chapter presents the development environment, software libraries, and hardware resources used during implementation and training. It also explains the training process, the optimization strategy, and the evaluation criteria used to measure recommendation quality.

Furthermore, we analyze the training dynamics using loss convergence and validation criteria such as HR@10 and NDCG@10 to study the model's learning behavior and its ability to generalize.

Finally, the performance of the proposed hybrid architecture is evaluated on the test set and compared to basic recommendation methods. An analytical study is also conducted to examine the contribution of each component of the architecture, including the LightGCN module, the Transformer serial encoder, and the interpretation mechanism.

4.2 Development Environment and Tools

For develop our hybrid recommendation system, various tools and environments were used to support model design, training, and evaluation.

4.2.1 Programming Language

The entire system data preprocessing, model architecture, training loop, evaluation pipeline, explainability framework, and REST API is implemented in Python 3.10. Python was

selected as the implementation language for several converging reasons. Because it is the most widely used language for deep learning research, and it contains a vast number of frameworks and libraries, especially for data processing and model interpretation.

As for development productivity, its excellent support in Jupyter environments such as Kaggle Notebooks made it perfectly suited to the iterative, experiment-based development process used in this project.

4.2.2 Libraries and Frameworks

These are the used libraries and their roles

- PyTorch: used for model definition, autograd, optimisation.
- torch-geometric: Sparse graph convolution primitives used in the LightGCN component
- Captum: used in GradientSHAP attribution for the explainability framework
- FastAPI: REST API server exposing recommendation and explanation endpoints
- Uvicorn: ASGI server running the FastAPI application
- NumPy: Array operations, negative sampling, bias table construction
- Pandas: Dataset loading, chronological sorting, train/val/test splitting
- SciPy: Sparse adjacency matrix construction for LightGCN graph normalisation
- cloudflared: Cloudflare tunnel exposing the local FastAPI server via a public URL

4.2.3 Development Platform

Development and experimentation were conducted entirely within Kaggle Notebooks, a cloud-based Jupyter environment. Kaggle was chosen for three practical reasons: it offers free access to T4 GPU sessions for up to 30 hours per week, provides a pre-configured Python environment with PyTorch and most scientific libraries pre-installed, and allows datasets to be stored permanently, eliminating the need to re-download MovieLens 1M between sessions.

4.2.4 Hardware

All training and experimentation were happened on a single NVIDIA Tesla T4 GPU (16 GB VRAM) provided through the Kaggle Notebooks platform.The NVIDIA T4 GPU accelerates diverse cloud workloads, including high-performance computing, deep learning

training and inference, machine learning, data analytics, and graphics,and based on the Turing architecture, offering 8.1 TFLOPS of FP32 performance and 65 TOPS of INT8 throughput.[72]The full training still 36 epochs required approximately 4 hours on this hardware.

4.3 Training Dynamics

4.3.1 Loss Convergence

The training continued for 36 of the planned 50 training sessions before being stopped early. the training loss rate dropped rapidly from 1.994 in the first session to approximately 1.037 in the tenth session.After this, the flat session maintained a stable learning rate, and the loss rate continued to decrease slowly but steadily. Validation loss tracks training loss closely throughout, with no sign of overfitting. at epoch 26 (the best checkpoint), training loss is 1.031 and validation loss is 0.954.

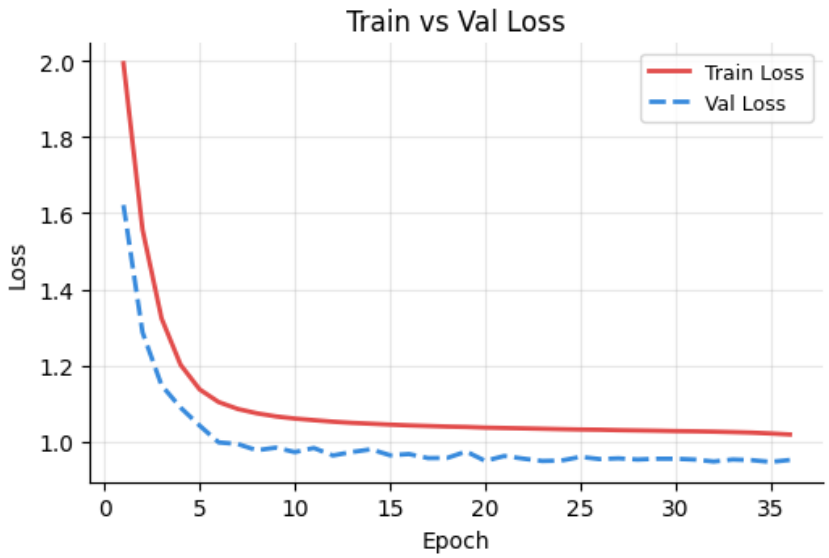


Figure 4.1: Train vs Val Loss Convergence

4.3.2 Validation Metric Progression

Validation HR@10 improves from 0.537 at epoch 1 to a peak of 0.847at epoch 26, As for NDCG, it starts from 0.299 to 0.6357 in the best epoch,The majority of gains are achieved in the first 10 epochs (HR@10 = 0.841),with subsequent training courses provide gradual but continuous improvements. This strong start followed by a slow improvement phase, is a characteristic of structures that benefit from a warm-up phase. The small

gap between training and validation curves suggests good generalization capability and limited overfitting.

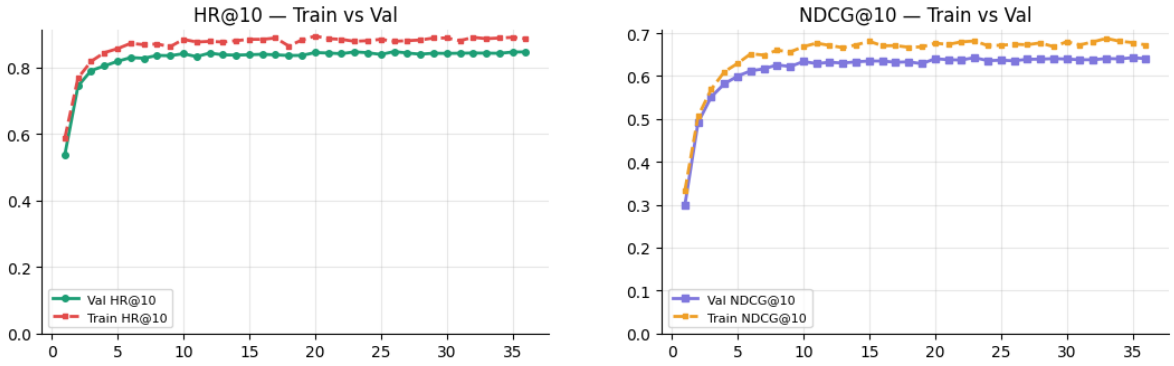


Figure 4.2: Train vs Val For HR@10 & NDCG@10

The two graphs show that the proposed hybrid recommendation model learns efficiently, converges quickly and maintains strong performance on invisible validation data. The close relationship between training and validation metrics indicates that the model achieves good classification quality while avoiding overfitting.

Table 4.1: Training and Validation Performance Across Epochs

Epoch	Train Loss	Val Loss	Val HR@10	Val NDCG@10
1	1.9936	1.6213	0.5374	0.2992
5	1.1370	1.0419	0.8180	0.5999
10	1.0604	0.9724	0.8409	0.6339
20	1.0366	0.9493	0.8443	0.6406
26 (best)	1.0311	0.9541	0.8471	0.6357
36 (stop)	1.0187	0.9517	0.8463	0.6408

4.4 Main Results

This section presents the final performance results obtained by the proposed hybrid recommendation system after the completion of the training phase. The evaluation was conducted on the test dataset using ranking-based recommendation metrics in order to measure the quality of the generated recommendations and the ability of the model to correctly predict future user interactions.

The obtained results demonstrate the effectiveness of combining collaborative graph learning, sequential modeling, and attention-based representation within a unified recommendation architecture.

4.4.1 Test Set Performance

After selecting the best validation point according to the validation criteria, we evaluate the final model on an invisible test set. The evaluation process was conducted using the HR@10, NDCG@10, and MRR@10 scales.

The results show that the model performed strongly in recommendations, demonstrating its ability to successfully identify relevant items and rank them higher within the recommendation list.

Table 4.2: Final Test Set Performance at Different Cutoff Values

Metric	**K=1**	**K=5**	**K=10**	**K=20**
HR@K	0.4046	0.7137	0.8114	0.8838
NDCG@K	0.4046	0.5715	0.6045	0.6211
MRR@K	0.4046	0.5239	0.5386	0.5418
Best Epoch: 26				

The HR@10 result shows that the model was capable of placing the correct item within the top-10 recommendations for a large proportion of users. In addition, the NDCG@10 score indicates that relevant items were generally ranked near the top positions, which improves recommendation usefulness from the user perspective.

The MRR@10 metric also confirms that the proposed architecture is capable of identifying relevant recommendations early in the ranked list rather than placing them in lower positions.

The strong performance obtained on the unseen test data suggests that the model generalizes effectively and does not simply memorize the training interactions.

4.4.2 Comparison with Baselines

To further evaluate the effectiveness of the proposed architecture, its performance was compared against several baseline recommendation models commonly used in recommendation system research.PopRec[73],BPR-MF[74],NCF[36],GRU4Rec[60],SASRec[61] and BERT4Rec[37]. And to make a fair comparison, we used the same parameters. The selected baselines include traditional collaborative filtering methods as well as deep learning and sequential recommendation architectures.

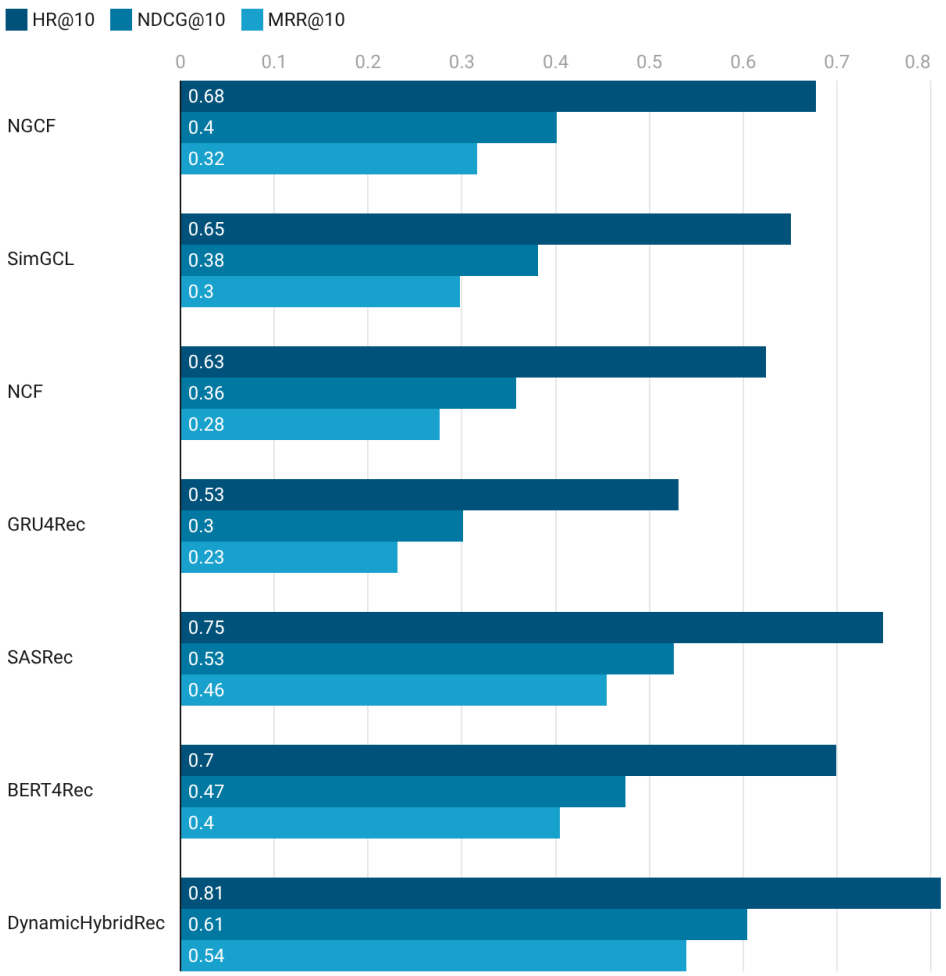


Figure 4.3: Comparison with Baseline Methods on MovieLens 1M

The results presented in Figure 4.3 demonstrate the effectiveness of the proposed DynamicHybridRec architecture compared with a diverse set of baseline recommendation models. A clear performance progression can be observed as the complexity of user preference modeling increases, moving from graph-based collaborative filtering approaches to deep learning and transformer-based sequential recommendation methods.

Graph neural network-based recommendation models, represented by NGCF and SimGCL, achieved competitive performance by leveraging user-item interaction graphs to learn collaborative representations. NGCF obtained an HR@10 of 0.677, an NDCG@10 of 0.401, and an MRR@10 of 0.316, demonstrating the effectiveness of graph convolutional propagation in capturing high-order connectivity information. SimGCL further enhanced representation learning through self-supervised contrastive learning, achieving HR@10 = 0.651, NDCG@10 = 0.381, and MRR@10 = 0.298. Although these methods effectively model collaborative relationships, they do not explicitly capture the temporal dynamics and sequential evolution of user preferences.

Neural recommendation approaches showed a noticeable improvement. NCF achieved an HR@10 of 0.63 and an NDCG@10 of 0.36, highlighting the benefits of nonlinear rep-

resentation learning compared to conventional matrix factorization techniques. Similarly, GRU4Rec leveraged sequential information through recurrent neural networks and achieved better ranking quality than traditional methods. These results confirm the importance of incorporating behavioral dynamics into recommendation models.

Transformer-based architectures further improved recommendation performance. BERT4Rec and SASRec significantly outperformed earlier methods due to their ability to model long-range dependencies and capture complex relationships within user interaction sequences through self-attention mechanisms. Among all baseline models, SASRec achieved the strongest performance, obtaining HR@10 = 0.75, NDCG@10 = 0.53, and MRR@10 = 0.46. This result demonstrates the effectiveness of attention-based sequential modeling for next-item recommendation tasks.

Despite the strong performance of SASRec, the proposed DynamicHybridRec model consistently achieved the best results across all evaluation metrics, reaching HR@10 = 0.81, NDCG@10 = 0.61, and MRR@10 = 0.54. The consistent improvement across all metrics indicates not only a higher probability of retrieving relevant items but also a better ranking quality within the top recommended positions.

The substantial improvement over both graph-based collaborative filtering and state-of-the-art sequential recommendation models suggests that user preferences cannot be fully represented by a single modeling paradigm. Instead, the integration of graph-based collaborative learning, sequential behavior modeling, and long-term preference memory provides a richer and more comprehensive understanding of user interests. These results demonstrate that the proposed hybrid architecture effectively captures multiple dimensions of user behavior, leading to more accurate, personalized, and robust recommendations.

4.4.3 Ablation Study

To determine the contribution of each element to the model, we conducted an elimination study by sequentially removing elements, starting with the Transformer only and adding an element after each evaluation until the full model.Table 4.3 reports the results.

Table 4.3: Ablation Study of the Proposed Recommendation Model

Configuration	**HR@10**	**NDCG@10**	**MRR@10**
GCN only	0.416	0.219	0.160
Transformer only	0.653	0.404	0.328
LightGCN + Transformer	0.703	0.477	0.406
LightGCN + Transformer + Memory (No CAF)	0.730	0.507	0.440
Full Model	**0.811**	**0.605**	**0.539**

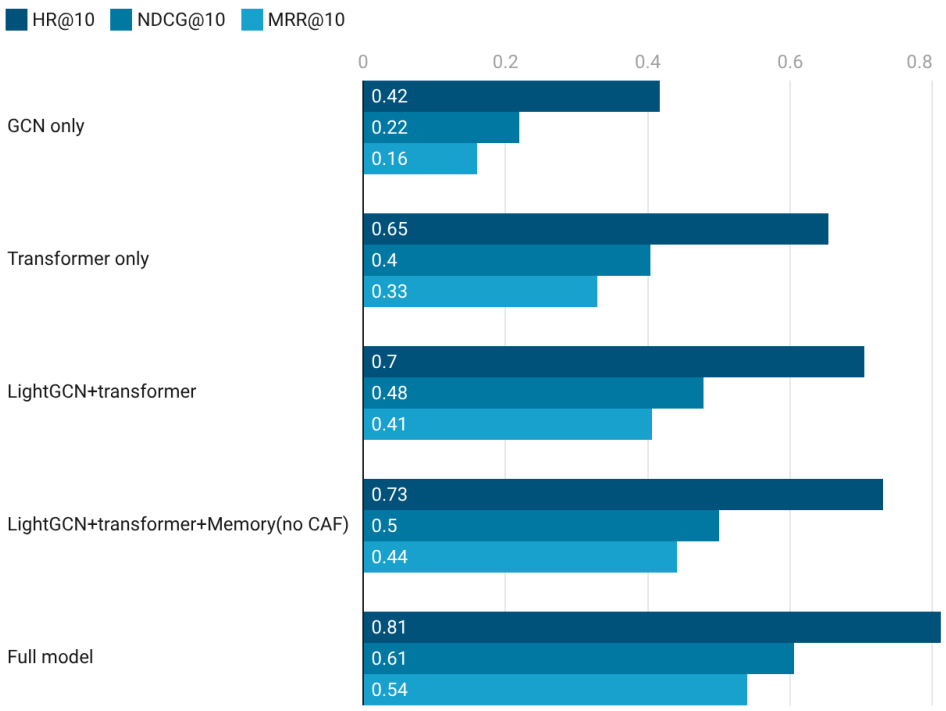


Figure 4.4: Component Ablation Study

The analysis reveals the contribution of each component to the proposed recommendations framework. The GCN configuration alone performed the lowest (HR@10 = 0.42, NDCG@10 = 0.22, MRR@10 = 0.16), indicating that modeling graph relationships between the user and the element alone is insufficient for effectively understanding sequential user preferences.

Replacing GCN with the Transformer architecture alone significantly improved performance (HR@10 = 0.65, NDCG@10 = 0.40, MRR@10 = 0.33), highlighting the importance of sequential behavior modeling. Combining LightGCN with Transformer resulted in even greater performance gains (HR@10 = 0.70, NDCG@10 = 0.48, MRR@10 = 0.41), demonstrating that collaborative graph-based information and sequential patterns offer complementary benefits.

Adding the memory module resulted in further improvements, with the metrics rising to HR@10 = 0.73, NDCG@10 = 0.50, and MRR@10 = 0.44. This indicates that the memory mechanism effectively accommodates long-term user interests not fully represented by the graph and transformer components. Finally, the complete model, incorporating the Cross Attention Fusion (CAF) mechanism, achieved the highest performance across all evaluation metrics (HR@10 = 0.81, NDCG@10 = 0.61, and MRR@10 = 0.54). These significant gains compared to previous configurations confirm that CAF successfully integrates information from the graph, sequence, and memory representations, enabling more accurate and personalized recommendations. Overall, the analysis confirms that each component contributes positively to the model, with peak performance achieved when all

modules are run together.

4.4.4 Generalization to Sparse Datasets

To evaluate the ability of the proposed architecture to generalize beyond MovieLens 1M, an additional experiment was conducted using the Yelp Reviews Dataset. Unlike MovieLens, which contains relatively dense user interaction histories, Yelp exhibits a significantly higher level of sparsity, with many users interacting with only a limited number of businesses.

This dataset represents a more challenging recommendation scenario, allowing us to investigate whether the proposed DynamicHybridRec architecture remains effective when long interaction sequences are not available.

User Interaction Distribution

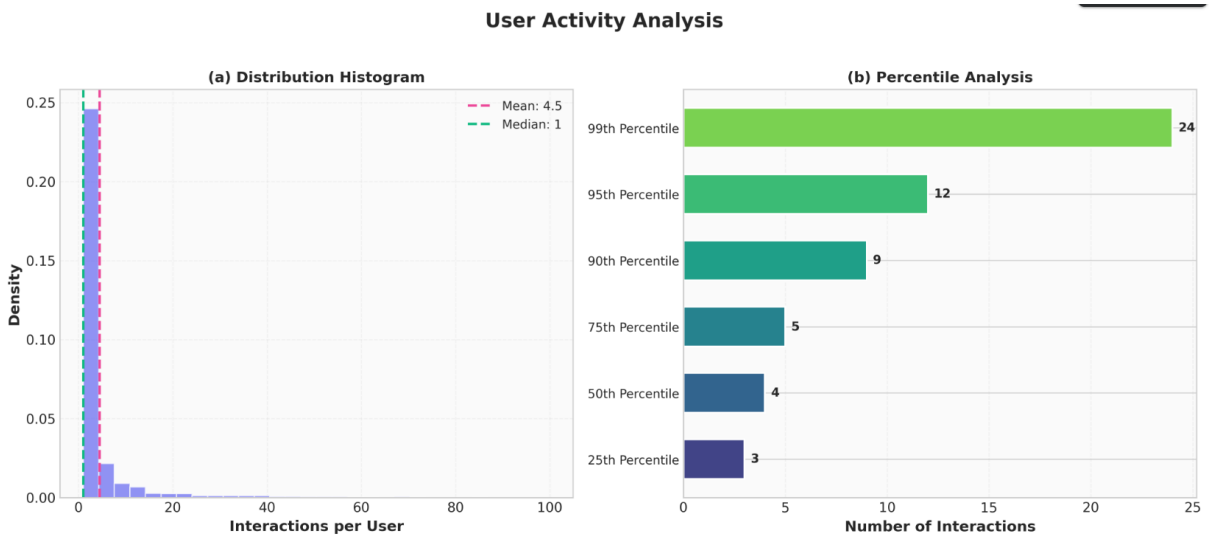


Figure 4.5: User interaction distribution in the Yelp Reviews dataset.

Figure 4.5 shows the highly skewed distribution of user interactions in the Yelp dataset. The median number of interactions is only four, while more than 90% of users have fewer than ten interactions. This confirms the extreme sparsity of the dataset and demonstrates that most users possess only short interaction histories. Consequently, Yelp provides a challenging benchmark for evaluating the generalization ability of recommendation models beyond dense datasets such as MovieLens 1M.

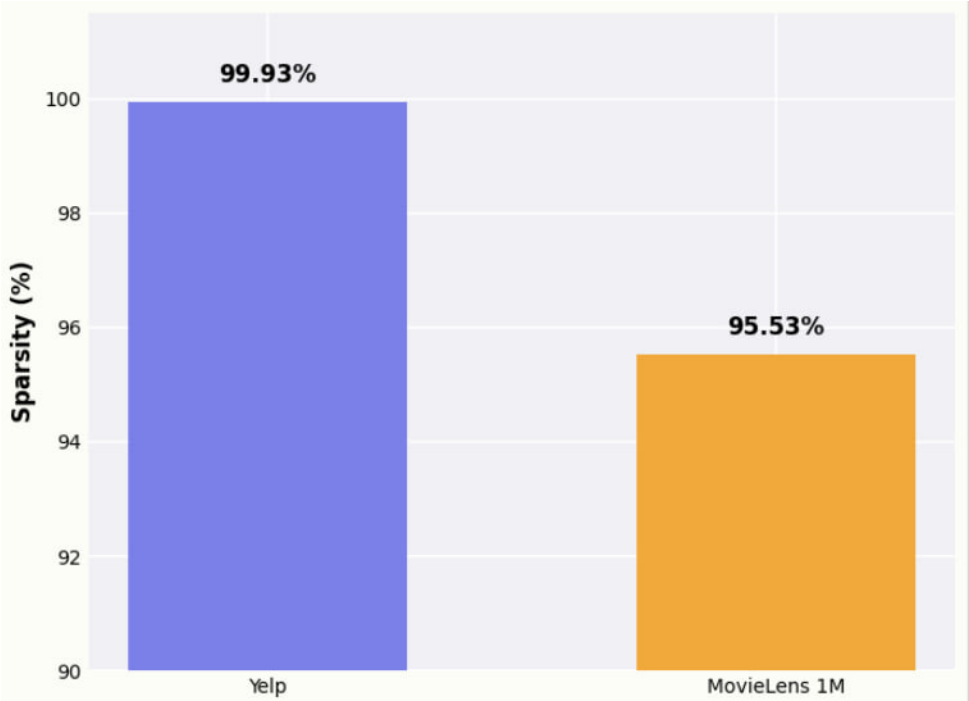


Figure 4.6: Comparison of Dataset Sparsity: Yelp (Preprocessed) vs. MovieLens 1M.

Performance on Yelp Dataset

Despite the increased sparsity of the Yelp dataset, DynamicHybridRec maintained strong recommendation quality, achieving an HR@10 of 0.784 and an NDCG@10 of 0.533.

Table 4.4: Performance comparison across datasets

Dataset	**HR@10**	**NDCG@10**
MovieLens 1M	0.811	0.604
Yelp Reviews	0.784	0.533

Discussion and Architectural Analysis

The experimental results provide useful insights into the role of the different architectural components under sparse conditions.

When user histories are short, the contribution of the Transformer encoder naturally decreases because fewer sequential patterns can be learned. Similarly, the External Memory module becomes less critical, since only a limited amount of long-term information exists beyond the sequence window.

However, the LightGCN component remains highly relevant. By propagating information through the user–item interaction graph, LightGCN can capture collaborative signals even when individual users have only a small number of interactions.

Consequently, while the relative importance of the sequential and memory components decreases in sparse environments, the graph-based collaborative representation continues

to provide valuable information. The obtained results suggest that DynamicHybridRec is capable of generalizing beyond dense recommendation datasets and remains applicable to more challenging sparse recommendation scenarios.

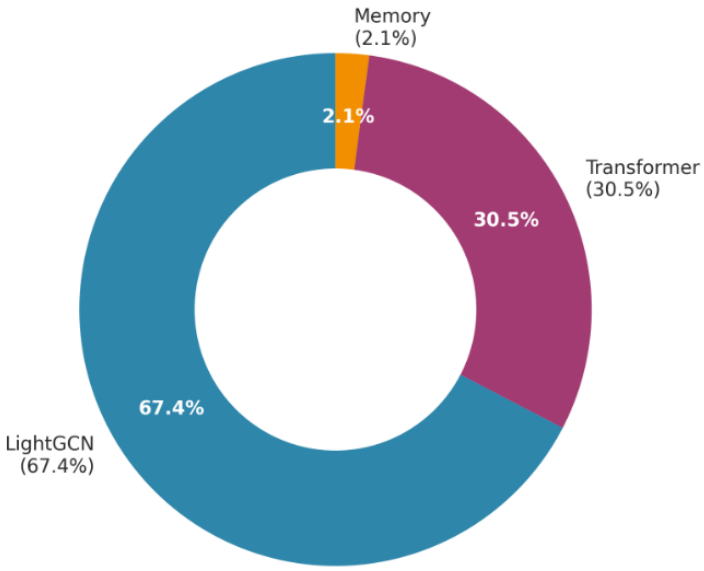


Figure 4.7: Component-wise contribution analysis.

4.5 Explainability Evaluation

4.5.1 Faithfulness

Faithfulness answers the question of whether the explanations accurately reflect the model's actual decision-making process. We assessed faithfulness using the Area Over the Perturbation Curve (AOPC)[75] protocol. The key idea behind AOPC is:the explanation is faithful if:

- Deletion test: the prediction score should be drop rapidly when we remove the most important items
- Insertion test: the prediction score should be rise rapidly when we add the most important items

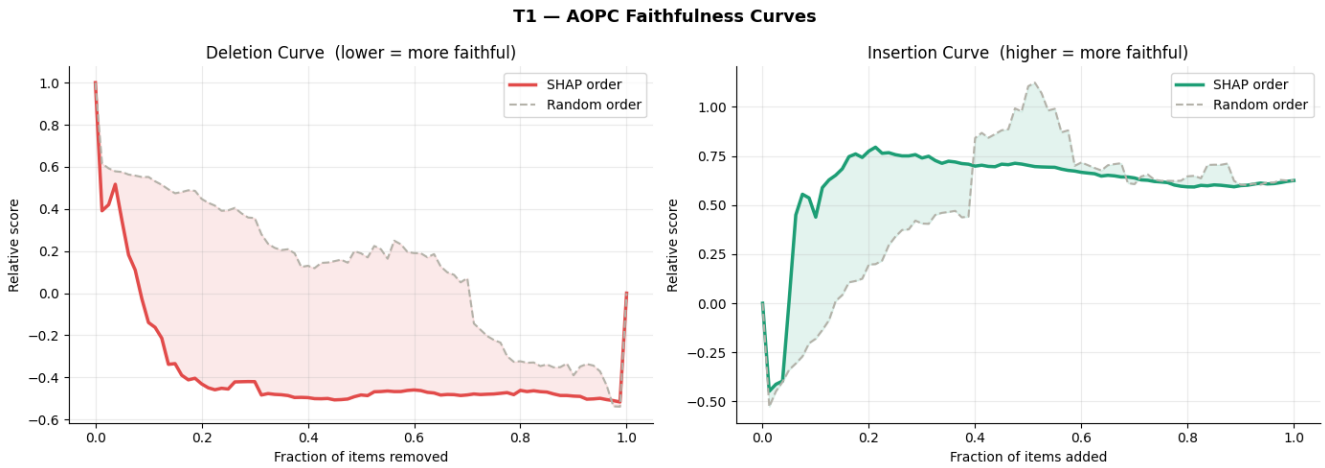


Figure 4.8: AOPC Faithfulness Curves

Deletion Curve

The SHAP order (red) drop rapidly from 1 to around -0.4 after we remove 20% of items,Unlike the random order (gray) decrease more slowly. This indicates that deleting elements identified as important by SHAP causes the model to collapse, proving that it has a real impact.

Insertion Curve

The SHAP order (green) rises rapidly. when we add only 15% or 20% of items We observe that the result rises significantly to 0.70-0.80,Unlike the random order (gray) increase more slowly.At about 40-60%, the random curve becomes unstable,even pass the SHAP curve.But that does not matter,The most important features identified by SHAP are enough to reconstruct most of the recommendation score very early.

also we provided a complementary counterfactual test at the ranking level. Removing the top-SHAP items from a user's history caused Hit Rate at 10 (HR@10) to drop by 55.2 percentage points on average, whereas removing the bottom-SHAP items caused no measurable degradation (0.0 pp). This near-perfect asymmetry constitutes strong evidence that the explanations identify the items that are causally responsible for the recommendation,not merely correlated with it.

4.5.2 Explanation Robustness

GradientSHAP is not a deterministic algorithm. Every time we run it, it adds a small amount of random noise to the input, which means that running it twice on the same user gives slightly different attribution scores each time. So we should ask a question ' are those differences big enough to change the explanation?' for know that we run SHAP

twice and get importance ranking for user's history items for each time. And compute Spearman ρ between the two rankings.

Spearman Rank Correlation

It is a non-parametric statistical measure that quantifies the degree of agreement between two ranked orderings of the same items, without requiring the underlying scores to share a common scale.the coefficient is computed as:

$$\rho = 1 - \frac{6 \sum_{i=1}^{n} d_i^2}{n(n^2 - 1)} \tag{4.1}$$

where ρ is the Spearman rank correlation coefficient, n is the number of ranked items, and d_i is the difference between the ranks of the i-th item in the two rankings being compared.

- If $\rho \approx 1.0$, the rankings are nearly identical despite the randomness, indicating that the explanation is stable.
- If $\rho \approx 0.0$, the rankings differ substantially across runs, indicating that the explanation is largely noise.

The mean Spearman rank correlation between the two runs was 0.980. This indicates that the importance ratings are highly stable and not dominated by randomness of the gradient sampling procedure.

4.5.3 Population-level Coverage

This part asks the most basic question about personalisation at the population level: "Does the model explain different things to different users, or does it keep citing the same items for everyone?" We quantified explanation diversity by computing the average pairwise overlap between the top-SHAP item sets of different users. How it works: For each user we take their top-3 SHAP items (the items SHAP says are most responsible for their recommendation),we compare every pair of users (how many of their top-3 SHAP items are the same?) and calculate the average that overlap across all possible user pairs. The pairwise overlap that we got was 0.2%,that is mean Almost no two users share the same important items in their explanations.This means the model is explaining recommendations based on each user's unique personal history not falling back on a small set of globally popular items as a one size fits all explanation.

4.6 Strengths and Limitations

4.6.1 Strengths

The proposed hybrid recommendation architecture demonstrated several important strengths during the experimentation phase. One of the main advantages of the model is its ability to combine collaborative graph learning and sequential behavioral modeling within a unified framework. The integration of the LightGCN component allowed the system to effectively capture collaborative relationships between users and items through graph propagation mechanisms, while the Transformer-based sequential encoder improved the understanding of temporal interaction patterns and recent user interests.

Another important strength of the proposed architecture is its strong recommendation performance on the MovieLens 1M dataset. The obtained HR@10, NDCG@10, and MRR@10 results indicate that the model can successfully rank relevant items near the top of the recommendation list while maintaining stable generalization capability on unseen data.

The proposed model also benefits from the use of attention mechanisms and explainability techniques such as GradientSHAP, which improve the interpretability of recommendations and provide additional insights into the factors influencing prediction decisions. This increases the transparency of the recommendation process and makes the system easier to analyze and evaluate.

In addition, the hybrid structure helps alleviate several classical recommendation problems such as sparse interactions and the inability of traditional collaborative filtering methods to model sequential dependencies between interactions.

4.6.2 Limitations

Despite these strengths, the proposed architecture still presents several limitations. First, the hybrid model requires relatively high computational resources because it combines multiple deep learning components including graph convolutions, Transformer blocks, and explainability mechanisms. As a result, the training process requires GPU acceleration and longer execution time compared to simpler recommendation models.

Another limitation is the dependency on historical interaction data. Although the sequential encoder improves temporal modeling, the system may still struggle in severe cold-start situations where very limited information is available for new users or new items.

Moreover, the Transformer-based sequential module increases architectural complexity and introduces several hyperparameters such as sequence length, attention heads, embedding dimensions, and dropout ratios, which require careful tuning in order to achieve optimal performance.

Finally, while the explainability framework improves transparency, explanation quality may still vary depending on the complexity of user behavior and the nonlinear nature of deep neural recommendation models.

4.7 CinéRec Interface Implementation

The system interface was implemented as a React application that communicates with the recommendation backend via a REST API. The interface is organized into two main modes:

4.7.1 Custom History mode

where the user manually creates a viewing history by searching for and selecting movies.

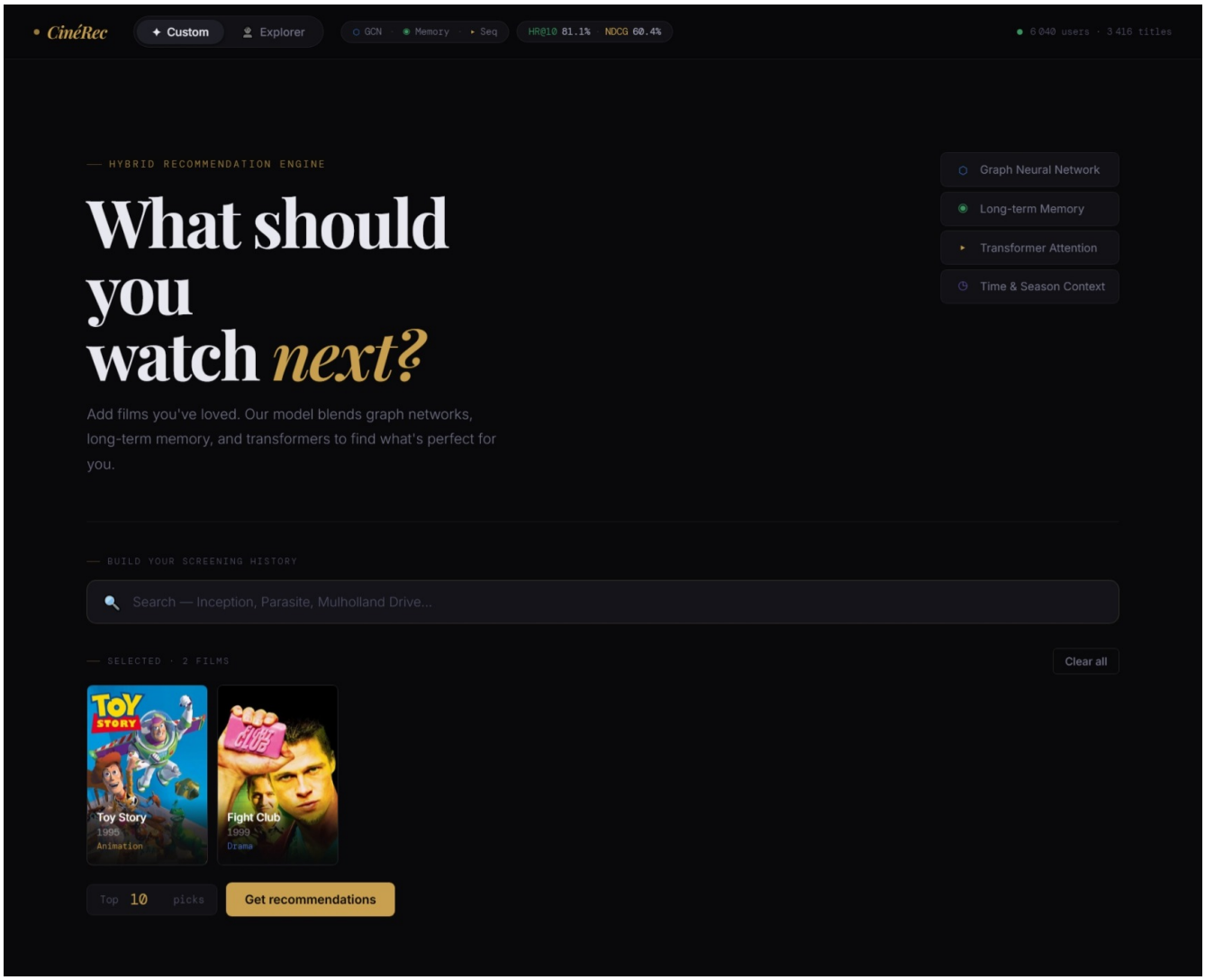


Figure 4.9: Comparison with Baseline Methods on MovieLens 1M

4.7.2 User Explorer mode

which loads any user from the test group and runs the model against their recorded history.

Profile Card

Once you select a user, the main panel opens with a profile card at the top. It contains: The user title (e.g., ”User 1042”) and a subtitle showing how many films are in their history and confirming they are a test-set user, A row of metric indicators showing Hit Rate and NDCG for this specific user, The target film the one film the model was trying to predict for this user, The rank at which that target film appeared in the recommendations, A Top N input to adjust how many recommendations to generate.

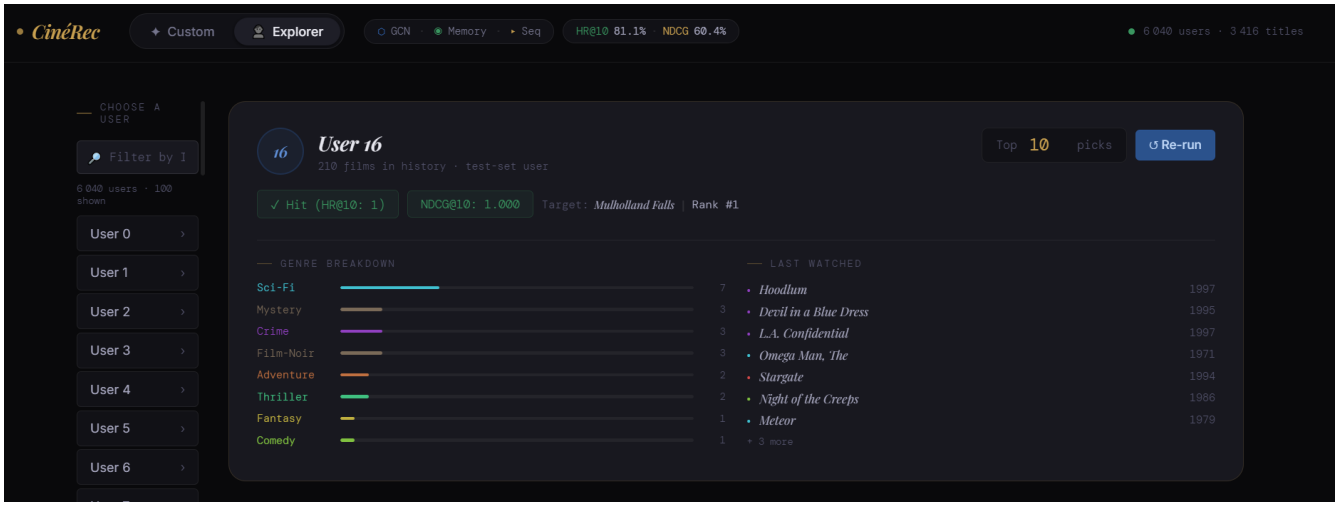


Figure 4.10: Profile Card

History Timeline and Recommendations Grid

The main output is a large-scale movie grid displaying all recommended titles, sorted from most relevant to least relevant. Each movie card displays the movie poster and title (automatically fetched from TMDB API).

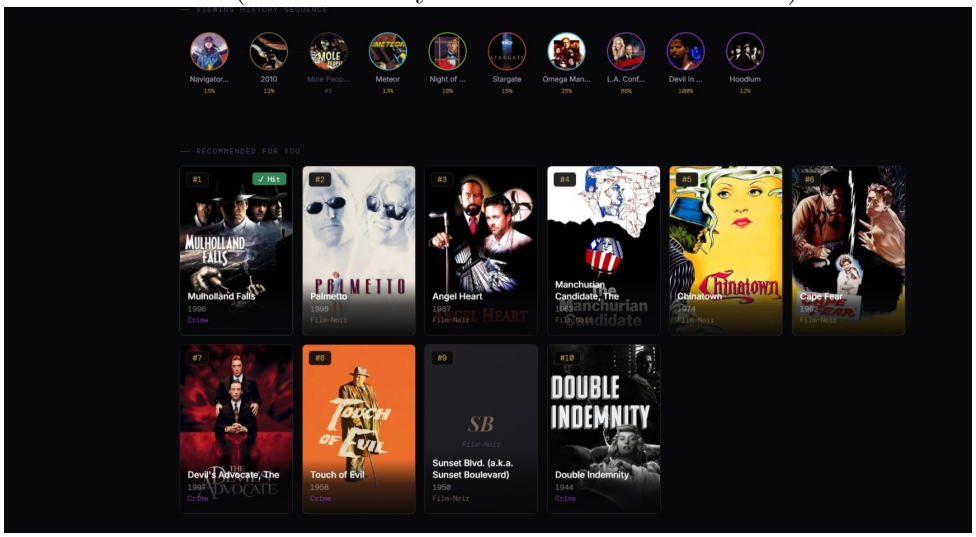


Figure 4.11: History Timeline and Recommendations Grid

AI Explanation Card

A narrative paragraph generated by the Gemini system explains why these films were recommended to this specific user. The explanation is generated by calling the Gemini API with a structured prompt containing the user's watched genres, the top-8 attention-weighted films, and the top-5 recommended films with their scores.And also we can switch the paragraph to arabic.

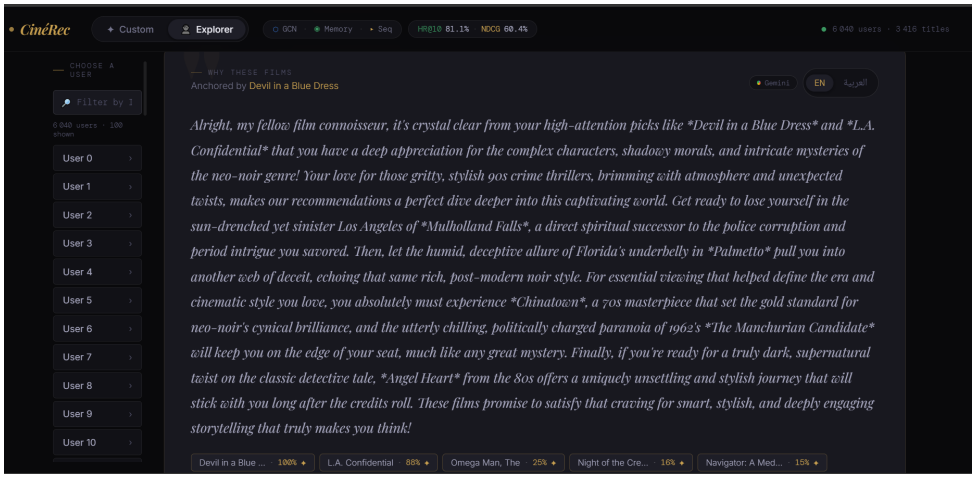


Figure 4.12: AI Explanation Card

The objective of this component is not to replace the quantitative explainability framework, but rather to transform complex attribution signals into more understandable tex-

tual explanations.

The `generateGeminiExplanation` function collects the most influential elements extracted from the hybrid explainability pipeline, including attention scores, GradientSHAP importance values, and the most relevant historical interactions from the `history_influence` array. The system also includes the top-ranked recommended items and the aggregated genre preference profile of the user in order to provide contextual information to the language model.

A structured prompt is then generated to guide the language model toward producing explanations aligned with the attribution outputs generated by the recommendation architecture.

Model Analytics Charts

Six charts organized in three rows of two.All six charts use Chart.js rendered into canvas elements through React refs.

- Model Component Blend + Signal Breakdown: The two graphs show the weights of the three gates, illustrating the contribution of each model to obtaining these recommendations.
- Attention Weights: The horizontal bar chart ranks the top-8 history films by attention score, each bar colored by that film's genre.
- Genre Spread: shows how the recommended titles distribute across genres.
- Score Chart: The bar chart shows raw relevance scores for the top-10 recommendations.
- Score Decay Curve: The line chart shows how scores fall off across all 15 recommendations, showing the confidence gradient from rank 1 to rank 15.

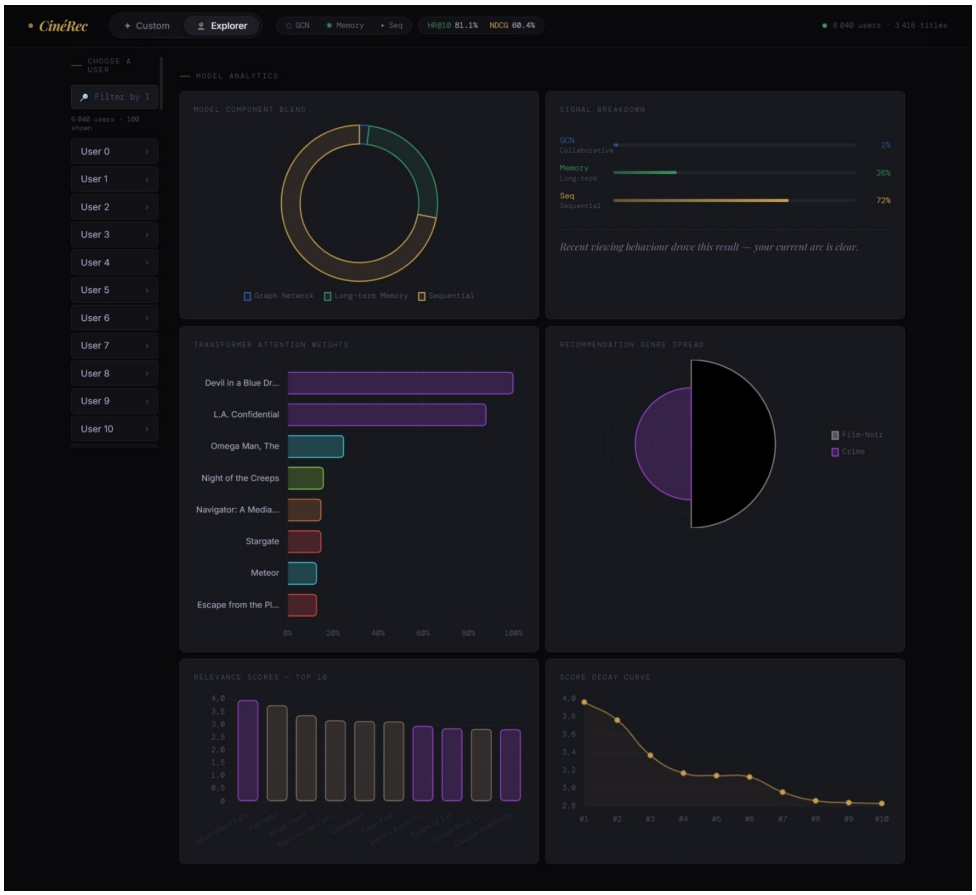


Figure 4.13: Model Analytics Charts

4.8 Conclusion

This chapter presented the experimental evaluation of the proposed DynamicHybridRec recommendation system. The results obtained on the MovieLens 1M dataset demonstrated that the model achieves strong recommendation performance according to HR@10, NDCG@10, and MRR@10 metrics. The training curves indicated fast convergence and stable learning behavior without significant overfitting. Comparisons with baseline approaches showed that the integration of LightGCN, the Transformer encoder, the External Memory module, and the Cross-Attention Fusion layer significantly improves recommendation quality compared to individual components. The ablation study further confirmed the contribution of each module to the overall effectiveness of the system. Additional experiments on sparse user histories revealed that the model remains capable of generating useful recommendations even when limited interaction data is available. The explainability evaluation demonstrated that the generated explanations are consistent and provide meaningful insights into the recommendation process, while the CinéRec user interface illustrated how recommendations and explanations can be presented in an interactive and user-friendly manner. Overall, the results indicate that DynamicHybridRec successfully combines recommendation accuracy, explainability, and practical usability, supporting the objectives of this work and highlighting its potential for real-world applications.

Despite these promising results, several directions remain for future research. A natural extension of the proposed framework is the integration of temporal context to capture the evolution of user preferences over time, enabling the system to model changing interests across different periods and situations more effectively. Another promising enhancement is the incorporation of viewing duration as an implicit feedback signal, inspired by large-scale recommendation platforms such as Netflix and YouTube. Rather than treating all interactions equally, the model could distinguish between briefly viewed and fully consumed content, leading to more accurate representations of user engagement and preference. Furthermore, future work could extend the proposed architecture beyond the movie domain toward cross-domain recommendation scenarios involving multiple content types such as movies, music, books, and other media. Additional improvements may include leveraging richer multimodal information, incorporating self-supervised learning techniques, and developing more advanced explanation mechanisms that further enhance transparency, personalization, and user trust in recommendation systems.

General Conclusion

Recommendation systems have become an essential part of modern digital platforms. They help users discover relevant content from large collections of items and improve the overall user experience. However, traditional recommendation methods still face several challenges, including data sparsity, cold-start situations, limited modeling of user behavior over time, and a lack of transparency in recommendation decisions.

The objective of this dissertation was to design and evaluate a hybrid recommendation system capable of improving recommendation quality while also providing understandable explanations. To achieve this goal, we proposed DynamicHybridRec, a hybrid architecture that combines LightGCN for collaborative learning, a Transformer encoder for sequential behavior modeling, an External Memory module for capturing long-term preferences, and a Cross-Attention Fusion mechanism for integrating information from different sources.

An explainability framework was also integrated into the system using attention analysis, GradientSHAP attribution, and component contribution analysis. In addition, a user interface called CinéRec was developed to present recommendations, visual analytics, and natural language explanations in an interactive way.

The proposed model was evaluated using the MovieLens 1M dataset. Experimental results showed that DynamicHybridRec achieved strong recommendation performance according to HR@10, NDCG@10, and MRR@10 metrics. The comparison with baseline models and the ablation study demonstrated the positive contribution of each architectural component. Additional experiments also showed that the model remains effective under sparse interaction conditions.

Despite these encouraging results, some limitations remain. The architecture requires significant computational resources and depends on historical interaction data. Furthermore, the explainability module was evaluated mainly from a technical perspective and could benefit from additional user-centered evaluation studies.

As future work, several improvements can be explored. The model could be evaluated on larger and more diverse datasets, additional contextual information could be incorporated into the recommendation process, and more advanced explainability techniques could be investigated. User studies could also be conducted to measure the usefulness and quality of the generated explanations.

In conclusion, the results obtained in this dissertation demonstrate that combining

collaborative learning, sequential modeling, memory mechanisms, and explainability techniques can lead to a recommendation system that is both effective and easier to understand. The proposed DynamicHybridRec architecture represents a promising step toward more accurate, transparent, and user-friendly recommendation systems.

Bibliography

[1] R. Burke, "Hybrid recommender systems: Survey and experiments," *User Modeling and User-Adapted Interaction*, vol. 12, no. 4, pp. 331–370, 2002.

[2] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 6, pp. 734–749, 2005.

[3] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[4] J.-W. Liu, J.-W. Liu, and X.-L. Luo, "Research progress in attention mechanism in deep learning," *Chinese Journal of Engineering*, vol. 43, no. 11, pp. 1499–1511, 2021.

[5] L. Lü, M. Medo, C. H. Yeung, Y.-C. Zhang, Z.-K. Zhang, and T. Zhou, "Recommender systems," *Physics Reports*, vol. 519, no. 1, pp. 1–49, 2012.

[6] P. Resnick and H. R. Varian, "Recommender systems," *Communications of the ACM*, vol. 40, no. 3, pp. 56–58, 1997.

[7] C. C. Aggarwal, *Recommender Systems: The Textbook*. Springer, 2016.

[8] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *IEEE Computer*, pp. 30–37, 2009.

[9] J. B. Schafer, D. Frankowski, J. Herlocker, and S. Sen, "Collaborative filtering recommender systems," in *The Adaptive Web*, pp. 291–324, Springer, 2007.

[10] P. Lops, M. de Gemmis, and G. Semeraro, "Content-based recommender systems: State of the art and trends," *Recommender Systems Handbook*, 2011.

[11] M. J. Pazzani and D. Billsus, "Content-based recommendation systems," in *The Adaptive Web*, pp. 325–341, Springer, 2007.

[12] M. Balabanović and Y. Shoham, "Fab: Content-based, collaborative recommendation," *Communications of the ACM*, vol. 40, no. 3, pp. 66–72, 1997.

[13] J. S. Breese, D. Heckerman, and C. Kadie, "Empirical analysis of predictive algorithms for collaborative filtering," in *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, pp. 43–52, 1998.

[14] J. L. Herlocker, J. A. Konstan, and J. Riedl, "Evaluating collaborative filtering recommender systems," *ACM Transactions on Information Systems*, vol. 22, no. 1, pp. 5–53, 2004.

[15] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrez, "Recommender systems survey," *Knowledge-Based Systems*, vol. 46, pp. 109–132, 2013.

[16] M. Etemadi, S. Bazzaz Abkenar, A. Ahmadzadeh, M. H. Kashani, P. Asghari, M. Akbari, and E. Mahdipour, "A systematic review of healthcare recommender systems: Open issues, challenges, and techniques," *Computer Methods and Programs in Biomedicine*, vol. 230, p. 107408, 2023.

[17] M. Wiesner and D. Pfeifer, "Health recommender systems: Concepts, requirements, technical basics and challenges," *International Journal of Environmental Research and Public Health*, vol. 11, no. 3, pp. 2580–2608, 2014.

[18] F. Ricci, L. Rokach, and B. Shapira, *Recommender Systems Handbook*. Springer, 2 ed., 2015.

[19] M. Schedl, H. Zamani, C.-W. Chen, Y. Deldjoo, and M. Elahi, "Current challenges and visions in music recommender systems research," *ACM Transactions on Intelligent Systems and Technology*, vol. 9, no. 3, pp. 1–21, 2018.

[20] C. A. Gomez-Uribe and N. Hunt, "The netflix recommender system: Algorithms, business value, and innovation," *ACM Transactions on Management Information Systems*, vol. 6, no. 4, pp. 1–19, 2015.

[21] D. Zibriczky, "Recommender systems meet finance: A literature review," *Expert Systems with Applications*, vol. 224, p. 119920, 2023.

[22] D. Buhalis and R. Law, "Twenty years on and 10 years after the internet: The state of etourism research," *Progress in Tourism Management*, vol. 30, no. 1, pp. 1–12, 2009.

[23] M. Madhukar, "Challenges & limitations in recommender systems," *International Journal of Latest Trends in Engineering and Technology (IJLTET)*, 2014.

[24] G. Shani and A. Gunawardana, "Evaluating recommendation systems," *Recommender Systems Handbook*, pp. 257–297, 2011.

[25] K. Järvelin and J. Kekäläinen, "Cumulated gain-based evaluation of ir techniques," *ACM Transactions on Information Systems*, 2002.

[26] S. Zhang, L. Yao, A. Sun, and Y. Tay, "Deep learning based recommender system: A survey and new perspectives," *ACM Computing Surveys*, vol. 52, no. 1, pp. 1–38, 2019.

[27] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*. Springer, 2009.

[28] X. Ning and G. Karypis, "Slim: Sparse linear methods for top-n recommender systems," in *ICDM*, 2011.

[29] S. Rendle, "Factorization machines," in *2010 IEEE International Conference on Data Mining*, pp. 995–1000, IEEE, 2010.

[30] scikit-learn developers, "Decision trees — scikit-learn documentation." `https://scikit-learn.org/stable/modules/tree.html`, 2026. Accessed: 2026-05-26.

[31] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *The Annals of Statistics*, vol. 29, no. 5, pp. 1189–1232, 2001.

[32] S. Rendle, "Factorization machines," in *2010 IEEE International Conference on Data Mining (ICDM)*, pp. 995–1000, IEEE, 2010.

[33] H.-T. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, M. Ispir, *et al.*, "Wide & deep learning for recommender systems," *arXiv preprint arXiv:1606.07792*, 2016.

[34] A. Aggarwal, "Evolution of recommendation systems in the age of generative AI," *International Journal of Science and Research Archive*, 2025.

[35] S. S. Aljameel *et al.*, "Matrix factorization collaborative-based recommender system for riyadh restaurants: Leveraging machine learning to enhance consumer choice," *Applied Sciences*, vol. 13, no. 17, p. 9574, 2023.

[36] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *Proceedings of the 26th International Conference on World Wide Web*, pp. 173–182, International World Wide Web Conferences Steering Committee, 2017.

[37] F. Sun, J. Liu, J. Wu, C. Pei, X. Lin, W. Ou, and P. Jiang, "BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer," in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pp. 1441–1450, ACM, 2019.

[38] M. Khoali, A. Tali, and Y. Laaziz, "A survey of artificial intelligence based e-commerce recommendation systems," *International Journal of Advanced Computer Science and Applications*, 2025.

[39] K. U. Kala and M. Nandhini, "Applicability of deep learning techniques in recommender systems," *IIOAB Journal*, vol. 10, no. 1, pp. 11–20, 2019.

[40] S. Lu, H. Chen, X. Zhou, B. Wang, H. Wang, and Q. Hong, "Graph-based collaborative filtering with MLP," *Mathematical Problems in Engineering*, vol. 2018, pp. 1–10, 2018.

[41] X. Zhou, "A brief review of basic deep learning models for recommendation systems," SciTePress, 2024.

[42] D. Liang, R. G. Krishnan, M. D. Hoffman, and T. Jebara, "Variational autoencoders for collaborative filtering," in *Proceedings of the 2018 World Wide Web Conference*, pp. 689–698, International World Wide Web Conferences Steering Committee, 2018.

[43] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *International Conference on Learning Representations (ICLR)*, 2017.

[44] U. A. Bhatti, H. Tang, G. Wu, S. Marjan, and A. Hussain, "Deep learning with graph convolutional networks: An overview and latest applications in computational intelligence," *International Journal of Intelligent Systems*, 2023.

[45] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *International Conference on Learning Representations (ICLR)*, 2017.

[46] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[47] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[48] F. Wu, A. Souza, T. Zhang, C. Fifty, T. Yu, and K. Weinberger, "Simplifying graph convolutional networks," in *Proceedings of the 36th International Conference on Machine Learning (ICML)*, pp. 6861–6871, 2019.

[49] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, and M. Wang, "LightGCN: Simplifying and powering graph convolution network for recommendation," in *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '20)*, pp. 639–648, ACM, 2020.

[50] X. Wang, X. He, M. Wang, F. Feng, and T.-S. Chua, "Neural graph collaborative filtering," in *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pp. 165–174, ACM, 2019.

[51] X. Wang, H. Jin, A. Zhang, X. He, T. Xu, and T.-S. Chua, "Disentangled graph collaborative filtering," in *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pp. 1001–1010, ACM, 2020.

[52] J. Wu, X. Wang, F. Feng, X. He, L. Chen, J. Lian, and X. Xie, "Self-supervised graph learning for recommendation," in *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pp. 726–735, ACM, 2021.

[53] J. Yu, H. Yin, X. Xia, T. Chen, L. Cui, and N. Q. V. Hung, "Are graph augmentations necessary? simple graph contrastive learning for recommendation," in *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pp. 1294–1303, ACM, 2022.

[54] B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk, "Session-based recommendations with recurrent neural networks," in *International Conference on Learning Representations (ICLR)*, 2016.

[55] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *Proceedings of the 30th International Conference on Machine Learning (ICML)*, pp. 1310–1318, 2013.

[56] L. Ben Allal, A. Lozhkov, E. Bakouch, *et al.*, "Smollm2: When smol goes big – data-centric training of a small language model," *arXiv preprint arXiv:2502.02737*, 2025.

[57] W.-C. Kang and J. McAuley, "Self-attentive sequential recommendation," in *IEEE International Conference on Data Mining (ICDM)*, 2018.

[58] F. Sun, J. Liu, J. Wu, C. Pei, X. Lin, W. Ou, and P. Jiang, "BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer," in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management (CIKM)*, pp. 1441–1450, ACM, 2019.

[59] A. Vaswani *et al.*, "Attention is all you need," in *NeurIPS*, 2017.

[60] B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk, "Session-based recommendations with recurrent neural networks," in *International Conference on Learning Representations (ICLR)*, 2016.

[61] W.-C. Kang and J. McAuley, "Self-attentive sequential recommendation," *arXiv preprint arXiv:1808.09781*, 2018.

[62] D. Soydaner, "Attention mechanism in neural networks: Where it comes and where it goes," *Neural Computing and Applications*, 2022.

[63] X. Chen *et al.*, "Sequential recommendation with user memory networks," in *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining (WSDM)*, 2018.

[64] J. Weston, S. Chopra, and A. Bordes, "Memory networks," *International Conference on Learning Representations (ICLR)*, 2015.

[65] A. Graves, G. Wayne, and I. Danihelka, "Neural turing machines," *arXiv preprint arXiv:1410.5401*, 2014.

[66] Y. Zhang and X. Chen, "Explainable recommendation: A survey and new perspectives," *Foundations and Trends in Information Retrieval*, vol. 14, no. 1, pp. 1–101, 2020.

[67] S. Jain and B. C. Wallace, "Attention is not explanation," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, (Minneapolis, Minnesota), pp. 3543–3556, Association for Computational Linguistics, 2019.

[68] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 4765–4774, 2017.

[69] S. Jain and B. C. Wallace, "Attention is not explanation," in *NAACL-HLT*, 2019.

[70] M. T. Ribeiro, S. Singh, and C. Guestrin, "Why should I trust you?: Explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1135–1144, 2016.

[71] "Precision and recall at k in ranking and recommendations," *Evidently AI*, 2025.

[72] NVIDIA, "NVIDIA T4 tensor core GPU datasheet." `https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/tesla-t4/t4-tensor-core-datasheet-951643.pdf`, 2019. Accessed: 2026-05-20.

[73] Y. Ji, A. Sun, J. Zhang, and C. Li, "A re-visit of the popularity baseline in recommender systems," in *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 1749–1752, ACM, 2020.

[74] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "BPR: Bayesian personalized ranking from implicit feedback," in *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence (UAI)*, pp. 452–461, AUAI Press, 2009.

[75] W. Samek, A. Binder, G. Montavon, S. Lapuschkin, and K.-R. Müller, "Evaluating the visualization of what a deep neural network has learned," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 11, pp. 2660–2673, 2017.