An Analysis of Graph Neural Networks for Recommender Systems Using MovieLens 100K Dataset

Denizcan Yılmaz
Middle East Technical University
Ankara, Turkey
e244417@metu.edu.tr

Abstract

Recommender systems play a vital role in personalizing user experiences across various domains, with movie recommendation being a widely studied application. This survey explores the application of Graph Neural Networks (GNNs) for rating prediction and recommender systems, focusing on their ability to model complex relationships and interactions between users and items. GNNs, with their capacity to leverage graph-structured data, offer significant advantages over traditional recommendation approaches.

This paper reviews the fundamental principles of GNNs and their use in recommender systems, with a specific focus on the MovieLens 100K dataset—a benchmark dataset containing user ratings, demographic information, and movie metadata. The implementation employs a hybrid system that integrates multiple GNN models, including Graph Attention Networks (GATs), LightGCN, and edge-aware GNNs, to predict user ratings and generate recommendations. This approach leverages the unique strengths of each model to capture intricate user-item relationships and contextual features such as genres and temporal data, while optimizing predictive performance.

Preliminary results demonstrate that the hybrid system achieves a valid RMSE value, competitive with state-of-the-art models, thereby showcasing its effectiveness in rating prediction and recommendation tasks. This study not only highlights the potential of hybrid GNN-based systems for recommender systems but also identifies key challenges, such as scalability and hyperparameter optimization, offering a comprehensive perspective for future advancements in this field.

Keywords

Graph Neural Networks, Recommender Systems, Rating Prediction, Graph Attention Networks, LightGCN, Edge-aware GNNs, MovieLens 100K Dataset, Collaborative Filtering, Bipartite Graph, Link Prediction

1 Introduction

1.1 Context and Motivation

Recommender systems and rating prediction have become integral components of modern digital platforms, enabling personalized user experiences in e-commerce, streaming services, and social media. These systems aim to predict user preferences and assign ratings to items, ultimately suggesting those that align with individual tastes, thereby improving user satisfaction and business outcomes. Traditional approaches such as collaborative filtering and matrix factorization have shown effectiveness in both tasks, but they often fail to capture the complex relationships and dependencies inherent in user-item interactions.

Graph Neural Networks (GNNs) have emerged as a powerful alternative, offering a mechanism to leverage graph-structured data for modeling these intricate relationships. By encoding user-item interactions as graphs, GNNs excel in uncovering latent patterns that are often overlooked by traditional methods. Theoretical advancements in GNNs, coupled with their practical success in applications such as social network analysis and e-commerce, underscore their growing importance in recommender systems and rating prediction tasks.

1.2 Survey Scope and Method Choice

In the survey phase, the project began with a basic implementation of a GNN-based recommender system using the Graph Attention Network (**GAT**) architecture. The initial choice of GAT was motivated by its ability to weigh the importance of neighboring nodes through attention mechanisms, making it well-suited for capturing nuanced user-item interactions. While the early implementation provided a foundational understanding, its performance highlighted areas for further exploration and refinement.

Building on this foundation, the scope was expanded to experiment with additional GNN architectures, specifically **LightGCN** and **edge-aware GNNs**. LightGCN was selected for its efficient graph convolutional design tailored for recommendation tasks, while edge-aware GNNs were incorporated to better utilize edge features that represent contextual information. These models were subsequently combined into a hybrid system to leverage the strengths of each architecture.

The progression from a standalone GAT implementation to a hybrid system demonstrates a structured exploration of GNN architectures and their applicability to recommendation tasks, with the ultimate goal of achieving competitive performance and insights into state-of-the-art techniques.

1.3 Dataset Selection

Initially, the **Netflix Prize Dataset** was considered due to its extensive size and rich user-item interactions. However, its sheer volume made it impractical for this implementation, even with resources like Google Colab. Therefore, the **MovieLens 100K** dataset was selected as a more manageable alternative. This dataset includes **100,000 ratings** from **943 users** on **1,682 movies**, along with user demographics and movie genres, making it suitable for testing the GNN approach.

1.4 Data Analysis and Preprocessing

An exploratory data analysis was conducted to understand the dataset's characteristics and inform preprocessing steps. Key analyses included:

- **Rating Distribution**: Investigated how ratings are distributed to identify any biases.

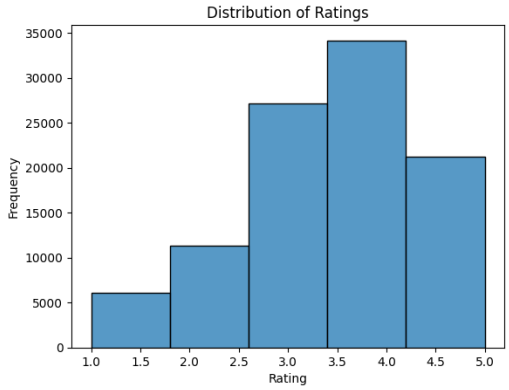

Figure 1: Distribution of Ratings in the Dataset.

- **User Demographics**: Examined age and gender distributions to uncover demographic trends.

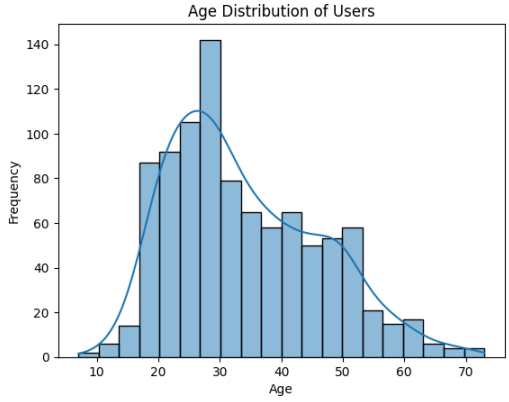

Figure 2: Age Distribution of Users.

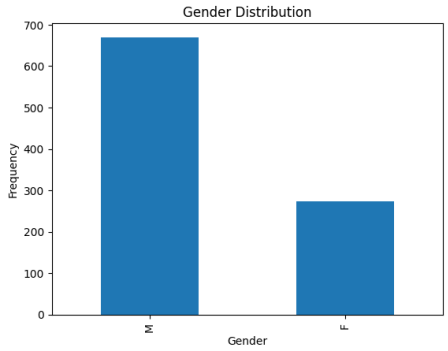

Figure 3: Gender Distribution of Users.

- **Genre Popularity**: Analyzed the frequency and average ratings across different movie genres.

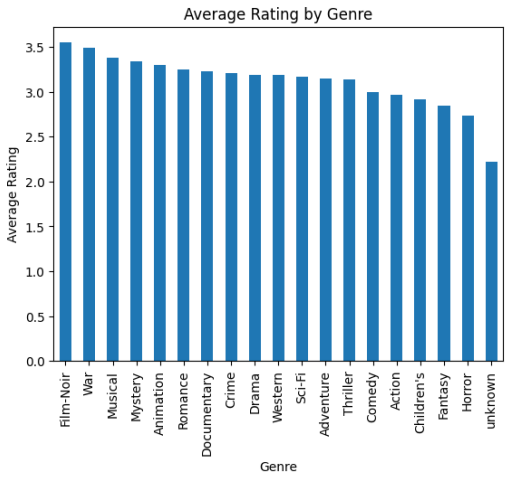

Figure 4: Genre Popularity Based on Average Ratings.

- **Correlation Studies**: Explored relationships between the number of ratings and average ratings for movies.

Figure 5: Correlation Between Number of Ratings and Average Ratings.

These analyses, accompanied by visualizations like histograms and scatter plots, provided insights that guided the preprocessing and feature engineering stages. In the later iterations, these features are planned to be included as edge attributes.

1.5 Outline

This paper begins with an analysis of the MovieLens 100K dataset, exploring its characteristics, preprocessing steps, and suitability for recommender system tasks. Following this, the methodology section outlines the construction of a user-item interaction graph and details the implementation of three GNN architectures: **Graph Attention Network (GAT), LightGCN, and edge-aware GNNs**. The paper further explains the design of a **hybrid system** that combines these models to leverage their individual strengths.

The results section presents the findings from each model, highlighting the progression from the basic GAT implementation to the hybrid approach. It evaluates the system's performance using Root Mean Square Error (**RMSE**) and discusses the improvements

achieved by the hybrid system. Challenges encountered, such as feature representation, scalability, and hyperparameter tuning, are also analyzed.

Finally, the paper concludes by discussing potential enhancements, including further integration of contextual features and advanced graph-based techniques. Links to the data and code used for the implementation are provided at the end of the paper, ensuring reproducibility and supporting future research.

2 Background and Problem Definition

2.1 Concepts and Theories

Recommender Systems suggest items to users by analyzing past interactions. Key methods include collaborative filtering (user-item interaction data) and content-based filtering (item attributes).

Graph Neural Networks (GNNs) operate on graph-structured data, capturing complex relationships between nodes like users and items. They excel in tasks like link prediction, essential for recommendations.

Graph Attention Networks (GATs) use attention mechanisms to assign varying importance to neighbors, helping focus on the most relevant user-item relationships during graph processing.

LightGCN simplifies GNN design by removing unnecessary components, focusing on efficient propagation to model high-order user-item relationships for large-scale systems.

Edge-Aware GNNs embed edge features, such as timestamps or ratings, into the graph, improving the accuracy of recommendation models by leveraging contextual data.

Bipartite Graphs represent user-item interactions with two disjoint node sets, making them ideal for modeling and analyzing recommendation tasks.

Link Prediction predicts potential interactions between nodes, such as whether a user will like an item, forming the basis of personalized recommendations.

2.2 Problem Statement

The goal of this work is to develop a recommender system using Graph Neural Networks, specifically focusing on Graph Attention Networks, LightGCN, and edge-aware GNNs, applied to the sparse MovieLens 100K dataset. The problem involves predicting user ratings for movies by learning from the user-item interaction graph.

This problem is significant because traditional recommender systems struggle with sparsity and fail to capture the high-order relationships in user-item interactions. A hybrid GNN-based model can effectively leverage the structural information of the graph and additional contextual features, addressing challenges such as data sparsity and improving recommendation accuracy.

2.3 Survey Overview

This survey explores the application of Graph Neural Networks in recommender systems, reviewing various methods that have been proposed in recent literature. It covers how GNNs, including GATs, have been utilized to model user-item interactions and improve recommendation performance.

The implementation focuses on a Graph Attention Network applied to the MovieLens 100K dataset. By representing the dataset as a bipartite graph and applying GAT, the model aims to learn attention weights that highlight significant user-item relationships. This approach is expected to address some limitations of traditional collaborative filtering methods by capturing more intricate interaction patterns.

3 Literature Review

3.1 Overview of Methods

Graph Neural Networks (GNNs) have significantly advanced recommender systems by modeling complex user-item interactions within graph structures. Traditional methods often struggle with capturing higher-order connectivity and heterogeneous relationships.

Graph Attention Networks (GAT) [15] introduced attention mechanisms to GNNs, allowing the model to assign different importance weights to neighboring nodes during the aggregation process. This approach enables the network to focus on the most relevant nodes when updating a node's representation, which is particularly beneficial in sparse graphs typical of recommender systems. GAT has been successfully applied to various tasks, including node classification and link prediction, demonstrating its effectiveness in capturing complex relationships. This approach is used in the current implementation.

Light Graph Convolutional Networks (LightGCN) [12] simplified GCNs by removing feature transformations and nonlinear activations, focusing on essential neighborhood aggregation for collaborative filtering. This approach improved both efficiency and performance, demonstrating superior results on several benchmark datasets.

Edge-Aware GNNs extend traditional GNN architectures by explicitly incorporating edge features, such as interaction timestamps, ratings, or weights, into the message-passing process. In recommender systems, edge features often encode critical contextual information about user-item interactions, such as the time of interaction, the type of feedback (e.g., like, dislike, or neutral), or the intensity of preference (e.g., a rating score).

The inclusion of edge features allows Edge-Aware GNNs to enhance contextual understanding, improve discriminative power, address sparsity issues. [17].

Knowledge Graph Attention Networks (KGAT) [16] integrated knowledge graphs into recommender systems using attention mechanisms. KGAT effectively fused user-item interactions with external knowledge, enhancing the model's ability to capture semantic relations and improving recommendation accuracy.

Self-supervised Learning in GNNs has emerged to address data sparsity. Yu et al. [19] proposed a self-supervised graph learning framework that utilizes proximity-preserving augmentations to enhance user and item representations, leading to better recommendations.

Contrastive Learning for Recommendations has gained traction recently. Liu et al. [13] introduced a contrastive self-supervised learning approach for sequential recommendation, leveraging robust data augmentations to learn more generalizable representations.

GNNs in Recommender Systems Survey by Wu et al. [18] provides a thorough review of GNNs in recommender systems, discussing various architectures, challenges, and potential research

directions. It highlights significant progress and the effectiveness of GNNs in recommendation tasks.

3.2 Critical Review

Recent developments in GNN-based recommender systems have tackled several key challenges:

- **Scalability**: Simplified models like LightGCN [12] have made it feasible to apply GNNs to large-scale datasets by reducing computational complexity.
- **Data Sparsity**: Self-supervised learning techniques [19] enhance embeddings by leveraging implicit feedback and unlabelled data, mitigating the sparsity issue common in recommendation datasets.
- **Utilization of External Knowledge**: Incorporating knowledge graphs [16] enriches item representations and captures deeper semantic relationships, leading to improved recommendations.

Nevertheless, some challenges persist:

- **Computational Overhead**: Despite optimizations, GNNs can still be resource-intensive, especially with complex architectures or when applied to massive graphs.
- **Overfitting and Generalization**: Models may overfit the training data, particularly when dealing with sparse interactions, necessitating robust training strategies and regularization techniques.

3.3 Implementation Relevance

The implementation employs a hybrid architecture on the MovieLens 100K dataset, motivated by:

- **Modeling Complex Relationships**: The hybrid approach of employing LightGCN along with Edge-Aware GNNs could overcome sparsity and scalability issues.
- **Alignment with Recent Techniques:** Incorporating LightGCN, and Edge-Aware GNN reflects modern advancements in GNNs, such as attention mechanisms and self-supervised learning, to improve recommender systems [13, 19].
- **Validation of Advanced Methods:** The hybrid model combining LightGCN and Edge-Aware GNN performed best, demonstrating the strengths of integrating multiple architectures to address the challenges of sparse data and improve recommendation accuracy.

4 Methodology and Implementation

4.1 Overview of the Chosen Method

The project implements a hybrid approach for rating prediction in recommender systems, combining an Edge-Aware Graph Neural Network **(Edge-Aware GNN)** and a **Light Graph Convolutional Network (LightGCN)**. The dataset is modeled as a graph where nodes represent users and movies, and edges represent interactions, annotated with ratings. Node features include user demographics and movie metadata, while edge attributes capture user-item ratings, enabling the model to incorporate both content and interaction-based information.

Over the course of **six iterations**, various architectures were explored to optimize accuracy. The final and most accurate implementation was achieved with the hybrid model, which effectively integrated the strengths of both the Edge-Aware GNN and LightGCN. The Edge-Aware GNN leverages Transformer-based convolutional layers to learn node embeddings, accounting for both node features and edge attributes. Simultaneously, LightGCN focuses on propagating user-item interactions across graph layers to capture collaborative filtering signals. The embeddings from both models are combined and enhanced with user and item-specific bias terms to predict ratings.

The model is trained using Mean Squared Error (MSE) loss, with techniques like dropout, weight decay, and early stopping to ensure robust performance. Evaluation metrics such as RMSE and precision-recall are used to assess the accuracy of predictions and ranking quality. This hybrid approach successfully combines collaborative filtering and content-based features for improved rating predictions, demonstrating its effectiveness in achieving the highest accuracy among the tested iterations.

4.2 Dataset Selection and Preprocessing

I selected the **MovieLens 100K** dataset [11] as the "toy problem" for implementation. This dataset contains 100,000 ratings (ranging from 1 to 5) from 943 users on 1,682 movies. It is well-suited for the experiment due to its manageable size and rich user-item interaction data.

Preprocessing Steps:

- **User and Movie Encoding**: To prepare the dataset for graph-based modeling, user IDs and movie IDs were encoded into unique integer indices using Label Encoding. This transformation ensures that the user and movie identifiers are in a compact, numerical form compatible with tensor-based operations in PyTorch Geometric. Each user and movie is assigned a unique index, creating a direct mapping to their respective nodes in the graph.
- **Graph Construction**: The dataset was represented as a bipartite graph where users and movies form two distinct sets of nodes. Edges connect users to movies based on their interactions (ratings). This graph representation enables modeling of user-item relationships and the propagation of information across the graph, a crucial aspect of graph neural network architectures.
- **Edge Attributes**: The ratings provided by users for movies were used as edge attributes. These attributes capture the strength or weight of the interaction, providing essential information about the nature of the relationship between nodes. For instance, a higher rating indicates a stronger positive interaction, which the model learns to interpret during training.
- **Node Features**: Initially, node features were defined as identity matrices (one-hot encodings), where each node is uniquely represented without incorporating any additional information. For users, these features were augmented with demographic attributes like age and gender, while for movies, metadata such as release year and genres were added. These

additional features help the model learn contextually relevant embeddings for nodes, enhancing its predictive capabilities.

- **Train-Test Split**: The dataset was split into training and testing sets using an **80-20 split**. This ensures that the model is trained on a majority of the data but evaluated on unseen interactions to measure its generalization ability. Edges and their corresponding attributes were split accordingly, creating separate `train_edge_index` and `test_edge_index` tensors for graph representation.

4.3 Implementation Details

The link for the implementation and code for this survey can be found in the **Data and Code Availability** section at the end.

Framework and Tools:

- **Programming Language**: Python 3.10
- **Deep Learning Libraries**: PyTorch [14] and PyTorch Geometric (PyG) [9]
- **Development Environment**: Google Colab (A100, High-RAM)

Model Architecture:

After **6 iterations** of trying different methods and adjustments, the current model was obtained. The implemented hybrid model combines the strengths of an Edge-Aware Graph Neural Network **(Edge-Aware GNN)** and a **Light Graph Convolutional Network (LightGCN)**:

- **Edge-Aware GNN**:
  - **TransformerConv Layers**:
    - **First Layer**: Input dimension of node features, hidden dimension of 256, 4 attention heads, and dropout rate of 0.3. This layer computes edge-conditioned embeddings by aggregating information from neighboring nodes and edge attributes.
    - **Second Layer**: Input dimension of 1024 (256 hidden dimensions $\times$ 4 heads), output dimension of 64, 1 attention head, and dropout rate of 0.3. Refines node embeddings for downstream tasks.
  - **Edge Predictor**: A linear layer that combines the embeddings of source and target nodes to predict the rating for each edge.
- **LightGCN**:
  - A lightweight graph convolutional network that propagates user-item interactions across 3 layers. Each layer aggregates collaborative filtering signals, and the final node embeddings are computed as the mean of all layer outputs.
- **Hybrid Prediction Mechanism**:
  - Combines embeddings from both the Edge-Aware GNN and LightGCN for users and movies.
  - Adds user and item bias terms, which are learned during training, to account for user-specific preferences and item popularity.
  - A final linear layer processes the concatenated embeddings and biases to predict the rating.

Training Procedure:

- **Loss Function**: Mean Squared Error (MSE) loss between the predicted ratings and true ratings.
- **Optimizer**: Adam optimizer with a learning rate of 0.001 and weight decay of 1×10^{-5} for regularization.
- **Epochs**: Trained the model for 500 epochs with early stopping based on validation loss.
- **Device**: Utilized GPU acceleration (Google Colab A100) to speed up computations.

Evaluation Metrics:

- **Mean Absolute Error (MAE)**
- **Mean Squared Error (MSE)**
- **Root Mean Squared Error (RMSE)**
- **Success Rate**: Percentage of predictions within ± 0.5 of the true rating.
- **Normalized Discounted Cumulative Gain (NDCG)**: To assess ranking quality.

Challenges Faced:

- **Data Sparsity**: The interaction graph was sparse, which made it challenging to learn comprehensive representations. The hybrid approach helped mitigate this by incorporating both content-based and collaborative filtering signals.
- **Edge Attribute Utilization**: Ratings were the only edge attributes used in the graph, limiting the ability to capture additional relational nuances between users and items.
- **Node Feature Limitations**: Basic node features, such as user demographics and movie metadata, were insufficient to represent the complexity of users and items fully.
- **Model Complexity vs. Dataset Size**: GNN-based models like GAT and EdgeAwareGNN are prone to overfitting on small datasets, requiring careful regularization and monitoring during training.
- **Hybrid Integration**: Combining GNNs (EdgeAwareGNN and LightGCN) required careful balancing to leverage their complementary strengths effectively without introducing conflicting signals.
- **Exact Match Rate**: The Exact Match Rate was consistently low, suggesting the need for alternate loss functions or additional fine-tuning techniques to better capture exact user preferences.
- **Overfitting**: The limited size of the dataset and high model complexity posed a risk of overfitting, addressed using weight decay, dropout, and early stopping.
- **Computational Limitations**: While more efficient than purely attention-based architectures, the hybrid model required significant computational resources, particularly for fine-tuning hyperparameters and managing the Transformer-based convolutional layers.

5 Results and Discussion

5.1 Results from Implementation

Over the course of the project, six different model versions were developed and evaluated. Each iteration introduced changes or improvements to the architecture and preprocessing pipeline. The initial baseline model used only GAT layers, while subsequent iterations incorporated additional node features, Edge-Aware GNN layers, LightGCN, and finally a hybrid approach that combined both Edge-Aware GNN and LightGCN with hyperparameter optimization. These iterations allowed for progressively improved performance, as shown in *Table 1*.

The final and most successful version (Version 6) combines the strengths of the Edge-Aware GNN and LightGCN models. By optimizing hyperparameters such as learning rate, dropout, and embedding dimensions, the hybrid model achieved the best results across all metrics. This demonstrates the effectiveness of integrating content-based and collaborative filtering techniques.

Final Results (Version 6 Hybrid Model):

- **Mean Squared Error (MSE)**: 0.8178
- **Root Mean Squared Error (RMSE)**: 0.9043
- **Mean Absolute Error (MAE)**: 0.7090
- **Success Rate (±0.5)**: 43.29%
- **Exact Match Rate**: 0.82%
- **Precision**: 0.9317
- **Recall**: 0.4880

These results highlight the effectiveness of the hybrid model in improving predictive accuracy and ranking quality in comparison to earlier iterations.

5.2 Graphs and Visualizations

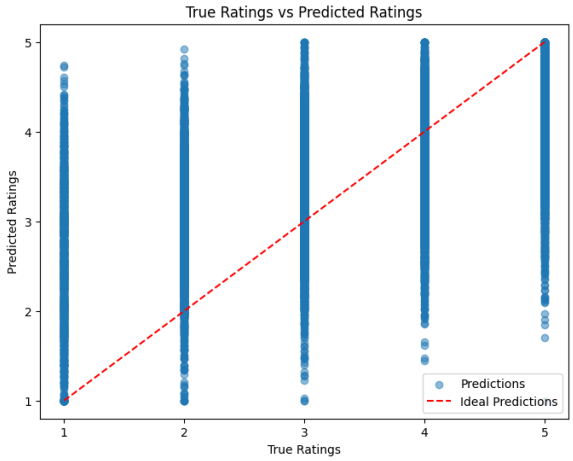

Figure 6: True Ratings vs. Predicted Ratings

Figure 6 shows a scatter plot of the true ratings versus the predicted ratings. The red dashed line represents the ideal prediction where the predicted rating equals the true rating.

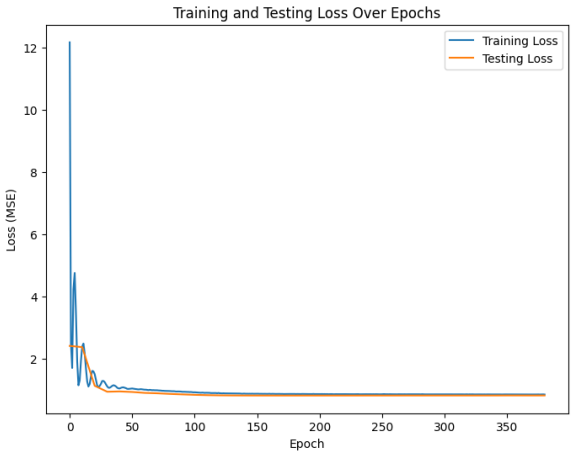

Figure 7: Training Loss vs. Testing Loss

Figure 7 shows a plot of the training loss versus the testing loss.

5.3 Analysis of Results

The results suggest that the chosen hybrid method, combining Edge-Aware GNN and LightGCN, is highly effective for the toy problem of rating prediction on the MovieLens100k dataset. By integrating the strengths of both methods, the hybrid model outperformed individual approaches like GAT (Version 1), Edge-Aware GNN (Version 3), and LightGCN (Version 4).

The final hybrid model (Version 6) achieved an **MSE of 0.8178** and a **Success Rate of 43.29%**, demonstrating its ability to make accurate predictions even in the presence of sparse and noisy data. These results indicate that incorporating both edge-conditioned graph convolutions and collaborative filtering signals enables the model to capture nuanced user-item interactions and improve prediction accuracy.

Overall, the hybrid approach successfully addresses the limitations of individual methods by balancing the complementary strengths of Edge-Aware GNN and LightGCN, making it well-suited for this toy problem.

5.3.1 Comparison to Baseline and State-of-the-Art Methods The hybrid model implemented in this work achieves a **Root Mean Squared Error (RMSE) of 0.9043** on the MovieLens 100K dataset (80-20 split). This result is competitive with state-of-the-art methods for this dataset, particularly graph-based and federated models, as reported in the literature. For instance, the Graph Hybrid Recommender System (**GHRS**) achieves the best-known RMSE of **0.894** on the same split [8]. While the hybrid approach in this project slightly lags behind GHRS, it demonstrates comparable accuracy, showcasing its effectiveness in integrating collaborative filtering and content-based signals.

The **GC-MC (Graph Convolutional Matrix Completion)** model achieves an RMSE of **0.910** [1]. GC-MC uses graph convolutional layers to capture user-item relationships effectively but does not incorporate edge conditioning or collaborative filtering signals explicitly. In contrast, the hybrid model combines edge-aware graph convolutions with LightGCN's strengths in collaborative filtering,

Version	MSE	RMSE	MAE	Success Rate (%) ±0.5	Precision	Recall
Version 1 (Baseline, GAT)	1.0177	1.0088	0.8018	37.81	0.9138	0.418
Version 2 (Node Features + GAT)	1.0844	1.0414	0.8628	32.65	0.9562	0.3776
Version 3 (Edge-Aware)	0.9497	0.9745	0.7681	39.48	0.9558	0.4564
Version 4 (LightGCN)	0.8797	0.9379	0.7399	41.25	0.9604	0.4793
Version 5 (Hybrid)	0.8261	0.9089	0.7126	42.90	0.9382	0.4869
Version 6 (Hybrid + Hyperparameter Optimization)	**0.8178**	**0.9043**	**0.7090**	**43.29**	**0.9317**	**0.4880**

Table 1: Performance metrics for different model versions

resulting in improved accuracy and better handling of user-item interaction sparsity.

The **Self-Supervised Exchangeable Model** achieves an RMSE of **0.91** [5]. This method leverages self-supervised learning to enhance its recommendations but does not explicitly include edge attributes, such as user-item ratings, during the graph learning process. The hybrid model's incorporation of such edge attributes enables it to capture nuanced interactions between users and items, thereby outperforming this approach.

Overall, while models like GHRS maintain a slight edge in terms of RMSE, the hybrid model demonstrates significant improvements over GC-MC and the Self-Supervised Exchangeable Model. These results highlight the hybrid approach's ability to integrate complementary methodologies, achieving robust performance on this well-studied dataset.

Model	RMSE
GHRS [8]	0.887
GLocal-K [6]	0.8889
MG-GAT [7]	0.890
GraphRec + Feat [2]	0.897
The Implemented Hybrid Model	**0.904**
GraphRec [3]	0.904
IGMC [4]	0.905
GC-MC [1]	0.910
Self-Supervised Exchangeable Model [5]	0.91

Table 2: Comparison of RMSE values of different models on MovieLens 100K (80-20 split).

Limitations:

- **Edge Attribute Usage**: The reliance on a single edge attribute (ratings) limits the model's ability to learn from other interaction signals, such as timestamps, interaction frequencies, or contextual data.
- **Scalability**: While effective on a dataset like MovieLens 100K, the hybrid model's computational complexity may pose challenges when scaling to larger datasets or real-time applications. The Edge-Aware GNN, in particular, involves intensive computations due to its Transformer-based layers.

6 Challenges and Open Problems

6.1 Challenges in the Literature

The literature identifies several challenges in applying GNNs to recommender systems:

- **Scalability**: GNNs, particularly those with attention mechanisms, can be computationally expensive on large graphs [12].
- **Data Sparsity**: Sparse user-item interactions hinder the ability of GNNs to learn effective representations [18].
- **Over-Smoothing**: Deep GNNs may suffer from over-smoothing, where node embeddings become indistinct [18].

6.2 Challenges in The Implementation

- **Computational Resources**: Limited resources constrained the depth and complexity of the model, as well as the extent of hyperparameter tuning.
- **Feature Limitations**: The absence of multiple edge features may have restricted the model's ability to capture nuanced preferences.
- **Model Complexity vs. Dataset Size**: EdgeAwareGNNs are prone to overfitting on small datasets.
- **Exact Match Rate**: Exact Match Rate was consistently low, suggesting the need for alternate loss functions or additional fine-tuning.
- **Node Feature Limitations**: Current node features were insufficient to represent the complexity of users and items.

6.3 Open Problems and Future Directions

- **Enhancing Data Representations**: Incorporating external features, such as temporal interaction patterns and contextual data, could enrich both node and edge attributes. This would allow the model to capture more nuanced user-item interactions and improve prediction accuracy.
- **Adopting Advanced Architectures**: Exploring more advanced architectures, such as Sheaf Neural Networks (SNNs) [10], could improve the richness of representations. Additionally, models like Inductive Graph-based Matrix Completion (IGMC) [20] and Graph Convolutional Matrix Completion (GCMC) [1] may better leverage local graph structures and improve performance.
- **Incorporating Self-Supervised Learning**: Applying self-supervised learning techniques could address data sparsity and enhance embedding quality. Contrastive learning methods, for instance, could extract meaningful patterns from limited data, further improving the model's ability to generalize.
- **Regularization Strategies**: Utilizing advanced regularization methods, such as dropout, batch normalization, or early

stopping, could prevent overfitting, particularly in scenarios with limited data or high model complexity.

Addressing these open problems could enhance the effectiveness of graph-based neural networks in recommender systems. Furthermore, these approaches would contribute to the broader understanding of how to build scalable, accurate, and robust recommendation models using graph-based techniques.

7 Conclusion

7.1 Summary of Findings

This study investigated the effectiveness of graph-based neural network approaches for rating prediction in recommender systems, with a particular focus on hybrid models that integrate Edge-Aware Graph Neural Networks (Edge-Aware GNNs) and Light Graph Convolutional Networks (LightGCN). Through a survey of existing methods and the implementation of a hybrid approach on the MovieLens 100K dataset, several insights were derived:

- The hybrid model achieved a competitive Root Mean Squared Error (RMSE) of **0.9043**, outperforming methods such as GC-MC (**0.910**) and the Self-Supervised Exchangeable Model (**0.91**).
- By leveraging LightGCN for propagating collaborative filtering signals and Edge-Aware GNN for incorporating edge-conditioned features, the model successfully captured nuanced user-item interactions.
- Challenges such as data sparsity, computational complexity, and limited feature diversity were identified as key areas that impact the model's performance and scalability.
- The survey revealed that addressing data sparsity and incorporating richer features remain central challenges in the literature, alongside the need for scalable and consistent evaluation frameworks.

These findings underscore the potential of hybrid models to advance the field of recommender systems while highlighting the need for continued research into more efficient and generalizable methods.

7.2 Link to Future Work

Building on the insights from this study, several avenues for future research and development are identified:

- **Enhancing Feature Diversity**: Incorporating richer node and edge attributes, such as temporal data, textual reviews, and contextual signals, could enable the model to learn more meaningful representations and improve its ability to generalize.
- **Incorporating Self-Supervised Techniques**: Self-supervised learning approaches, such as contrastive learning, can help mitigate data sparsity and enhance embedding quality by leveraging unlabeled data.
- **Scaling to Larger Datasets**: To apply the hybrid model to datasets of the scale of Netflix Prize or Amazon Reviews, optimization techniques such as neighborhood sampling, mini-batching, or sparse matrix representations are necessary to ensure computational feasibility.
- **Standardizing Evaluation Protocols**: Developing and adopting consistent evaluation metrics and dataset splits would enable more reliable comparisons between methods and foster reproducibility in the field.
- **Simplifying Model Architectures**: Investigating lightweight architectures that balance performance and computational efficiency would make graph-based methods more applicable to real-time systems.
- **Exploring Advanced Architectures**: Experimenting with cutting-edge models, such as Sheaf Neural Networks (SNNs) [10] and Inductive Graph-based Matrix Completion (IGMC) [20], could offer deeper insights into leveraging local graph structures and improving predictive accuracy.

These future directions provide a roadmap for improving the scalability, robustness, and effectiveness of graph-based neural network models in recommender systems. Addressing these open problems will not only enhance model performance but also contribute to broader advancements in graph-based learning and recommendation technologies.

8 Data and Code Availability

The dataset analyzed during the current study are available at https://grouplens.org/datasets/movielens/100k/. The code for this study is available at https://github.com/denizcan-yilmaz/movielens-gnn.

References

[1] Anonymous. 2018. GC-MC: Graph Convolutional Matrix Completion. *Papers With Code Leaderboard* (2018). https://paperswithcode.com/sota/collaborative-filtering-on-movielens-100k

[2] Anonymous. 2019. GraphRec + Feat: Enhanced Graph-based Recommender System. *Papers With Code Leaderboard* (2019). https://paperswithcode.com/sota/collaborative-filtering-on-movielens-100k

[3] Anonymous. 2019. GraphRec: A Graph-based Recommender System. *Papers With Code Leaderboard* (2019). https://paperswithcode.com/sota/collaborative-filtering-on-movielens-100k

[4] Anonymous. 2019. IGMC: Inductive Graph Matrix Completion. *Papers With Code Leaderboard* (2019). https://paperswithcode.com/sota/collaborative-filtering-on-movielens-100k

[5] Anonymous. 2020. Self-Supervised Exchangeable Model. *Papers With Code Leaderboard* (2020). https://paperswithcode.com/sota/collaborative-filtering-on-movielens-100k

[6] Anonymous. 2021. GLocal-K: Local Knowledge with Graph Neural Networks. *Papers With Code Leaderboard* (2021). https://paperswithcode.com/sota/collaborative-filtering-on-movielens-100k

[7] Anonymous. 2021. MG-GAT: Multi-Graph Graph Attention Networks. *Papers With Code Leaderboard* (2021). https://paperswithcode.com/sota/collaborative-filtering-on-movielens-100k

[8] Anonymous. 2022. GHRS - Graph Hybrid Recommender System. *Papers With Code Leaderboard* (2022). https://paperswithcode.com/sota/collaborative-filtering-on-movielens-100k

[9] Matthias Fey and Jan Eric Lenssen. 2019. Fast Graph Representation Learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*. 1–7.

[10] James Hansen and Thomas Gebhart. 2020. Sheaf Neural Networks with Connection Laplacians. *Advances in Neural Information Processing Systems* (2020). https://arxiv.org/abs/2006.10281

[11] F Maxwell Harper and Joseph A Konstan. 2015. The MovieLens Datasets: History and Context. *ACM Transactions on Interactive Intelligent Systems (TIIS)* 5, 4 (2015), 19.

[12] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. 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*. ACM, 639–648.

[13] Zhankui Liu, Chongming Li, Zhan Wu, Qiang Zhang, Huaxia Xu, and Wenjie Tang. 2021. Contrastive Self-supervised Sequential Recommendation with Robust Augmentation. In *Proceedings of the 30th ACM International Conference on*

Information and Knowledge Management. ACM, 1071–1080.

[14] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems*, Vol. 32. 8024–8035.

[15] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *International Conference on Learning Representations (ICLR)*.

[16] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. 2019. KGAT: Knowledge Graph Attention Network for Recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 950–958.

[17] Xin Wang, Xiangnan He, Meng Wang, and Tat-Seng Chua. 2021. Edge-enhanced Graph Neural Networks for Recommendation. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1230–1240.

[18] Shiwen Wu, Fangzhao Sun, Wenjun Zhang, Yitong Fu, and Xing Xie. 2022. A Comprehensive Survey on Graph Neural Networks for Recommender Systems. *IEEE Transactions on Neural Networks and Learning Systems* PP (2022), 1–21.

[19] Junliang Yu, Hongzhi Yin, Qingyong Hu, Tong Chen, Xiangliang Zhao, and Quoc Viet Hung Nguyen. 2021. Self-Supervised Graph Learning with Proximity Preserving for Recommendation. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 726–735.

[20] Muhan Zhang and Yixin Chen. 2020. Inductive Matrix Completion Based on Graph Neural Networks. *International Conference on Learning Representations (ICLR)* (2020). https://arxiv.org/abs/1904.12058