ISTANBUL TECHNICAL UNIVERSITY
FACULTY OF COMPUTER AND
INFORMATICS

Advancing Recommendation Systems Using Graph Neural Networks

Graduation Project Interim Report

Mustafa Can Çalışkan 150200097
Yusuf Şahin 150200016

Department: Computer Engineering
Division: Computer Engineering

Advisor: Assoc. Prof. Dr. Yusuf YASLAN

June 2025

Statement of Authenticity

I/we hereby declare that in this study

1. all the content influenced from external references are cited clearly and in detail,
2. and all the remaining sections, especially the theoretical studies and implemented software/hardware that constitute the fundamental essence of this study is originated by my/our individual authenticity.

İstanbul, June 2025

Mustafa Can Çalışkan 150200097
Yusuf Şahin 150200016

Acknowledgments

Advancing Recommendation Systems Using Graph Neural Networks

(SUMMARY)

The project is based on advancing an existing recommendation system using different techniques, and improving its approaches. This project requires solid information about *Graph Neural Networks, Vector Embeddings, Recommender Systems* and *Graph Theory.*

The article Graph Neural Network for Recommendation with Diversified Embedding Generation (DGRec) [1] is used as the base article. This article emphasizes the importance of the diversity which means that users are recommended with new interests or products. It uses GNNs since they are well suited for capturing user-item interactions. The flow of the article as basically follows:

- **Selecting a Diverse Subset** of neighbors to reduce redundancy via facility location function
- **Assigning attention weights** to embeddings from different GNN layers in order to solve over-smoothing problem
- **Re-weighting the Training Loss** to emphasize long-tail categories

The article mainly issues some problems for example *Accuracy - Diversity Dilemma* which refers increasing diversity leads to decreasing accuracy. The problems will be detailedly examined in the section 2.1.3.

For any kind of learning-based application, testing the model on an unseen data is crucial. Therefore, in this article 2 datasets are used as follow:

- **TaoBao** is a large-scale e-commerce dataset
- **Beauty** is an Amazon product review dataset

The results are useless by themselves until they are compared to others. In the end, this approach is compared to some state-of-the-art models like the Light Graph Convolution Network (LightGCN) and the Diversified Graph Convolution Network (DGCN), and this article outperforms.

Future directions for this article can be thought as follow:

- **Optimizing Hyperparameters** allow fine-tuning the accuracy - diversity dilemma
- **Applying to Different Domains** like social media or knowledge graphs
- **Testing on Different Dataset and Metrics** lead this research to get much more realistic feedbacks

This is for the base article. Our project focuses on improving this performance. As can be seen, the general outline is determined and prepared to be applied. The detailed plan is in section 1.

Graf Sinir Ağlarını Kullanarak Öneri Sistemlerini Geliştirme

(ÖZET)

Proje, mevcut bir öneri sistemini farklı teknikler kullanarak geliştirmeye ve yaklaşımlarını iyileştirmeye dayanmaktadır. Bu proje, *Graf Sinir Ağları, Vektör Gömmeleri, Öneri Sistemleri* ve *Graf Teorisi* hakkında yeterli bilgi gerektirmektedir. Çeşitlendirilmiş Gömme Üretimi ile Graf Sinir Ağı Önerisi (DGRec) [1] temel makale olarak kullanılmıştır. Bu makale, kullanıcılara yeni ilgi alanları veya ürünler önerilmesi anlamına gelen çeşitliliğin önemini vurgulamaktadır. Kullanıcı-ürün etkileşimlerini yakalamada iyi oldukları için GNN'leri kullanmaktadır. Makalenin akışı temel olarak şu şekildedir:

- Gereksiz tekrarı azaltmak için tesis konumlandırma fonksiyonu aracılığıyla **Çeşitli Komşu Alt Kümesi Seçimi**
- Aşırı düzleştirme problemini çözmek için farklı GNN katmanlarından gelen gömmelere **Dikkat Ağırlıkları Atama**
- Uzun kuyruk kategorilerine vurgu yapmak için **Eğitim Kaybını Yeniden Ağırlıklandırma**

Makale temel olarak *Doğruluk - Çeşitlilik İkilemi* gibi bazı sorunları ele almaktadır; bu, çeşitliliği artırmanın doğruluğu azaltmaya yol açtığı anlamına gelir. Bu sorunlar 2.1.3 bölümünde detaylı olarak incelenecektir. Öğrenme tabanlı herhangi bir uygulama için, modeli görülmemiş veriler üzerinde test etmek çok önemlidir. Bu nedenle, bu makalede aşağıdaki 2 veri seti kullanılmıştır:

- **TaoBao**, büyük ölçekli bir e-ticaret veri setidir
- **Beauty**, bir Amazon ürün inceleme veri setidir

Sonuçlar, başkalarıyla karşılaştırılmadığı sürece tek başlarına anlamsızdır. Sonuç olarak, bu yaklaşım, Hafif Graf Evrişim Ağı (LightGCN) ve Çeşitlendirilmiş Graf Evrişim Ağı (DGCN) gibi en son teknoloji modellerle karşılaştırılmış ve bu makale daha iyi performans göstermiştir. Bu makale için gelecek yönler şu şekilde düşünülebilir:

- **Hiperparametreleri Optimize Etmek** doğruluk - çeşitlilik ikilemini ince ayarlamaya olanak sağlar
- **Farklı Alanlara Uygulamak** sosyal medya veya bilgi grafları gibi
- **Farklı Veri Seti ve Metrikler Üzerinde Test Etmek** bu araştırmanın çok daha gerçekçi geri bildirimler almasını sağlar

Bu temel makale içindir. Projemiz bu performansı geliştirmeye odaklanmaktadır. Görüldüğü gibi, genel çerçeve belirlenmiş ve uygulanmaya hazır hale getirilmiştir. Detaylı plan 1 bölümündedir.

Contents

1 Introduction and Problem Definition

This project's aim to enhance a selected GNN-based recommendation system's performance on the chosen datasets and metrics. This is crucial since as discussed in the Summary, the article [1] has some problems, and some aspects available to be improved.

To form a frame, the planned flow of the project as follows:

- GNN
  - Allows representing user-item interactions as a bipartite graph
  - Enables embedding generation

  We will change the embedding strategies to improve the kept information of properties and relationships.

- Submodular Neighbor Selection
  - Chooses varied group of neighbors to avoid repetitive information
  - Enhances the diversity of embeddings

  We will try different submodular functions like category covarage or bucket covarage and try to enhance neighbor selection algorithms.

- Layer Attention
  - Merges embeddings from multiple GNN layers using dynamically learned attention weights

  We will try different attention mechanisms like multi-head and self-attenion. Also, we will work on improving readout function for better layer embedding combinations.

- Different Datasets
  - Datasets are vital to evaluate any model's performance
  - TaoBao and Beauty datasets are used in the article

  We will test our model using different datasets to get more comprehensive results. Example datasets are Steam, Last.fm and Goodreads.

2 Technical Details

In this section, some detailed technical background of the project such as problems and techniques.

2.1 GNN

GNNs are a family of deep learning models designed to process graph-structured data. They have become a powerful tool for tasks that require capturing relationships between interconnected entities, such as users and items in recommender systems.

2.1.1 How Works

2.1.1.1 Graph Representation Data is represented as a graph $G = (V, E)$, where:

- V: The set of nodes (e.g., users and items).
- E: The set of edges (e.g., interactions such as clicks, purchases, or ratings).

For recommendation systems, this is typically a **bipartite graph**, where:

- Nodes represent users ($u \in U$) and items ($i \in I$).
- Edges represent user-item interactions, such as a user clicking on or purchasing an item.

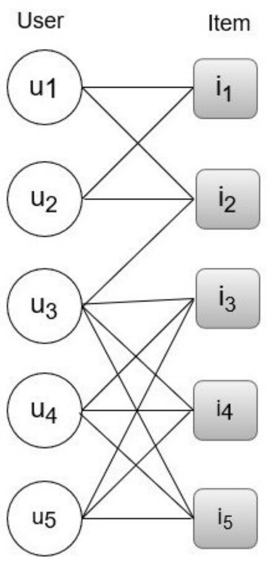

Figure 2.1: An example bipartite user-item graph [2]

2.1.1.2 Node Embeddings

- Each node is assigned an **embedding**, a dense vector that captures its attributes or properties.
- Embeddings are iteratively updated based on information from neighboring nodes in the graph.

2.1.1.3 Neighbor Aggregation

- At each GNN layer, a node aggregates the embeddings of its neighbors using an **aggregation function** (e.g., mean, sum, max, or learnable functions).
- The updated embedding for a node u at layer $l+1$ can be expressed as:

$$e_u^{(l+1)} = \text{COMBINE}\left(e_u^{(l)}, \text{AGGREGATE}\left(\{e_v^{(l)} : v \in N_u\}\right)\right)$$

  where N_u is the set of neighbors of u, and l represents the layer index.

2.1.1.4 Stacking Layers

- Multiple GNN layers are stacked to allow nodes to aggregate information from **higher-order neighbors**, such as neighbors of neighbors.
- This enables the model to capture both local and global relationships in the graph.

2.1.1.5 Output Embeddings

- The final embeddings encode the **structural** and **attribute information** of each node.
- These embeddings can be used for downstream tasks such as:
  - Classification
  - Regression
  - Recommendation

2.1.2 LightGCN

LightGCN is a simplified and efficient version of Graph Convolutional Networks (GCNs) tailored for recommendation systems. It is designed to capture user-item interactions in a graph-based structure while avoiding the complexities of traditional GCNs.

- **Feature Transformation Removal**
  - Traditional GCNs apply a learnable weight matrix for feature transformation at each layer.
  - LightGCN omits this, as feature transformation has little contribution to collaborative filtering tasks.
- **No Nonlinear Activation**
  - Nonlinear activation functions -e.g., rectified linear unit (ReLU)- are removed to preserve the raw collaborative signal in the embeddings.
  - This helps retain meaningful relationships between nodes.

2.1.3 Problems

2.1.3.1 Over-Smoothing

- As the number of layers in a GNN increases, the embeddings of nodes in the same neighborhood tend to become too similar.
- This phenomenon, known as **over-smoothing**, causes nodes to lose their uniqueness, reducing the effectiveness of recommendations.
- **Impact on Recommendations**:
  - User and item embeddings may no longer capture their distinct preferences and attributes.
  - Leads to less personalized recommendations.

2.1.3.2 Redundancy in Neighbor Aggregation

- GNNs aggregate information from all neighbors of a node, but not all neighbors contribute equally to the final embedding.
- Redundancy arises when popular nodes (e.g., frequently interacted items) dominate the aggregation process.

- **Impact on Recommendations**:
  - Reduces diversity in embeddings as multiple neighbors may carry redundant information.
  - Over-representation of popular items leads to a lack of novelty in recommendations.

2.1.3.3 Long-Tail Neglect

- In real-world datasets, user-item interactions often follow a power-law distribution, where a small number of items (head items) dominate interactions.
- GNNs naturally focus on these head items, while long-tail items (less popular ones) receive less attention during training.
- **Impact on Recommendations**:
  - Long-tail items are underrepresented in embeddings, making them less likely to be recommended.
  - Limits the diversity of recommendations, reducing user satisfaction.

3 Literatre Survey

Throughout the literature review, we read more than 50 articles. We have an idea about what state-of-art approaches are, which techniques are traditional and which methods are ignored mistakenly. A broad summary is available below:

3.1 Overview of the Articles

The articles mostly focus on the GNNs since the main aim is to enhance the use of GNNs. However, during our literature review we have seen that reinforcement learning (RL) is one of the other methods to be used with GNNs for recommendation systems. Until this point of the report, RL is not mentioned because we want to firstly inform the reader about main components. After this point our ideas, problems, perspectives, and more generally our future direction can be seen clearer. Detailed information on those ideas will be provided.

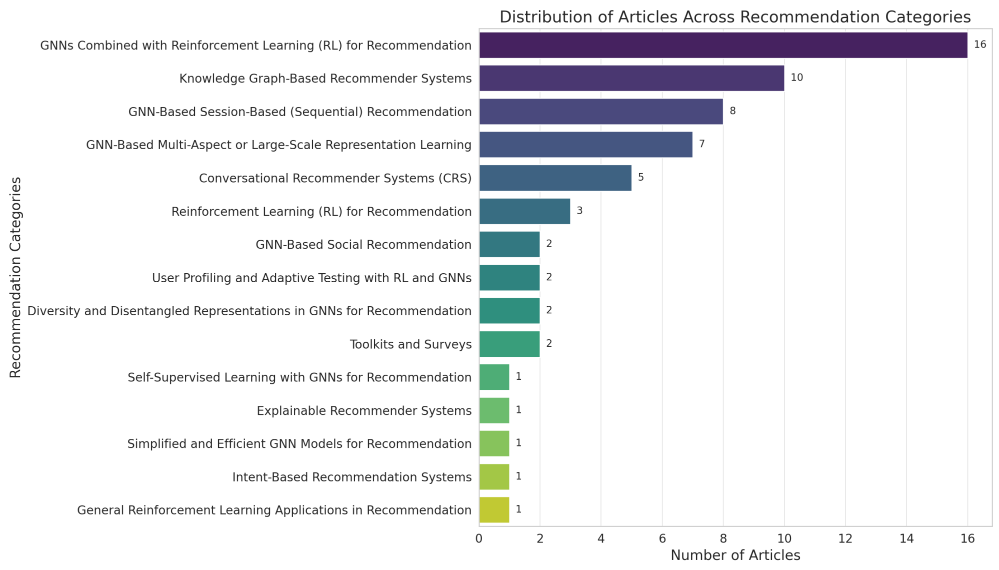

Figure 3.1: Distribution of the Article Topics

Figure 3.1 indicates the following:

- As mentioned, primary interest is in recommendation systems using GNN
- RL is also one of the used methods, and this combination provides acceptable results
- Self-supervised learning is used, however; it is not one of the first preferred methods
- Efficiency, explainability and diversity are other concerns
- Surveys are good to see trends

Year distribution of the articles can be seen in Figure 3.2.

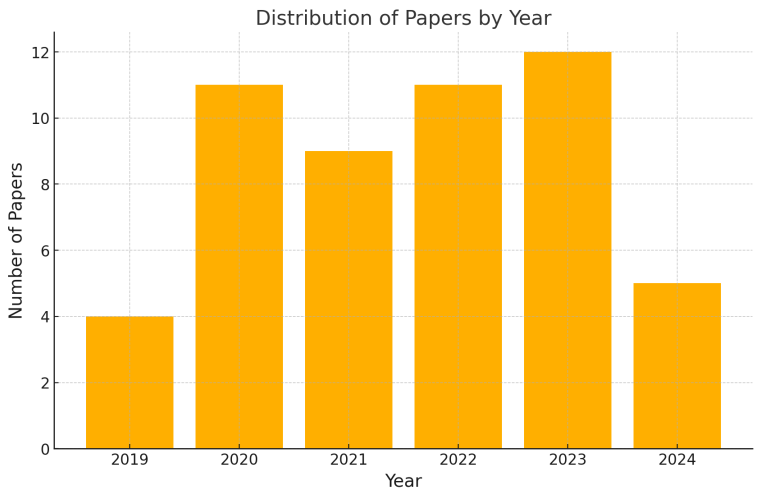

Figure 3.2: Distribution of the Article Release Years

3.2 GNN-based Publications

This section synthesizes the findings from prominent works in GNN-based recommendation systems to present a cohesive understanding of the field.

3.2.1 Core Methodological Advances

The integration of GNNs into recommendation systems has brought transformative changes by leveraging graph structures to model user-item interactions. Fan et al. [3] demonstrated how incorporating social networks into GNNs enhances recommendation accuracy, particularly by mitigating data sparsity through social signals. On the other hand, He et al. [4] introduced LightGCN, which simplified GNN architectures by focusing solely on neighborhood aggregation. This approach not only improved computational efficiency but also outperformed state-of-the-art models like Neural Graph Collaborative Filtering.

Dynamic GNNs, as proposed by Zhang et al. [5], expanded the scope to sequential recommendations by integrating temporal dynamics, enabling personalized and time-sensitive recommendations. Meanwhile, the base article [1] addressed the critical accuracy-diversity trade-off through DGRec, which incorporates modules such as submodular neighbor selection and layer attention to enhance diversity without sacrificing accuracy.

Last but not least, Gao et al. [6] provided a comprehensive survey of the field, identifying challenges in scalability, embedding propagation, and multimodal data integration. Their work serves as a critical reference point for future research directions.

3.2.2 Best Practices Across Studies

- **Simplification of Architectures:** LightGCN [4] illustrates the benefits of removing non-essential components to enhance performance and scalability.
- **Temporal and Contextual Adaptation:** Dynamic GNNs [5] show the importance of capturing evolving user preferences.
- **Balancing Accuracy and Diversity:** DGRec [1] exemplifies effective strategies to address the accuracy-diversity dilemma using advanced optimization techniques.

3.2.3 Comparative Performance and Applications

- **Accuracy:** LightGCN consistently outperforms in terms of precision and recall, particularly for collaborative filtering tasks [4].
- **Diversity:** DGRec leads in diversity metrics while maintaining competitive accuracy, addressing the limitations of earlier methods such as DGCN [1].

- **Sequential Recommendations:** Dynamic GNNs [5] effectively capture temporal dynamics, enhancing predictions for time-sensitive tasks.

3.2.4 Future Directions and Challenges

The evolution of GNN-based recommendation systems highlights promising areas for further exploration:

- **Scalability:** Developing models capable of handling large-scale graphs efficiently.
- **Enhanced Diversity:** Building on DGRec [1] to further improve diversity without compromising accuracy.
- **Multimodal Integration:** Leveraging text, images, and videos to enrich user-item representations [6].
- **Advanced Optimization Techniques:** Addressing oversmoothing and improving layer-wise aggregation, as seen in LightGCN [4] and DGRec [1].

3.3 Recommender Systems with RL

Advancements in RL for recommender systems revolve around innovations in graph-structured data processing, policy optimization, and sequential behavior modeling. The following provides a synthesized view of methodologies and best practices.

3.3.1 Graph Neural Network Integration with RL

Jung et al. (2023) proposed the Dual Policy Aggregation Optimization (DPAO) framework to enhance GNN-based recommendations by introducing dual Deep-Q-Networks (DQN). This approach optimizes user-item interaction aggregations through adaptive policy learning, achieving up to a 63.7% improvement in nDCG and addressing explainability challenges by incorporating Knowledge Graphs (KGs) [7].

Similarly, Hu et al. (2024) extended the scope of GNN applications by combining deep reinforcement learning (DRL) with semantic feature attribute adjacency graphs (MSFAAGs). This approach improves classification accuracy for MBD product model recommendations by leveraging both geometric and non-geometric attributes [8].

3.3.2 Sequential and Temporal Modeling

Hou et al. (2023) introduced a hybrid framework combining GRU and attention mechanisms to effectively capture sequential dependencies in user interactions. This model dynamically adjusts recommendations based on evolving user behavior [9].

3.3.3 State and Policy Optimization Frameworks

Lei et al. (2020) developed the Graph Convolutional Q-network (GCQN), leveraging graph-structured state and action representations to optimize recommendation policies. This model demonstrates substantial improvements in recommendation precision by harnessing user-item bipartite graphs [10].

Afsar et al. (2022) presented a comprehensive survey identifying four essential components for RL-based recommender systems: state representation, policy optimization, reward formulation, and environment building. The survey emphasizes the scalability of DRL methods in handling large action spaces [11].

3.3.4 Best Practices and Key Takeaways

3.3.4.1 Methodological Innovations

- **Graph Representations:** Leveraging graph-based representations (e.g., GCQN [10]) enhances the capture of high-order connectivity in user-item interactions.
- **Sequential Adaptation:** Incorporating temporal models like GRUs [9] improves personalization by accounting for time-sensitive user preferences.
- **Adaptive Aggregation:** Dual policy frameworks [7] optimize message passing in GNNs, enhancing both accuracy and explainability.

3.3.4.2 Performance Benchmarks

- **Accuracy:** Models like GCQN [10] and GRU-based frameworks [9] demonstrate high precision in sequential and graph-based scenarios.
- **Diversity:** Combining DRL with GNNs [8] improves model diversity by leveraging non-geometric attributes.
- **Explainability:** Knowledge graph-enhanced frameworks [7] provide interpretable recommendations by explicitly modeling context.

3.3.5 Challenges and Opportunities

The future of RL-based recommender systems lies in addressing existing limitations while exploring novel directions:

- **Scalability:** Efficiently scaling RL algorithms to handle large datasets and real-time interactions [11].

- **Reward Design:** Developing robust reward functions to optimize long-term engagement [11].
- **Data Integration:** Combining textual, visual, and contextual data for richer recommendations [8].
- **Explainability and Fairness:** Enhancing user trust by addressing biases and improving transparency [7].

3.4 Self-supervised Learning Centered Recommender Systems

This section consolidates findings on the integration of self-supervised learning into graph neural networks (GNNs) for recommendation systems, highlighting methodological innovations and empirical results.

3.4.1 Self-Supervised Graph Learning Framework

Wu et al. (2021) proposed the Self-supervised Graph Learning (SGL) framework to improve the robustness and accuracy of GNN-based recommendation systems. By supplementing supervised learning with a self-supervised task, SGL enhances node representation learning through contrastive objectives. Specifically, SGL generates multiple views of a node using three operators:[12]

- **Node Dropout:** Randomly removes nodes to simulate structural changes in the graph.
- **Edge Dropout:** Randomly drops edges to reduce noise and prevent overfitting.
- **Random Walk:** Samples subgraph structures to capture local connectivity patterns.

The framework maximizes the agreement between different views of the same node while minimizing similarities with other nodes, effectively self-discriminating nodes.

3.4.2 Addressing Long-Tail Recommendations and Noisy Interactions

Key Innovations:

- **Long-Tail Items:** SGL mitigates the dominance of high-degree nodes, improving recommendations for low-degree items.

- **Robustness to Noise:** By leveraging the auxiliary self-supervised task, SGL reduces the impact of noisy edges, ensuring more robust representations.
- **Hard Negative Mining:** The contrastive loss in SGL naturally mines hard negatives during training, as shown in the theoretical analyses [12].

3.4.3 Performance Comparisons and Results

SGL was implemented on LightGCN and evaluated on three benchmark datasets, yielding [12]:

- **Improved Accuracy:** Significant enhancements in top-K recommendation metrics, particularly for long-tail items.
- **Increased Robustness:** Superior performance in scenarios with noisy user-item interactions.

3.4.4 Best Practices and Implications

The SGL framework introduces key practices for robust and accurate recommendations:

- Incorporate self-supervised tasks as auxiliary objectives to reinforce node representation learning.
- Utilize graph augmentation techniques such as node and edge dropout for regularization.
- Leverage contrastive learning to improve robustness and enhance long-tail recommendations.

3.4.5 Challenges and Future Directions

The following research opportunities are identified [12]:

- **Adaptive Graph Augmentation:** Exploring dynamic strategies for selecting node and edge drop rates.
- **Multimodal Data Integration:** Extending SGL to incorporate textual, visual, and contextual information.
- **Model Generalization:** Applying SGL to diverse GNN-based architectures beyond LightGCN.

3.5 Efficiency and Explainability

Reinforcement learning (RL) integrated with graph neural networks (GNNs) represents a transformative approach for enhancing the efficiency and explainability of recommender systems. This section consolidates key contributions and insights from recent researches, with a focus on improving computational efficiency and providing interpretable recommendations.

3.5.1 Efficiency Improvements through RL-GNN Integration

Sharifbaev et al. (2024) introduced an efficient RL-GNN framework using Double Deep Q-Networks (DDQN). This integration addresses key challenges in scalability and computational complexity. [13]

- **Training Efficiency:** DDQN mitigates overestimation bias in reinforcement learning, leading to faster and more stable convergence.
- **Heuristic Adaptations:** Dynamically applied heuristics enhance the GNN training process, ensuring effective utilization of computational resources.
- **Scalability:** The framework achieves efficiency gains that make it applicable to large-scale datasets without compromising performance.

Empirical results demonstrate improvements in precision, recall, and overall computational efficiency, positioning the model as a practical solution for real-world recommendation systems.

3.5.2 Explainability Through Sentiment-Enriched Knowledge Graphs

Park et al. (2022) developed a Sentiment-Aware Policy Learning (SAPL) framework that integrates sentiment data into knowledge graphs to enhance the interpretability of recommendations. [14]

- **Sentiment-Aware Knowledge Graph (SAKG):** Constructed by analyzing user reviews and ratings, the SAKG incorporates nuanced sentiment data, providing richer context for user-item relationships.
- **Interactive Explanations:** SAPL generates textual explanations and highlights sentiment-driven interactions, significantly improving user trust and satisfaction.
- **Explainable Reasoning:** RL is utilized to traverse the SAKG, allowing the system to provide transparent and interpretable recommendations.

Experimental results indicate that SAPL improves both recommendation accuracy and the quality of explanations, setting a benchmark for explainable recommendation systems.

3.5.3 Key Insights and Best Practices

- **Efficiency:** Leveraging DDQN and heuristic adaptation strategies [13] improves computational scalability while maintaining recommendation accuracy.
- **Explainability:** Integrating sentiment analysis into knowledge graphs, as demonstrated by SAKG [14], enhances the interpretability of recommendations.
- **Unified Learning Objectives:** Combining efficiency-driven RL models with explainability-focused frameworks ensures both practical applicability and user satisfaction.

3.5.4 Performance and Challenges

- **Efficiency Gains:** The DDQN-GNN framework achieves significant reductions in computational overhead while delivering accurate recommendations.
- **Enhanced User Trust:** The explainability provided by SAPL improves user engagement and satisfaction.
- **Challenges:** Balancing the trade-offs between efficiency and explainability remains an open research question, particularly in dynamic and large-scale environments.

3.5.5 Future Directions

- **Real-Time Explainability:** Developing interactive interfaces for real-time explanation delivery could further enhance user engagement [14].
- **Efficiency Optimizations:** Exploring adaptive RL strategies to further reduce computational complexity [13].
- **Sentiment Dynamics:** Investigating the impact of evolving sentiment trends on long-term recommendation accuracy and trust.

4 Novel Aspects and Technological Contributions

As discussed in the section 3, our project is available to apply and integrate different techniques. In this section, techniques beside GNN (it is explain in the section 2.1).

4.1 RL

4.1.1 Reinforcement Learning in Recommender Systems

Reinforcement Learning offers a framework where an agent learns to make decisions by interacting with an environment to maximize cumulative rewards. In the context of recommender systems, the agent is the recommendation model, the environment encompasses the users and their interactions, actions correspond to the items recommended, states represent the current context or user state, and rewards are derived from user feedback such as clicks, purchases, or ratings.

4.1.1.1 Markov Decision Process The recommendation problem can be formalized as a Markov Decision Process (MDP), characterized by:

- **State Space** ($\mathcal{S}$): Represents the user's current context, preferences, and interaction history.
- **Action Space** ($\mathcal{A}$): The set of items that can be recommended to the user.
- **Transition Function** ($\mathcal{T}$): Models the probability of transitioning from one state to another after an action is taken.
- **Reward Function** (R): Assigns a reward based on the user's response to the recommendation.
- **Policy** (π): A strategy that the agent follows to decide actions based on states.

4.1.1.2 Advantages of RL-based Recommenders

- **Sequential Decision Making**: RL models the recommendation process over time, considering the long-term effects of actions.
- **Dynamic Adaptation**: The system can adapt to changing user preferences and contextual factors.
- **Personalization**: RL enables the creation of personalized recommendation policies tailored to individual user behaviors.

4.1.2 Integration of Graph Neural Networks with Reinforcement Learning

Graph Neural Networks (GNNs) have demonstrated their efficacy in modeling complex relational data by capturing the structural information of user-item interaction graphs. Integrating GNNs with RL can enhance the recommendation process by leveraging the rich connectivity patterns inherent in the data.

4.1.2.1 Graph-based State Representation GNNs can be employed to construct rich state representations by aggregating information from a user's local neighborhood in the interaction graph. This enables the RL agent to make informed decisions based on the broader context of user-item relationships.

4.1.2.2 Enhanced Policy Learning By utilizing embeddings generated by GNNs, the policy network within the RL framework can better capture the nuanced interactions between users and items. This leads to more accurate and contextually relevant recommendations.

4.1.3 Challenges and Problems

Despite the promising integration of RL and GNNs in recommender systems, several challenges persist:

4.1.3.1 Scalability Both RL and GNNs are computationally intensive, especially when dealing with large-scale user-item graphs. Ensuring scalability requires efficient algorithms and optimization techniques.

4.1.3.2 Sparse and Delayed Rewards User feedback can be sparse or delayed, making it difficult for the RL agent to learn effective policies. Designing appropriate reward mechanisms that can handle such sparsity is crucial.

4.1.3.3 Exploration vs. Exploitation Balancing the need to explore new recommendations with exploiting known user preferences remains a significant challenge. Excessive exploration can degrade user experience, while insufficient exploration may limit the discovery of optimal recommendations.

4.1.3.4 Cold Start Problem New users or items with limited interaction history pose difficulties for RL-based recommenders, as the agent lacks sufficient data to learn effective policies for these cases.

4.1.3.5 Integration Complexity Combining RL with GNNs introduces additional layers of complexity in model design and training. Ensuring seamless integration while maintaining performance is non-trivial.

4.1.3.6 Evaluation Metrics Traditional evaluation metrics may not adequately capture the long-term benefits of RL-based recommenders. Developing metrics that reflect user satisfaction and engagement over extended interactions is necessary.

4.2 Self-Supervised Learning in Recommender Systems

Self-Supervised Learning (SSL) is a subset of unsupervised learning where the data itself provides the supervision. By designing pretext tasks, SSL models can learn useful representations that can be fine-tuned for downstream tasks such as recommendation.

4.2.1 Motivation for SSL in Recommender Systems

- **Data Efficiency**: SSL reduces the reliance on labeled data, which is often scarce or expensive to obtain in recommender systems.
- **Robust Representations**: By learning from the inherent structure and patterns in the data, SSL can produce more robust and generalizable representations.
- **Cold Start Problem**: SSL can help mitigate the cold start problem by leveraging auxiliary information and unlabeled interactions to better represent new users or items.
- **Enhanced Feature Learning**: SSL encourages models to capture deeper semantic and structural features, improving the quality of recommendations.

4.2.2 Common SSL Techniques

Several SSL techniques have been adapted for recommender systems, particularly when combined with GNNs:

4.2.2.1 Contrastive Learning Contrastive learning involves learning representations by distinguishing between similar (positive) and dissimilar (negative) pairs. In recommender systems, positive pairs can be user-item interactions, while negative pairs can be non-interactions.

4.2.2.2 Masking Strategies Inspired by techniques like Bidirectional Encoder Representations from Transformers (BERT) in Natural Language Processing (NLP), masking certain parts of the input (e.g., user or item features) and training the model to predict the masked elements can help in learning robust representations.

4.2.2.3 Graph Augmentation Applying various augmentations to the interaction graph, such as node dropping, edge perturbation, or subgraph sampling, allows the model to learn invariant features under these transformations.

4.2.3 Integration of Self-Supervised Learning with Graph Neural Networks

Graph Neural Networks (GNNs) are adept at modeling the relational structure of data, making them ideal for capturing user-item interactions in recommender systems. Integrating SSL with GNNs can significantly enhance the representation learning process.

4.2.3.1 Graph-Based SSL Objectives

- **Node Classification**: Predicting node attributes or labels based on the graph structure.
- **Link Prediction**: Predicting the existence of edges between nodes, which aligns well with recommendation tasks.
- **Graph Reconstruction**: Reconstructing the graph from learned representations to ensure that the structural information is preserved.

4.2.4 Contrastive GNNs for Recommendations

Contrastive methods for GNNs involve creating multiple views of the interaction graph through augmentations and training the model to maximize agreement between representations of the same node across different views while minimizing agreement with representations of other nodes.

4.2.5 Masked Graph Autoencoders

Masked graph autoencoders involve masking a subset of node features or edges and training the GNN to reconstruct the masked parts. This encourages the model to learn comprehensive representations that capture both local and global graph structures.

4.2.6 Graph Augmentation Techniques

Graph augmentation involves creating modified versions of the original interaction graph to provide diverse training signals. Techniques include:

- **Node Dropping**: Randomly removing nodes to simulate missing interactions.
- **Edge Perturbation**: Adding or removing edges to introduce noise.
- **Subgraph Sampling**: Extracting subgraphs to focus on local structures.

These augmentations help the GNN learn invariant and robust features, improving the quality of recommendations.

4.2.7 Advantages of SSL with GNNs in Recommender Systems

- **Enhanced Representation Learning**: SSL enables GNNs to learn more expressive and meaningful representations by leveraging the underlying graph structure without relying solely on labeled data.
- **Improved Generalization**: Models trained with SSL objectives tend to generalize better to unseen data, leading to more accurate and reliable recommendations.
- **Data Efficiency**: By utilizing unlabeled data, SSL reduces the dependency on labeled interactions, making it feasible to train models in environments with limited labeled data.
- **Mitigation of Overfitting**: SSL introduces additional training signals that can help prevent overfitting, especially in scenarios with sparse interactions.
- **Robustness to Noise**: Learning through SSL can make GNN-based recommender systems more robust to noisy or incomplete interaction data.

4.2.8 Challenges and Problems

Despite the promising advantages, integrating SSL with GNNs in recommender systems presents several challenges:

4.2.8.1 Designing Effective Pretext Tasks

Creating pretext tasks that align well with the downstream recommendation objectives is non-trivial. The pretext tasks must encourage the model to learn representations that are beneficial for making accurate recommendations.

4.2.8.2 Scalability

GNNs, especially when combined with SSL techniques like contrastive learning, can be computationally intensive. Scaling these models to handle large-scale user-item interaction graphs requires efficient algorithms and optimization strategies.

4.2.8.3 Negative Sampling in Contrastive Learning

Effective negative sampling is crucial for contrastive learning. Poorly chosen negative samples can lead to suboptimal representations. Balancing the number and diversity of negative samples remains a challenge.

4.2.8.4 Balancing SSL and Supervised Objectives

Integrating SSL with supervised learning objectives requires careful balancing to ensure that the model benefits from both types of signals without one overshadowing the other.

4.2.8.5 Handling Dynamic Interactions

User preferences and interactions are dynamic. Ensuring that SSL-based GNN models can adapt to evolving interaction patterns without retraining from scratch is essential for real-world applications.

4.2.8.6 Evaluation Metrics

Traditional evaluation metrics may not fully capture the benefits of SSL-enhanced representations. Developing comprehensive evaluation frameworks that consider both recommendation accuracy and representation quality is necessary.

5 System Requirements

Our GNN-based recommendation system enhancement project requires both functional and non-functional capabilities to effectively process user-item interaction data, generate meaningful recommendations, and evaluate performance improvements.

5.1 Functional Requirements

5.1.1 Data Processing and Management

The system must be able to:

- Import and process large-scale user-item interaction datasets
- Handle different data formats (CSV, JSON) for input datasets
- Preprocess and clean interaction data to remove inconsistencies
- Convert raw data into graph structure for GNN processing

5.1.2 Model Training and Optimization

The system should:

- Train GNN models on user-item interaction graphs
- Support multiple GNN architectures (LightGCN, DGCN)
- Implement various optimization techniques (neighbor selection, layer attention)
- Allow hyperparameter tuning for model optimization

5.1.3 Recommendation Generation

The system must:

- Generate personalized recommendations for users
- Support batch processing for multiple users
- Provide recommendation scores or rankings
- Allow filtering of recommendations based on various criteria

5.1.4 Evaluation and Analysis

The system should:

- Calculate standard recommendation metrics (precision, recall, nDCG)
- Generate performance comparison reports
- Support A/B testing between different model variants
- Visualize performance metrics and results

5.2 Non-Functional Requirements

5.2.1 Performance

5.2.1.1 Training Performance

- Model training time should not exceed 24 hours on standard datasets
- Support for GPU acceleration
- Memory usage should not exceed 32GB RAM

5.2.1.2 Inference Performance

- Recommendation generation should take less than 100ms per user
- System should handle at least 1000 requests per minute
- Support batch processing of at least 100 users simultaneously

5.2.2 Scalability

The system must:

- Handle datasets with millions of user-item interactions
- Scale horizontally for larger datasets
- Support distributed training for large models

5.2.3 Reliability

System reliability requirements:

- 99.9% uptime for recommendation service
- Automatic error recovery during training
- Data consistency checks during processing
- Regular model checkpointing

5.2.4 Maintainability

Code and Documentation:

- Comprehensive API documentation
- Clear code structure and organization
- Unit tests with at least 80% coverage
- Version control and change tracking

5.3 Use Cases / User Stories

5.3.1 Data Scientists

As a data scientist, I want to:

- Experiment with different GNN architectures
- Tune model hyperparameters easily
- Compare performance metrics across models
- Generate visualization reports

5.3.2 Researchers

As a researcher, I want to:

- Implement new optimization techniques
- Test novel attention mechanisms
- Evaluate model performance on different datasets
- Generate publication-quality results

5.3.3 Model Developers

As a model developer, I want to:

- Add new model architectures to the system
- Optimize existing implementations
- Debug model behavior effectively
- Profile model performance

5.3.4 System Administrators

As a system administrator, I want to:

- Monitor system resource usage
- Scale computing resources as needed
- Manage model deployments
- Handle system maintenance

These requirements ensure that the system can effectively serve its purpose as a research platform for advancing recommendation systems while maintaining practical usability and performance standards. The use cases cover the primary stakeholders who will interact with the system during development, research, and maintenance phases.

6 Project Plan

6.1 Resource Requirements

We utilize high-performance GPU computers in the LFD and Optimization Laboratory to implement our project, as it requires significant GPU capabilities.

6.2 Time Plan

In Figure 6.1 you can see the Gantt diagram of the project.

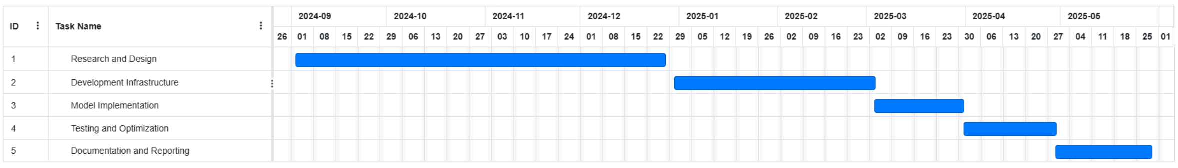

Figure 6.1: The Gantt diagram of the project

7 Goals and Evaluation Criteria

This project aims to enhance the existing DGRec recommendation system by implementing novel GNN architectures and integrating reinforcement learning techniques. Our primary focus is on improving both recommendation accuracy and diversity while maintaining computational efficiency.

7.1 Implementation Goals

1. Development of an enhanced GNN architecture incorporating:
   - Multi-head attention mechanism for better neighbor selection
   - Improved layer attention mechanism to address over-smoothing
   - Integration of reinforcement learning for dynamic user preference modeling
2. Implementation of novel diversity enhancement techniques:
   - Advanced submodular function design for neighbor selection
   - Dynamic reweighting strategy for long-tail item recommendations
   - Self-supervised learning components for better representation learning

7.2 Performance Criteria

1. **Accuracy Metrics**:
   - Achieve at least 15% improvement in NDCG@10 compared to base DGRec
   - Maintain precision@10 above 0.25 on TaoBao dataset
   - Achieve recall@20 of at least 0.30 on Beauty dataset
2. **Diversity Metrics**:
   - Increase category coverage by minimum 20% compared to LightGCN
   - Improve long-tail item recommendation ratio by at least 25%
   - Maintain intra-list diversity score above 0.7
3. **Efficiency Metrics**:
   - Training time should not exceed 1.5x of base DGRec
   - Memory consumption should stay within 32GB during training
   - Inference time should be under 100ms per user for top-k recommendations
4. **Scalability Benchmarks**:
   - Successfully process graphs with over 1 million nodes
   - Handle batch sizes of at least 1024 during training
   - Support parallel processing of minimum 100 concurrent requests

7.3 Evaluation Methodology

The system will be evaluated using:

- 5-fold cross-validation on both TaoBao and Beauty datasets
- A/B testing against baseline models (LightGCN, DGCN, DGRec)
- Additional testing on Steam and Last.fm datasets for generalization
- Comprehensive ablation studies for each new component

Success metrics will be measured through:

- Standard ranking metrics (NDCG, Precision, Recall)
- Diversity measurements (coverage, long-tail ratio)
- Efficiency metrics (training time, memory usage, inference speed)
- Statistical significance testing with p-value ¡ 0.05

References

[1] L. Yang, S. Wang, Y. Tao, J. Sun, X. Liu, P. S. Yu, and T. Wang, "Dgrec: Graph neural network for recommendation with diversified embedding generation," in *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*, ser. WSDM '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 661–669. [Online]. Available: https://doi.org/10.1145/3539597.3570472

[2] T. Lakshmi and D. Surampudi, "Link prediction approach to recommender systems," 02 2021.

[3] W. Fan, Y. Ma, Q. Li, J. Wang, G. Cai, J. Tang, and D. Yin, "A graph neural network framework for social recommendations," *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 5, pp. 2033–2047, 2022.

[4] 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*, ser. SIGIR '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 639–648. [Online]. Available: https://doi.org/10.1145/3397271.3401063

[5] M. Zhang, S. Wu, X. Yu, Q. Liu, and L. Wang, "Dynamic graph neural networks for sequential recommendation," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 5, pp. 4741–4753, 2023.

[6] C. Gao, Y. Zheng, N. Li, Y. Li, Y. Qin, J. Piao, Y. Quan, J. Chang, D. Jin, X. He, and Y. Li, "A survey of graph neural networks for recommender systems: Challenges, methods, and directions," *ACM Trans. Recomm. Syst.*, vol. 1, no. 1, Mar. 2023. [Online]. Available: https://doi.org/10.1145/3568022

[7] H. Jung, S. Kim, and H. Park, "Dual policy learning for aggregation optimization in graph neural network-based recommender systems," in *Proceedings of the ACM Web Conference 2023*, ser. WWW '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 1478–1488. [Online]. Available: https://doi.org/10.1145/3543507.3583241

[8] M. Y. M. Z. Yuying Hu, Zewen Sheng and C. Jian, "Gnn-based deep reinforcement learning for mbd product model recommendation," *International Journal of Computer Integrated Manufacturing*, vol. 37, no. 1-2, pp. 183–197, 2024. [Online]. Available: https://doi.org/10.1080/0951192X.2023.2258090

[9] Y.-e. Hou, W. Gu, K. Yang, and L. Dang, "Deep reinforcement learning recommendation system based on gru and attention mechanism." *Engineering Letters*, vol. 31, no. 2, 2023.

[10] Y. Lei, H. Pei, H. Yan, and W. Li, "Reinforcement learning based recommendation with graph convolutional q-network," in *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*,

ser. SIGIR ’20. New York, NY, USA: Association for Computing Machinery, 2020, p. 1757–1760. [Online]. Available: https://doi.org/10.1145/3397271.3401237

[11] M. M. Afsar, T. Crump, and B. Far, “Reinforcement learning based recommender systems: A survey,” *ACM Comput. Surv.*, vol. 55, no. 7, Dec. 2022. [Online]. Available: https://doi.org/10.1145/3543846

[12] 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*, ser. SIGIR ’21. New York, NY, USA: Association for Computing Machinery, 2021, p. 726–735. [Online]. Available: https://doi.org/10.1145/3404835.3462862

[13] A. Sharifbaev, M. Mozikov, H. Zaynidinov, and I. Makarov, “Efficient integration of reinforcement learning in graph neural networks-based recommender systems,” *IEEE Access*, vol. 12, pp. 189 439–189 448, 2024.

[14] S.-J. Park, D.-K. Chae, H.-K. Bae, S. Park, and S.-W. Kim, “Reinforcement learning over sentiment-augmented knowledge graphs towards accurate and explainable recommendation,” in *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*, ser. WSDM ’22. New York, NY, USA: Association for Computing Machinery, 2022, p. 784–793. [Online]. Available: https://doi.org/10.1145/3488560.3498515