HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

REPORT PROJECT 3

KGAT: Knowledge Graph Attention Network for Recommendation Implementation

TRẦN THỊ TƯỜNG VÂN
van.ttt225590@sis.hust.edu.vn

Major: Cyber Security

Supervisor: **Assoc. Prof. Dr. Nguyen Thi Kim Anh** _______________

Signature

Department: Computer Engineering

School: The School of Information and Communications Technology

HANOI, 02/2026

ACKNOWLEDGMENTS

I would like to extend my most sincere gratitude to Mrs. Nguyen Thi Kim Anh for her diligent attention and steadfast support in helping me navigate the complexities of Project 3. Her profound dedication and expertise played a truly pivotal role in the success of this work, and I am deeply privileged to have had the opportunity to learn under her mentorship. I am particularly grateful for her immense patience in deconstructing difficult theoretical concepts and for her constant vigilance in ensuring that my research was always aligned with the right objectives. Her unwavering willingness to address my numerous inquiries, provide access to essential academic resources, and motivate me to persevere through the most taxing technical challenges was instrumental to my overall progress.

Beyond technical guidance, this project has also been a transformative experience for me in terms of self-reliance. I recognize that the rigorous process of independent research and active knowledge-seeking was essential for the project's completion. This journey has not only solved the problems at hand but also equipped me with a versatile set of skills and a resilient mindset that will undoubtedly serve as a cornerstone for my future professional endeavors. Once again, I wish to express my heartfelt appreciation for her exceptional mentorship. Her invaluable guidance has left a lasting impact on my academic development, and I look forward to the possibility of learning from her again in the future.

ABSTRACT

Knowledge Graph Attention Network (KGAT) is a state-of-the-art recommendation framework designed to leverage high-order relations within a hybrid structure of knowledge graphs and user-item interaction graphs. Unlike traditional methods that treat interactions as independent instances, KGAT explicitly models the collaborative signal by linking items with their attributes. This report details the implementation and performance evaluation of the KGAT model, focusing on its core mechanism: recursive embedding propagation with an attention-based aggregator.

We explore how the model discriminates the importance of neighbors (users, items, and attributes) to refine node embeddings in an end-to-end fashion. The implementation is conducted on two benchmark datasets: Last.FM (music listening data) and MovieLens (movie ratings). We describe the background of knowledge-aware recommendation, the architectural components of KGAT—including the TransR-based embedding layer, the attentive delivery layer, and the experimental setup. Our study aims to verify the efficacy of KGAT in capturing high-order connectivity and its ability to provide superior recommendation accuracy compared to baseline models. The results provide insights into the interpretability benefits of the attention mechanism and the impact of the knowledge graph on resolving data sparsity issues in collaborative filtering.

TABLE OF CONTENTS

CHAPTER 1. INTRODUCTION

1.1 Motivation

In the era of information explosion, recommendation systems are essential for navigating vast datasets. Traditional methods, such as Collaborative Filtering (CF), primarily rely on historical user-item interactions. However, CF often suffers from data sparsity and the cold-start problem because it only considers direct, independent instances ($r = 0$) and fails to incorporate rich side information.

This project focuses on the impact of **High-order Connectivity**. By expanding the relation set from basic interactions ($r = 0$) to more complex attribute-based relations ($r = 1, 2$), the KGAT model can propagate preference signals across multiple hops in the Collaborative Knowledge Graph (CKG). Our goal is to demonstrate that increasing the diversity and depth of relations significantly enhance recommendation accuracy and robustness against sparse data.

- **High-order Connectivity:** High-order relations connect entities through one or multiple linked attributes, forming long-range connectivities in the graph. For instance, a user might be linked to a movie because they have watched other movies by the same director. This "High-order Connectivity" provides a powerful way to represent complex dependencies. Existing methods often fail to unify behavior-based (user-item) and attribute-based (KG) information effectively, or they lack the capability to model these multi-hop connections in an end-to-end manner.

1.2 Project Goals

This project aims to bridge the gap between user-item interactions and knowledge graph structures. The specific goals are:

- **Unified Structure:** To formulate the analysis task on a unified graph structure, called the Collaborative Knowledge Graph (CKG), which merges the user-item interaction graph with the entity-attribute knowledge graph.
- **Integration of Information:** To ensure that user behaviors and diverse side information can be encoded into a single relational graph, allowing the model to perceive the multi-step paths between users and items.
- **End-to-End Learning:** To learn a prediction function based on this graph structure that can automatically perform feature engineering via embedding propagation rather than relying on manually designed meta-paths.

- **Attentive Modeling:** To incorporate an attention mechanism that can discriminate the importance of different neighbors in the high-order connectivity, ensuring more precise representation learning.

1.3 Report Layout

The remainder of this report is organized into six chapters, detailing the transition from theoretical research to practical implementation and evaluation:

- **Chapter 1: Introduction -** Provides the motivation, research objectives, core contributions, and a high-level overview of the project's scope.
- **Chapter 2: Theoretical Background –** Delves into the fundamental concepts of Knowledge Graphs (KG) and the Collaborative Knowledge Graph (CKG). It further explains the architecture of the Knowledge Graph Attention Network (KGAT), including the TransR embedding layer, the attentive embedding propagation mechanism using Bi-Interaction aggregators, and the prediction layer.
- **Chapter 3: Dataset Analysis –** Introduces the Last.fm dataset used for experiments. This chapter analyzes the characteristics of the data, focusing on challenges such as high noise levels, data sparsity, and the computational risks associated with large-scale embedding matrices.
- **Chapter 4: Proposed Methodology –** This is the core of the project, detailing the technical improvements made to the source code. It covers the 3-Step Standardization process (Aggregation, Global Filtering, and Temporal/Intensity Splitting) and the Graph Construction strategies, including ID mapping and Neighbor Sampling to handle "Hub Nodes."
- **Chapter 5: Implementation and Evaluation –** Describes the experimental setup, including hyperparameters and the two-phase training strategy (KG Training and CF Training). It also presents the evaluation metrics (Recall@20, NDCG@20) and demonstrates the real-world utility of the system through the Inference Module.
- **Chapter 6: Conclusion and Future Work –** Summarizes the achieved results, discusses the effectiveness of the proposed preprocessing and sampling methods, and suggests potential directions for future enhancements, such as integrating multi-modal features or more complex aggregators.

1.4 Contribution

In modern data analysis, the explicit modeling of high-order relations is critical for achieving effective predictions. To exploit these complex interactions efficiently, it is essential to utilize end-to-end models rather than fragmented approaches. In this context, this project proposes the application of the Knowledge Graph Attention Network (KGAT) to capture the intricate patterns inherent in interconnected data. The main contributions include:

- **Explicit High-order Modeling:** By employing an attentive embedding propagation layer, the model explicitly captures high-order connectivities in a CKG, which is more effective than traditional regularization-based methods.
- **Attention-based Aggregation:** The model utilizes a knowledge-aware attention mechanism to weigh the influence of neighbors, providing better interpretability by identifying which relations contribute most to a recommendation.
- **Superior Performance:** We demonstrate that by combining the strengths of Graph Neural Networks (GNN) and Knowledge Graphs, the model can significantly outperform state-of-the-art methods like FM and NFM in terms of accuracy and robustness against sparse data.

CHAPTER 2. THEORETICAL BACKGROUND

The system is modeled as a Collaborative Knowledge Graph (CKG), which integrates a user–item interaction graph with a knowledge graph.

2.1 Knowledge Graph - KG

The system is modeled as a Collaborative Knowledge Graph (CKG), which integrates a user–item interaction graph with a knowledge graph.

CKG encodes user behaviors and item knowledge as a unified relational graph.

The key to successful recommendation is to fully exploit the high-order relations in CKG.

2.1.1 User–Item Graph:

- Interaction data is represented as a user-item bipartite graph.
- Defined as $(u, y_{ui}, i) \mid u \in \mathcal{U}, i \in \mathcal{I}$ where $\mathcal{U}$ and $\mathcal{I}$ separately denote the user and item sets, and a link (y_{ui}) = 1 indicates that there is an observed interaction between user u and item i; otherwise (y_{ui}) = 0.

2.1.2 Knowledge Graph:

- In addition to user-item interactions, side information for items, such as item attributes and external knowledge, is available.
- Such auxiliary data consist of real-world entities and relationships among them to profile an item.
- Presented as $(h, r, t) \mid h, t \in E,\ r \in R$, where *h* and *t* are entities and *r* is the relation between them.

2.2 Knowledge Graph Attention Network

The KGAT model consists of three core components:

1. **Embedding Layer (TransR)**
    - Knowledge graph embedding is employed to parameterize entities and relations as vector representations while preserving the graph structure.
    - This layer encodes Users, Items, and Relations into continuous vector representations.
    - The loss function calc_kg_loss is employed to preserve the structural properties of the knowledge graph triples (h,r,t), enforcing the constraint:

$$e_h + e_r \approx e_t \tag{2.1}$$

2. **Attentive Embedding Propagation Layer**
    - The model recursively propagates embeddings from a node's neighbors to update its representation.
    - An attention mechanism is used to compute the importance weights of neighboring entities during propagation.
    - The Bi-Interaction Aggregator (aggregator_type='bi') is adopted, which combines both linear summation and element-wise multiplication of the entity embedding and its neighbor embeddings.
3. **Prediction Layer**

- Multiple representations obtained from different propagation layers are concatenated to form the final user and item embeddings.
- The prediction score is computed by taking the inner product of the final user and item representations.
- This layer computes the similarity score between a User and an Item using the dot product of their corresponding embeddings.

2.3 Technology Stack

- **Backend:** Python.

CHAPTER 3. DATASET

To evaluate the effectiveness and generalization capability of the KGAT model, this project conducts experiments on two benchmark datasets with distinct characteristics: MovieLens1M (Explicit Feedback) and Last.fm1K (Implicit Feedback).

3.1 MovieLens1M Dataset

The MovieLens1M Dataset was collected by GroupLens Research (University of Minnesota) in 2000. This dataset represents Explicit Feedback, where users actively express their preference through specific ratings.

Dataset Overview:

- **Total rating:** 1,000,029 ratings.
- **Users:** 6040 users.
- **Movies:** Approximately 3,900 movies.
- **Rating scale:** 1 to 5 stars (integers only).
- **Density:** Each user has rated at least 20 movies, indicating a relatively dense dataset.

The Movies include Title and Genres(Action, Comedy, Drama, etc.).

The Users include Gender, Age, occupation, and Zip-code.

File Structure:

- **ratings.dat:** UserID::MovieID::Rating::Timestamp
- **users.dat:** UserID::Gender::Age::Occupation::Zip-code
- **movies.dat:** MovieID::Title::Genres

3.2 Last.fm1K Dataset

The Last.fm1K Dataset was collected from the Last.fm API (user.getRecentTracks) by Oscar Celma, covering listening habits up to May 2009. This dataset represents Implicit Feedback. The system only records the action of "listening" (tuples of <user, artist, song, timestamp>). There are no specific ratings; instead, listening frequency is used as a proxy for preference.

Dataset Overview:

- **Total Lines(Raw interactions):** approximately 19,150,6 lines.
- **Users:** 992 users.

- **Artists:** 107,528 (with MBID) and 69,420 (without MBID).

The primary relation used to construct the Knowledge Graph is the link between Track and Artist (Is_Sung_By).

User demographic profiles (Gender, Age, Country) are also available.

File Structure:

- **userid-timestamp-artid-artname-traid-traname.tsv:** Detailed listening logs.
- **userid-profile.tsv:** Contains user demographic information.

3.3 Dataset Comparison

Feature	MovieLens1M	Last.fm1K
FeedBack Type	Explicit (Rating 1-5)	Implicit(Listen Count)
Interaction Scale	1 Million	19 Million (Raw Logs)
Users Count	6,040	992
Sparsity	Dense Data	Sparse & Noise
Primary Challenge	Accurate Rating Preduction	Noise Filtering & Scalability

Table 3.1: Comparison Table

CHAPTER 4. METHODOLOGY

4.1 Data preprocessing

To address the recommendation problem on sparse and noisy datasets, this project proposes a rigorous Data Pipeline combined with the KGAT model. This chapter details the preprocessing steps, entity definition strategies, and knowledge graph construction methods for both Last.fm and MovieLens 1M datasets.

4.1.1 Preprocessing for MovieLens 1M

The MovieLens 1M dataset contains specific Explicit Ratings ranging from 1 to 5 stars. The processing pipeline focuses on converting the problem into a Ranking prediction task and leveraging rich movie attributes.

a, Label Binarization

The KGAT model in this project is designed for Top-K Recommendation tasks (predicting the probability of user interaction) rather than exact rating prediction (Regression). Therefore, the 1-5 star scale is converted into binary labels:

- **Positive Interaction(1):** Only ratings with a score of ≥ 4 are retained. This ensures the model learns only from movies the user truly enjoyed.
- **Removal(0):** Ratings < 4 are considered noise or dislike and are removed from the positive training set.

b, Entity Definition Strategy (User-Movie Modeling)

Determining the granularity of entities in the MovieLens graph is crucial for the GNN to operate effectively.

- **User:** Defined as the Viewer.
- **Item:** Maintained as the individual Movie.

Since MovieLens data possesses high interaction density and quality. Therefore, the project maintains the granularity at the Movie level to ensure the most detailed personalization.

However, to enhance knowledge propagation between movies, the project constructs an auxiliary entity called Genre Cluster:

- Instead of treating genres individually (e.g., separating Action and Comedy), the project treats the entire genre combination string (e.g., "Animation|Children's|Comedy") as a unique entity.
- **Significance:** Movies sharing the exact same genre "formula" are tightly linked

via this intermediate Genre Cluster node. This allows the KGAT model to recognize complex user tastes (e.g., distinguishing a user who prefers Action-Comedy from one who strictly prefers pure Action).

c, Data Splitting Strategy (Leave-One-Out)

To evaluate the model's ability to predict the user's next behavior, the project applies a time-based Leave-One-Out strategy:

- **Sorting:** All interactions for each user are sorted in ascending order by Timestamp.
- **Test Set:** Exactly 1 movie (highly rated, ≥ 4) that was rated most recently is selected for each user (args.test_size = 1). This is a standard evaluation protocol in Recommender Systems research.
- **Train Set:** All remaining positive historical interactions are used to train the model.

4.1.2 Preprocessing for Last.fm1K

The raw data from Last.fm consists of event logs containing significant noise regarding formatting and random listening behaviors. The processing pipeline is divided into four main stages:

a, Robust Parsing and Cleaning

Since the raw TSV data often contains formatting errors (e.g., extra tab characters within track or artist names), reading files using standard methods can lead to data misalignment. The project implemented a robust file reading module (lastfm_clean_from_ts
with a flexible string splitting mechanism:

- The first three fields (user_id, timestamp, artist_id) are split from the left.
- The last two fields (track_id, track_name) are split from the right.
- The remaining middle segment is merged into the artist name, ensuring no information loss and maintaining structural integrity.

b, Entity Definition Strategy (User-Artist Modeling)

A critical technical decision in this project is determining the data granularity for Last.fm:

- **User:** Defined as the Listener.
- **Item:** Defined as the Artist rather than the individual Track.

In the Last.fm 1K dataset, individual Tracks lack side-information such as genres or tags. If the Knowledge Graph were built based on Tracks, the network would become extremely sparse, lacking semantic links, making it difficult for the model to find commonalities between users. By elevating the granularity to Artist, the Knowledge Graph becomes a Denser Graph. Tracks by the same singer are grouped, creating more "Common Interest Points" between users. This enables the Attention mechanism of the KGAT model to propagate preference information more effectively.

c, Aggregation and Global Pruning

- **Aggregation:** Listening events are grouped by the (User, Artist) triplet to calculate the listen_count (total number of listens).
- **Global Pruning:** A threshold of MIN_GLOBAL_COUNT = 2 is applied. Artists with a total listen count across the entire system that is too low (e.g., listened to only once) are treated as noise or typos and are completely removed from the Master Data. This step reduces the Embedding space and prevents Out-Of-Memory (OOM) errors.

d, Temporal Splitting and Intensity Filtering

Instead of a Random Split, the project splits data based on the time axis to simulate a real-world future prediction scenario:

- **Sorting:** Interactions for each user are sorted by time (last_time descending).
- **Test Set:** The top $K = 5$ Artists listened to most recently by each user are selected.
- **Train Set:** The remaining interactions are used, but an intensity filter MIN_TRAIN_LISTEN = 3 is applied.

Only Artists that the user listens to repeatedly (≥ 3 times) are considered true preferences (Positive Samples). Exploratory listens (1-2 times) are removed to avoid False Positives during training.

Dataset	#Users (raw)	#Items (raw)	#Interactions (raw)	#Users (after)	#Items (after)	#Interactions (after)
MovieLens	6,040	3,952	1,000,209	6,040	3,883	1,000,209
Last.fm	992	83,982 artists	19,150,869	991	83,982	740,960

Table 4.1: Dataset statistics before and after preprocessing

4.2 Knowledge Graph Construction

After pre-processing, the data is converted into a Collaborative Knowledge Graph (CKG) structure.

4.2.1 Collaborative Knowledge Graph (CKG) Structure

To evaluate the effectiveness of adding more relations, we categorize the graph structure into three levels of connectivity:

1. **Level 1 ($r = 0$ - Baseline):** Includes only the direct interaction triples (User, interact, Item).
2. **Level 2 ($r = 1$):** Integrates direct item attributes to reduce sparsity.
   - Movie $\xrightarrow{belong\_to}$ Genre Cluster.
   - Artist $\xrightarrow{is\_sung\_by}$ Track.
3. **Level 3 ($r = 2$):** Incorporates advanced semantic or demographic relations.
   - MovieLens: Includes (User $\xrightarrow{has\_job}$ Occupation) and (Movie $\xrightarrow{released\_in}$ Year).
   - Last.fm: Includes the "Co-listening" relation (Artist $co - listen$ Artist), which links artists frequently listened to by the same user groups to capture hidden similarities.

4.2.2 Graph Structure for MovieLens1M

The MovieLens graph is richer due to available metadata:

1. **Interaction Relation:** User $\xrightarrow{interact}$ Movie.
2. **Attribute Relations:**
   - Movie $\xrightarrow{belong\_to}$ Genre Cluster.
   - Movie $\xrightarrow{released\_in}$ Year.
   - User $\xrightarrow{has\_job}$ Occupation (Leveraging demographic information).

4.2.3 Graph Structure for Last.fm

The graph consists of triples (h, r, t) with the following relations:

1. **Interaction Relation:** User $\xrightarrow{interact}$ Artist (Based on listening history).
2. **Co-listening Relation:** (Artist A $co - listen$ Artist B).
   - Automatically built from behavioral data via the build_kg_colisten.py module.
   - Two artists are linked if they are frequently listened to by the same group of users (exceeding min_edge_users). This adds hidden semantic information (e.g., genre similarity) that the raw metadata lack.

4.3 KGAT Implementation

The project applies the KGAT (Knowledge Graph Attention Network) architecture to learn representations on the graphs constructed above.

4.3.1 Embedding Layer (TransR)

Before entering the neural network, all Users, Items (Songs/Movies), and Attributes (Artists/Genres) must be transformed into vector representations (Embeddings).

Instead of using random initialization without semantic meaning, we employ the TransR mechanism to "teach" the model the semantics of relationships from the start.

4.3.2 Attentive Embedding Propagation

As the number of relations (r) increases, the Knowledge-aware Attention Mechanism becomes critical for discriminating the importance of various neighbors.

- In a low-connectivity graph ($r = 0$), the weights are relatively uniform.
- In a high-connectivity graph ($r \in \{0, 1, 2\}$), the model automatically assigns higher weights (π) to influential relations (e.g., an Artist link) while attenuating irrelevant ones (e.g., a specific Release Year), ensuring more precise representation learning.

This process involves two steps:

Step 1: Attention Mechanism Not all neighbors are equally important. For example, a user might like a song because of the Artist, which is more significant than the Release Year.

- The model uses an Attention network to automatically assign Weights (π) to each connection.
- Connections carrying useful information receive a high weight π, while irrelevant ones are attenuated.

Step 2: Information Aggregation (Bi-Interaction Aggregator) After determining the importance of neighbors, the model needs to aggregate their information into the node itself. We utilize the Bi-Interaction Aggregator (aggregator_type='bi') – the most robust method in the KGAT family. The aggregation formula is:

$$e_{new} = LeakyReLU(W_1(e_{old} + e_{neighbor}) + W_2(e_{old} \odot e_{neighbor}))$$

Its unique strength lies in combining two types of signals:

- **Addition ($+$):** Captures similarity in features.
- **Multiplication ($\odot$):** Captures strong interaction affinity between features. This enriches the user's representation vector after each propagation layer.

4.3.3 Prediction & Optimization

Prediction: After passing through 3 GNN layers (layer_size=[64, 32, 16]), the User and Item vectors have absorbed information from the entire graph. The compatibility score is simply calculated using the Dot Product:

$$Score = Vector(User) \cdot Vector(Item)$$

Loss Function: We do not train the model to predict exact scores (1-5 stars) but rather to rank items using the BPR Loss (Bayesian Personalized Ranking).

- **Principle:** In each training step, we sample a pair: (Item user likes) and (Item user dislikes).
- **Objective:** Force the model to score the (Liked Item) higher than the (Disliked Item). The larger this margin, the better.

CHAPTER 5. IMPLEMENTATION AND EVALUATION

5.1 Experimental Setup

5.1.1 Environment & Technology Stack

. The system is implemented using Python 3.8+ and PyTorch, optimized for GPU acceleration.

Data Structure: As defined in README.txt, standardized output formats are used for both datasets to ensure consistent loading:

- interactions_union.csv: Contains weighted interactions with temporal splits.
- kg_triples.csv: Stores KG structure (h, r, t) with unified entity indexing.

Hardware: Experiments were conducted on Local Enviroment

5.1.2 Hyperparameters

Based on config.py, the optimal configurations are:

- **Embedding Size:** 64 (for both Users and Entities).
- **Layer Size:** [64, 32, 16] (3 layers of high-order propagation).
- **Batch Size:** 512
- **Learning Rate:** $1e - 4$.
- **Aggregator:** Bi-Interaction (bi).

5.2 Training Strategy

Unlike standard implementations, our train.py employs an Alternating Optimization Strategy to balance structural learning and preference learning within each epoch:

1. **Phase 1: Knowledge Graph Embedding (TransR)**
    - Objective: Optimize the geometric structure of the graph.
    - Loss: $\mathcal{L}_{KG} = \sum ||h + r - t||^2$.
    - Process: Randomly samples triplets (h, r, t) and negative tails t' to refine entity embeddings.
2. **Phase 2: Collaborative Filtering (BPR)**
    - Objective: Optimize the ranking of items for users.
    - Loss: Bayesian Personalized Ranking (BPR) Loss.

- Process: Updates user and item embeddings based on the propagated information from the graph.

This separation ensures that the gradients from the recommendation task do not overwhelm the semantic structure of the Knowledge Graph.

5.3 Evaluation Protocol

5.3.1 Metrics

We use standard Top-K ranking metrics (implemented in utils.py):

- **Recall@K:** Measures the proportion of relevant items found in the top-K recommendations.
- **NDCG@K:** Measures the quality of ranking (position-aware).
- **K = 20** is used for all experiments.

5.3.2 Testing Procedure (Masking Strategy)

To ensure realistic evaluation, we implement a strict Masking Mechanism in test.py:

- Before ranking, the scores of all items that the user has seen in the Training Set are set to $-\infty$ (-np.inf).
- **Significance:** This forces the model to recommend new items (Discovery) rather than memorizing history.

5.4 Experimental Results

The following table illustrates the performance of the KGAT model as we increase the complexity of the relation set r. All experiments used fixed hyperparameters: Layer_Size = [64, 32, 16] and $Learning\_Rate = 10^{-4}$

Relation Level	Recall@20 (Last.fm)	NDCG@20 (Last.fm)	Recall@20 (MovieLens)	NDCG@20 (MovieLens)
Only $r = 0$ (CF Baseline)	0.0301	0.0207	0.0573	0.0200
$r \in \{0, 1\}$ (Basic Attributes)	0.0303	0.0213	0.0580	0.0205
$r \in \{0, 1, 2\}$ (Full CKG)	0.0303	0.0203	0.0559	0.0209

Table 5.1: Performance Comparison by Relation Levels

Key Observations:

1. **Effectiveness of High-order Connectivity:** In MovieLens, adding basic attributes $(r = 0, 1)$ yields a modest improvement, while adding additional relations $(r = 0, 1, 2)$ does not consistently improve Recall@20, suggesting that relation design and weighting need careful tuning. In Last.fm, gains are small,

consistent with the dataset's high noise level; stronger pruning and more reliable semantic relations are likely required to obtain larger improvements.

2. **Performance in Sparse/Noisy Data:**For Last.fm, the performance gains across relation levels were relatively marginal. While the $r = 2$ (co-listen) relation was intended to provide a significant boost , the decrease in NDCG@20 from 0.0213 to 0.0203 indicates that the "co-listen" relations may contain high noise levels or "Hub Nodes" that dominate the attention mechanism.
3. **The Role of the Attention Mechanism:** Despite the fluctuations in specific metrics, the attention mechanism successfully prevented a significant drop in performance when adding complex relations ($r = 1, 2$). This ensures that the model can weigh neighbors effectively, even if certain relations do not provide high-quality signals for every user.
4. **Model Sensitivity:** The results highlight that simply increasing the number of relations does not guarantee linear performance growth. Effective relation pruning and more rigorous semantic filtering are necessary to fully exploit the potential of High-order Connectivity in noisy environments like Last.fm.

To further validate the effectiveness of the Knowledge Graph Attention Network, we compare it against LightGCN, a state-of-the-art collaborative filtering model that utilizes graph convolution but lacks the ability to incorporate side information (Knowledge Graphs).

The following table compares the performance of LightGCN against KGAT at the $r = 1$ level (Basic Attributes).

Relation Level	Recall@20 (Last.fm)	NDCG@20 (Last.fm)	Recall@20 (MovieLens)	NDCG@20 (MovieLens)
LightGCN (Pure CF)	0.0739	0.0276	0.0936	0.0463
KGAT ($r = 1$) (With KG)	0.0303	0.0213	0.0580	0.0205

Table 5.2: Performance Comparison Between LightGCN & KGAT

Discussion and Analysis:

1. **Overall Comparison (Baseline Strength)** Based on Table 2, LightGCN consistently outperforms KGAT (r = 1) across all metrics and both datasets. Specifically, LightGCN achieves Recall@20 / NDCG@20 = 0.0739 / 0.0276 on Last.fm and 0.0936 / 0.0463 on MovieLens, whereas KGAT (r = 1) reaches 0.0303 / 0.0213 (Last.fm) and 0.0580 / 0.0205 (MovieLens). This indicates that, in our current implementation and preprocessing setting, the collaborative filtering signals captured by LightGCN provide stronger recommendation

performance than the KG-enhanced approach used by KGAT.

2. **Performance Gap on MovieLens:** The gap is especially noticeable on MovieLens. LightGCN's Recall@20 = 0.0936 is substantially higher than KGAT's 0.0580, and the difference is even larger for NDCG (0.0463 vs 0.0205). This suggests that, for MovieLens-1M, the user–item interaction patterns alone are highly informative, and the added knowledge relations in KGAT (r = 1) do not provide sufficient additional signal to surpass the strong CF baseline. Possible contributing factors include: (i) the constructed KG relations being too sparse or weakly aligned with user preference, (ii) noise introduced by imperfect side information, and/or (iii) KGAT being more sensitive to hyperparameters and training stability under constrained settings.

3. **Ranking Quality (NDCG)**: The NDCG results also favor LightGCN on both datasets. On Last.fm, LightGCN achieves NDCG@20 = 0.0276, higher than KGAT's 0.0213; similarly on MovieLens, 0.0463 is higher than 0.0205. Therefore, KGAT does not improve ranking quality in our current setup, and the attention-based propagation does not translate into better top-ranked recommendations under the tested configuration.

4. **Conclusion on Architecture:**The comparison suggests that the presence of side information alone is not sufficient for KGAT to outperform LightGCN. KGAT's advantage depends strongly on the quality, density, and relevance of KG relations, as well as careful tuning. In this experiment, LightGCN's simpler uniform neighborhood aggregation on the interaction graph appears more effective, while KGAT may require a more informative KG construction (e.g., more reliable item semantics or stronger user/context relations) and more extensive hyperparameter search to realize consistent gains.

5.5 Limitation

- Relation quality: weak/noisy relations may not help and can dilute attention.
- Hub nodes: popular entities can dominate neighborhoods; sampling and pruning are necessary.
- Hardware constraints: embedding size and batch size are limited by memory.

CHAPTER 6. CONCLUSION AND FUTURE WORK

6.1 Conclusion

In this project, we implemented and evaluated KGAT on MovieLens-1M and Last.fm and investigated the effect of increasing relation richness in the knowledge graph. The results in Table 5.1 show that adding relations does not guarantee consistent improvements across datasets or metrics. On MovieLens-1M, extending from r=0 to r=0,1 yields a small gain, while adding r=2 leads to mixed outcomes (e.g., Recall@20 decreases but NDCG@20 slightly increases). On Last.fm, expanding relations similarly results in non-uniform changes, where the best configuration depends on the metric, and additional relations can even reduce NDCG@20. Therefore, the main takeaway is that relation quality and alignment with user preferences matter more than relation quantity.

Moreover, our experiments show that a strong graph-based collaborative filtering baseline, LightGCN, outperforms KGAT under our current preprocessing and training constraints. This suggests that, in practice, KG-enhanced recommenders may require (i) more informative and denser side relations, (ii) careful indexing and graph construction, and (iii) more extensive hyperparameter tuning to consistently surpass well-optimized CF baselines. For future work, we plan to refine KG construction (e.g., more reliable user context features and stronger item semantics), conduct multi-seed evaluation for stability, and explore relation weighting/selection strategies to reduce noise and improve KGAT's effectiveness.

6.2 Future work

- Explore higher-order relations ($r \geq 3$) by integrating multi-modal features such as visual or textual descriptions.
- Optimize the training strategy to handle even larger knowledge graphs without incurring Out-Of-Memory (OOM) errors.

REFERENCE

[1] XiangWang, *KGAT: Knowledge Graph Attention Network for Recommendation*. Available: `https://arxiv.org/abs/1905.07854`.

[2] Harshal, *Last.FM Dataset*. Available: `https://www.kaggle.com/datasets/harshal19t/lastfm-dataset`.

[3] Sachin Sarkar, *Movielens Movie Recommendation System*. Available: `https://www.kaggle.com/code/sachinsarkar/movielens-movie-recommendation-system`.