Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed Computing

Transfer Learning for Graph-Based Recommendation Systems

Bachelor's Thesis

Elia Trachsel

trachsels@ethz.ch

Distributed Computing Group
Computer Engineering and Networks Laboratory
ETH Zürich

Supervisors:
Florian Grötschla, Luca Lanzendörfer
Prof. Dr. Roger Wattenhofer

March 6, 2025

Acknowledgements

I would like to express my gratitude to my supervisors, Florian and Luca, for their guidance throughout this thesis. At the same time, I appreciate the freedom they provided, allowing me to implement my ideas independently.Their encouragement and reassuring words were especially helpful during moments of frustration. I am also grateful to the professor for the opportunity to conduct my thesis in his lab.

Finally, I would like to thank my girlfriend, Livi, and my best friend, Vincent. Although they occasionally distracted me from working on my thesis, spending time with them always helped me recharge and regain motivation.

Abstract

Recommender systems play a crucial role in modern life, driving services ranging from streaming platforms to e-commerce. Traditional recommender models are typically trained end-to-end on static datasets, limiting their adaptability to the dynamic nature of real-world applications, where new users and items continually emerge. In this work, we introduce GRUTL (Graph-based Recommendation Using Transfer Learning), an inductive link prediction model that leverages transfer learning to generalize across multiple recommender datasets with minimal fine-tuning. By integrating graph convolutional networks with edge features—such as interaction weights and contextual information—GRUTL captures both the structural and semantic nuances of user-item interactions. Extensive experiments on diverse datasets, including MovieLens-1M, LastFM, Amazon, and Gowalla, demonstrate that a pretrained GRUTL backbone performs competitively in a zero-shot setting and that minimal fine-tuning further enhances its performance, often surpassing state-of-the-art end-to-end models. Our findings highlight the potential of transfer learning in recommender systems, paving the way for more resource-efficient and adaptive recommendations.

Contents

Chapter 1
Introduction

Recommender systems have become an integral part of our daily lives. Whether it is the next song queued on Spotify, product suggestions while online shopping, recommendations on a streaming platform, or content appearing on a social media feed—these systems are ubiquitous. The primary task of a recommender system is to suggest an item to a user based on past interactions and those of other users. A common way to model such recommendation tasks is as a bipartite graph, where nodes represent users and items, and edges indicate past interactions between them. One can then use modern graph-based learning techniques to generate high-quality recommendations.

However, these models are typically trained *end-to-end* on a fixed dataset, meaning they learn from a specific set of users, items, and interactions. During inference, recommendations are often made for the same set of users from the same pool of items. While this approach works in some domains, it fails to capture the *dynamic nature* of real-world recommender systems, where new users register and new items—such as songs, movies, or products—are introduced daily.

To keep up with these changes, many platforms retrain their models frequently, updating their graphs and re-optimizing the model parameters. However, this process is computationally expensive, as training a model is significantly more resource-intensive than inference. Existing models attempt to address this issue by incorporating techniques that handle the evolving nature of recommender systems without requiring frequent retraining. However, we take this idea one step further:

> *Can we train a backbone model on multiple recommender datasets, enabling it to generalize to new, unseen datasets with minimal fine-tuning?*

Since recommender systems are naturally represented as bipartite graphs, there is an inherent similarity between different recommendation settings. Instead of learning a recommendation model from scratch for each dataset, we propose leveraging *transfer learning* to create a generalizable model.

Our proposed model, GRUTL (Graph-based Recommendation Using Transfer Learning), is designed precisely for this purpose. At its core, GRUTL is an *inductive link prediction model.* Unlike traditional models that learn a direct mapping from users to recommended items, GRUTL *learns how to learn* this mapping. More specifically, we train a graph convolutional aggregation function, which can be applied to new datasets without retraining the entire model from scratch.

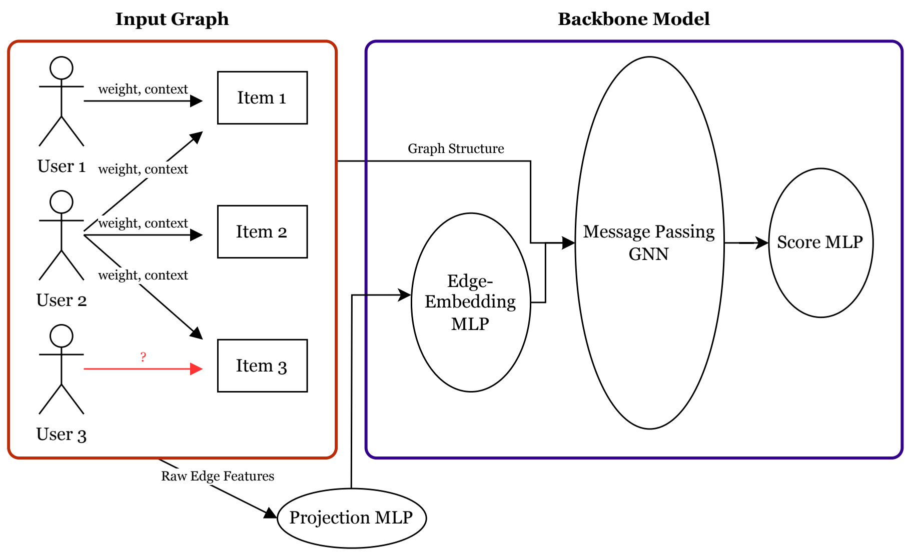


Figure 1.1: A high-level overview of GRUTL. The input graph captures user-item interactions and associated edge features (e.g., weight, context). A Projection MLP processes raw edge features, which are then embedded by an Edge-Embedding MLP. The graph is subsequently passed through a Message Passing GNN, and a Score MLP computes relevance scores for link prediction.

GRUTL builds upon NBFnet, an inductive link prediction model originally designed for knowledge graphs. However, NBFnet relies solely on the *structural properties* of the graph (i.e., edge types and connectivity patterns), which is sufficient for knowledge graphs but not for recommendation tasks. In contrast, real-world recommender systems often incorporate rich edge features—such as interaction weights (e.g., ratings or timestamps) and, when available, user/item attributes To bridge this gap, GRUTL leverages edge features throughout the convolution phase.

To ensure inductivity with respect to edge features, our backbone model expects feature vectors of a predefined dimension. These dimensions are carefully chosen to capture the most relevant information across various datasets. To accommodate different feature spaces, we use shallow MLPs to project initial embeddings into the feature space of GRUTL, allowing for fine-tuning on new datasets.

We evaluate the effectiveness of transfer learning for graph-based recommendation using GRUTL through extensive experiments across real-world recommendation datasets, covering various domains, including movies (MovieLens 1M), music streaming (LastFM), e-commerce (Amazon), and location-based recommendations (Gowalla), among others. The implementation is publicly available at [1].

Our results demonstrate that with a pretrained backbone, GRUTL is competitive in a 0-shot setting, sometimes outperforming state-of-the-art end-to-end recommender models. Furthermore, with minimal fine-tuning, its performance is further enhanced. These results underscore the potential of transfer learning in graph-based recommendation systems and pave the way for more adaptive, resource-efficient models.

[1]https://github.com/Elematre/tl4rec

Chapter 2

Related Work

In this section, we review works related to GRUTL. We first discuss inductive graph learning, covering models like NBFNet and ULTRA, which can generalize to new graphs and perform link prediction. Since GRUTL is built upon NBFNet, understanding its underlying principles is crucial. Additionally, we explore general inductive graph learning models and their relevance to recommendation systems.

Next, we examine cross-domain recommendation, a subfield of recommender systems where knowledge is transferred from a data-rich source domain to a data-scarce target domain. Most cross-domain approaches assume some form of similarity between domains, as outlined by Zhu et al. [1], who classify these methods based on whether they share the same users, items, or certain attributes such as keywords or tags. However, these methods fail when no such assumptions hold.

Finally, we review current transfer learning approaches for recommender systems that do not rely on cross-domain assumptions. These models typically leverage advances in natural language processing (NLP) and computer vision, utilizing pretrained representations from large language models (LLMs) and image-based embeddings to facilitate recommendation in unseen domains.

2.1 Inductive Graph Learning

2.1.1 General Overview

Inductive graph learning methods are designed to generalize to unseen nodes and graphs, a crucial property for dynamic real-world applications. Traditional approaches, such as LightGCN [2], perform message passing to obtain final node embeddings. However, they often rely on generating static entity and relation embeddings, which are inherently transductive and thus unable to generalize beyond the training dataset. To achieve node-inductivity, researchers have focused on path-based methods, where relational semantics are defined by edges rather than entities, allowing generalization to unseen nodes.

One of the earliest path-based inductive models is MINERVA [3], which uses reinforcement learning (RL) to explore relational paths relevant to a query. However, its scalability is limited due to the exponential growth of paths with graph size.

A*Net [4] addresses this limitation by leveraging a neural priority function, inspired by the A* search algorithm, to efficiently select the most informative paths. Unlike traditional A* search, which relies on handcrafted heuristics, A*Net learns a query-dependent priority function. It then applies message passing over selected paths, significantly improving efficiency on large knowledge graphs.

Despite their advantages, purely path-based methods often fail to capture rich local subgraph structures beyond sequential dependencies [5]. This motivates the development of subgraph-based methods, such as GraIL [6], which samples k-hop enclosing subgraphs around a query node pair and applies a graph neural network (GNN) to infer relationships. GraIL has been theoretically shown to capture probabilistic logical rules (e.g., Horn clauses), making it an effective method for inductive link prediction.

RED-GNN [5] combines the strengths of path-based and subgraph-based methods. It introduces r-digraphs, which extend relational paths into structured subgraphs by aggregating multiple overlapping paths. RED-GNN employs dynamic programming and a query-dependent attention mechanism to encode r-digraphs efficiently, achieving strong performance while maintaining node-inductive capabilities.

NBFNet, the foundation of GRUTL, shares similarities with RED-GNN in that it constructs pairwise representations by aggregating all path representations between a node pair, effectively forming a path-based subgraph representation.

2.1.2 NBFNet

The backbone model of GRUTL, which handles the graph convolution, can be seen as a specific instantiation of Neural Bellman-Ford Networks (NBFNet) [7], one of the state-of-the-art models for inductive link prediction with respect to node features. The generic NBFNet algorithm generates node representations based on a given query-head node h and query-relationship type q as follows:

$$h_v^{(0)} \leftarrow \text{INDICATOR}(u, v, q) \tag{2.1}$$

$$M_v^{(t)} \leftarrow \left\{ \text{MESSAGE}\left(h_x^{(t-1)}, \mathbf{w}_q(x, r, v)\right) \middle| (x, r, v) \in \mathcal{E}(v) \right\} \tag{2.2}$$

$$h_v^{(t)} \leftarrow \text{AGGREGATE}\left(M_v^{(t)} \cup \{h_v^{(0)}\}\right) \tag{2.3}$$

The INDICATOR function captures the boundary condition of the traditional Bellman-Ford algorithm. In practice, we often define it as follows:

$$\text{INDICATOR}(u, v, q) = \mathbb{1}(u = v) \cdot \mathbf{1}$$

This means that the head node h is initialized with ones while all other nodes receive a zero vector. With this approach, we guide the information flow from the head node outward to other nodes.

For updating the embedding of some node v at layer t, we apply the MESSAGE function to each outgoing edge $(x, r, v) \in \mathcal{E}(v)$, using the embedding of the previous layer and the edge embedding $\mathbf{w}_q(x, r, v)$. The aggregated result is combined with the boundary condition $h_v^{(0)}$.

There is a lot of flexibility in choosing implementations for AGGREGATE, MESSAGE, and INDICATOR. As demonstrated later, GRUTL utilizes DistMult [8] for the MESSAGE function and summation for the AGGREGATE function.

It is important to note that NBFNet is inductive on node types, meaning it can generate node representations for new nodes not seen in training. It achieves this by not learning fixed embeddings for the nodes in the graph but instead learning how to leverage the structural information of the graph to compute node embeddings. In other words, it learns an aggregation strategy rather than static node representations. By making the convolution process relative to the head node and relationship type, it is a powerful method for generating inductive node embeddings that can be used for various tasks.

2.1.3 ULTRA

A powerful adaptation of NBFNet by the same authors is ULTRA [9], designed for zero-shot link prediction in knowledge graphs. ULTRA demonstrated impressive results in inductive learning settings, particularly for knowledge graph completion tasks.

ULTRA extends the capabilities of NBFNet by introducing a two-step reasoning process:

1. It first applies the Neural Bellman-Ford algorithm on a **Graph of Relations** to obtain embeddings for relationship types.
2. It then applies the same algorithm on the actual knowledge graph, using the learned relation embeddings to guide the message passing.

This approach allows ULTRA to be inductive not only on entity types but also on relation types. Unlike standard transductive models, which fail to generalize to unseen nodes and relations, ULTRA can dynamically construct embeddings for new relations using its learned relational reasoning framework.

ULTRA's methodology was a key inspiration for our work, demonstrating the potential of transfer learning on graph-based models. Our goal was to adapt ULTRA's ideas to the domain of recommender systems, which we explore in later sections.

2.2 Cross Domain Recommendation

HeroGRAPH [10] is a multi-target cross-domain recommendation (MTCDR) model that constructs a large heterogeneous graph by merging multiple domains. Instead of explicitly modeling pairwise relations, HeroGRAPH aggregates user-item interactions across all domains in a shared graph. The model uses node aggregation and an attention mechanism to generate recommendation scores. However, it requires access to the target domain during training and assumes structural similarity between datasets, as evidenced by its evaluation solely on Amazon datasets.

Unlike traditional cross-domain methods that rely on overlapping users or extensive pre-processing, Krishnan et al. [11] trains a meta-layer to capture domain-invariant user-item interaction patterns and context-aware representations. However, this approach still requires partial access to the target domain during training.

Similarly, Zhao et al. [12] avoids direct overlap by learning a correspondence mapping between entities in the source and target domains. This mapping is then used within a transfer learning-based collaborative filtering model for predictions. Yet, target domain data is still included in training, limiting its applicability to unseen domains.

2.3 NLP-Based Transfer Learning in Recommender Systems

The model that makes the most extensive use of large language models (LLMs) in recommendation is Chat-Rec [13]. It converts user profiles and interaction histories into prompts, which are then processed by an LLM to generate recommendations. Since LLMs are trained on diverse domains, the authors claim that Chat-Rec should generalize well to new domains. However, this was not explicitly tested in a transfer learning setup; instead, all experiments were conducted end-to-end on MovieLens, limiting the evaluation of its cross-domain generalizability.

Ding et al. [14] and U-BERT [15] also leverage NLP-based embeddings for transfer learning in recommendation. Ding et al. builds a zero-shot recommendation system by encoding items as embeddings from textual or visual features and representing users through their interaction history. This allows the model to utilize sequential recommendation techniques. While it performs well in zero-shot settings, end-to-end models significantly outperform it.

U-BERT follows a pretrain-finetune paradigm, leveraging BERT-based user and review encoders learned from a text-rich source domain. During fine-tuning, it incorporates an item encoder and a review co-matching layer to connect user and item reviews, forming three key representations used for recommendations.

Both models depend on textual or visual features, making them inapplicable to domains lacking such structured inputs, thereby limiting their full transfer potential.

Chapter 3

Methodology

Recommender systems are commonly represented as bipartite graphs, where users and items are nodes, and interactions form edges. These interactions often include side information, such as ratings or timestamps. For instance, in the MovieLens dataset, users correspond to individuals who rate movies, while items represent the movies themselves.

As is standard in graph-based recommendation, we model this system as a bipartite graph where users are positioned on one side and items on the other. An edge between a user and an item exists if the user has interacted with the item in the past. The side information associated with these interactions is incorporated as edge features in the graph. Given this setup, our goal is to recommend new items to users by leveraging message passing in the graph, utilizing edge embeddings relative to the query user to compute a relevance score for each item.

For a new dataset, we first project its raw edge features into a unified edge embedding space using a shallow edge-projection MLP. This projection function is dataset-specific, allowing the model to handle raw edge features of arbitrary dimensions. The projected edge features are then passed through a deeper, shared edge-embedding MLP. These edge embeddings serve as inputs to the message passing mechanism in the graph, which is based on an adjusted version of NBFNet [7]. By initializing the projection MLP in a meaningful way, we can perform zero-shot recommendation with a pretrained backbone model.

3.1 Graph Propagation

More formally, given a user u and an interaction context (raw edge feature) r, we proceed as follows:

Initialization: Each node v is initialized with the standard NBFNet scheme:

$$h_v^{(0)} = \mathbb{1}(u = v) \tag{3.1}$$

This ensures that all information originates from the query user u. Moreover, let g denote the function that generates the edge embedding for a given raw edge feature r.

$$g(r) = \mathrm{MLP}_{\mathrm{emb}}\Big(\mathrm{MLP}_{\mathrm{proj}}(r)\Big).$$

This function incorporates both the dataset-specific projection and the backbone embedding MLPs.

Message Computation: At each layer t, the set of messages for node v is computed as:

$$M_v^{(t)} = \Big\{\mathrm{MESSAGE}\Big(h_x^{(t-1)}, \mathbf{w}_q(x, r, v)\Big) \mid (x, r, v) \in \mathcal{E}(v)\Big\}, \tag{3.2}$$

where the edge weight is defined as:

$$\mathbf{w}_q(x, r, v) = \mathrm{MLP}_t\big(g(r)\big).$$

Here, edge features r are transformed into a layer-specific embedding via MLP_t. The message function is implemented as a non-parametric *DistMult* operation [8].

Node Update: The node update is performed by aggregating messages together with the initial boundary condition:

$$h_v^{(t)} = \mathrm{UPDATE}\Big(h_v^{(t-1)},\ \mathrm{AGGREGATE}\Big(M_v^{(t)} \cup \{h_v^{(0)}\}\Big)\Big). \tag{3.3}$$

Summation is used as the aggregation function, followed by a linear transformation, layer normalization, and an activation function, as specified by the model's hyperparameters.

Score Generation: After the message passing procedure, we compute logit scores for all items v by applying a final MLP that projects the node embeddings from the final layer T, concatenated with the boundary condition, to a scalar value:

$$\mathrm{score}(u, q, v) = \mathrm{MLP}_{\mathrm{score}}\Big(\mathrm{concat}\big(h_v^{(T)}, h_v^{(0)}\big)\Big). \tag{3.4}$$

Crucially, our model does not learn node-specific parameters, ensuring node-inductive generalization. Unlike traditional embedding-based methods such as LightGCN [2], which precompute all node embeddings and then use simple similarity functions for recommendation, our method performs message passing dynamically for each query.

3.2 Runtime Analysis

In the initialization phase of GRUTL, we initialize all nodes and gather edge embeddings from the raw edge features. Assuming that the MLP operations run in constant time, this phase incurs a runtime cost of $O(|V| + |E|)$.

For each layer, we perform the following operations:

1. **Edge Projection:** We project the edge embeddings to a layer-specific embedding, incurring a cost of $O(|E|)$.
2. **Message Gathering:** For every node, we collect messages from its neighboring nodes. Since each edge is processed twice (once for each direction), this results in an additional cost of $O(|E|)$.

Thus, assuming that $|V| \leq |E|$, the total runtime per layer is $O(|E|)$, and for T layers, the overall cost is $O(T \cdot |E|)$.

Finally, computing the score for each node incurs a cost of $O(|V|)$. Therefore, the overall complexity of a forward pass in GRUTL is:

$$O\big((T+1) \cdot |E|\big)$$

3.2.1 Scalability

With a runtime complexity of $O\big((T+1) \cdot |E|\big)$, our model is reasonably scalable, as demonstrated by experiments on large graphs such as Yelp, which contains approximately 1 million edges. However, it is important to note that compared to traditional GNNs like LightGCN—which precompute node embeddings during training and achieve constant-time inference (or $O(|V|)$, depending on the implementation)—our method incurs a slightly higher cost at inference. This increased cost is a reasonable trade-off for the enhanced adaptability and transferability offered by our approach.

3.3 Training Procedure

To optimize our model, we remove easy edges (batch edges) from the graph during training. This ensures that the model learns from non-trivial paths and does not overfit to direct interactions.

Following the standard practice [16], we train the model by minimizing the cross-entropy loss over positive and negative triplets:

$$\mathcal{L} = -\log p(u, q, v) - \sum_{i=1}^{n} \frac{1}{n} \log(1 - p(u_i', q, v_i')), \tag{3.5}$$

where (u, q, v) is a positive interaction, and $\{(u'_i, q, v'_i)\}_{i=1}^{n}$ are negative samples obtained by corrupting either the head u or the tail v of the positive sample.

This loss encourages the model to assign higher probabilities to positive interactions while minimizing the likelihood of negative interactions. By training the model in this way, we ensure that it can generalize to unseen users and items, enabling zero-shot recommendation.

Chapter 4

Experiments

4.1 Datasets

Our experiments are conducted on several real-world datasets:

- **Yelp18**: A dataset containing user reviews and ratings of businesses. The dataset splits were obtained from Hugging Face[1], and the edge features were sourced from the RecBole dataset collection[2]. The current best-performing model for this dataset is NESCL [17].
- **MovieLens 1M**: A dataset of movie ratings provided by GroupLens[3], with the top benchmarked model being BPRFM [18].
- **Gowalla**: A location-based social networking dataset. The dataset splits were obtained from Hugging Face[4], and the edge features were sourced from the SNAP dataset collection [19]. The leading model for this dataset is NESCL [17].
- **LastFM**: A dataset of music listening events published by GroupLens[5], with the most competitive model being SLIM [20].
- **Epinions**: A product review social network sourced from the Trust dataset[6], where the strongest model to date is NGCF [21].
- **BookX**: A book review dataset available on Kaggle[7], for which BPRMF [18] presents the most effective approach.

[1] https://huggingface.co/datasets/reczoo/Yelp18_m1
[2] https://recbole.io/dataset_list.html
[3] https://grouplens.org/datasets/movielens/1m/
[4] https://huggingface.co/datasets/reczoo/Gowalla_m1/raw/main/train.txt
[5] https://grouplens.org/datasets/hetrec-2011/
[6] https://www.cse.msu.edu/~tangjili/trust.html
[7] https://www.kaggle.com/datasets/syedjaferk/book-crossing-dataset

- **Amazon**: A collection of datasets containing user interactions and reviews across various product domains (Fashion, Games, Men, Beauty). The datasets, along with edge features, were sourced from [22]. The best-performing models for these domains are as follows:
  - **Beauty**: CARCA [23]
  - **Fashion**: ProxyRCA [24]
  - **Men**: CARCA [23]
  - **Games**: ProxyRCA [24]

For ML-1M, BookX, LastFM, and Epinions the state-of-the-art models are reported in [25], which provides extensive benchmarks on recommender systems. We also used the codebase of that paper to obtain the dataset splits[8]. For the other datasets, the splits and evaluation settings were obtained according to the benchmark available on Papers with Code. For more details on preprocessing and hyperparameter tuning, please refer to the relevant sections in Appendix A.1 and Appendix A.2.

Table 4.1: Dataset statistics. As a side note, we report the average degrees, and density always refers to the overall graph density with respect to user-item graphs, meaning it is computed as $\frac{|E|}{|U|\cdot|I|}$ rather than $\frac{|E|}{n(n-1)}$.

Dataset	# Edges	# Users	# Items	UserDeg	ItemDeg	Density
Amazon Beauty Test	50498	52204	57289	1	1	1.00e-04
Amazon Beauty Train	293912	52204	57289	6	5	1.00e-04
Amazon Fashion Test	45184	45184	166270	1	0	4.77e-05
Amazon Fashion Train	267635	45184	166270	6	2	4.77e-05
Amazon Games Test	30901	31013	23715	1	1	4.00e-04
Amazon Games Train	225305	31013	23715	7	10	4.00e-04
Amazon Men Test	34244	34244	110636	1	0	6.73e-05
Amazon Men Train	186382	34244	110636	5	2	6.73e-05
BookX Test	88639	12720	18318	7	5	1.90e-03
BookX Train	276334	12720	18318	22	15	1.90e-03
Epinions Test	84884	21008	13887	4	6	1.50e-03
Epinions Train	266791	21008	13887	13	19	1.50e-03
Gowalla Test	217242	29858	70839	7	3	5.00e-04
Gowalla Train	712504	29858	70839	24	10	5.00e-04
LastFM Test	12596	1867	1530	7	8	2.20e-02
LastFM Train	39717	1867	1530	21	26	2.20e-02
ML-1M Test	114309	5950	2811	19	41	3.42e-02
ML-1M Train	364654	5950	2811	61	130	3.42e-02
Yelp18 Test	324147	31668	38048	10	9	1.30e-03
Yelp18 Train	1097007	31668	38048	35	29	1.30e-03

[8] https://github.com/recsys-benchmark/DaisyRec-v2.0/tree/dev

The general statistics are shown in Table 4.1. Notice that the graph structure varies considerably: we have very small, dense graphs like LastFM and larger, sparser graphs like Gowalla. The average user degree also varies significantly. In Table 4.2, the edge-attribute descriptions for each dataset are listed. Notably, there are substantial differences in their richness: some datasets, such as Epinions, include expressive interaction contexts, whereas others (e.g., BookX or LastFM) provide only basic attributes like ratings or listening counts

Table 4.2: Descriptions of edge attributes for each dataset

Dataset	**Edge Attribute Descriptions**
Epinions	category, rating, helpfulness, timestamp
LastFM	listening count
BookX	rating
ML-1M	rating and timestamp
Gowalla	date, longitude, latitude
Amazon Beauty	context (already processed)
Amazon Fashion	context (already processed)
Amazon Men	context (already processed)
Amazon Games	context (already processed)
Yelp18	rating, date, useful, funny, cool

4.1.1 Sequential Recommendation on Amazon Datasets

In the Amazon datasets, the recommendation task is generally approached from a sequential perspective, which is more time-aware than conventional recommendation. For instance, the model should learn that if a user purchases a razor, they might subsequently buy shaving foam. State-of-the-art models [24] employ multi-head attention layers to capture time-aware item sequences for each user.

In our approach, we used a different initialization technique in the graph convolution step for these datasets, where all head-nodes were initialized not with ones but with the context (edge features) of the query edge. Unfortunately, our model could not compete with time-aware recommendation methods. A potential improvement would be to include a separate multi-head attention layer that learns an embedding for the query-edge context, rather than relying solely on the edge-embedding MLP.

4.2 General Setting

The general experimental setting is as follows. We used the processed datasets to pretrain different single-graph and multi-graph checkpoints of our models. Subsequently, we evaluated all datasets using various evaluation modes: end-to-end, zero-shot, and fine-tuning with various checkpoints. In addition to evaluating our model, GRUTL, we also assessed a simple instantiation of NBFNet that relies solely on the structural information of the graphs. For more details on NBFNet, please refer to the corresponding section in Appendix A.3.

As evaluation metrics, we use the hit rate at K (Hits@K) and the normalized discounted cumulative gain at K (NDCG@K). Let n denote the number of users and define the function relevance(i_j, u) to be 1 if and only if user u interacted with the j-th recommended item i_j in the test set, and 0 otherwise.

$$\text{Hits@K} = \frac{1}{n}\sum_{u=1}^{n}\sum_{j=1}^{K}\text{relevance}(i_j, u) \tag{4.1}$$

$$\text{DCG@K} = \frac{1}{n}\sum_{u=1}^{n}\sum_{j=1}^{K}\frac{\text{relevance}(i_j, u)}{\log_2(j+1)} \tag{4.2}$$

$$\text{NDCG@K} = \frac{\text{DCG@K}}{\text{IDCG@K}} \tag{4.3}$$

Here, IDCG@K denotes the ideal discounted cumulative gain, computed with respect to the optimal ordering. Let u_i denote the number of relevant items for user u. Then,

$$\text{IDCG@K} = \frac{1}{n}\sum_{u=1}^{n}\sum_{j=1}^{\min(u_i,K)}\frac{1}{\log_2(j+1)} \tag{4.4}$$

It is important to note that Hits@K is binary in nature and measures whether a relevant item (i.e., an item with which the user has interacted) appears in the top K recommendations. In contrast, NDCG@K not only considers all relevant items for a user but also accounts for their ranking in the recommendation list.

4.3 Results

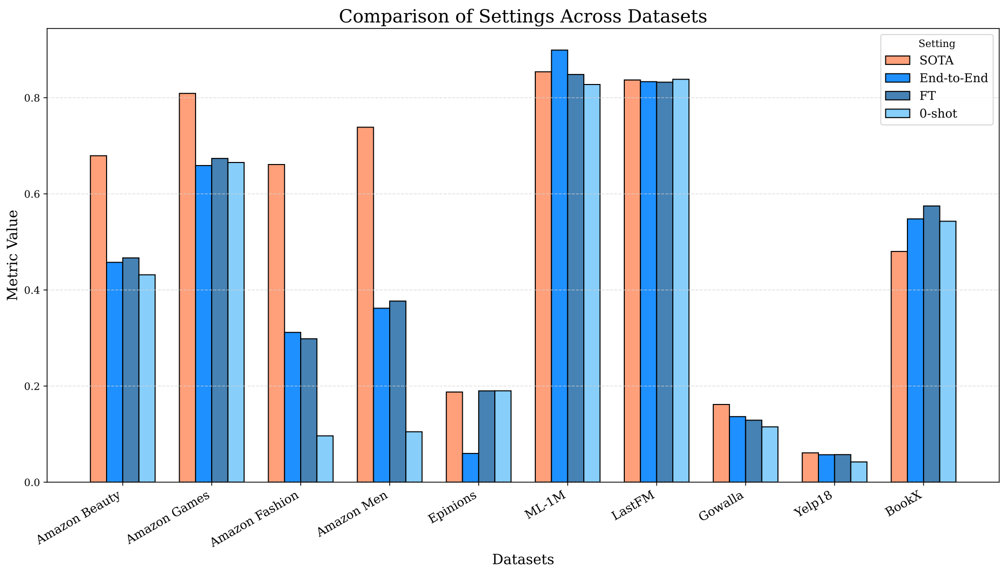


Figure 4.1: Comparison of different GRUTL model settings across various datasets, measured by Hits@10. For the Fine-Tuning and 0-shot settings, all datasets use the model pretrained on Epinions and Amazon Beauty.

Figure 4.1 provides a quick overview of the results. For the complete results in tabular form, please refer to Appendix A.4. We observe significant variation: on some datasets, the End-to-End setting achieves the highest performance, while on others, Fine-Tuning or 0-shot works best. Notably, for BookX, LastFM, and Epinions, our zero-shot model outperforms the SOTA baseline. This suggests that the previous SOTA models may not have been sufficiently expressive or well-optimized for these datasets. In general, Fine-Tuning provides a benefit over 0-shot, often matching or surpassing the End-to-End results. This strongly suggests that the model generalizes better when trained on diverse data. However, our model cannot compete with the SOTA methods on the Amazon datasets due to their sequential recommendation setup, as previously discussed. Furthermore, on Epinions, the model tends to overfit, which may be attributed to the richness of its edge features combined with a powerful edge embedding MLP. Adjusting hyperparameters could help mitigate this issue.

In 4.2, we further compare the performance of GRUTL with NBFnet. Although End-to-End NBFNet is often competitive with GRUTL, transfer learning (Fine-Tuning and 0-shot) tends to be more effective with GRUTL. This suggests that incorporating edge features helps the model capture transferable knowledge across datasets, albeit at a potential cost to end-to-end performance.

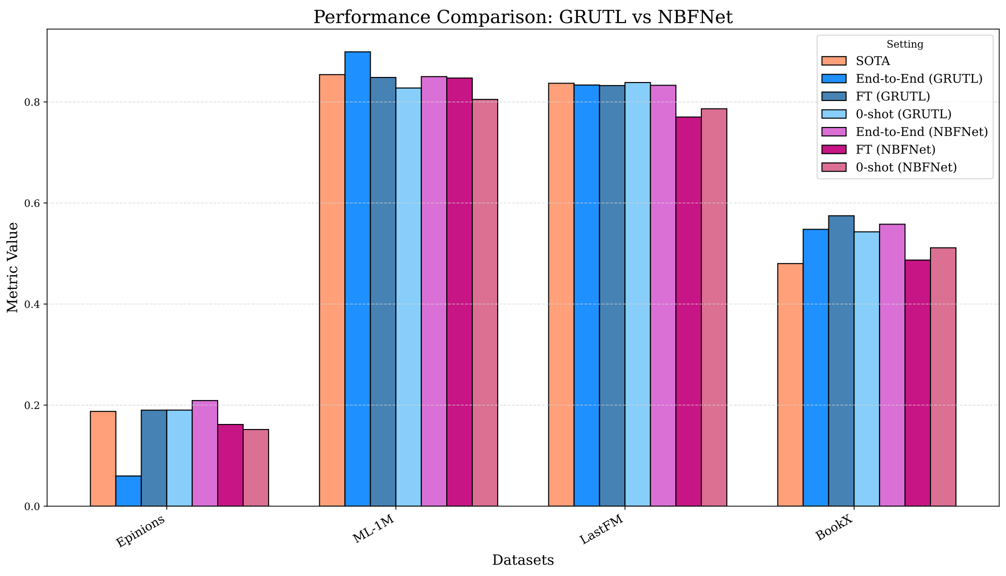


Figure 4.2: Evaluation of GRUTL and NBFNet across multiple datasets, comparing different training strategies measured by Hits@10. The Fine-Tuning and 0-shot models are initialized with pretraining on Epinions and Amazon Beauty.

Table 4.3: Performance comparison for BookX models. The best results in each column are highlighted in bold. For GRUTL, we used the Fine-Tuned model with a pretrained checkpoint on Beauty and Epinions, while for NBFNet, we used the End-to-End model.

Model	Hits@10	NDCG@10
MostPop	0.1081	0.063
ItemKNN	0.278	0.1921
PureSVD	0.474	**0.3216**
SLIM	0.3475	0.2163
BPRMF	0.4801	0.2938
BPRFM	0.4474	0.2719
NeuMF	0.4582	0.2791
NFM	0.345	0.1882
NGCF	0.3647	0.2018
Multi-VAE	0.3867	0.207
GRUTL	**0.575**	0.25
NBFnet	0.558	0.238

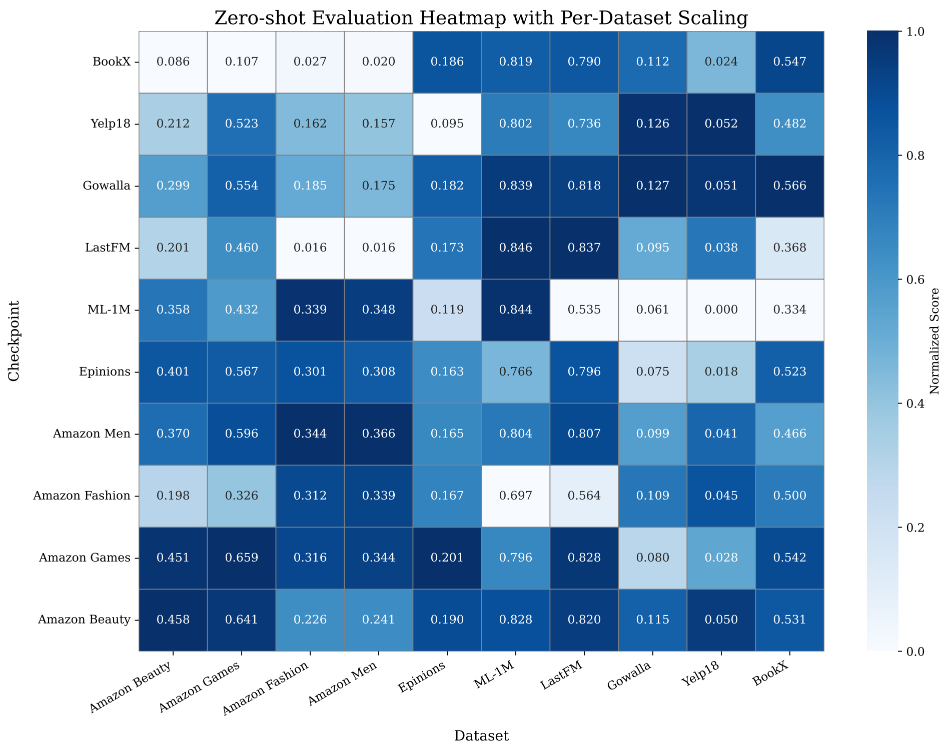
We observe from Table 4.3 that GRUTL and NBFNet exhibit a larger discrepancy between the Hits and NDCG metrics compared to the other models (MostPop [26], ItemKNN [26], PureSVD [27], SLIM [20], BPRFM [18], BPRFM [28], NeuMF [29], NFM [29], NGCF [21], Multi-VAE [30]). A similar observation can be made from Table 4.4, where our models are not as competitive in the NDCG metric as they are in the Hits metric.

Table 4.4: Performance comparison for different settings of Multi-Graph Checkpoints on ML-1M. The best results per column are highlighted in bold. F.T. stands for fine-tuned

Setting	Hits@10	NDCG@10	F.T. Hits@10	F.T. NDCG@10
SOTA	0.854	**0.647**	0.854	**0.647**
End-to-End	**0.899**	0.429	**0.899**	0.429
Ckpt: Epi-AB-M1	0.489	0.192	0.812	0.354
Ckpt: AB-AG	0.811	0.324	0.862	0.409
Ckpt: AB-Epi-Bx	0.805	0.325	0.871	0.385
Ckpt: AB-Epi-Bx-AG	0.822	0.355	0.856	0.381
Ckpt: AB-Epi	0.828	0.395	0.848	0.407
Ckpt: AM-Epi-Gw-Bx	0.825	0.345	0.833	0.354

This suggests that GRUTL and NBFNet excel at ranking at least one relevant item within the top 10, but may be less effective at identifying and ranking all relevant items for a user. One possible explanation is that this behavior could be inherent to link prediction models in recommender systems, although further investigation is needed. Nonetheless, even for the NDCG metric, GRUTL still outperforms many baseline models.

In Table 4.4 and Figure 4.3, we observe a substantial discrepancy among different checkpoints. While some checkpoints appear to perform better overall, no single checkpoint consistently outperforms the others across all datasets. Notably, as shown in Table 4.4, adding more graphs to a checkpoint does not necessarily lead to improved performance. For example, the checkpoint based on Amazon Beauty and Epinions performs better in the 0-shot setting on ML-1M; however, when BookX (which also transfers well to ML-1M, as seen in Figure 4.3) is included, the 0-shot performance declines. Conversely, in the fine-tuning setting, the checkpoint that includes Amazon Beauty, Epinions, and BookX outperforms the one with only Amazon Beauty and Epinions. Overall, we deem the Amazon Beauty and Epinions checkpoint to be the best, although the performance of a checkpoint heavily depends on the evaluation setting (0-shot vs. fine-tuning) and the specific dataset.

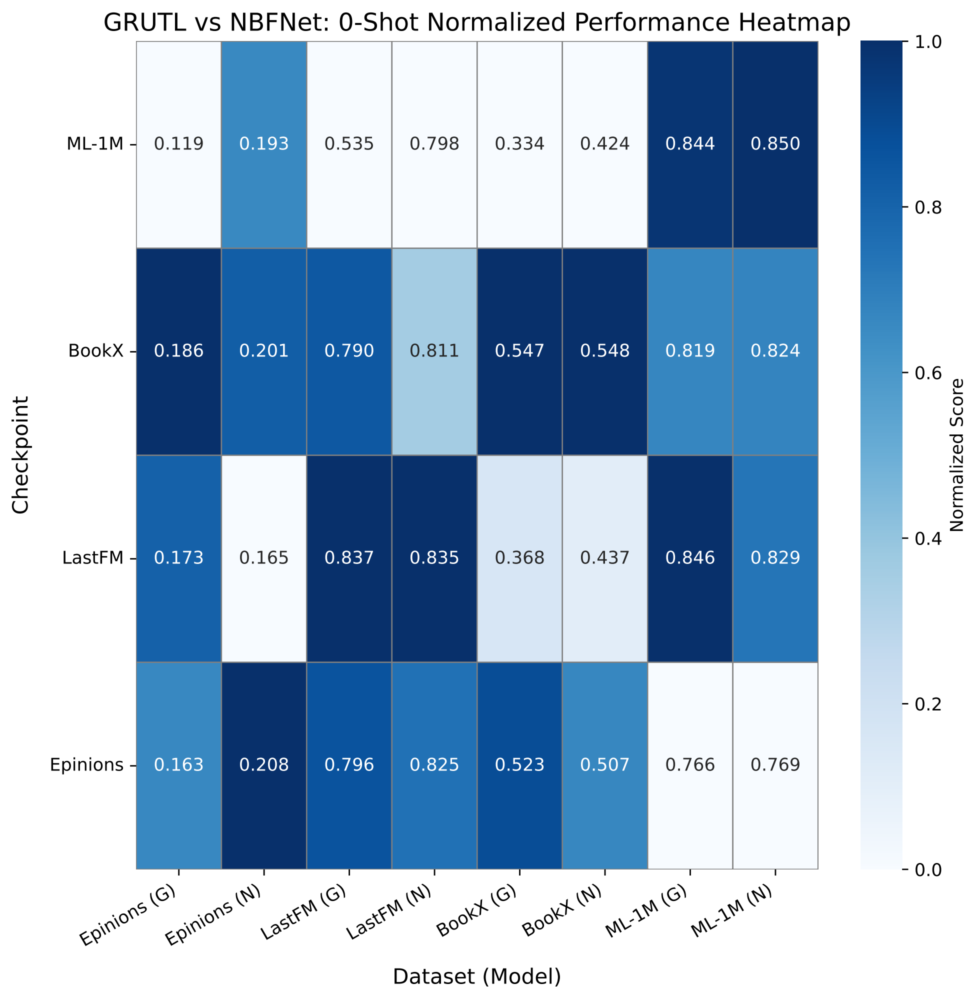


Checkpoint	Amazon Beauty	Amazon Games	Amazon Fashion	Amazon Men	Epinions	ML-1M	LastFM	Gowalla	Yelp18	BookX
BookX	0.086	0.107	0.027	0.020	0.186	0.819	0.790	0.112	0.024	0.547
Yelp18	0.212	0.523	0.162	0.157	0.095	0.802	0.736	0.126	0.052	0.482
Gowalla	0.299	0.554	0.185	0.175	0.182	0.839	0.818	0.127	0.051	0.566
LastFM	0.201	0.460	0.016	0.016	0.173	0.846	0.837	0.095	0.038	0.368
ML-1M	0.358	0.432	0.339	0.348	0.119	0.844	0.535	0.061	0.000	0.334
Epinions	0.401	0.567	0.301	0.308	0.163	0.766	0.796	0.075	0.018	0.523
Amazon Men	0.370	0.596	0.344	0.366	0.165	0.804	0.807	0.099	0.041	0.466
Amazon Fashion	0.198	0.326	0.312	0.339	0.167	0.697	0.564	0.109	0.045	0.500
Amazon Games	0.451	0.659	0.316	0.344	0.201	0.796	0.828	0.080	0.028	0.542
Amazon Beauty	0.458	0.641	0.226	0.241	0.190	0.828	0.820	0.115	0.050	0.531

Figure 4.3: Zero-shot evaluation heatmap with per-dataset scaling. Results are measured in NDCG@20 for Yelp18 and Gowalla, and Hits@10 for all other datasets. The color intensity is adjusted within each dataset to highlight relative performance differences.

In Figure 4.3, we further analyze dataset-to-dataset transfers. Interestingly, the diagonal does not always yield the best performance—for example, on BookX or Amazon Fashion. This suggests that the model performs better when trained on a different dataset and evaluated in a zero-shot setting on this dataset, rather than using its own checkpoint. It is important to note that "its own checkpoint" refers not to the full model checkpoint but only to the checkpoint of the backbone model. Similar results were also observed in Figure 4.1, where the End-to-End trained model was outperformed in a zero-shot setting on LastFM.

We observe that sparse edge features, as seen in BookX and LastFM, hinder overall transferability. However, having rich edge features does not guarantee successful transfer; for example, Yelp does not transfer well, which may be related to its graph size. In contrast, Gowalla, despite being a relatively large graph, transfers effectively.



	Epinions (G)	Epinions (N)	LastFM (G)	LastFM (N)	BookX (G)	BookX (N)	ML-1M (G)	ML-1M (N)
ML-1M	0.119	0.193	0.535	0.798	0.334	0.424	0.844	0.850
BookX	0.186	0.201	0.790	0.811	0.547	0.548	0.819	0.824
LastFM	0.173	0.165	0.837	0.835	0.368	0.437	0.846	0.829
Epinions	0.163	0.208	0.796	0.825	0.523	0.507	0.766	0.769

Figure 4.4: Comparison of GRUTL and NBFNet in a 0-shot setting across multiple datasets and checkpoints. Each dataset's results for GRUTL (G) and NBFNet (N) are displayed side by side. Scores are normalized per dataset to highlight relative performance differences

It is also noteworthy that transfer is not always symmetric. For instance, LastFM transfers well to MovieLens, whereas ML-1M does not transfer effectively to LastFM—even though both graphs are dense—likely due to substantial differences in graph size.

In Figure 4.4, we analyze the impact of using only structural information (as in NBFNet) versus additionally incorporating edge features (as in GRUTL) on dataset-to-dataset 0- shot transfer performance. First, we observe that NBFNet is highly competitive and, in general, slightly outperforms GRUTL in terms of absolute values in the 0-shot transfer setting. However, to fully leverage GRUTL's potential, it is necessary to employ multi-graph checkpoints—this is also where GRUTL outperforms NBFNet. Notably, the LastFM checkpoint performs better for GRUTL than for NBFNet, despite LastFM lacking rich edge features. This suggests that even less expressive edge features can enhance the model's generalization. Overall, the checkpoints tend to yield better transfer performance with GRUTL when using edge features, with the only exception being the Epinions checkpoint, where overfitting may be a factor. For the complete results in tabular form, please refer to Appendix A.1.

Chapter 5

Conclusion

Graph-based recommendation datasets naturally share a common structure by representing interactions as bipartite user-item graphs. In this work, we have clearly demonstrated that this shared structure can be effectively exploited. We designed GRUTL—a model that leverages both the consistent graph topology and the interaction features inherent in nearly every user-item interaction. Our experiments show that when pretrained and applied in a transfer learning setting, GRUTL can outperform state-of-the-art models in a zero-shot scenario on several datasets, with fine-tuning further boosting performance. This approach is not only resource-efficient but also better aligned with the dynamic nature of real-world recommender systems, where new items and users are introduced daily, rendering a strictly end-to-end recommender system unsuitable.

While GRUTL exhibits strong generalization capabilities, we acknowledge that its transfer performance is not universally optimal. For instance, in sequential recommendation settings, our model did not perform as well as specialized approaches. Nevertheless, our findings indicate that with the appropriate architecture, it is possible to generalize across diverse datasets—sometimes even surpassing the performance of end-to-end models trained on individual datasets. These promising results open avenues for future research, such as adapting other inductive graph models to the recommender setting and further enhancing performance in dynamic and sequential recommendation scenarios.

Bibliography

[1] F. Zhu, Y. Wang, C. Chen, J. Zhou, L. Li, and G. Liu, "Cross-domain recommendation: challenges, progress, and prospects," *arXiv preprint arXiv:2103.01696*, 2021.

[2] 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*, 2020, pp. 639–648.

[3] R. Das, S. Dhuliawala, M. Zaheer, L. Vilnis, I. Durugkar, A. Krishnamurthy, A. Smola, and A. McCallum, "Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning," *arXiv preprint arXiv:1711.05851*, 2017.

[4] Z. Zhu, X. Yuan, M. Galkin, L.-P. Xhonneux, M. Zhang, M. Gazeau, and J. Tang, "A* net: A scalable path-based reasoning approach for knowledge graphs," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[5] Y. Zhang and Q. Yao, "Knowledge graph reasoning with relational digraph," in *Proceedings of the ACM web conference 2022*, 2022, pp. 912–924.

[6] K. Teru, E. Denis, and W. Hamilton, "Inductive relation prediction by subgraph reasoning," in *International Conference on Machine Learning*. PMLR, 2020, pp. 9448–9457.

[7] Z. Zhu, Z. Zhang, L.-P. Xhonneux, and J. Tang, "Neural bellman-ford networks: A general graph neural network framework for link prediction," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[8] B. Yang, W.-t. Yih, X. He, J. Gao, and L. Deng, "Embedding entities and relations for learning and inference in knowledge bases," *arXiv preprint arXiv:1412.6575*, 2014.

[9] M. Galkin, X. Yuan, H. Mostafa, J. Tang, and Z. Zhu, "Towards foundation models for knowledge graph reasoning," in *The Twelfth International Conference on Learning Representations*, 2024. [Online]. Available: https://openreview.net/forum?id=jVEoydFOl9

[10] Q. Cui, T. Wei, Y. Zhang, and Q. Zhang, "Herograph: A heterogeneous graph framework for multi-target cross-domain recommendation." in *ORSUM@ RecSys*, 2020.

[11] A. Krishnan, M. Das, M. Bendre, H. Yang, and H. Sundaram, "Transfer learning via contextual invariants for one-to-many cross-domain recommendation," in *Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval*, 2020, pp. 1081–1090.

[12] L. Zhao, S. J. Pan, and Q. Yang, "A unified framework of active transfer learning for cross-system recommendation," *Artificial Intelligence*, vol. 245, pp. 38–55, 2017.

[13] Y. Gao, T. Sheng, Y. Xiang, Y. Xiong, H. Wang, and J. Zhang, "Chat-rec: Towards interactive and explainable llms-augmented recommender system," *arXiv preprint arXiv:2303.14524*, 2023.

[14] H. Ding, Y. Ma, A. Deoras, Y. Wang, and H. Wang, "Zero-shot recommender systems," *arXiv preprint arXiv:2105.08318*, 2021.

[15] Z. Qiu, X. Wu, J. Gao, and W. Fan, "U-bert: Pre-training user representations for improved recommendation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 5, 2021, pp. 4320–4327.

[16] Z. Zhu, C. Shi, Z. Zhang, S. Liu, M. Xu, X. Yuan, Y. Zhang, J. Chen, H. Cai, J. Lu *et al.*, "Torchdrug: A powerful and flexible machine learning platform for drug discovery," *arXiv preprint arXiv:2202.08320*, 2022.

[17] P. Sun, L. Wu, K. Zhang, X. Chen, and M. Wang, "Neighborhood-enhanced supervised contrastive learning for collaborative filtering," *IEEE Transactions on Knowledge and Data Engineering*, vol. 36, no. 5, pp. 2069–2081, 2023.

[18] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "Bpr: Bayesian personalized ranking from implicit feedback," *arXiv preprint arXiv:1205.2618*, 2012.

[19] J. Leskovec and A. Krevl, "SNAP Datasets: Stanford large network dataset collection," http://snap.stanford.edu/data, Jun. 2014.

[20] X. Ning and G. Karypis, "Slim: Sparse linear methods for top-n recommender systems," in *2011 IEEE 11th international conference on data mining*. IEEE, 2011, pp. 497–506.

[21] X. Wang, X. He, M. Wang, F. Feng, and T.-S. Chua, "Neural graph collaborative filtering," in *Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval*, 2019, pp. 165–174.

[22] A. Rashed, S. Elsayed, and L. Schmidt-Thieme, "Context and attribute-aware sequential recommendation via cross-attention," in *Proceedings of the 16th ACM conference on recommender systems*, 2022, pp. 71–80.

[23] A. Lopez-Avila, J. Du, A. Shimary, and Z. Li, "Positional encoding is not the same as context: A study on positional encoding for sequential recommendation," *arXiv preprint arXiv:2405.10436*, 2024.

[24] J. Seol, M. Gang, S.-g. Lee, and J. Park, "Proxy-based item representation for attribute and context-aware recommendation," in *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, 2024, pp. 616–625.

[25] Z. Sun, H. Fang, J. Yang, X. Qu, H. Liu, D. Yu, Y.-S. Ong, and J. Zhang, "Daisyrec 2.0: Benchmarking recommendation for rigorous evaluation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 7, pp. 8206–8226, 2022.

[26] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *Proceedings of the 10th international conference on World Wide Web*, 2001, pp. 285–295.

[27] P. Cremonesi, Y. Koren, and R. Turrin, "Performance of recommender algorithms on top-n recommendation tasks," in *Proceedings of the fourth ACM conference on Recommender systems*, 2010, pp. 39–46.

[28] S. Rendle, "Factorization machines," in *2010 IEEE International conference on data mining.* IEEE, 2010, pp. 995–1000.

[29] X. He and T.-S. Chua, "Neural factorization machines for sparse predictive analytics," in *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*, 2017, pp. 355–364.

[30] D. Liang, R. G. Krishnan, M. D. Hoffman, and T. Jebara, "Variational autoencoders for collaborative filtering," in *Proceedings of the 2018 world wide web conference*, 2018, pp. 689–698.

Appendix A

Experiments

A.1 Preprocessing

For the ML-1M, LastFM, Epinions, and BookX datasets, we applied the 10-filter preprocessing procedure from [25]. This procedure discards all items with fewer than 10 interactions. Additionally, for ML-1M, we retained only those edges where a user rated a movie with at least 4 (on a 1-to-5 scale).

For all datasets, numerical attributes were scaled to have a mean of 0 and a standard deviation of 1. Categorical features were processed using one-hot encoding, and date features were transformed into numerical values by taking the earliest date as a reference and computing the time elapsed since that date in days (and, if necessary, in seconds).

A.2 Hyperparameters

To determine the optimal hyperparameters, we performed a search over 100 runs, optimizing the validation NDCG for the multi-graph Amazon Beauty and Games datasets. Although this search did not directly include transfer learning, the model's performance improved in the transfer learning setting with these hyperparameters, and we deemed them sufficient. A more extensive hyperparameter search could be conducted in future work.

A.3 NBFNet Instantiation

In this model, the graph is assumed to have two types of directed edges: one for each direction (i.e., user-to-item and item-to-user edges). Given a user u and an edge type r, the difference compared to GRUTL lies in how the edge weight is defined in the message computation step:

$$\mathbf{w}_q(x, r, v) = \text{Embedding}_t(r).$$

This formulation implies that the model learns a layer-specific embedding for the two edge types, rather than using transformed edge feature embeddings as GRUTL does. The edge weight is then used in the message computation step as follows:

$$M_v^{(t)} = \left\{ \text{MESSAGE}\big(h_x^{(t-1)}, \mathbf{w}_q(x, r, v)\big) \mid (x, r, v) \in \mathcal{E}(v) \right\}. \tag{A.1}$$

The remainder of the model is analogous to GRUTL; that is, we use the same message function (DistMult), aggregation method (sum), and score generation procedure.

A.4 All Results

Table A.1: All results for Epinions. Metrics are computed on a per-user basis rather than a per-edge basis.

Model	Setting	Ckpt	Hits@1	Hits@3	Hits@10	NDCG@10
GRUTL	0-shot	AB-Epi	0.058	0.107	0.190	0.052
GRUTL	0-shot (best)	AB	0.056	0.111	0.200	0.054
GRUTL	End-to-End	-	0.012	0.027	0.060	0.012
GRUTL	FT	AB-Epi	0.052	0.104	0.190	0.052
GRUTL	FT (best)	LF	0.058	0.114	0.199	0.054
NBFnet	0-shot	AB-Epi	0.048	0.088	0.152	0.043
NBFnet	End-to-End	-	**0.061**	**0.117**	**0.209**	0.057
NBFnet	FT	AB-Epi	0.048	0.089	0.162	0.041
NGCF	End-to-End	-	-	-	0.188	**0.105**

Table A.2: All results for BookX. Metrics are computed on a per-user basis rather than a per-edge basis.

Model	Setting	Ckpt	Hits@1	Hits@3	Hits@10	NDCG@10
GRUTL	0-shot	AB-Epi	0.259	0.392	0.543	0.233
GRUTL	0-shot (best)	AB	0.269	0.408	0.566	0.246
GRUTL	End-to-End	-	0.259	0.402	0.548	0.234
GRUTL	FT	AB-Epi	**0.272**	**0.417**	**0.575**	0.250
GRUTL	FT (best)	AB-Epi	0.269	0.415	0.575	0.249
NBFnet	0-shot	AB-Epi	0.223	0.355	0.511	0.209
NBFnet	End-to-End	-	0.260	0.399	0.558	0.238
NBFnet	FT	AB-Epi	0.230	0.348	0.487	0.196
BPRMF	End-to-End	-	-	-	0.480	**0.298**

Table A.3: All results for ML-1M. Metrics are computed on a per-user basis rather than a per-edge basis.

Model	Setting	Ckpt	Hits@1	Hits@3	Hits@10	NDCG@10
GRUTL	0-shot	AB-Epi	0.444	0.671	0.828	0.395
GRUTL	0-shot (best)	AF	0.357	0.642	0.831	0.362
GRUTL	End-to-End	-	**0.489**	**0.733**	**0.899**	0.429
GRUTL	FT	AB-Epi	0.470	0.674	0.848	0.407
GRUTL	FT (best)	Gw	0.408	0.691	0.886	0.393
NBFnet	0-shot	AB-Epi	0.385	0.601	0.805	0.354
NBFnet	End-to-End	-	0.445	0.677	0.850	0.385
NBFnet	FT	AB-Epi	0.424	0.652	0.847	0.371
BPRFM	End-to-End	-	-	-	0.854	**0.647**

Table A.4: All results for LastFM. Metrics are computed on a per-user basis rather than a per-edge basis.

Model	Setting	Ckpt	Hits@1	Hits@3	Hits@10	NDCG@10
GRUTL	0-shot	AB-Epi	**0.430**	0.634	0.838	0.318
GRUTL	0-shot (best)	AG	0.389	0.632	**0.839**	0.312
GRUTL	End-to-End	-	0.416	0.626	0.833	0.314
GRUTL	FT	AB-Epi	0.421	0.629	0.832	0.317
GRUTL	FT (best)	AM-Epi-Gw-Bx	0.422	**0.639**	0.836	0.321
NBFnet	0-shot	AB-Epi	0.324	0.557	0.786	0.260
NBFnet	End-to-End	-	0.403	0.632	0.833	0.310
NBFnet	FT	AB-Epi	0.282	0.496	0.770	0.246
SLIM	End-to-End	-	-	-	0.837	**0.602**

Table A.5: All results for Yelp. Metrics are computed on a per-user basis rather than a per-edge basis.

Model	Setting	Ckpt	Hits@1	Hits@3	Hits@10	NDCG@20
GRUTL	0-shot	AB-Epi	0.044	0.109	0.250	0.042
GRUTL	0-shot (best)	AB-Epi-Bx-AG	0.046	0.114	0.261	0.052
GRUTL	End-to-End	-	0.053	0.129	0.284	0.057
GRUTL	FT	AB-Epi	0.054	0.126	0.280	0.057
GRUTL	FT (best)	AB	**0.056**	**0.134**	**0.292**	0.060
NESCL	End-to-End	-	-	-	-	**0.061**

Table A.6: All results for Gowalla. Metrics are computed on a per-user basis rather than a per-edge basis.

Model	Setting	Ckpt	Hits@1	Hits@3	Hits@10	NDCG@20
GRUTL	0-shot	AB-Epi	0.121	0.231	0.410	0.115
GRUTL	0-shot (best)	AM-Epi-Gw-Bx	0.138	0.256	0.445	0.134
GRUTL	End-to-End	-	**0.160**	**0.292**	**0.477**	0.136
GRUTL	FT	AB-Epi	0.124	0.238	0.429	0.129
GRUTL	FT (best)	M1	0.138	0.261	0.456	0.137
NESCL	End-to-End	-	-	-	-	**0.162**

Table A.7: All results on Amazon Beauty. Metrics are computed on a per-user basis rather than a per-edge basis.

Model	Setting	Ckpt	Hits@1	Hits@3	Hits@10	NDCG@10
GRUTL	0-shot	AB-Epi	0.182	0.292	0.432	0.296
GRUTL	0-shot (best)	AG	0.180	0.296	0.454	0.304
GRUTL	End-to-End	-	0.184	0.302	0.457	0.310
GRUTL	FT	AB-Epi	**0.188**	**0.307**	0.467	0.314
GRUTL	FT (best)	AB-Epi-Bx	0.184	0.303	0.467	0.312
CARCA	End-to-End	-	-	-	**0.679**	**0.449**

Table A.8: All results on Amazon Fashion. Metrics are computed on a per-user basis rather than a per-edge basis.

Model	Setting	Ckpt	Hits@1	Hits@3	Hits@10	NDCG@10
GRUTL	0-shot	AB-Epi	0.053	0.072	0.096	0.073
GRUTL	0-shot (best)	AB-Epi-Bx	0.131	0.214	0.329	0.220
GRUTL	End-to-End	-	**0.153**	**0.249**	0.312	0.272
GRUTL	FT	AB-Epi	0.150	0.244	0.298	0.269
GRUTL	FT (best)	Yp	0.131	0.229	0.360	0.241
ProxyRCA	End-to-End	-	-	-	**0.661**	**0.446**

Table A.9: All results for Amazon Games. Metrics are computed on a per-user basis rather than a per-edge basis.

Model	Setting	Ckpt	Hits@1	Hits@3	Hits@10	NDCG@10
GRUTL	0-shot	AB-Epi	0.290	0.473	0.665	0.466
GRUTL	0-shot (best)	AB-Epi	0.290	0.473	0.665	0.466
GRUTL	End-to-End	-	0.287	0.468	0.659	0.462
GRUTL	FT	AB-Epi	**0.296**	**0.480**	0.674	0.474
GRUTL	FT (best)	AB-Epi	0.296	0.480	0.674	0.474
ProxyRCA	End-to-End	-	-	-	**0.809**	**0.611**

Table A.10: All results for Amazon Men. Metrics are computed on a per-user basis rather than a per-edge basis.

Model	Setting	Ckpt	Hits@1	Hits@3	Hits@10	NDCG@10
GRUTL	0-shot	AB-Epi	0.051	0.074	0.105	0.076
GRUTL	0-shot (best)	AF	0.135	0.237	0.373	0.249
GRUTL	End-to-End	-	**0.147**	**0.255**	0.362	**0.268**
GRUTL	FT	AB-Epi	0.145	0.253	0.377	0.264
GRUTL	FT (best)	M1	0.135	0.246	0.390	0.253
CARCA	End-to-End	-	-	-	**0.739**	-