Spring 2023

RelationalNet using Graph Neural Networks for Social Recommendations

Dharahas Tallapally
San Jose State University

RelationalNet using Graph Neural Networks for Social Recommendations

A Project

Presented to

The Faculty of the Department of Computer Science

San José State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

by

Dharahas Tallapally

May 2023

The Designated Project Committee Approves the Project Titled

RelationalNet using Graph Neural Networks for Social Recommendations

by

Dharahas Tallapally

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

SAN JOSÉ STATE UNIVERSITY

May 2023

Dr. Katerina Potika	Department of Computer Science
Dr. Magdalini Eirinaki	Department of Computer Engineering
Dr. Genya Ishigaki	Department of Computer Science

ABSTRACT

RelationalNet using Graph Neural Networks for Social Recommendations

by Dharahas Tallapally

Traditional recommender systems create models that can predict user interests based on the user-item relationships. However, these systems often have limited performance due to sparse user behavior data. To address this challenge, researchers are now exploring models for social recommendation that can account for both user-user and user-item relationships based on social networks, and user past behavior, respectively. These models aim to understand each user's behavior by considering their trusted neighbors and their influence on each other. Specifically, the potential embedding of each user is influenced by their trusted neighbors, who are, in turn, influenced by their own trusted neighbors and social connections. Users' interests evolve over time as social influence propagates recursively in the social network.

In this project, we propose the RelationalNet model, which creates graphs not only for the user-item and user-user relationships but also for the item-item relationships. We learn the user interest predictions by using Graph Neural Networks(GNNs). By incorporating social influence into recommendation models, we can capture more accurate user interests, especially when traditional methods fall short due to data sparsity. Such models improve the accuracy and effectiveness of recommendation systems by leveraging social connections and interactions. Moreover, we perform experiments that demonstrate that incorporating the item graph information into GNNs produces better results compared to the current cutting-edge social recommendation models.[1, 2, 3, 4].

ACKNOWLEDGMENTS

I would like to express my deepest gratitude and appreciation to Prof. Katerina Potika, whose guidance, support, and encouragement have made this project possible. My deep gratitude also goes to Prof. Magdalini Eirinaki and Prof. Genya Ishigaki for being on my defense committee. I would like to thank them for their time and efforts.

I would also like to thank my professors who have been instrumental in shaping my academic and intellectual journey. Their guidance and mentorship have been invaluable in shaping my research interests and methodologies.

Furthermore, I would like to extend my sincere gratitude to my friends whose encouragement, support, and feedback have been invaluable throughout this project. Their insights and perspectives have helped me refine my ideas and strengthen my arguments.

Finally, I would like to acknowledge the invaluable contributions of all those individuals who have directly or indirectly supported me in completing this project. Their guidance, support, and encouragement have been essential to my success, and I am deeply grateful for their contributions.

TABLE OF CONTENTS

CHAPTER

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

Introduction

With the tremendous increase in digital data, recommender systems provide a unique experience to users for personalizing content and services. Recommender systems (RS) play a crucial role in numerous domains for providing personalized recommendations to individual users, such as product recommendations in e-commerce platforms (e.g., Amazon, Walmart, Target), curated playlists in streaming platforms (e.g., YouTube, Spotify), targeted ads in online advertising and many others. The primary purpose of recommender systems is to recommend a product or service to a user, and recommender systems do this by consuming the users' historical data to find patterns, learn users' preferences, and predict the likelihood of the user liking the product or service. Users' historical data is available in many formats, e.g., purchase history, travel locations, likes and followers in the social app, rating and reviews of products, and browsing activity. Recommender systems analyze a specific part of users' data based on the goal of recommendations through model training. Researchers have published many research papers on recommender systems in the past decade. Still, with the latest technologies and advances, researchers are learning new techniques and ideas to optimize and provide better recommendations to users. Graph Neural Network(GNN) models have recently gained much traction in social recommendations. The later part of the project discusses a new approach for a recommender systems model using GNN for social recommendations.

To tackle the recommendation problem, it is often necessary to forecast ratings for items a user has yet to encounter. One approach to accomplish this is to estimate a user's possible ratings for all unrated items and suggest the ones expected to receive the highest ratings. A simple example of a recommender system with five users and

five items is provided in Table 1. The table represents the rating profiles of each user, where each row gives a user's ratings for a subset of available Songs. User preferences for songs are quantified using discrete numerical values ranging from 1-5, with 5 representing a high liking for the corresponding item. It should be noted that these ratings are subjective and may be interpreted differently by different users. Our process for recommending songs to Alice involves initially approximating her prospective ratings for the unrated songs and then proposing the songs that are projected to receive the highest ratings.

	Song-1	Song-2	Song-3	Song-4	Song-5
Alice	5	?	3	?	?
Bob			4	4	
John		3		2	1
Mike	3		3		2
Wayne	5	3		4	2

Table 1: Example recommender systems' rating table for user-items.

The example given above uses explicit ratings, which are ratings that users directly provide for songs. These ratings are considered a strong source of information because users can precisely convey their feelings about a specific item. On the other hand, implicit ratings refer to ratings that are deduced by analyzing a user's actions and behavior. A case in point of an implicit rating could be deducing a user's interest in an item by the length of time they spend reading its online description, which might suggest that they view it as valuable. However, this inference can sometimes be incorrect, as the user may have been interrupted or distracted. Implicit ratings can be collected from various sources, including usage frequency, bookmarking, and printing. It's worth noting that our research work in this project does not differentiate between explicit and implicit ratings. Nonetheless, the data sets we used for our experiments only included explicit ratings. In future studies, we may further explore using implicit

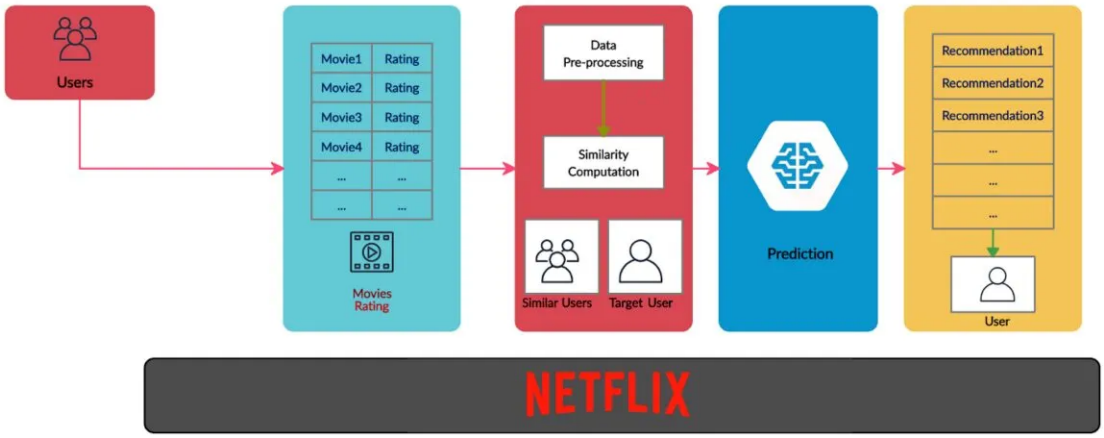

Figure 1: Example top-level pipeline of Netflix recommender system[5]

ratings to enhance recommendation models.

In the pursuit of providing enhanced recommendations to users, researchers have proposed and developed various variations of recommender systems. One prevalent approach is item-based neighborhood methods, which recommend items to users based on their similarity to previously interacted items. These methods directly leverage the historical item-user interactions to identify items that exhibit similarities, thereby facilitating personalized recommendations.

Another prominent direction in recommender systems is representation learning-based methods, which aim to encode both users and items as continuous vectors or embeddings in a shared space. By mapping users and items to a common latent space, these models enable direct comparisons and similarity computations. The popularity of representation-based models surged after the Netflix Prize competition [7], where matrix factorization models showcased their superiority over classic neighborhood

methods in terms of recommendation quality.

In recent years, deep learning models have emerged as the dominant methodology for recommender systems, garnering substantial interest in both academic research and industrial applications. The allure of deep learning lies in its ability to capture complex and non-linear relationships between users and items. By leveraging sophisticated neural architectures, deep learning models excel at capturing intricate user-item dynamics, thus enhancing the accuracy and relevance of recommendations. Additionally, these models offer the flexibility to seamlessly integrate diverse data sources, such as contextual information, textual data, and visual cues, thereby enriching the recommendation process with a wealth of information.

The integration of deep learning techniques into recommender systems has paved the way for significant advancements in recommendation research and has found wide-ranging applications across various domains. The inherent capacity of deep learning models to capture intricate user-item relationships and effectively incorporate diverse data sources has positioned them as a key enablers in the pursuit of more accurate and comprehensive recommendation systems.

Within the realm of deep learning algorithms, there exists a distinct category known as graph learning-based methods, which offer a unique perspective on information processing in recommender systems. In these methods, recommender system data is represented and analyzed through the lens of graph structures. Specifically, the interactions between users and items can be depicted as interconnected nodes in a graph, where the links reflect the relationships between them. By leveraging graph-based representations, recommender systems gain the advantage of incorporating structured external information, such as social relationships among users, into

the recommendation process. This integration of graph learning provides a unified framework for modeling the diverse and abundant data present in recommender systems.

Early explorations in graph learning-based recommender systems have focused on utilizing graph embedding techniques to capture the relationships between nodes. These techniques can be further categorized into factorization-based methods, distributed representation-based methods, and neural embedding-based methods [8]. These approaches aim to learn meaningful representations of nodes in the graph that capture their inherent relationships and characteristics.

Recently, there has been a surge of interest in employing Graph Neural Networks (GNNs) for recommendation tasks, owing to their exceptional ability to learn from graph-structured data. GNN-based recommendation models have attracted significant attention due to their capacity to effectively capture the complex relationships and dependencies among users, items, and other relevant features within the graph structure. By leveraging the expressive power of GNNs, these models hold great promise for enhancing the accuracy and effectiveness of recommender systems.

The utilization of graph learning techniques in recommender systems provides a valuable avenue for leveraging the rich and interconnected nature of user-item interactions. By incorporating graph structures and employing advanced methods like GNNs, recommender systems can effectively harness the power of heterogeneous and interconnected data sources to generate more accurate and personalized recommendations. As the field continues to advance, graph learning-based methods are poised to play a pivotal role in the evolution of recommender systems, offering novel approaches to address the challenges posed by diverse and interconnected data.

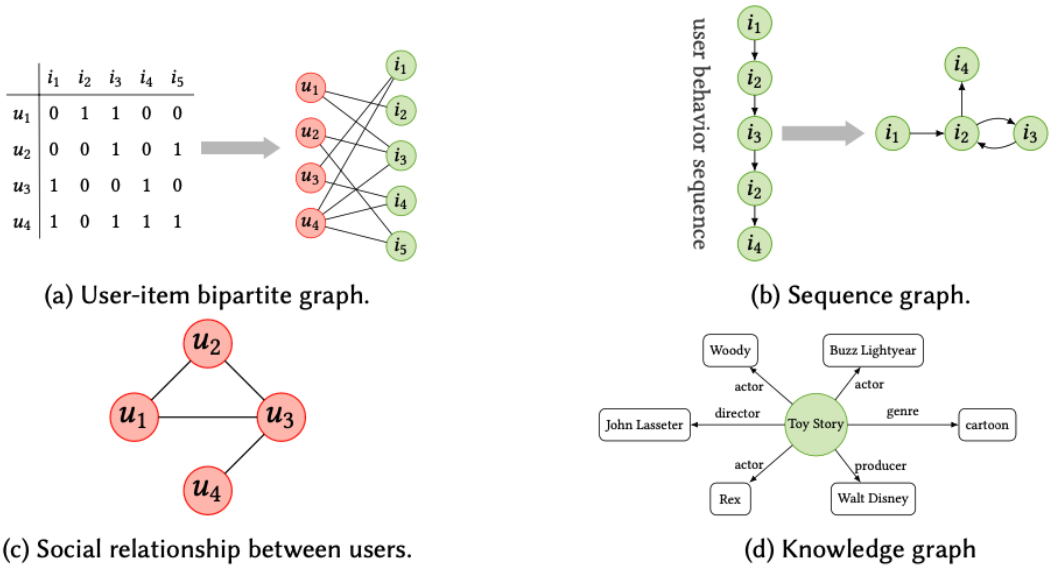

Figure 2: Representative graph structures in recommender systems[6]

Training graph learning-based recommender systems necessitate access to datasets that offer comprehensive information about users and items. However, not all publicly available datasets possess the necessary level of detail to construct the required graphs for training these models. In the context of this project and the development of the RelationalNet model, we employed the Yelp dataset, which proved to be a valuable resource in constructing the diverse graphs essential for our model's training. By leveraging the Yelp dataset, we were able to derive the necessary information and form various graphs that underpin the RelationalNet model's architecture and functionality. In Chapter 3, we delve into a comprehensive exploration of the different graphs employed within the RelationalNet model, shedding light on their significance and contribution to the overall recommendation process.

CHAPTER 2

Related Work

With the widespread adoption of online social platforms, the social recommendation has emerged as a highly promising approach that leverages the social networks among users to enhance recommendation performance [9, 10]. In fact, researchers have long recognized the influence of social connections on individuals, highlighting the phenomenon of similar preferences among social neighbors as information diffuses within social networks[11, 10, 12, 13, 14, 2, 15]. Empirically, social regularization [10] has been demonstrated to be effective in social recommendation scenarios, operating under the assumption that users with similar preferences exhibit shared latent preferences within popular latent factor-based models [16].

In the realm of social recommendations, graph neural networks (GNNs) have emerged as a powerful tool for capturing the intricate relationships between users, items, and other contextual features such as time and location[2, 3, 4, 14]. By leveraging the expressive capabilities of GNNs, personalized recommendations can be generated by considering not only the user-item interactions but also the influence and information propagation within the social network. The incorporation of GNNs into social recommendation models allows for a comprehensive understanding of the complex dynamics present in social networks, resulting in more accurate and contextually relevant recommendations.

In this chapter, we provide a comprehensive overview of recent studies that have explored the application of GNNs in the field of social recommendations. These studies investigate various aspects, such as modeling user-item interactions, capturing social influence, incorporating contextual information, and addressing scalability challenges.

By summarizing these advancements, we aim to present a holistic view of the current state-of-the-art approaches that utilize GNNs in social recommendation systems. Through a deeper understanding of these developments, researchers and practitioners can gain insights into the potential benefits and challenges associated with integrating GNNs into social recommendation frameworks, thereby fostering further innovation and advancements in this exciting research area.

Ying et al. [14] introduced PinSage, a novel framework based on Graph Neural Networks (GNNs), designed specifically for personalized feed recommendations. The motivation behind PinSage stems from the scalability limitations observed in traditional Collaborative Filtering (CF) methods. PinSage revolutionizes the landscape of personalized feeds(news) recommendations by constructing a graph representation that encompasses both items and users. Leveraging the expressive power of GNNs, PinSage efficiently learns personalized feed representations for each individual user. This graph-based approach enables the model to capture the complex relationships between items and users, thereby facilitating accurate and relevant pin recommendations.

One of the key advantages of PinSage lies in its ability to scale seamlessly to web-scale graphs. By leveraging the power of GNNs, PinSage efficiently processes and learns from large-scale graph data, making it well-suited for handling the massive amounts of information present in news recommendation scenarios. This scalability feature allows PinSage to handle the complexities of web-scale datasets, ensuring an effective and efficient solution for personalized news recommendations.

The integration of GNNs within the PinSage framework signifies a significant step forward in personalized news recommendation research. By leveraging the inherent structure and interconnections within the news graph, PinSage brings about

improvements in recommendation quality and scalability, paving the way for more accurate, personalized, and efficient news recommendation systems.

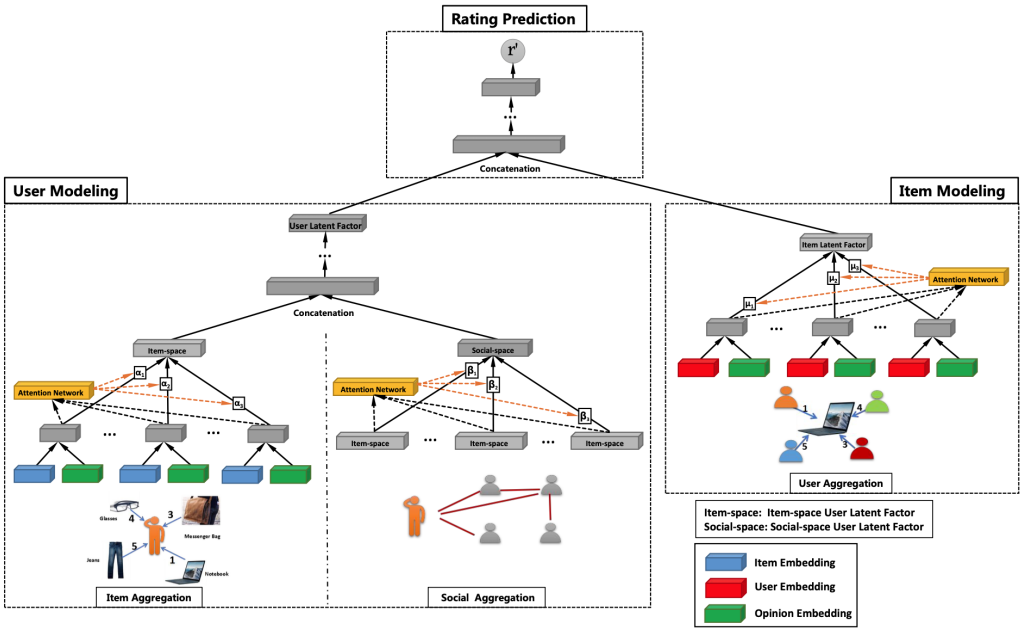

Figure 3: Model architecture of GraphRec for Social Recommendations [2]

In their paper, Fan et al. [2] presented GraphRec, an innovative recommendation algorithm that leverages the power of graphs, as depicted in Figure 9. The authors highlight the limitations of traditional recommendation techniques, particularly in dealing with the cold-start problem and effectively capturing intricate user-item connections. GraphRec aims to overcome these challenges by introducing a graph neural network (GNN) to model user-item interactions in the form of a diverse graph.

The constructed graph incorporates nodes representing both users and items, with edges symbolizing their interactions and relationships. By employing a GNN, GraphRec effectively learns compact representations, referred to as embeddings, for both users and items within this graph structure. These learned embeddings

capture rich information about users' preferences and item characteristics, enabling the algorithm to provide personalized and context-aware recommendations.

The GNN component of GraphRec undergoes training to refine its ability to extract meaningful patterns and relationships from the user-item interaction graph. By optimizing the embeddings, the model can capture the underlying structure and dynamics of the user-item ecosystem. These learned representations serve as the foundation for generating recommendations tailored to individual users, taking into account their preferences and the intricate interplay between users and items.

Through its utilization of graph-based modeling and the integration of GNNs, GraphRec offers a promising approach to recommendation systems, addressing the limitations of conventional techniques and effectively tackling the cold-start problem. By leveraging the power of graphs and learning compact embeddings, GraphRec opens up new avenues for providing accurate and personalized recommendations to users in diverse domains.

Wu et al. [3] proposed Diffnet, a neural influence diffusion model for social recommendation. Diffnet utilizes a user's social network data to provide personalized recommendations. Its neural architecture comprises four main components: the embedding layer, the fusion layer, the layer-wise influence diffusion layers, and the prediction layer.

The embedding layer takes relevant inputs and generates embeddings for users and items. The fusion layer combines a user's (or item's) free embedding with associated features, resulting in a hybrid user (or item) embedding. This fused embedding is then fed into the influence diffusion layers. The influence diffusion layers are structured hierarchically to capture the iterative social diffusion process within the

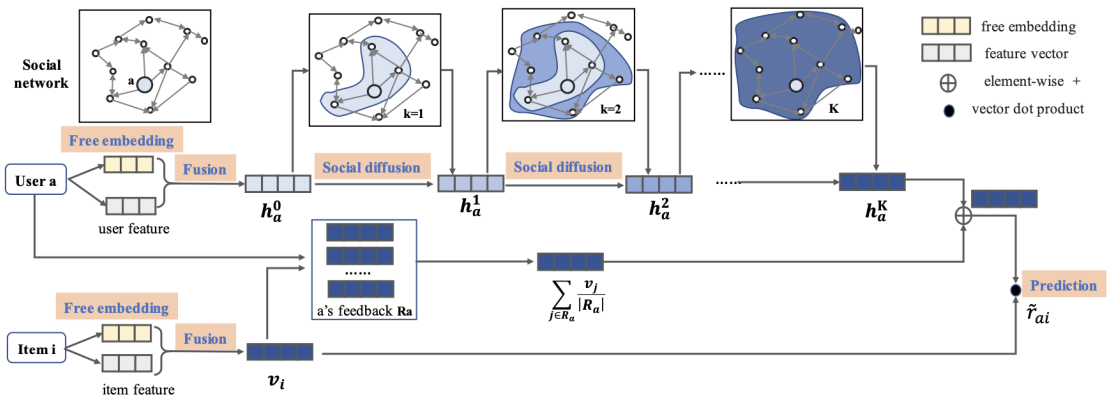

Figure 4: Model architecture of Neural Influence Diffusion Network (Diffnet)[3]

social network, which is a fundamental aspect of Diffnet. Once the influence diffusion process reaches stability, the output layer predicts the final preference for a user-item pair, as illustrated in Figure 4.

Compared to existing social recommendation models, which often rely on shallow network structures that struggle to capture complex diffusion patterns, the Diffnet model aims to overcome these limitations. Leveraging both user-item interaction data and social network information enhances recommendation accuracy.

In subsequent research work, Wu et al. [4] introduced an enhanced version of the Diffnet model [3], called Diffnet++. This advanced model builds upon the neural influence diffusion framework for social recommendations. In addition to learning user embeddings through influence diffusion from their social network, Diffnet++ incorporates user interest embeddings acquired through interest diffusion from user-item interactions.

Figure 5 provides a visual representation of Diffnet++'s approach. At the top of the figure, the user is connected to their social connections, forming a user-user graph used to learn the user influence embeddings. Similarly, at the bottom part

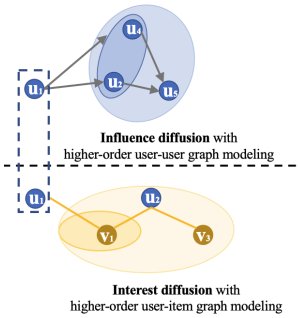

Figure 5: Influence and Interest Diffusion Network (Diffnet++)[4]

of the figure, the user is connected to items, enabling the learning of user interest embeddings from item interactions, represented as user-consumed-items graphs.

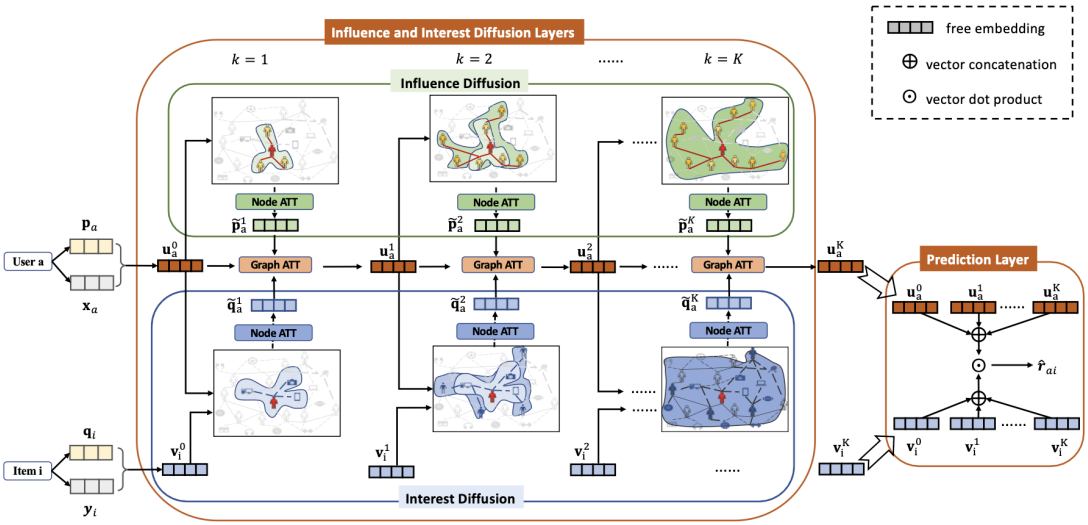

Figure 6: Model architecture of Neural Influence and Interest Diffusions (Diffnet++)[4]

Diffnet++ incorporates a neural architecture consisting of five essential components: the embedding layer, the fusion layer, the layer-wise influence diffusion layers, the layer-wise interest diffusion layer, and the prediction layer, as illustrated in Figure 6. To ensure the efficacy of the user embeddings in both influence and interest diffusion graphs, a node attention layer is employed to selectively emphasize the most relevant information from the surrounding connections. Subsequently, after

training the user influence embeddings and user interest embeddings separately, they are aggregated in a graph attention layer to generate the final user embeddings from the influence and interest perspectives.

As depicted in Figure 6, the model predicts the final preference for a user-item pair once the influence and interest diffusion processes have reached a stable state. This comprehensive architecture empowers the model to effectively capture both influence and interest dynamics. Through extensive evaluation of diverse benchmark datasets, Diffnet++ consistently demonstrates superior performance compared to the original Diffnet model and other cutting-edge recommendation methods.

The fundamental concept behind the Diffnet [3] and Diffnetplus [4] models revolves around training user embeddings through diffusion from graph relations. In the original Diffnet, only the influence graph (user-user) is utilized to learn user embeddings, whereas Diffnetplus incorporates both the influence graph and the interest graphs (user-item) for training user embeddings. In this project, we propose a novel model called RelationalNet, which builds upon the ideas of Diffnet++. However, in addition to the user-user and user-item graphs, we introduce an item-item graph and an item-consumed-users graph.

By incorporating item relations, the RelationalNet model places emphasis on both user and item influences and interests. This extension enables the model to capture and leverage the complex relationships between users and items. Consequently, the model aims to provide enhanced user recommendations by considering the individual interests of users and the intricate interplay between users and items.

The inclusion of the item-item graph and the item-consumed-users graph in RelationalNet further enrich the modeling capabilities, allowing for a more compre-

hensive representation of the user-item ecosystem. Through the integration of these additional relational graphs, the RelationalNet model aspires to deliver more accurate and contextually relevant recommendations to users, considering a broader spectrum of influences and connections.

CHAPTER 3

Methodology

The current chapter presents the problem statement and the proposed architecture of the RelationalNet model for social recommendations utilizing graph neural networks.

3.1 Problem Statement

Within all recommender systems, there exist two groups of entities:

- Users set: U with M users denoted as $|U| = M$
- Items set: V with N items denoted as $|V| = N$

Users interact with items to show their preferences in many forms, e.g., like/dislike, ratings, and purchasing an item. In our approach, we consider a user interaction with items as $r_{ui} = 1$ if user u liked item i; otherwise, it is equal to 0.

Social recommender systems differentiate themselves from other recommender systems by using user-user relations. A user-user directed graph $G_S = [U, U_S \in R^{M \times M}]$ where S represents the social connections between users, i.e., for user u, u_s are the users' neighbors. If a user a follows or trusts user b then $s_{ab} = 1$; otherwise, it is equal to 0.

Similar to the user-user social connections, item-item relations are defined as graph $G_T = [V, V_T \in R^{N \times N}]$ where T represents the item-item connections between items, i.e., for item i, i_t are the similar item neighbors. if a item p is similar to item q then $t_{pq} = 1$; otherwise, it is equal to 0.

Every user is also associated with real-valued embedding denoted as x_a in the user embedding matrix $X \in R^{D_1 \times M}$ where D_1 is the number of embedded features(or

dimensions) for each user a. Every item is also associated with real-valued embedding denoted as y_i in the item embedding matrix $Y \in R^{D_2 \times N}$ where D_2 is the number of embedded features(or dimensions) for each item i.

3.1.1 Definition

The objective is to predict the preferences of users towards unknown items, represented by matrix $\hat{R}$ with dimensions $M \times N$, given a rating matrix R consisting of users (M) and items (N). Additionally, there is user-user social network data (S), item-item network data (T), and two associated real-valued embedded matrices for users (X) and items (Y). The goal is to achieve this prediction through a function $f(R, S, T, X, Y)$.

$$\hat{R} = f(R, S, T, X, Y) \tag{1}$$

3.2 RelationalNet Model Architecture

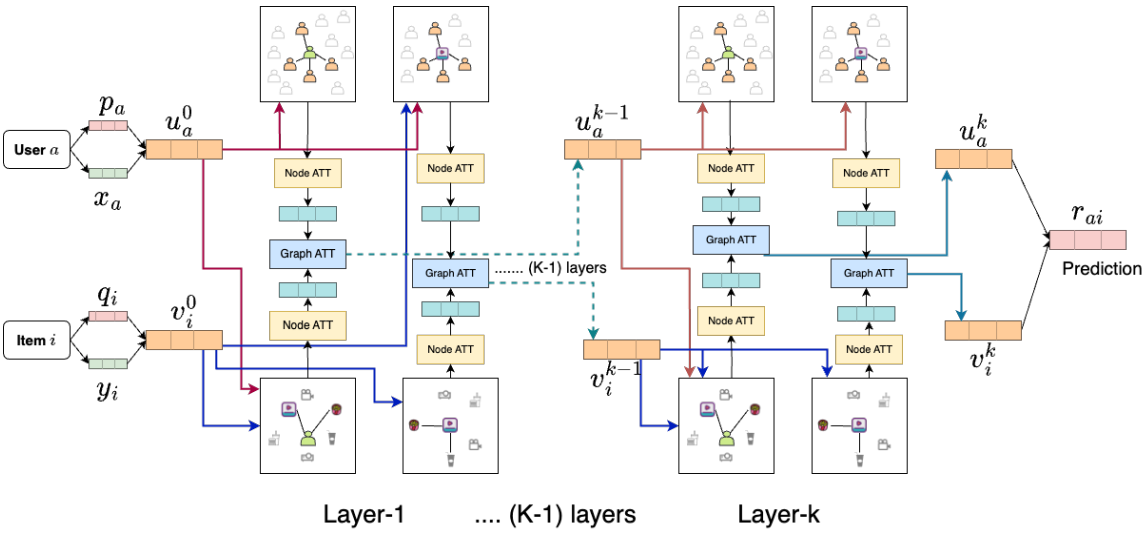

Figure 7: RelationalNet Model

The following subsections will describe the proposed RelationalNet model in

Figure 8, highlighting how each layer is interconnected to produce the final rating for a given user and item. The functionality of each layer and the flow of information between them will be discussed in detail in the following sections.

3.2.1 Relations

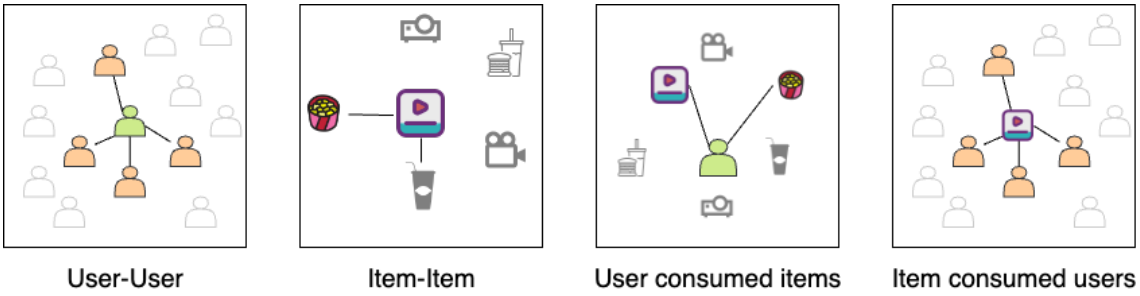

Figure 8: Graphs used in RelationalNet model

The proposed RelationalNet model incorporates four distinct graph structures to facilitate the training process for predicting unknown items, as illustrated in Figure 8. These graph structures encompass the following:

- **User-User Graph**: The User-User graph captures the social interactions between users by establishing links between them. This graph emphasizes the significance of user preferences influenced by their social connections. For instance, within the Yelp dataset, if user A follows users B and C, we create directed graph edges (links) from user A to user B and user A to user C.
- **User-Consumed-Items Graph**: The User-Consumed-Items graph is formed based on the interactions between users and items derived from each user's past behavior. It represents the items that a user has interacted with. For example, if user A watches movies Avengers and Avatar, connections (links) are established between user A and the movie Avengers, as well as between user A and the movie Avatar.

- **Item-Consumed-Users Graph**: The Item-Consumed-Users graph connects items with the users who have interacted with them. It signifies the users who have engaged with a particular item. For instance, if the movie Avengers is watched by users A and B, links are created between the Avengers movie and user A, as well as between the Avengers movie and user B.
- **Item-Item Graph**: The Item-Item graph establishes links between items that exhibit similarities to each other. This graph captures the relationship between items based on shared characteristics. For example, within the Yelp dataset, if two restaurants serve similar food categories, a link is established between these restaurants.

Compared to existing models, the RelationalNet model introduces the incorporation of item relations through the item-item graph and item-consumed-users graph. Similar to how users learn from their neighboring users in the user-user graph, items also have the ability to learn from their neighboring items. This inclusion of item relations enhances the model's capacity to capture complex item-item dynamics, thereby improving the quality of recommendations generated by the RelationalNet model.

3.2.2 Embedding Layer

The utilization of embedding layers is commonplace in the field of Natural Language Processing (NLP), as evidenced by numerous works such as [17, 18, 19]. The embedding layer is part of the hidden layers in a deep neural network, taking high-dimension input and outputs into a lower dimension. Recommender systems utilize this technique to represent users and items with respective free vector encodings.

- Let $P \in R^{M \times D}$ represents the free latent embedding matrices of users with D-dimensions. The embedding layer executes an index selection operation to generate the unrestrained latent vector of the user p_a for user a at a^{th} row.
- Similarly, Let $Q \in R^{N \times D}$ represents the free latent embeddings matrices of items with D-dimensions. The embedding layer executes an index selection operation to generate the unrestrained latent vector of the item q_b for item b at b^{th} row.

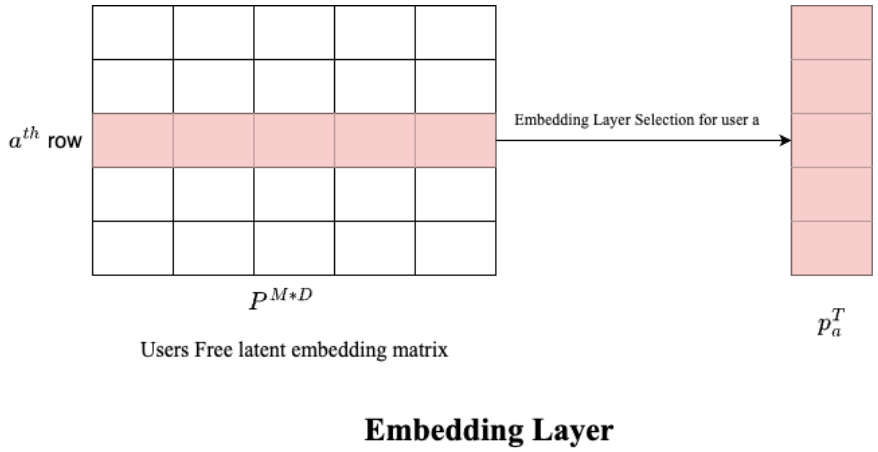

Figure 9: Embedding Layer example for users' free latent embedding matrix

3.2.3 Fusion Layer

The fusion layer merges the latent free vector with the real-valued embedding vector to capture diverse initial interests from the input data. The combination of these two vectors is essential for effective information integration.

- The fusion layer processes latent free vectors p_a from the embedding layer and real-valued embedding vectors x_a from input data X^{M*D} for each user a. The output is the users' initial interests u_a^0 across various input types.

$$u_a^0 = g(W_1 \times [p_a, x_a]) \tag{2}$$

Where W_1 is a trainable weights matrix and $g(x)$ is a function of the transformation matrix.

- For each item i, the inputs are the latent free vector q_i and the real-valued embedding vectors y_i, and the output is the initial interests v_i^0 of the items.

$$v_i^0 = g(W_2 \times [q_i, y_i]) \tag{3}$$

Where W_2 is a trainable weights matrix and $g(x)$ is a function of the transformation matrix.

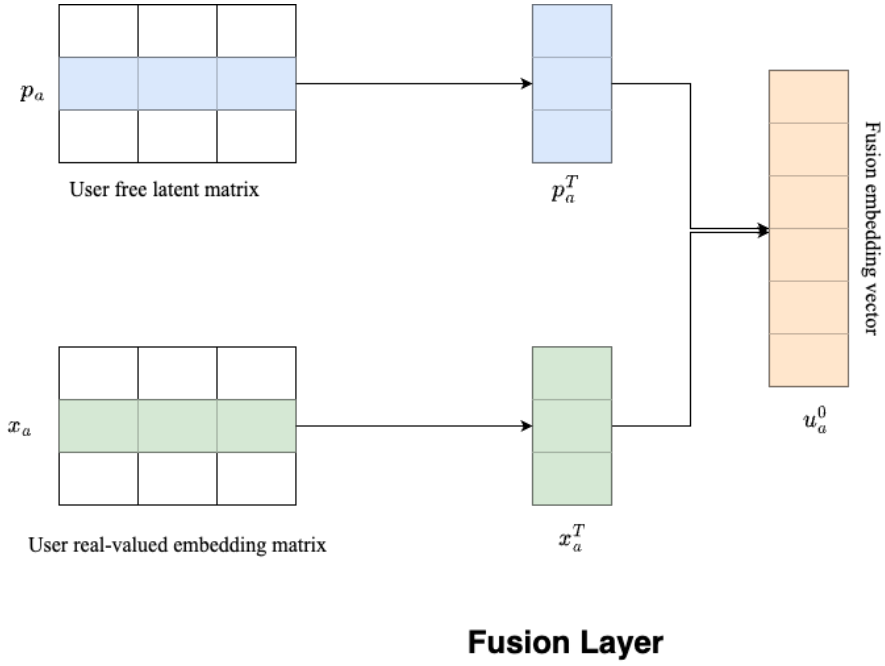

Figure 10: Fusion Layer example

3.2.4 Node Attention Layer

Graph neural networks (GNNs) consist of nodes with connections to other nodes, forming a graph. These connected nodes are referred to as neighboring nodes. In GNNs, each node receives and combines features from its neighboring nodes to represent the

local structure of the graph. Different types of GNN layers use various aggregation techniques for this purpose.

The social recommendations user-user graph employs attention from neighboring nodes to enhance each user's training. This attention can be computed in various ways, such as taking the mean, concatenating it into vectors, or selecting the maximum value from neighboring values. By repeatedly diffusing users' preferences through propagation in the graph, their preferences are effectively spread out as shown in figure 11.

- Let each user a has h_k^a as latent embedding at k^{th} layer, then the user a latent embeddings for the neighboring nodes at layer $k+1$ is

$$h_{Sa}^{k+1} = g(h_b^k | b \in S_a) \tag{4}$$

  Where $g(x)$ is an attention function. The user a latent embeddings can be trained for layer $k+1$ with combination of h_a^k and h_{Sa}^{k+1} neighboring latent embeddings:

$$h_a^{k+1} = \alpha(W^k \times [h_{Sa}^{k+1}, h_a^k]) \tag{5}$$

  where $\alpha(x)$ is non-linear transformation function.

- Let each item i has l_k^i as latent embedding at k^{th} layer, then the item i latent embeddings for the neighboring nodes at layer $k+1$ is

$$l_{Ti}^{k+1} = g(l_b^k | b \in T_i) \tag{6}$$

  Where $g(x)$ is an attention function. The item i latent embeddings can be trained for layer $k+1$ with combination of l_i^k and l_{Ti}^{k+1} neighboring latent embeddings:

$$l_i^{k+1} = \alpha(W^k \times [l_{Ti}^{k+1}, l_i^k]) \tag{7}$$

where $\alpha(x)$ is non-linear transformation function.

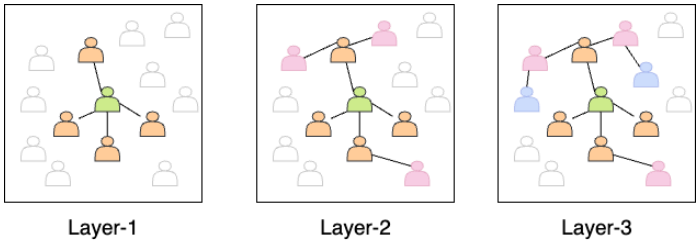

Figure 11: Diffusion at Node attention layer

The RelationalNet model trains the social graph for user-user and item-item connections and incorporates a graph of additional items consumed by each user and the users consumed by each item. Like the user neighbor attention network, consumed items by each user and those consumed for each item are trained for k layers.

- Let each user a has $\tilde{h}_k^a$ as latent embedding at k^{th} layer, then the user a latent embeddings for the consumed item nodes at layer $k+1$ is

$$\tilde{h}_{Va}^{k+1} = g(\tilde{h}_b^k | b \in V_a) \tag{8}$$

  where V_a represents the items consumed by user a and $g(x$ is a transformation function. The user a latent embeddings can be trained for layer $k+1$ with combination of $\tilde{h}_a^k$ and $\tilde{h}_{Va}^{k+1}$ consumed items latent embeddings:

$$\tilde{h}_a^{k+1} = \beta(\tilde{W}^k \times [\tilde{h}_{Va}^{k+1}, \tilde{h}_a^k]) \tag{9}$$

  where $\beta(x)$ is non-linear transformation function.

- Let each item i has $\tilde{l}_k^i$ as latent embedding at k^{th} layer, then the item i latent embeddings for the consumed user nodes at layer $k+1$ is

$$\tilde{l}_{Si}^{k+1} = g(\tilde{l}_b^k | b \in S_i) \tag{10}$$

Where S_i represents users who consumed item i and $g(x)$ is an attention function. The item i latent embeddings can be trained for layer $k+1$ with combination of $\tilde{l}_i^k$ and $\tilde{l}_{S_i}^{k+1}$ consumed users latent embeddings:

$$\tilde{l}_i^{k+1} = \beta(\tilde{W}^k \times [\tilde{l}_{Si}^{k+1}, \tilde{l}_i^k]) \tag{11}$$

where $\beta(x)$ is non-linear transformation function.

3.2.5 Graph Attention Layer

The graph Attention Layer generates the latent embeddings for each user and item using the node attention layer's at every layer by a combination of suited graph embeddings.

- For user a, using graph attention layer at $k+1$ layer to combine the latent embeddings learned from neighbour users h_a^{k+1} and from item consumed users $\tilde{l}_i^{k+1}]$ for each item i and item latent embeddings from k^{th} layer i.e. v_i^k is

$$u_a^{k+1} = MLP_2([h_a^{k+1}, MLP_1([\tilde{l}_i^{k+1}, v_i^k])]) \tag{12}$$

  where MLP_1, MLP_2 are the multi-layer perceptions used to learn the complex relationship between social relations and item embeddings.
- For item i, using graph attention layer at $k+1$ layer to combine the latent embeddings learned from neighbour items l_i^{k+1} and from users consumed item $\tilde{h}_a^{k+1}]$ for each user a and item latent embeddings from k^{th} layer i.e. u_a^k is

$$v_i^{k+1} = MLP_2([l_i^{k+1}, MLP_1([\tilde{h}_a^{k+1}, u_a^k])]) \tag{13}$$

  where MLP_1, MLP_2 are the multi-layer perceptions used to learn an understanding of the intricate connections between item relations and user embeddings.

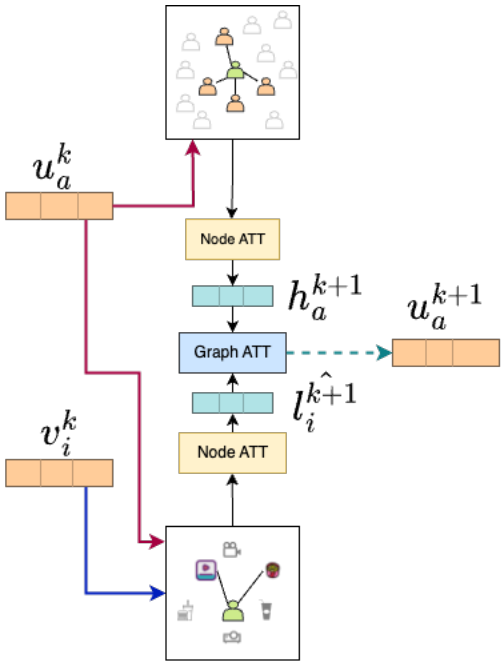

Figure 12: Graph Attention at $k+1$ Layer for user embeddings example

3.2.6 Prediction Layer

To predict a user's liking for an item, we consider the latent embedding of each user a from the final graph attention layer [12] denoted as u_a^k at the k^{th} layer after repeated diffusion of neighbors from eq.4. Similarly, for every item i latent embeddings from graph attention layer [13] denoted as v_i^k.

$$\hat{r}_{ai} = (v_i^k)^T \times u_a^k \tag{14}$$

3.2.7 Loss Function

Our focus is on the implicit feedback of users. In line with the commonly used ranking-based loss function in Bayesian personalized ranking[20], we have designed a loss function based on pairwise ranking for optimization purposes:

$$minimize(L(R, \hat{R})) = \sum_{a=1}^{M} \sum_{i \in D_a}^{N} \sigma(\hat{r}_{ai} - r_{ai}) \tag{15}$$

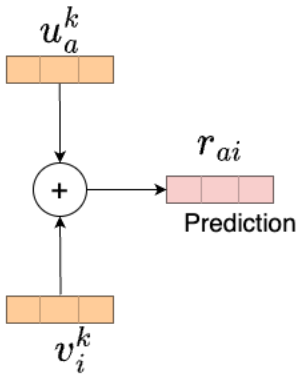

Figure 13: Prediction Layer

where $\sigma(x)$ is a sigmoid function. D_a represents the known rated items of user a.To implement it, we utilize TensorFlow to execute the suggested model. We optimize the model parameters using a mini-batch Adam optimizer.

CHAPTER 4

Results

4.1 Dataset

The Yelp dataset is a vast collection of information on businesses, reviews, users, check-ins, and tips accessible for research purposes. It contains data on various types of businesses, including restaurants, bars, and cafes. These details include their names, addresses, phone numbers, categories, ratings, operational hours, and reviews. Individuals have the ability to provide a rating of 1-5 stars and compose written evaluations for businesses. Moreover, the dataset incorporates social network information, such as the followers of each Yelp user and the quantity of fans subscribed to a business. The Yelp dataset can be downloaded from the Yelp website, but users must abide by the Yelp Dataset License Agreement[21].

Yelp reviews are valuable for gaining insight into businesses, as they provide star ratings and user-written feedback. To analyze this data, a preprocessing step is performed by converting star ratings of 3 or higher to a rating of 1, indicating a positive sentiment towards the business, and ratings below 3 are converted to 0, indicating negative sentiment. In order to obtain insights from the written reviews, the gensim tool and Word2vec model are employed to capture the embedding of each word. By doing so, it becomes possible to produce a feature vector for each user by computing the mean of the trained word vectors that are correlated to their reviews. A similar process is employed to create feature vectors for every business. The feature vectors for both users and items serve as inputs to the model, denoted as X and Y, respectively.

The Yelp dataset contains user followers' information, creating a user-user graph.

In this graph, a link is established between user a and user b if user a follows user b, and the link is assigned a weight of 1 which is used in input as S in problem definition. The dataset also contains information on businesses and their categories. Businesses with at least seven common categories are considered similar, and a link is formed between them to create a business-business graph. The user-user and business-business graphs are used as inputs s and T respectively for the neural network to capture graph relationships.

During the training and testing process, the dataset is filtered to exclude information that may not be reliable or useful. Users with inadequate information, such as those with less than ten reviews or ten followers, are removed from the dataset. Users with certain conditions that may skew the results are also excluded. By applying these filters, the dataset is refined to ensure that only relevant and reliable information is utilized for the analysis.

Dataset	**Yelp**
Users	1,987,897
Businesses	150,346
Reviews	6,990,280
Filtered Users	15,519
Filtered Business	24,648
Filtered Reviews	815,777
Social Links (user-user)	836,186
Item links (business-business)	196,010
Reviews Sparsity	0.213%

Table 2: Yelp Dataset.

4.2 Training Settings

The Yelp dataset, after being preprocessed, is divided into three distinct subsets - the training, validation, and test datasets, respectively. These subsets are created using a ratio of 7:1:2. At the initialization of the model, there are fixed input values

provided for various parameters such as the user feature matrix, items feature matrix, user-consumed items, item-consumed users, user-user graph data, and item-item graph data. The fixed input values are utilized in various layers of the training model. In the training process, the hyperparameters are fine-tuned using the validation dataset.

During each epoch, different mini-batches of users with varying sizes (100, 250, 500, 1000) are tested, and it is found that a batch size of 500 yields comparatively better results than other batch sizes. The Adam optimizer is utilized to optimize the model, with an initial learning rate of [0.001,0.0025,0.005] and a decay learning rate to minimize the loss function given by Equation (15). To train the model to be unbiased, a certain number of false negative ratings are added for each user from randomly selected unrated items.

The GCN models utilize the depth parameter K to gauge the impact of diffusion on the overall model (as shown in Eq.(12, 13)). To evaluate the performance of the model, it is trained with different values of $K = 2, 3$. The size of the user and item-free embeddings, denoted as D, is determined by the number of dimensions in the fusion layer as well as the subsequent diffusion layers. For the fusion layers, we use a sigmoid function as the non-linear function $g(x)$ to transform each value into the range (0, 1) (as shown in Eq.(2, 3)). The output size of each layer is set to $D = 64$ dimensions.

4.3 Performance Metrics

The models' effectiveness is evaluated using top-N recommendation metrics, which are specifically designed to measure the system's ability to predict the items that a user may find interesting. The two main metrics used for this purpose are Hit Rate (HR) and Normalized Discounted Cumulative Gain (NDCG).

4.3.1 Hit Rate (HR)

HR is a metric used to evaluate the performance of the model by calculating the percentage of times at least one item from the test set of a user was recommended in the top-N recommendations suggested by the model. Essentially, HR assesses whether the model is able to recommend at least one relevant item to the user.

The Hit Rate (HR) at N is calculated as follows:

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

where a "hit" is defined as the recommended item being among the ground truth items for a user, and "top-N" refers to the top N recommended items for each user.

For example, let's consider a test set of 100 users and their corresponding ground truth sets of movies. The recommender system generates a list of top-5 movie recommendations for each user. After evaluating the recommendations, it is found that 30 users have at least one recommended movie in their ground truth sets. In this case, the hit rate would be calculated as $30/100 = 0.3$ or 30%.

A higher hit rate serves as an indicator of a recommender system's effectiveness, as it suggests that the system is successfully recommending items that match users' preferences. However, it is essential to recognize that the hit rate metric solely focuses on whether the recommended items are present in the user's ground truth set, without considering their ranking or relevance. As a result, it is common practice to complement the hit rate with other evaluation metrics to obtain a more comprehensive assessment of the recommender system's performance. By incorporating additional metrics, the evaluation process gains insights into the system's ability to provide

accurate and highly relevant recommendations to users.

4.3.2 Normalized Discounted Cumulative Gain (NDCG)

Normalized Discounted Cumulative Gain (NDCG) evaluates the quality of the recommended items' ranking. This metric accounts for the relevance of items based on their position in the recommendation list, with items at higher positions being deemed more relevant. The metric is normalized based on the ideal DCG (Discounted Cumulative Gain). The ideal DCG represents the cumulative gain of the perfectly ordered items that are most relevant to each user.

$$NDCG@k = \frac{DCG@k}{IDCG@k}$$

where k represents the position in the recommendation list. Discounted Cumulative Gain (DCG) is defined as:

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

where rel_i is the score that indicates how relevant the item recommended at position i is, and IDCG (Ideal Discounted Cumulative Gain) is the DCG score of the ideal (perfectly ordered) list of recommended items.

For example, let's consider a scenario where a user is looking for movie recommendations. The recommender system generates a list of recommended movies based on the user's preferences. The relevance of each recommended movie is assessed based on factors such as the user's past interactions, ratings, or explicit feedback.

Suppose the system recommends a list of movies with their corresponding relevance

scores:

Movie	Relevance Score
Movie A	0.9
Movie B	0.6
Movie C	0.8
Movie D	0.4
Movie E	0.7

Table 3: Relevance score for movies from recommender system

The ideal or perfect ranking of the movies, according to the user's preferences, would be:

Movie	Relevance Score
Movie A	0.9
Movie C	0.8
Movie E	0.7
Movie B	0.6
Movie D	0.4

Table 4: Ideal Relevance score for movies from recommender system in order

To calculate NDCG, we take the relevance scores of the recommended movies and discount them based on their positions in the ranking. The discounted relevance scores are then summed up and normalized.

In this example, the NDCG would be calculated as follows:

$$DCG = 0.9 + 0.8/log2(3) + 0.7/log2(4) + 0.6/log2(5) + 0.4/log2(6)$$
$$DCG = 0.9 + 0.8/1.585 + 0.7/2 + 0.6/2.322 + 0.4/2.585$$
$$DCG = 0.9 + 0.505 + 0.35 + 0.258 + 0.154$$
$$DCG = 2.167$$

$$IdealDCG(IDCG) = 0.9 + 0.8/log2(2) + 0.7/log2(3) + 0.6/log2(4) + 0.4/log2(5)$$

$$IDCG = 0.9 + 0.8 + 0.7/1.585 + 0.6/2 + 0.4/2.322$$
$$IDCG = 0.9 + 0.8 + 0.441 + 0.3 + 0.173$$
$$IDCG = 2.614$$

$$NDCG = DCG/IDCG = 2.167/2.614 \approx 0.828$$

In this case, the NDCG score is approximately 0.828, indicating the quality of the recommended list based on both ranking and relevance. The higher the NDCG score (closer to 1), the better the recommender system is performing in terms of providing relevant and well-ranked recommendations.

In this project, HR and NDCG are evaluated on the test data to determine how well the models perform. By utilizing these top-N metrics, we can effectively gauge the ability of the model to make accurate recommendations and compare the effectiveness of different models.

4.4 Observations

We conducted several experiments to evaluate the effectiveness of our proposed RelationalNet model, where we trained the model using various sets of hyperparameters, as described in the training settings section. We aimed to find the hyperparameter settings that would yield the best performance for the model. To compare our RelationalNet model results, we used the Diffnet++ model[4], which is the latest recommender system for social recommendations. The Diffnet++ model has demonstrated superior performance compared to other existing models in this field.

To ensure fairness in our experiments, we employed the identical datasets and evaluation metrics used by the Diffnet++[4] paper. Specifically, we used the Yelp

datasets and evaluated our model using HR and NDCG metrics. Our findings indicate that our RelationalNet model outperformed the Diffnet++ model to a small extent, with respect to all the evaluation metrics mentioned above.

Model	HR(5)	HR(10)	HR(15)
Diffnet++[4] (GNN Layers K=2)	0.2700	0.3112	0.3650
Diffnet++[4] (GNN Layers K=3)	0.2455	0.2824	0.3359
RelationalNet Model (GNN Layers K=2)	**0.2914**	**0.3255**	**0.3718**
RelationalNet Model (GNN Layers K=3)	0.2749	0.3123	0.3608

Table 5: Results metrics: Hit Rate(HR) for top 5,10 and 15 recommendations

Table 5 presents the results obtained by training our RelationalNet model with various combinations of hyperparameters, as outlined in the training settings section. The RelationalNet model, consisting of 2 GNN layers, demonstrated better performance in terms of hit rate metrics for top-k=[5,10,15] recommendations. Our analysis revealed that increasing the number of layers led to a decrease in metrics, indicating a potential negative impact on both our RelationalNet model and the Diffnet++ model.

Model	NDCG(5)	NDCG(10)	NDCG(15)
Diffnet++[4] (GNN Layers K=2)	0.4773	0.5124	0.5242
Diffnet++[4] (GNN Layers K=3)	0.4644	0.4979	0.5090
RelationalNet Model (GNN Layers K=2)	**0.5040**	**0.5328**	**0.5407**
RelationalNet Model (GNN Layers K=3)	0.4911	0.5217	0.5303

Table 6: Results metrics: Normalized Discounted Cumulative Gain(NDCG) for top 5,10 and 15 recommendations

Furthermore, Table 6 shows the NDCG metrics for different models, including the RelationalNet model with 2 GNN layers. Our analysis revealed that the RelationalNet model demonstrated promising results and outperformed other NDCG metrics for top-k recommendations. The findings indicate that our RelationalNet model may improve the precision of recommendations in social recommendation systems.

Model	MAE	RMSE
SocialMF[16]	0.3697	2.1990
GraphRec[2]	0.5442	0.6640
Diffnet++[4] (GNN Layers K=2)	0.0283	0.1685
Diffnet++[4] (GNN Layers K=3)	0.0273	0.1654
RelationalNet Model (GNN Layers K=2)	0.0283	0.1684
RelationalNet Model (GNN Layers K=3)	**0.0273**	**0.1653**

Table 7: Results metrics: Mean Average Error(MAE) and Root Mean Square Error (RMSE)

Table 7 shows a comparison between SocialMF[16], GraphRec[2], Diffnet++[4], and RelationalNet model in terms of their mean average error (MAE) and root mean square error (RMSE) values. It is worth mentioning that there are considerable variations between the metrics obtained by GraphRec and the other models. Specifically, while GraphRec predicts ratings on a scale of 1-5, the other models predict whether the user likes an item or not, with binary values of 1 or 0. Hence, comparing the performance of these models based on the same metric can be challenging due to significant differences in the metrics obtained by GraphRec and the other models. Nevertheless, we can observe that the RelationalNet model and Diffnet++ outperform GraphRec[2] and SocialMF[16] which is a social recommender system without GNNs on both MAE and RMSE metrics. The RelationalNet model with 3 GNN layers achieved the best performance among these two models, with the lowest MAE and RMSE values.

The outcomes from our experiments presented in the above tables illustrate the efficacy of our RelationalNet model in tackling the social recommendation problem. Our model has achieved accurate user preference prediction and outperformed existing models in multiple evaluation metrics. The outcomes indicate that integrating GNN layers to facilitate interest and influence diffusion has led to an enhancement in

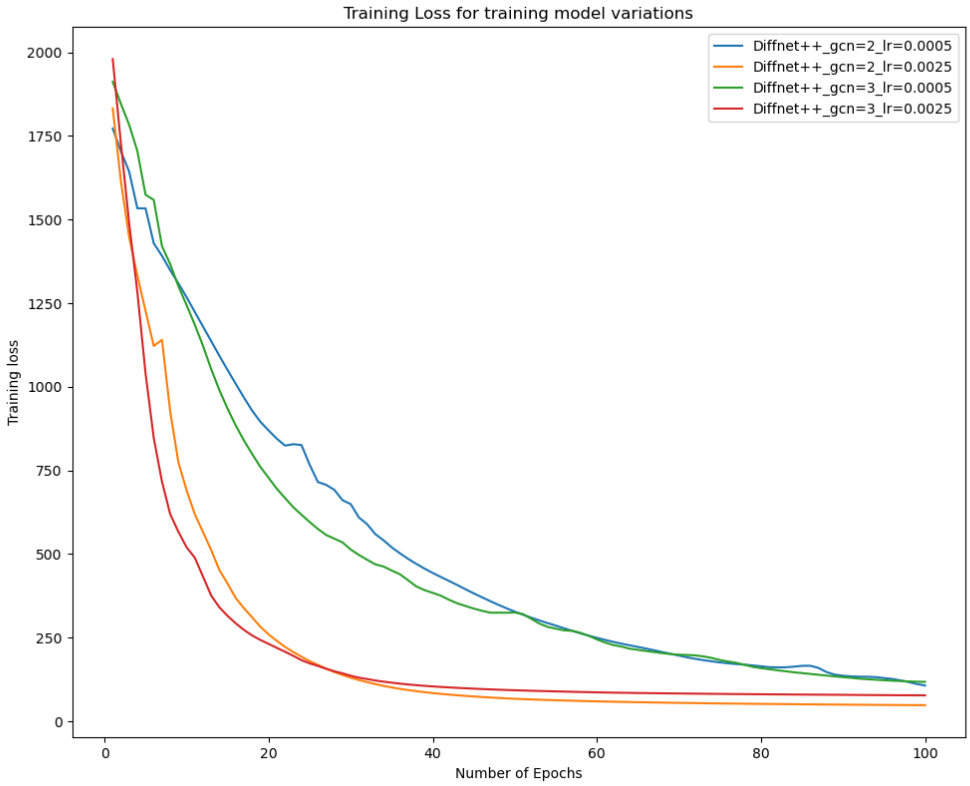

Figure 14: Training loss of the Diffnet++ model[4]

recommendation precision. The RelationalNet model can potentially enhance the recommendation accuracy in social recommendation systems and can be effectively used in real-world applications. In general, our findings emphasize the practicality and efficiency of the model we introduced in tackling the difficulties encountered in social recommendation systems.

Figures 14 and 15 illustrate the comparison of the proposed RelationalNet model's training loss with different learning rates to analyze its performance under various hyperparameters. The visual representation of the loss function enables us to evaluate the model's convergence rate. The figures reveal that a learning rate of 0.0025 leads to

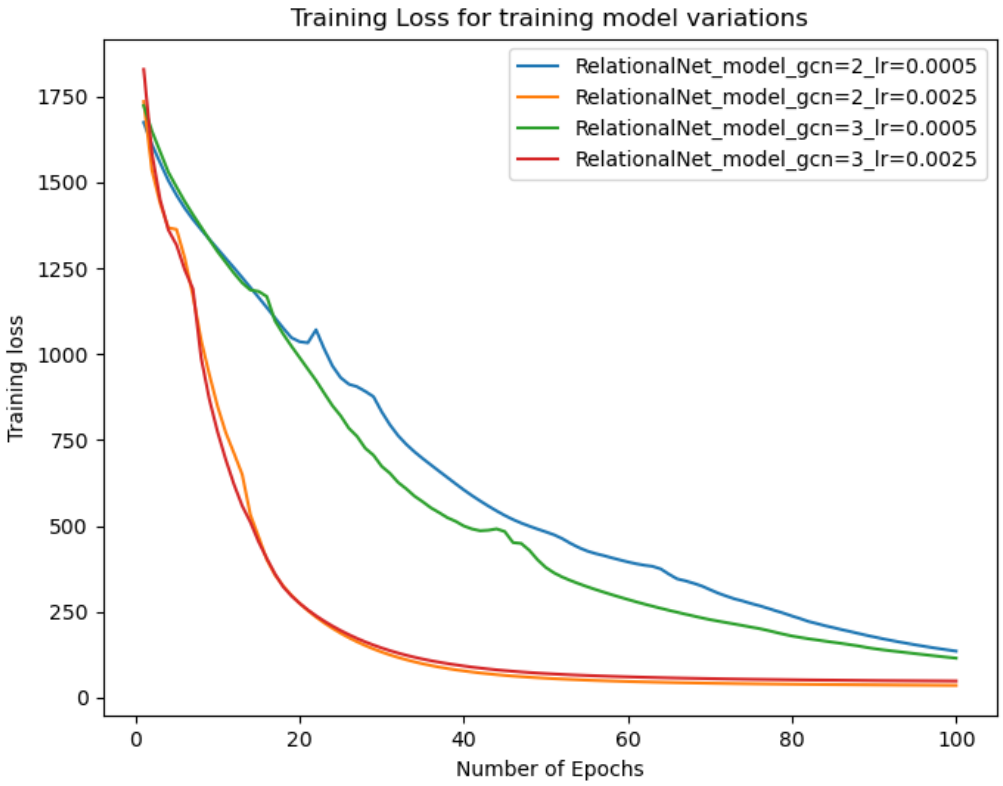

Figure 15: Training loss of RelationalNet model

a smoother decrease in training loss than a learning rate of 0.005, indicating that the former results in better convergence. Moreover, the RelationalNet model outperforms the Diffnet++ model in terms of training loss for most learning rates, indicating its ability to learn from input data effectively and suggesting that it can achieve higher accuracy than the Diffnet++ model.

CHAPTER 5

Conclusion

This project introduced an improved version of the neural influence and diffusion model for social recommendations, named RelationalNet based on the foundation of the Diffnet++[4] approach. Our model utilized social connections to create a user-user graph. To further enable the learning of user and item embeddings using graph neural networks, we created item-item graphs by linking similar items. We created a multi-layer diffusion network that employs graph attention to combine graph and node-level representations, with the aim of enhancing user and item modeling.

Our experiments using the Yelp dataset showed that our RelationalNet model achieved impressive results, indicating its effectiveness in addressing the social recommendation problem. However, there is still significant scope to explore different mechanisms for forming connections between items and investigating graph reasoning models to learn users' preferences better.

Overall, our RelationalNet model provides a significant step forward in social recommendation systems by taking advantage of social connections and leveraging the power of graph neural networks. We hope our work will inspire further research and development of even more powerful and effective recommendation systems for social networks.

LIST OF REFERENCES

[1] R. v. d. Berg, T. N. Kipf, and M. Welling, ‘‘Graph convolutional matrix completion,’’ *arXiv preprint arXiv:1706.02263*, 2017.

[2] W. Fan, Y. Ma, Q. Li, Y. He, E. Zhao, J. Tang, and D. Yin, ‘‘Graph neural networks for social recommendation,’’ in *The World Wide Web Conference*, ser. WWW ’19. New York, NY, USA: Association for Computing Machinery, 2019, p. 417–426. [Online]. Available: https://doi.org/10.1145/3308558.3313488

[3] L. Wu, P. Sun, Y. Fu, R. Hong, X. Wang, and M. Wang, ‘‘A neural influence diffusion model for social recommendation,’’ in *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR’19. New York, NY, USA: Association for Computing Machinery, 2019, p. 235–244. [Online]. Available: https://doi.org/10.1145/3331184.3331214

[4] L. Wu, J. Li, P. Sun, R. Hong, Y. Ge, and M. Wang, ‘‘Diffnet++: A neural influence and interest diffusion network for social recommendation,’’ *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 10, pp. 4753--4766, 2022.

[5] B. Leder. ‘‘How to implement a product recommendation system with snowflake.’’ [Online]. Available: https://www.phdata.io/blog/how-to-implement-a-product-recommendation-system-with-snowflake/

[6] S. Wu, F. Sun, W. Zhang, X. Xie, and B. Cui, ‘‘Graph neural networks in recommender systems: a survey,’’ *ACM Computing Surveys*, vol. 55, no. 5, pp. 1--37, 2022.

[7] Y. Koren, R. Bell, and C. Volinsky, ‘‘Matrix factorization techniques for recommender systems,’’ *Computer*, vol. 42, no. 8, pp. 30--37, 2009.

[8] S. Wang, L. Hu, Y. Wang, X. He, Q. Z. Sheng, M. A. Orgun, L. Cao, F. Ricci, and P. S. Yu, ‘‘Graph learning based recommender systems: A review,’’ in *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, Z.-H. Zhou, Ed. International Joint Conferences on Artificial Intelligence Organization, 8 2021, pp. 4644--4652, survey Track. [Online]. Available: https://doi.org/10.24963/ijcai.2021/630

[9] M. Jiang, P. Cui, F. Wang, W. Zhu, and S. Yang, ‘‘Scalable recommendation with social contextual information,’’ *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 11, pp. 2789--2802, 2014.

[10] H. Ma, D. Zhou, C. Liu, M. R. Lyu, and I. King, ‘‘Recommender systems with social regularization,’’ in *Proceedings of the Fourth ACM International Conference on Web Search and Data Mining*, ser. WSDM ’11. New York, NY, USA: Association for Computing Machinery, 2011, p. 287–296. [Online]. Available: https://doi.org/10.1145/1935826.1935877

[11] A. Anagnostopoulos, R. Kumar, and M. Mahdian, ‘‘Influence and correlation in social networks,’’ in *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD ’08. New York, NY, USA: Association for Computing Machinery, 2008, p. 7–15. [Online]. Available: https://doi.org/10.1145/1401890.1401897

[12] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, ‘‘Graph neural networks: A review of methods and applications,’’ *AI Open*, vol. 1, pp. 57--81, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2666651021000012

[13] A. Micheli, ‘‘Neural network for graphs: A contextual constructive approach,’’ *IEEE Transactions on Neural Networks*, vol. 20, no. 3, pp. 498--511, 2009.

[14] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, ‘‘Graph convolutional neural networks for web-scale recommender systems,’’ in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, NY, USA: Association for Computing Machinery, 2018, p. 974–983. [Online]. Available: https://doi.org/10.1145/3219819.3219890

[15] A. Gulati and M. Eirinaki, ‘‘Influence propagation for social graph-based recommendations,’’ in *2018 IEEE International Conference on Big Data (Big Data)*, 2018, pp. 2180--2189.

[16] M. Jamali and M. Ester, ‘‘A matrix factorization technique with trust propagation for recommendation in social networks,’’ in *Proceedings of the fourth ACM conference on Recommender systems*, 2010, pp. 135--142.

[17] O. Levy and Y. Goldberg, ‘‘Dependency-based word embeddings,’’ in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 2014, pp. 302--308.

[18] S. Rendle, ‘‘Factorization machines with libfm,’’ *ACM Trans. Intell. Syst. Technol.*, vol. 3, no. 3, may 2012. [Online]. Available: https://doi.org/10.1145/2168752.2168771

[19] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, ‘‘Neural collaborative filtering,’’ in *Proceedings of the 26th International Conference on World Wide*

Web, ser. WWW '17. Republic and Canton of Geneva, CHE: International World Wide Web Conferences Steering Committee, 2017, p. 173–182. [Online]. Available: https://doi.org/10.1145/3038912.3052569

[20] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, ''Bpr: Bayesian personalized ranking from implicit feedback,'' in *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, ser. UAI '09. Arlington, Virginia, USA: AUAI Press, 2009, p. 452–461.

[21] Yelp. ''Yelp dataset.'' [Online]. Available: https://www.yelp.com/dataset