A Survey on the Applications of Graph Neural Networks in Recommender Systems
Group 10

Qian Long*
UCLA
USA

Yujun Zhao*
UCLA
USA

Ruochen Wang*
UCLA
USA

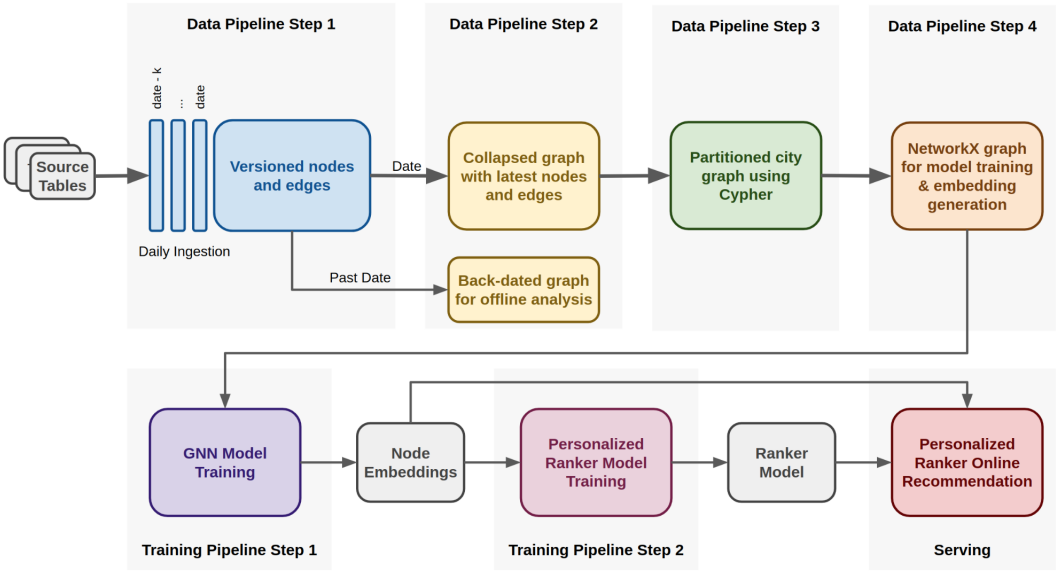

Figure 1: Graph Learning pipeline that powers Uber Eats

ABSTRACT

Deep neural networks for graph structured data (Graph Neural Network) has witness state-of-the-art performance in recommender system tasks. This survey provides a comprehensive overview of the recent developments in applying graph neural networks to recommendation tasks. We focus on categorizing existing methods into high level categorizes, discussing main research concerns related to methods in these categorizes. Representative methods in each category are also presented and discussed.

KEYWORDS

recommendation system, neural networks, graph neural networks

CS249 '21, Mar 01–03, 2021, Los Angeles, CA

ACM ISBN xxxx. . .$99999.00
https://doi.org/xx/xxx

ACM Reference Format:
Qian Long, Yujun Zhao, and Ruochen Wang. 2021. A Survey on the Applications of Graph Neural Networks in Recommender Systems Group 10. In *CS249 '21: Graph Neural Networks, Mar 01–03, 2021, Los Angeles, CA.* ACM, New York, NY, USA, 9 pages. https://doi.org/xx/xxx

1 INTRODUCTION

1.1 Motivation

Recommender systems plays a crucial role in the era of mobile internet. It has become the fundamental technology that powers a majority of the applications we use everyday, including Amazon, Uber Eats, TikTok, Youtube, to name a few. With ever increasing amount of information generated everyday at an unprecedented pace, recommender system becomes an efficient way to actively distribute relevant information to users based on their historical activities and social connections, compared with passive search that dominants the PC internet era.

In recent years, Graph Neural Network has witness much success in learning representations from graph data. There has been an increasing interest in applying GNN to recommendation system task. First of all, the user and item interaction data used in recommender systems are essentially graph data. Secondly, recommender system benefits from side information including social networks

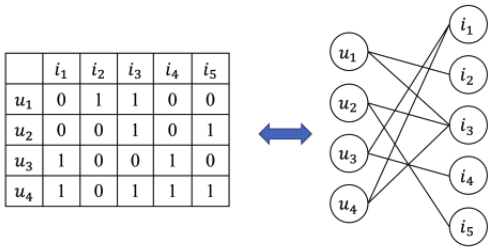

Figure 2: User-item interaction matrix represented as a bipartite graph [13]

and knowledge graph, which are also graph data. In this survey, we aim at providing a review of the applications of GNNs in the recommendation system, focusing on categorizing existing development and discussing representative methods.

1.2 Scope

In this survey, we focus on discussing the application of graph neural network models in the recommender system tasks, including practical concerns in designing model architectures as well as algorithms. The formulation of recommender system tasks will be laid out in high level. From a top-down perspective, we discuss major research areas or concerns and organize different methods accordingly. We provide several representative methods in each category, of which the key contributions will be summarized.

1.3 Organization

In this survey, we divide the recommendation system task into the following two categories:

- **General Recommendation System** assumes static user preference. The user-item interaction are represented as matrix and the recommendation system task is formulated as matrix completion. Concretely, it decomposes the user-item interaction matrix into matrices of user and item latent vectors, where the similarity between the user and item latent vectors represents the user preference over an item.
- **Sequential Recommendation System** formulates the recommendation system task as next-item recommendation. It captures user behavior over time, in the form of item to item transition. In this case, the recommendation task is essentially a time series prediction. Typical models used in this category includes Markov Chain, RNN and Transformers.

Section 2 discusses the formulation of general recommendation systems without side information. We then discuss general recommendation system augmented with various side information in section 3. We separate the discussion on side information into its own category because it is generally applicable to both general RS and sequential RS. We introduce sequential recommendation system in detail in section 4. We then discuss briefly future directions and conclude the survey.

2 GENERAL RECOMMENDATION

In this section, we focus on general recommendation systems (General RS) without side information. We first provide an overview of the formulation of General RS. Then we discuss four main research issues of applying Graph Neural Network to General RS and provide corresponding solutions and design choices.

2.1 Overview

General Recommender System infers static user preference over items utilizing user-item interaction data. Mathematically, it can be formulated as a matrix completion task [13]. Given user-item interactions as a matrix where each row represents a user, each column represents an item, and each entry represents the user-item interaction (e.g. 1 if user purchases an item and 0 otherwise), recommendation task can be formulated as decomposing the matrix into user and item latent vectors. The similarity between user and item latent vectors represents the user preference over an item, which can be used to recommend new items to the user, and thus complete missing entries in the interaction matrix. Two most popular traditional recommender system algorithms include Matrix Factorization [13] and Collaborative filtering [18].

Recently there has been a growing body of research works that study the application of GNN to recommender systems. The reasons are as follows: Firstly, the data used in recommender system, i.e., user-item interaction, are essentially graphs. Secondly, graph Neural Networks have demonstrated their superiority in representation learning on Graph data over traditional methods [20]. Thirdly, recommender systems benefit from side information such as knowledge graphs and social networks, which are also mostly graph data. The formulation is rather straightforward: View users and items as nodes and user-item interaction as edges, we can build a bipartite graph from the data and directly apply GNN to infer latent representations for users and items as shown in Figure 2.

2.2 Main Concerns of Research

2.2.1 Graph Construction. Real world data for recommendation system are typically large scale and sparse. For example, there could be millions of users and items on Amazon (a large number of nodes), but chances are each user might only purchase a few hundred different items over their lifetime (a relative small number of edges). Therefore, naively applying GNN to these graph data leads to extremely high computation efficiency, especially for GNNs with more layers. Methods alone this line primarily differ from each other in what to sample. PinSage [15] samples fixed size neighbors via random-walk, and use the sampled neighbor alone for message passing. This method incurs much randomness, with the advantage of being able to model global information. On the other hand, IG-MC [17] samples relatively small subgraphs from the supergraph and train their models on these subgraphs. Compared with PinSage, IG-MC transfers better to different graphs, at the expense of losing global information during subgraph sampling.

2.2.2 Neighbour Aggregation. A key component of GNN design is the aggregation function, which largely affects its performance. The aggregation function can be divided into two categories based on whether they differentiate the influence of neighbors or treat

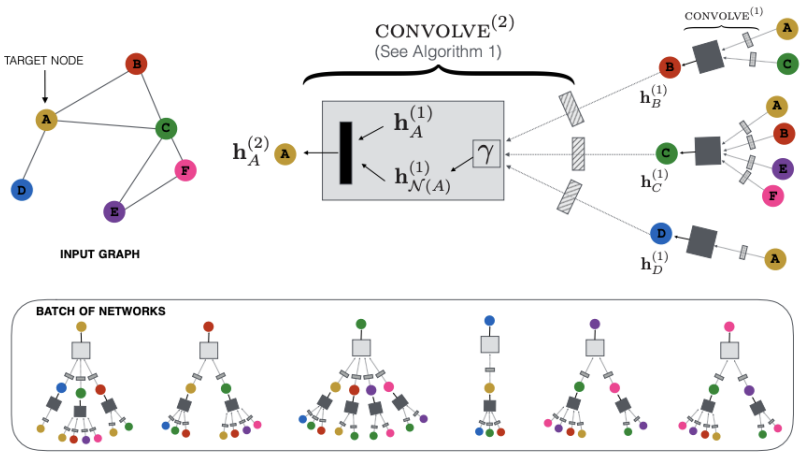

Figure 3: Overview of PinSage architecture with depth-2 convolutions [15]

all neighbors equally. Representative members of the first category includes attentive pooling [8], which relies on learned attention mechanism to assign different weights to the neighbors, and central node augmentation [9], which uses central node to filter messages from the neighbors. The second category includes many commonly used aggregation functions such as mean pooling and central degree normalization. For recommender system tasks, differentiating the importance of neighbor nodes generally improve the performance [8]. Intuitively, items the user interacts do not represent the user preference equally. For example, one might purchase bottled water on amazon every other week, which might not offer much information about the user's preference compared with an one-time purchase of a kindle.

2.2.3 Information Update. After aggregated the neighbor messages into one embedding, we need to combine it with the embedding of the current node. This is the information update step. Typical information update methods include pooling and concatenation. Concatenation allows the neighbor feature and current node feature to be separated during MLP transformation, however the dimensionality also doubles. This part is generally studied in the GNN architecture design literature, and not specific to the recommendation system tasks.

2.2.4 Final Node Representation. As discussed before, the purpose of applying GNN to recommender system is to harness the information in the graph data and produce a feature embedding for each user and item. There are two ways to obtain such node representations. One can either simply use the embedding from the last layer of GNN, or use embeddings from all layers. Similar to information Update, this part is most related to general GNN design and is not specific to the recommender task at hand.

2.3 Representative Methods

In this section, we present two representative GNN methods used in Recommender Systems - STAR-GCN and PinSage.

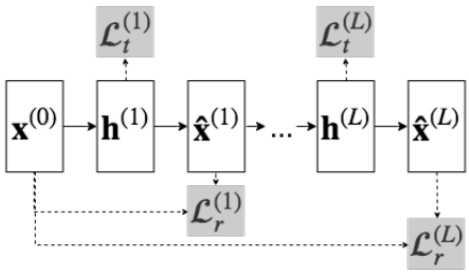

Figure 4: STAR-GCN architecture with L blocks [16]

2.3.1 STAR-GCN. In feed forward neural networks, the model capacity is mainly controlled by the number of layers. However, directly increasing the layers of GCN leads to oversmoothing problem [16] - embeddings of different nodes become similar to each other at convergence, which is not ideal for recommender systems that rely on the similarity between these embeddings. STAR-GCN [16] solves this issue by stacking multiple GCN blocks instead of adding extra layers to each of these block. Figure 4 illustrates a high level structure of STAR-GCN. As we can see, a reconstruct loss is added to the output of each GCN block, which enforces the GCN blocks to learn identity mapping. This structure is similar in design principle to ResNet [3] and Stacked Hourglass [5] used in computer vision.

Furthermore, STAR-GCN also identifies the training leakage issue when applying GNN to Recommender Systems. Concretely, the rating values, which the model predicts, are included in the graph data and thus leaked into the GNN model during message passing, leading to overestimating the model's predictive ability. They propose to solve it by sample-and-removing, i.e., masking out relevant rating values in the data.

2.3.2 PinSage. To scale up GNN models to real world large graph datasets, PinSage [15] resorts to random walk to improve the computation efficiency of GNN models. Concretely, starting from the current node, PinSage applies random walk and count the number of visits to each surrounding node, and then constructs a fixed-size neighbor for the current node by grouping surrounding nodes with a normalized visit count less than the predefined threshold. This way, the neighborhood of a node might not be directly connected with the current node (i.e., N-hop). The process incurs much randomness, while significantly improved the efficiency.

In terms of Neighbor aggregation, PinSage differentiates between the importance of each neighbor. For final node representation, PinSage simply uses the last-layer embedding. PinSage is one of the first GNN methods that scales to graphs with billions of nodes. For exmale, experiment on Pinterest dataset with 3 billion of nodes demonstrates the effectiveness and scalability of the proposed method. Figure 3 shows the architecture design of this method.

3 NETWORK WITH SIDE INFORMATION

We have discussed a lot about the pure user-item bipartite graph and learned some techniques about it. But in the real world, sometimes we have more information to help us get broader and more accurate prediction. Very often, they are social network and knowledge graph.

Take the example of Facebook, it has large database and information system about the social network. If you are a basketball player, you certainly will have many friends that also like or play basketball. They will watch videos, go shopping and read news. With high probability (compare to other irrelevant people), the videos, store and news they like may also suit you. In this way, the recommender system can take the information from your friends and recommend the similar things to you. We know from the above part that the recommendation system neural network suffers from sparse user-item interaction matrix and "cold start" a lot. With the information of social network, this situation could be greatly alleviated.

Then let's think about an even more complex graph – in the Linkin network. You may work for some company, be colleague with some other users, graduate from some University, live in some city. Here, we can find that there are many different entities and different relationship in the same graph. For example, there are entities like companies, universities, people and cities. In this heterogeneous graph, we have even more rich semantic information than in the homogeneous graph like social network. It's more complex, harder to deal with, but also has the potential to give us more information and therefore we may be able to predict more accurately by using this kind of graph.

In the following subsections, we will discuss the social network enhanced recommendation system at first, see what it contains and how current work utilize this kind of information. Then, we will step into the knowledge graph part and also see how we can extract the information we want efficiently.

3.1 Social Network Enhanced Recommendation System

Social network is an homogeneous graph. In this graph, all entities represent users and all edges represents friend relationships between users.

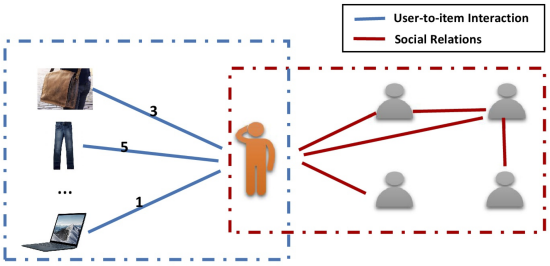

Figure 5: Social network and bipartite

Social network is a good side information system for the recommendation. With the social network, we have the potential to learn more accurate user embedding and therefore deal with the "cold start" problem in the recommender system and alleviate the data sparsity. For example, figure 5 represents a typical user-item bipartite system with the side information of social network. On the one hand, the user is connected to his friends in the social network. On the other hand, the user also connects to the items that he interacted with.

Social network is homogeneous graph and so is user-item interaction graph. This means we have two choice. One is that we can gather information from these two graph separately and then merge the information together. Another possibility is that we can unify these two graphs into one at the first edge. Then this graph will become a heterogeneous graph like knowledge graph. We can also try to extract useful information and do prediction directly on this new graph.

Besides that, My friend's friend may also have influence on me. When we apply GNN to social network, this recursive propagation is an intrinsic property of the GNN technique. Therefore, by propagate the information in each node more than one time, we can learn our friend's friends information.

3.1.1 The Major Issues. How to use the social network as side information is the key in this part. There are two major issues:

- **Influence of different friends** Do friends have equal influence on a certain user? Should we differentiate friends' influence? Furthermore, for a certain user and her certain friend, will the influence be different because of different items? These are questions we can ask when it comes to the influence of different friends. In the following works, we can see that different works also put great attention to this part and try hard to get the most useful information.
- **Merge Graph Information** Should we unify the social network graph and user-item interaction graph into one? Or should we learn something from the two graph separately and then merge the information together? Will one of these two choices always be better? How can we develop some methods for each of these two paths? These are the question we want to know when we consider how to merge the information from these two different graph.

3.1.2 Methods. In the following parts, we will introduce some representative methods in this area. In the next section, we will conclude our findings for the above questions.

GraphRec. [1] This work learns the user/item embedding from the two graphs separately. In the social network, for the question of how to differentiate the importance of friends, it uses the attention mechanism. More specifically, the influence of the friends depends on the similarity between their latent vectors[13]:

$$\alpha_{uv}^{*} = v_a^T ReLU(W_a[h_v^{(l)}||h_u^{(l)}]) \tag{1}$$

Besides from learning user embedding from the social network, this work also learns it from the user-item bipartite graph. To get the overall user representation, it concatenates these two embedding together by:

$$\begin{aligned} c_1 &= [h_u^I \oplus h_u^S] \\ c_2 &= \sigma(W_2 \cdot c_1 + b_2)... \\ h_u &= \sigma(W_l \cdot c_{l-1} + b_l) \end{aligned} \tag{2}$$

By using this attention mechanism, we find that it's a good way to differentiate the influence of different friends. By calculating the "similarity" between my friends and me, I can see that who is more "similar" to me. For the "closer" friends, it's more likely that this friend has more influence than other friends and when they gather the information for the neighbors of a certain user, this friend deserves more attention. In the architecture of GraphRec. To get user embedding, they apply attention network to item aggregation and user aggregation. After that, we concatenate the result and put it into a neural network to get the final result. For the item embedding, they apply attention network to item aggregation and get the embedding.

DiffNet++. [10] This work models this problem in the same network, which means it unifies the two graphs, social network graph and user-item bipartite graph into one. Therefore, the influence of friends and items are now flowing in the same graph. It also uses GAT mechanism to get the embedding for user and item.

3.1.3 Summary. We summarize the work and conclude some findings that how they deal with the above questions for social network enhanced system.

- **Influence of different friends** Most existing work utilize the attention mechanism to differentiate the influence of different friends. Furthermore, some work explore the possibility of costomizing the friend's influence for different items like [11].
- **Gather Both Graphs' Information** Some work utilize the two graphs separately and then merge the information from them. Some other work tries to unify the two graph into one and then learn the information from the big graph. So far we don't know which methods is better and it still needs time to verify.

3.2 Knowledge graph enhanced Recommendation System

Let's move into an even more complex side information graph – Knowledge Graph. It's a harder problem. There are multiple kinds of entities and relationship in the same graph, which make it heterogeneous and complex to deal with. But sometimes difficulty is not a bad thing, because we may get more from it.
We are promised to get better prediction results because of the rich

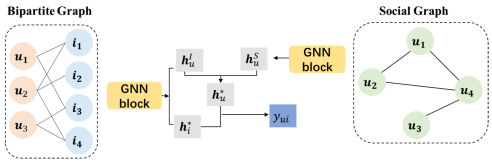

Figure 6: Gather information separately from two graphs[13]

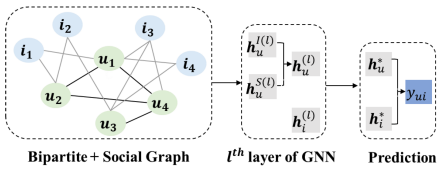

Figure 7: Merge graph into one and learn information from it[13]

semantic information in the knowledge graph. There is more information for us to explore. We can use this information to improve the prediction results and reasonably know a broader interest of our users.

3.2.1 The Major Issues. How to use the social network as side information is the key in this part. There are two major issues:

- **How to merge the information from two graphs** Like the question we proposed in the social network section, how can we combine the information from both knowledge graph and user-item bipartite graph?
- **Do we need to simplify the knowledge graph** As discussed, the knowledge graph is complex. Then can we just extract some information from it to construct a sub-graph? Will it be helpful?
- **Should we differentiate the influence of different relationship** This question is also like what we have discussed in the social network section. There are multiple kinds of relationship in the knowledge graph. Then should we differentiate their influence, and how?

3.2.2 Methods. In the following parts, we will introduce some representative methods in this area. In the next section, we will conclude our findings for the above questions.

KGAT. [7] As figure 8 shows, in this work, the authors unified the two graphs into one. Let's say the users are audience, items are movies and entities may be director, editor and actor. In this graph, there will be multiple relationship like "Directed By", "Acted By" and "Genre". Different audiences may emphasize on different relationship. For example, some people may care about whether their favorite star takes part in this movie, while some may care more about who this movie is directed. Therefore, we need find a way to represent the different importance of relation to a certain audience.
In KGAT, this is accomplished by attention network. If we have a

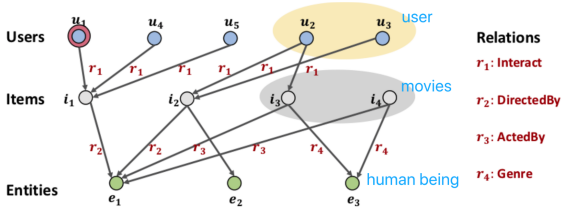

Figure 8: Two graphs are merged into one in KGAT[7]

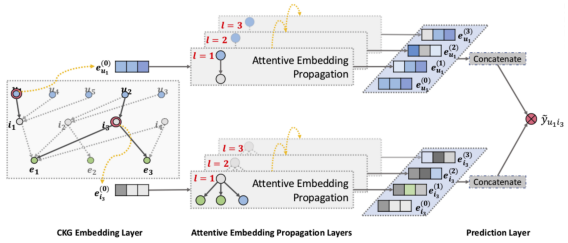

Figure 9: Multiple jump to get neighbors' information in KGAT[7]

look at the formula, it looks like:

$$g(h, r, t) = \|W_r e_h + e_r - W_r e_t\|^2 \tag{3}$$

In figure 9 we can find, the author used attentive embedding propagation to not only get my neighbor's information, but also my neighbor's neighbor's information. Therefore, by this attention mechanism, this work can gather more information.

KGCN. The interesting part of this work is in the relation part. Instead of considering (entity1, relation, entity2), this work considers only (entity, relation). Let's show the key idea by formula:

$$\begin{aligned} \pi_r^u &= g(r, u) \\ v_{N(v)}^u &= \sum_{e \in N(v} \widetilde{\pi}_{r_{v,e}}^u e \end{aligned} \tag{4}$$

IntentGC. [19] The interesting part of this work is the simplification of the knowledge graph. It creates two subgraph: user-user and item-item graphs. For example, look at the figure 10, in the user-user graph, different users may have different relationship. For example, they may buy the same brand, or visit the same shop. Notice that both graphs are homogeneous since the types of entities are the same in both graphs. By this simplification, this work transform the complex knowledge graph into two simplified homogeneous graphs.

3.2.3 Summary. We summarize the work and conclude some findings that how they deal with the above questions for social network enhanced system.

- **How to merge the information from two graphs** Some works integrate the two graphs together and do learning based on the new graph. But some other works learn user/item embeddings separately from the two graphs and then merge the information together.

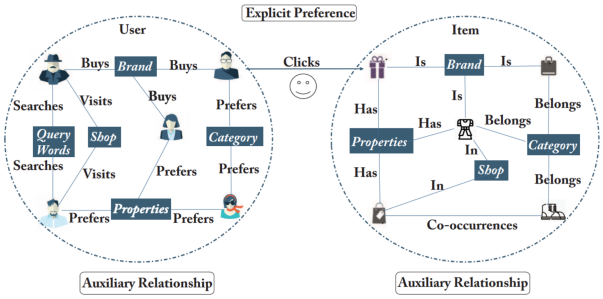

Figure 10: In IntentGC, the knowledge graph is simplified into two graphs[19]

- **Do we need to simplify the knowledge graph** Some works do this, while some others do not. For the works who try to do that, they may extract user-user/item-item information from the original knowledge graph. Besides that, some works construct the subgraph based on the shortest path algorithm[13].
- **Should we differentiate the influence of different relationship** Most works use attention mechanism to consider both the connected nodes and their relation. But there are also some works trying to find a different way. They may emphasize on only the user-relation part – the importance of one relation to a certain user.

4 SEQUENTIAL RECOMMENDATION

The sequential recommendation system tries to predict user's actions based on previous sessions. The conventional methods using recurrent neural networks such as LSTM or GRU to capture the transition of item choices. There are also works employ attention mechanism for pattern capture. However, GNN can capture the complex transitions of items, which are difficult to be revealed by previous conventional sequential methods. Recent works adopts the advantage of GNN by modeling each behavior sequence as a graph to achieve the state-of-the-art performance. We are gonna talk about the basic architecture and several typical GNN based algorithms in the sequential recommendation field.

4.1 Overall Framework

We can model the sequence as a graph, thus GNN can capture complex transitions of items which will be beneficial in sequence recommendation. The main issue is to solve the following parts[13]:

Graph Construction. How to transfer the sequence data to a graph is very important since the performance of the GNN model is highly relied on the sequence graph. Most of the papers model the sequence by applying items as nodes and build directed edges based on the selection[12],[14]. However, there are methods divide the sequence into long and short-term interests which is more straightforward[4].

Information Propagation. This is to update nodes and edges information based on the given graph structure. Many papers[12],[14]

Figure 11: The overall framework of GNN in sequential recommendation.

are using GRU to update the central node. [4] and [6]used an attention mechanism to assign weights for the neighbor nodes based on their relevance.

Sequential Preference. This is to get the user's final preference. Most of the time is a sequence of ranking items.

The overall structure is shown in Fig.11.

4.2 Methods

In this section, I will briefly talk about a few representative methods in sequential representation.

SR-GNN. In this paper[12], they dynamically construct a graph structure for session sequences and capture a graph structure for session sequences and capture rich local dependencies via GNN. For graph construction, they treat each item s_i as a node and (s_{i-1}, s_i) as an edge which represents a user clicks item s_i after s_{i-1} in the session S. Since this is a directed graph, they use M^I and M^O to denote weighted connections of outgoing and incoming edges in the session graph. For node vectors updating, the information propagation between different nodes can be formalized as:

$$a_t = Concat(M_t^I([s_1, ..., s_n]W_a^I + b^I), M_t^O([s_1, ..., s_n]W_a^O + b^O)) \quad (5)$$

The final output of h_t, they use GRU mechanism to decide what information from neighbors to be propagated and the information from current node to be maintained:

$$h_t = GRU(a_t, h_{t-1}) \quad (6)$$

Where h_{t-1} is the previous state.

Next, they combine the long-term preference and the current interest of the session, and they use the combined embedding as the session representation. They weight the last clicked-item vector as the local embeddingss_l with the global embedding h_g:

$$h_s = W_3[s_l; s_g] \quad (7)$$

The global embedding is achieved in a soft-attention manner:

$$\alpha_i = q^T \delta(W_1 v_n + W_2 v_i + c)$$
$$s_g = \sum_{i=1}^{n} \alpha_i v_i \quad (8)$$

GC-SAN. In this paper [14], the architecture is based on the SR-GNN and the main difference is they use self-attention for to generate the session representation. There is a little boost on the final performance.

NISER. The architecture[2] is also the same as SR-GNN, they find that GNN is not good at handling long-tail recommendations. They solve this problem by using l2 normalization on both the session and item embeddings. The models using normalized item and sessiongraph representations perform significantly better: i. for the less popular long-tail items in the offline setting, and ii. for the less popular newly introduced items in the online setting.

MA-GNN. In this paper [4], they proposed a memory augmented graph neural network (MA-GNN) to capture both the long and short-term user interests. Specifically, they applied a graph neural network to model the item contextual information within a short-term period and utilize a shared memory network to capture the long-range dependencies between items. For short-term preference, they use a sliding window strategy and only focus on the most recent items. They only build edges between the most recent items (three in the paper).

$$h_i = tanh(W^1[\sum_k e_k A_{i,k}; e_i])$$
$$P_{u,l}^S = tanh(W^2[1/|L| \sum_i h_i; P_u]) \quad (9)$$

For the long-term memory, they model it by employ the attention mechanism. After that, they use the LSTM to bridge the gap between short-term memory and long-term memory. The whole architecture is shown in Fig.12

FGNN. This paper [6] studied the item transition pattern by constructing a session graph and propose a novel model which collaboratively considers the sequence order and the latent order in the session graph for a session-based recommender system. They formulate the next item recommendation within the session as a graph classification problem. Specifically, they propose a weighted attention graph layer and a Read-out function to learn embeddings of items and sessions for the next item recommendation. The graph construction is the same as SR-GNN, they propose a weighted graph attentional layer(WGAT), which simultaneously incorporates the edge weight when performing the attention aggregation on neighboring nodes. They calculate the weights using attention mechanism and use linear combination to update features of nodes. They also adopt Set2Set to learn a query vector indicating the order of reading from the memory for an undirected graph, which combines attention and GRU together.

4.3 Evaluation

We first compare SR-GNN, GC-SAN and NISER since they have similar architecture. They study the effectiveness of GC-SAN on two real-world datasets, Diginetica and Retailrocket. For evaluation they adopt three common metrics Hit Rate (HR@N), Mean Reciprocal Rank (MRR@N) and Normalized Discounted Cumulative Gain (NDCG@N). I only list part of the results of Diginetica in Fig.13 Adopting the self-attention in the session representation improve the results. For NISER, they find out that normalization of embeddings (L2 norm) contributes the most to performance improvement. The positonal embedding is also helpful shown in Fig.14. This can be considered a helpful engineering trick.

We then compare the MA-GNN with GC-SAN. They evaluated the methods on five real-world datasets from various domains with

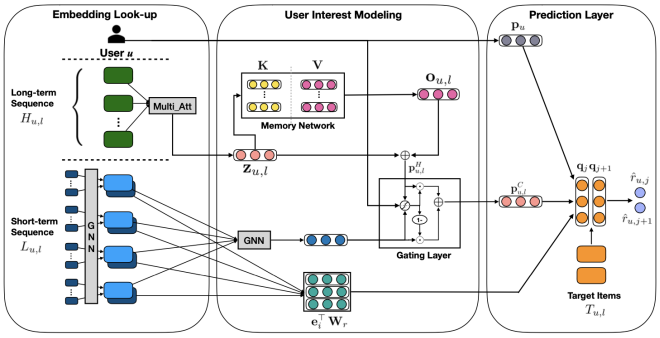

Figure 12: The overall structure of MA-GNN [4].

Datasets	Diginetica					
Measures	HR@5	HR@10	MRR@5	MRR@10	NDCG@5	NDCG@10
Pop	0.0036	0.0077	0.0019	0.0025	0.0023	0.0037
BPR-MF	0.1060	0.1292	0.0789	0.0842	0.0586	0.0672
IKNN	0.1407	0.2083	0.0776	0.0867	0.0693	0.0902
FPMC	0.1855	0.2309	0.0875	0.0986	0.0811	0.1037
GRU4Rec	0.2577	0.3657	0.1434	0.1577	0.1276	0.1607
STAMP	0.3998	0.5014	0.2357	0.2469	0.2039	0.2394
SR-GNN	0.4082	0.5269	0.2439	0.2599	0.2078	0.2443
GC-SAN	**0.4280**	**0.5351**	**0.2694**	**0.2838**	**0.2223**	**0.2552**
Improv.	4.84%	1.56%	10.46%	9.20%	6.98%	4.44%

Figure 13: The performance of SR-GNN and GC-SAN on Diginetica dataset[14]

Method	DN	RR	YC-1/64	YC-1/4
Recall@20				
NISER+	**53.39±0.06**	**61.41±0.09**	**71.27±0.05**	71.80±0.09
-L_2 norm	52.23±0.10	59.16±0.10	71.10±0.09	71.46±0.19
-Dropout	52.81±0.12	60.99±0.09	71.07±0.13	**71.90±0.03**
-PE	53.11±0.12	61.22±0.03	71.13±0.04	71.70±0.11
MRR@20				
NISER+	**18.72±0.06**	**36.50±0.05**	31.61±0.02	31.77±0.10
-L_2 norm	18.11±0.05	33.78±0.04	30.90±0.07	31.49±0.07
-Dropout	18.43±0.11	35.99±0.02	31.56±0.06	**31.93±0.17**
-PE	18.60±0.09	36.32±0.03	**31.68±0.05**	31.71±0.06

Figure 14: Experiments show that L2 normalization, dropout and positional embedding contribute to the performance improvement

different sparsities: MovieLens-20M, Amazon-Books and Amazon-CDs, Goodreads-Children and Goodreads-Comics. The result is shown in Fig.15. There is a large boost in the final performance. The large bias between the two methods is the graph construction part witch they split the graph into short-term preference and long-term preference. We can conclude that adopting short-term preference is vital in the sequential recommender system since it is not modeled in GC-SAN or other baselines.

Last, we compare the performance between FGNN and SR-GNN. They test them on the Yoochoose and Diginetica. The result is

Methods	CDs		Books		Children		Comics		ML20M	
	R@10	N@10	R@10	N@10	R@10	N@10	R@10	N@10	R@10	N@10
BPRMF	0.0269	0.0145	0.0260	0.0151	0.0814	0.0664	0.0788	0.0713	0.0774	0.0785
GRU4Rec	0.0302	0.0154	0.0266	0.0157	0.0857	0.0715	0.0958	0.0912	0.0804	0.0912
GRU4Rec+	0.0356	0.0171	0.0301	0.0171	0.0978	0.0821	0.1288	0.1328	0.0904	0.0946
GC-SAN	0.0372	0.0196	0.0344	0.0256	0.1099	0.0967	0.1490	0.1563	0.1161	0.1119
Caser	0.0297	0.0163	0.0297	0.0216	0.1060	0.0943	0.1473	0.1529	0.1169	0.1116
SARSRec	0.0341	0.0193	0.0358	0.0240	0.1165	0.1007	0.1494	0.1592	0.1069	0.1014
MARank	0.0382	0.0151	0.0355	0.0223	0.1092	0.0980	0.1325	0.1431	0.1002	0.1031
MA-GNN	**0.0442***	**0.0214***	**0.0432***	**0.0279***	**0.1215**	**0.1137***	**0.1617***	**0.1655**	**0.1236***	**0.1272***
Improvement	15.7%	9.2%	20.7%	9.0%	4.3%	12.9%	8.23%	4.0%	5.7%	13.7%

Figure 15: The performance of MA-GNN and GC-SAN [4]

shown in Fig.16. The difference of those two models is that FGNN consider more of the order of the sequence. There is little improvement on the results.

Method	*Yoochoose1/64*		*Yoochoose1/4*		*Diginetica*	
	R@20	MRR@20	R@20	MRR@20	R@20	MRR@20
POP	6.71	1.65	1.33	0.30	0.89	0.20
S-POP	30.44	18.35	27.08	17.75	21.06	13.68
Item-KNN	51.60	21.81	52.31	21.70	35.75	11.57
BPR-MF	31.31	12.08	3.40	1.57	5.24	1.98
FPMC	45.62	15.01	-	-	26.53	6.95
GRU4REC	60.64	22.89	59.53	22.60	29.45	8.33
NARM	68.32	28.63	69.73	29.23	49.70	16.17
STAMP	68.74	29.67	70.44	30.00	45.64	14.32
SR-GNN	70.57	30.94	71.36	31.89	50.73	17.59
(ours)						
FGNN-Gated	70.85	31.05	71.5	32.17	51.03	17.86
FGNN-ATT-OUT	70.74	31.16	71.68	32.26	50.97	18.02
FGNN	**71.12**	**31.68**	**71.97**	**32.54**	**51.36**	**18.47**

Figure 16: The performance of FGNN and SR-GNN [6]

4.4 Future works

For the graph construction part, few of the recommender system fully use the side information. For example, given the knowledge that there is a great dependency between item A and item B, how will that change the initiation of nodes and edges? The model of the item interactions are implicit. Better initiation method that model the relations will surely boost the performance.

Inspired by the MA-GNN structure, they model both the short-term preference and long-term preference. For the short-term preference, it is also possible to measure the flow of the change. Since with the aid of the flow, we can update and modify the model as the selection change along the way. For the long-term preference, they only adopt the attention mechanism, I wonder if we also handle the sequence like SR-GNN or GC-SAN, will it be beneficial to the final performance.

5 CONCLUSION

GNN has been popular used in recommender system. In this paper, we split the methods into general recommendation and sequential recommendation and for each part, we briefly clarify the main issues, summarize the corresponding algorithm framework and give a detailed introduction to the strategy for the main issues adopted by the representative models.

REFERENCES

[1] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. 2019. Graph neural networks for social recommendation. In *The World Wide Web Conference*. 417–426.

[2] Priyanka Gupta, Diksha Garg, Pankaj Malhotra, Lovekesh Vig, and Gautam Shroff. 2021. NISER: Normalized Item and Session Representations to Handle Popularity Bias. arXiv:1909.04276 [cs.IR]

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep Residual Learning for Image Recognition. *arXiv preprint arXiv:1512.03385* (2015).

[4] Chen Ma, Liheng Ma, Yingxue Zhang, Jianing Sun, Xue Liu, and Mark Coates. 2019. Memory Augmented Graph Neural Networks for Sequential Recommendation. arXiv:1912.11730 [cs.IR]

[5] Alejandro Newell, Kaiyu Yang, and Jia Deng. 2016. Stacked Hourglass Networks for Human Pose Estimation. In *ECCV*.

[6] Ruihong Qiu, Jingjing Li, Zi Huang, and Hongzhi YIn. 2019. Rethinking the Item Order in Session-based Recommendation with Graph Neural Networks. *Proceedings of the 28th ACM International Conference on Information and Knowledge Management* (Nov 2019). https://doi.org/10.1145/3357384.3358010

[7] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. 2019. Kgat: Knowledge graph attention network for recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 950–958.

[8] Yiwei Wang, Wei Wang, Yuxuan Liang, Yujun Cai, Juncheng Liu, and Bryan Hooi. 2019. Understanding Attention and Generalization in Graph Neural Networks. In *NeurIPS*.

[9] Yiwei Wang, Wei Wang, Yuxuan Liang, Yujun Cai, Juncheng Liu, and Bryan Hooi. 2020. NodeAug: Semi-Supervised Node Classification with Data Augmentation. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

[10] Le Wu, Junwei Li, Peijie Sun, Richang Hong, Yong Ge, and Meng Wang. 2020. DiffNet++: A Neural Influence and Interest Diffusion Network for Social Recommendation. *IEEE Transactions on Knowledge and Data Engineering* (2020).

[11] Qitian Wu, Hengrui Zhang, Xiaofeng Gao, Peng He, Paul Weng, Han Gao, and Guihai Chen. 2019. Dual graph attention networks for deep latent representation of multifaceted social effects in recommender systems. In *The World Wide Web Conference*. 2091–2102.

[12] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. 2019. Session-Based Recommendation with Graph Neural Networks. *Proceedings of the AAAI Conference on Artificial Intelligence* 33 (Jul 2019), 346–353. https://doi.org/10.1609/aaai.v33i01.3301346

[13] Shiwen Wu, Wentao Zhang, Fei Sun, and Bin Cui. 2020. Graph Neural Networks in Recommender Systems: A Survey. arXiv:2011.02260 [cs.IR]

[14] Chengfeng Xu, Pengpeng Zhao, Yanchi Liu, Victor S. Sheng, Jiajie Xu, Fuzhen Zhuang, Junhua Fang, and Xiaofang Zhou. 2019. Graph Contextualized Self-Attention Network for Session-based Recommendation. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*. International Joint Conferences on Artificial Intelligence Organization, 3940–3946. https://doi.org/10.24963/ijcai.2019/547

[15] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, and Jure Leskovec. 2018. Graph Convolutional Neural Networks for Web-Scale Recommender Systems. In *KDD*.

[16] Jiani Zhang, Xingjian Shi, Shenglin Zhao, and Irwin King. 2019. STAR-GCN: Stacked and Reconstructed Graph Convolutional Networks for Recommender Systems. In *The 28th International Joint Conference on Artificial Intelligence*. 4264–4270.

[17] Muhan Zhang and Yixin Chen. 2020. Inductive Matrix Completion Based on Graph Neural Networks. In *International Conference on Learning Representations*. https://openreview.net/forum?id=ByxxgCEYDS

[18] Ruisheng Zhang, Qi dong Liu, Chun-Gui, Jia-Xuan Wei, and Huiyi-Ma. 2014. Collaborative Filtering for Recommender Systems. In *2014 Second International Conference on Advanced Cloud and Big Data*.

[19] Jun Zhao, Zhou Zhou, Ziyu Guan, Wei Zhao, Wei Ning, Guang Qiu, and Xiaofei He. 2019. Intentgc: a scalable graph convolution framework fusing heterogeneous information for recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2347–2357.

[20] Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. 2018. Graph Neural Networks: A Review of Methods and Applications. *arXiv preprint arXiv:1812.08434* (2018).