Graph Neural Networks in Recommender Systems: A Survey

SHIWEN WU, Peking University
WENTAO ZHANG, Peking University
FEI SUN, Alibaba Inc.
BIN CUI, Peking University

With the explosive growth of online information, recommender systems play a key role to alleviate such information overload. Due to the important application value of recommender system, there have always been emerging works in this field. In recent years, graph neural network (GNN) techniques have gained considerable interests which can naturally integrate node information and topological structure. Owing to the outperformance of GNN in learning on graph data, GNN methods have been widely applied in many fields. In recommender systems, the main challenge is to learn the efficient user/item embeddings from their interactions and side information if available. Since most of the information essentially has graph structure and GNNs have superiority in representation learning, the field of utilizing graph neural network in recommender systems is flourishing. This article aims to provide a comprehensive review of recent research efforts on graph neural network based recommender systems. Specifically, we provide a taxonomy of graph neural network based recommendation models and state new perspectives pertaining to the development of this field.

ACM Reference Format:
Shiwen Wu, Wentao Zhang, Fei Sun, and Bin Cui. 2020. Graph Neural Networks in Recommender Systems: A Survey. *J. ACM* 37, 4, Article 111 (August 2020), 27 pages. https://doi.org/XX.XXXX/XXXXXXX.XXXXXXX

1 INTRODUCTION

With the rapid development of e-commerce and social media platforms, recommender systems have become indispensable tools for many business [78]. Users rely on recommender systems to filter out the numerous uninformative messages and facilitate decision making. An efficient recommender system should accurately capture users' preferences and suggest items that the users are potentially interested in, which can enhance users' satisfactory towards platform and improve user retention.

Recommender systems evaluate users' preferences for the items based on their interests and item properties. Both the user interests and the item properties are represented with compressed vectors. Hence, to learn user/item embeddings with historical interactions and other side information such as social relationship and knowledge graph [49] is the main challenge in this field. In recommender systems, most information has graph structure. For example, social relationship among users and knowledge graph related to items are naturally graph data. In addition, the interactions between users and items can be considered as the bipartite graph, and the item transitions in sequences can be constructed as graphs as well. Therefore, graph learning approaches have been leveraged to get user/item embeddings. Among graph learning methods, graph neural network (GNN) enjoys a massive hype at the moment.

The past few years have witnessed the tremendous success of graph neural networks in many application domains such as relation extraction and protein interface prediction [82]. Recent works

0004-5411/2020/8-ART111 $15.00
https://doi.org/XX.XXXX/XXXXXXX.XXXXXXX

have showed a boost of performance in recommender when introducing the interactions between users/items and side information in the form of graphs[41] and utilizing graph neural network techniques to get better user/item representations. Graph neural network is able to capture the higher-order interaction in user-item relationships through iterative propagation. Moreover, if the information of social relationship or knowledge graph is available, it enables to integrate such side information in network structure efficiently.

Contribution of this survey This survey aims to thoroughly review literature on the advances of graph neural network based recommender systems. The researchers and practitioners who are interested in recommender systems could have a general understanding about the latest developments in the field of graph neural network based recommendation and how to utilize graph neural network to solve recommendation tasks. The key contributions of this survey are summarized as follows:

- **New taxonomy** We propose a systematic classification schema to organize the existing graph neural network based recommendation models. One can easily step into this field and make a distinction between different models.
- **Comprehensive review** For each category, we demonstrate the main issues to deal with and summarize the overall framework of the models. Moreover, we briefly introduce the representative models and illustrate how they address these issues.
- **Future research** We discuss the limitations of current methods and propose four potential future directions in terms of efficiency, multi-graph integration, scalability and sequence graph construction.

The remaining of this article is organized as follows: Section 2 introduces the preliminaries for recommender systems and deep neural networks. Section 3 presents the classification framework. Section 4 and Section 5 summarize the main issues for each category and give a detailed introduction to representative models for general recommendation and sequential recommendation respectively. Section 5 discusses the challenges and points out future directions in this field. Finally, we conclude the survey in Section 6.

2 OVERVIEW OF RECOMMENDER SYSTEMS AND GRAPH NEURAL NETWORKS

Before we dive into the details of this survey, we give a brief introduction to recommender systems and graph neural network techniques. We also discuss the motivation of utilizing graph neural network techniques in recommender systems.

2.1 Recommender Systems

Recommender systems infer users' preferences and items' properties from their attributes or user-item interactions, and further recommend items that users might be interested in [1]. Recommendation has been a popular research area for decades because it has great application value and the challenges in this field are still not well addressed. According to whether considering the order of the items, recommender systems can be divided into general and sequential recommendation tasks [1, 25].

The general recommendation assumes the users have static preferences and models the compatibility between users and items based on either implicit (clicks) or explicit (ratings) feedback. It predicts the user's rating for the target item, i.e., rating prediction or recommends top-N items the user might be interested in, i.e., top-N recommendation. Most studies consider user-item interactions in the form of matrix, and take the recommendation as the matrix completion task [29]. Matrix factorization (MF) [30, 40], one of the most traditional collaborative filtering methods, learns user/item latent vectors to reconstruct the interaction matrix. Due to the success of deep learning

techniques, recent works use neural components [9, 24], such as multi-layer perceptions (MLP) to replace the conventional MF [20]. From the perspective of graph, the user-item interactions can be considered as a bipartite graph, where the nodes are users and items involved in the system and the user nodes are linked with those clicked item nodes. Given the user-item bipartite graph, graph neural network techniques can be utilized to capture the interactions of user-item relationships and learn efficient user/item embeddings. In addition to users' feedbacks, side information, such as social relationship and knowledge graph, is leveraged to improve recommendation performance, especially for data sparsity scenarios. The common strategies to incorporate side information, are either adding regularization terms or fusing the representations learnt from side information.

Sequential recommendation captures sequential patterns among successive items and recommends what the users might click next, i.e., next item(s) recommendation. Some works adopt Markov Chain (MC) [18, 44] to capture the item-to-item transition based on the assumption that the most recent clicked item reflects the user's dynamic preference. For example, FPMC [44] fuses MF and MC to combine users' static and dynamic preferences. Owing to the advantage of Recurrent Neural Network (RNN) in sequence modeling, some works employ the RNN unit to capture sequence patterns [21, 52]. To enhance the session representation, attention mechanism is leveraged to integrate the whole sequence in addition to the most recent item [32, 35]. Inspired by the outperformance of Transformer in NLP tasks, SASRec [25] leverages self-attention technique to model item interactions, which allows more flexibility to item-to-item transitions. With the emerging of GNNs, Some works transform a sequence of items into the graph data and utilize GNN to capture transition patterns.

2.2 Graph Neural Network Techniques

Graph neural network (GNN) techniques can capture the dependence of graphs via message passing between the nodes of graphs [82]. The main idea of GNN is how to iteratively aggregate feature information from neighbors and integrate the aggregated information with the current central node representation [71]. Recently, systems based on variants of GNN have demonstrated ground-breaking performance on many tasks related to graph data, such as physical systems [2, 45], protein structure [11], knowledge graph [15] and other areas [3, 4, 26]. In terms of whether using spectral convolution operation, existing GNN methods can be categorized into spectral methods and non-spectral methods. For spectral methods, the convolution operation is defined in the Fourier domain by computing the eigendecomposition of the graph Laplacian [5]. Since the operation of eigendecomposition requires high computational resources, researchers adopt the Chebyshev polynomials as the approximation [17, 27]. For example, graph convolutional network [27] uses first-order localized convolution operation to iteratively propagate the information from neighbors. All of the spectral models use the original graph structure to denote relations between nodes. The non-spectral methods design aggregator and updater respectively. The aggregator is responsible for collecting and aggregating the message of neighbors, while the updater aims to merge the message of the central node and its neighbors. In the aggregation step, existing works either treat each neighbor equally with mean-pooling operation [16, 34], or differentiate the importance of neighbors with attention mechanism [56]. In the update step, there are various ways to integrate the two representations, such as GRU mechanism [34], concatenation [16] and sum operation[56].

Among various GNN frameworks, graph convolutional network (GCN), graph attention network (GAT), gated graph neural network (GGNN), GraphSage are widely adopted. We briefly summarize these four methods:

- **GCN** [27] updates the embedding by $\mathbf{H}^{(l+1)} = \sigma\left(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{H}^{(l)}\mathbf{W}^{(l)}\right)$, where $\sigma(\cdot)$ is the activation function, $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ is the adjacency matrix of the undirected graph with added self-connections, and $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$.
- **GraphSage** [16] samples a fixed size of neighborhood, proposes mean/LSTM/pooling aggregator and adopt concatenation operation for update, $\mathbf{n}_v^{(l)} = \text{AGGREGATE}_t\left(\left\{\mathbf{h}_u^l, \forall u \in \mathcal{N}_v\right\}\right), \mathbf{h}_v^{(l+1)} = \sigma\left(\mathbf{W}^{(l)} \cdot \left[\mathbf{h}_v^{(l)} \| \mathbf{n}_v^{(l)}\right]\right)$.
- **GAT** [56] differentiates the influence of neighbors and updates the vector of each node by attending over its neighbors, $\alpha_{vj} = \frac{\exp\left(\text{LeakyReLU}\left(\mathbf{a}^T\left[\mathbf{W}^{(l)}\mathbf{h}_v^{(l)} \| \mathbf{W}^{(l)}\mathbf{h}_j^{(l)}\right]\right)\right)}{\sum_{k\in\mathcal{N}_v}\exp\left(\text{LeakyReLU}\left(\mathbf{a}^T\left[\mathbf{W}^{(l)}\mathbf{h}_v^{(l)} \| \mathbf{W}^{(l)}\mathbf{h}_k^{(l)}\right]\right)\right)}$, $\mathbf{h}_v^{(l+1)} = \sigma\left(\sum_{j\in\mathcal{N}_v}\alpha_{vj}\mathbf{W}^{(l)}\mathbf{h}_j^{(l)}\right)$.
- **GGNN** [34] adopts Gate Recurrent Units (GRU) in the update step, $\mathbf{n}_v^{(l)} = \frac{1}{|\mathcal{N}_v|}\sum_{j\in\mathcal{N}_v}\mathbf{h}_j^{(l)}$, $\mathbf{h}_v^{(l+1)} = \text{GRU}(\mathbf{h}_v^{(l)}, \mathbf{n}_v^{(l)})$.

2.3 Why Graph Neural Network for Recommendation

In the past few years, many graph neural network based recommender systems have been proposed. The motivation of applying graph neural network methods to recommender systems lies in two facets: (1) Most of the data in RS has essentially a graph structure. (2) GNN techniques are powerful in capturing connections among nodes and representation learning for graph data.

	i_1	i_2	i_3	i_4	i_5
u_1	0	1	1	0	0
u_2	0	0	1	0	1
u_3	1	0	0	1	0
u_4	1	0	1	1	1

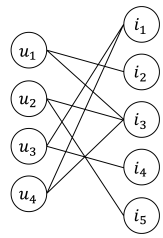
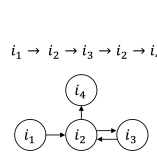


Fig. 1. User-item bipartite graph.



	Outgoing edges				Incoming edges			
	1	2	3	4	1	2	3	4
1	0	1	0	0	0	0	0	0
2	0	0	1/2	1/2	1/2	0	1/2	0
3	0	1	0	0	0	1	0	0
4	0	0	0	0	0	1	0	0

Fig. 2. Sequence graph and adjacency matrice.

	u_1	u_2	u_3	u_4
u_1	0	1	1	0
u_2	1	0	1	0
u_3	1	1	0	1
u_4	0	0	1	0

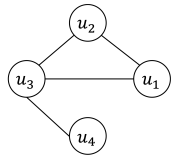


Fig. 3. Social relationship between users.

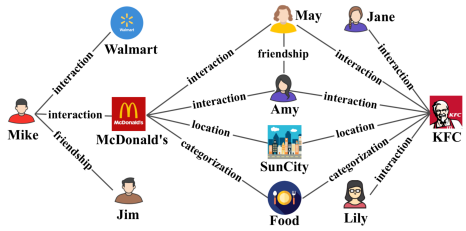


Fig. 4. Knowledge graph.

Recommender systems aim to recommend items for users based on user-item interactions and side information if available. The main point of RS is to learn efficient user/item representations with available information. Most of the existing works consider the data in RS from the perspective

of matrices. The users' implicit or explicit feedbacks towards items can be formatted as a matrix, denoted as $\mathcal{R}^{m \times n}$, where m is the total number of users and n is the total number of items. From the perspective of graph structure, users are connected with items if they once clicked or bought those items. Thus, the user-item interactive relationships can be regarded as the user-item bipartite graph, shown in Figure 1. For the sequential recommendation, the item can be connected with one or more subsequent items and a sequence of items can be transformed into the sequence graph. Figure 2 shows an example of a sequence graph where there is an edge between consecutive items, and the corresponding in and out adjacent matrice. Compared to the sequence data, the sequence graph allows more flexibility to item-to-item relatedness. Beyond that, some side information has naturally graph structure, such as social relationship and knowledge graph, shown in Figure 3,4.

Recently, GNN has been demonstrated to perform better than traditional graph learning methods, e.g., random-walk and graph embedding, in various domains [82]. For the user-item bipartite graph, GNN iteratively propagates information from interacted items and updates the user vector (the same for item), which can enhance user/item representations. Moreover, GNN could be leveraged to learn the compressed representations of the side information with graph structure as well. Further, the learnt embeddings of side information could be integrated with the embeddings from interaction data to boost the overall recommendation performance. Another strategy is to combine multiple graphs into one heterogeneous graph, then propagate information over the full graph.

3 CATEGORIES OF GRAPH NEURAL NETWORKS BASED RECOMMENDATION

In this section, we firstly propose a new taxonomy to classify the existing graph neural network based models. For each category, we summarize the main issues and the overall framework of models. Furthermore, we introduce representative models and illustrate what strategies they adopt to address those issues correspondingly. Before going further into different sections, we give the notations that will be used throughout the paper. The detailed descriptions of the notations could be found in Table 1.

We classify the graph neural network based models based on the types of information used. We further divide the existing models into the general and sequential recommendation in terms of whether to consider the order of items. Figure 5 summarizes the classification scheme.

The rationale of classification lies in two aspects: (1) different types of information have different characteristics of graph structure, which require corresponding GNN strategies; (2) the assumption behind the general and sequential recommendation is different since the former one considers static users' preferences while the latter one intends to capture users' dynamic preferences.

The information used in GNN based RS includes users' feedbacks towards items (pairwise user-item interaction or sequence behavior), social relationship between users and knowledge graph. The user-item bipartite graph has two types of nodes and the neighbors of each node are homogeneous. Sequence graph is a directed graph and the influence of both the previous item and the next item should be modeled. Social graph is homogeneous and the importance of neighbors should be differentiated since the social influence of friends depends on the strength of their relationship. Knowledge graph has various entities and relations, which increases the challenge of capturing information.

Here, we give a brief introduction to the categories of general and sequential recommendation:

- **General RS**: In these categories, models assume that users' preferences are invariant with time. We further categorize them into three subcategories based on the types of the information used. Without side information, existing models consider the user-item relationships as a user-item bipartite graph. With social relationship information, the GNN techniques are leveraged to mine social influence to augment users' representations. With knowledge graph

Table 1. Key notations used in this paper

Notations	Descriptions
$\mathcal{U}/\mathcal{I}$	The set of users/items
$\mathbf{u}$	Hidden vector of user u
$\mathbf{i}$	Hidden vector of item i
$\mathcal{R} = \{r_{u,i}\}$	Interaction between users and items
$\mathcal{G}_S$	Social relationship between users
$\mathcal{G}_{KG}$	Knowledge graph
$\mathcal{E}_{KG} = \{e_i\}$	The set of entities in Knowledge graph
$\mathcal{R}_{KG} = \{r_{e_i,e_j}\}$	The set of relations in Knowledge graph
$\mathbf{A}$	Adjacency matrix of graph
$\mathbf{A}^{in}/\mathbf{A}^{out}$	In & out adjacency matrix of directed graph
$\mathcal{N}_v$	Neighborhood set of node v
$\mathbf{h}_v^{(l)}$	Hidden state of node embedding at layer l
$\mathbf{n}_v^{(l)}$	Aggregated vector of node v's neighbors at layer l
$\mathbf{h}_u^*$	Final representation of user u
$\mathbf{h}_i^*$	Final representation of item i
$\mathbf{h}_u^S$	Final representation of user u in the social space
$\mathbf{h}_u^I$	Final representation of user u in the item space
$\mathbf{W}^{(l)}$	Transformation matrix at layer l
$\mathbf{W}_r^{(l)}$	Transformation matrix of relation r at layer l
$\mathbf{b}^{(l)}$	Bias term at layer l
$\|\|$	Vector concatenation
$\odot$	Element-wise multiplication operation

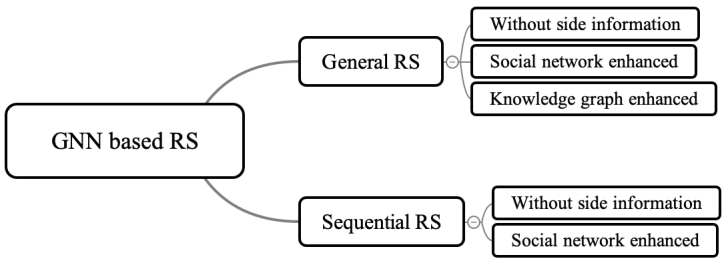


Fig. 5. Categories of graph neural network based recommendation models

information, the GNN methods are adopted to capture item-to-item relatedness to enhance items' representations.

- **Sequential RS**: For sequential recommendation, the core idea is to capture transition patterns in sequences for next item(s) recommendation. Most the existing works capture users' dynamic preferences only based on the sequences. They construct a sequence graph and adopt GNN methods to capture transition patterns. With social relationship information, GNN is utilized to model social influence.

Table 2 lists the representative models. We summarize them from the following perspectives: the types of information utilized, the types of graph used, and the adopted GNN techniques. Some of them will be discussed in detail in the following sections.

Table 2. Graph neural network based recommender systems.

Model	Venue	Year	Information				Graph Type	GNN Framework
			Pair.	Seq.	SN	KG		
GC-MC [54]	KDD	2018	√				Bipartite graph	variant of GCN (mean-pooling)
PinSage [73]	KDD	2018	√				Bipartite graph	variant of GraphSage (importance-pooling)
SpectralCF [81]	RecSys	2018	√				Bipartite graph	variant of GCN
STAR-GCN [76]	IJCAI	2019	√				Bipartite graph	variant of GCN (mean-pooling)
Bi-HGNN [31]	IJCAI	2019	√				Bipartite graph	GraphSage
DGCN-BinCF [57]	IJCAI	2019	√				Bipartite graph	variant of GCN (CrossNet)
NGCF [63]	SIGIR	2019	√				Bipartite graph	variant of GraphSage (sum updater)
IG-MC [77]	ICLR	2020	√				Bipartite graph	variant of GraphSage (sum updater)
LR-GCCF [6]	AAAI	2020	√				Bipartite graph	variant of GCN (w/o activation)
MCCF [65]	AAAI	2020	√				Bipartite graph	GAT
LightGCN [19]	arxiv	2020	√				Bipartite graph	variant of GCN (w/o activation & transformation)
HashGNN [51]	WWW	2020	√				Bipartite graph	variant of GCN (mean-pooling)
DiffNet [67]	SIGIR	2019	√		√		Social graph	GraphSage
GraphRec [10]	WWW	2019	√		√		Bipartite + Social graph	variant of GAT
DANSER [68]	WWW	2019	√		√		Social + Item-to-item graph	variant of GAT
DiffNet++ [66]	TKDE	2020	√		√		Bipartite + Social graph	variant of GAT
KGCN [60]	WWW	2019	√			√	Knowledge graph	variant of GAT
KGNN-LS [58]	KDD	2019	√			√	Knowledge graph	variant of GCN (user-specific adjacent matrix)
KGAT [62]	KDD	2019	√			√	Bipartite + Knowledge graph	variant of GAT
IntentGC [80]	KDD	2019	√			√	User-to-user + Item-to-item graph	variant of GraphSage (sum updater)
AKGE [46]	arxiv	2019	√			√	Knowledge graph	variant of GAT
SR-GNN [69]	AAAI	2019		√			Sequence graph	variant of GGNN (in & out adjacent matrice)
NISER [14]	CIKM	2019		√			Sequence graph	variant of GGNN (in & out adjacent matrice)
FGNN [42]	CIKM	2019		√			Sequence graph	variant of GAT
GC-SAN [72]	IJCAI	2019		√			Sequence graph	variant of GGNN (in & out adjacent matrice)
A-PGNN [70]	TKDE	2019		√			Sequence graph	variant of GGNN (in & out adjacent matrice)
MA-GNN [36]	AAAI	2020		√			Sequence graph	GraphSage
MGNN-Spred [61]	WWW	2020		√			Sequence graph	variant of GraphSage (sum updater)
DGRec [48]	WSDM	2019		√	√		Social graph	variant of GAT

4 GENERAL RECOMMENDATION

General recommendation aims to model user preferences by leveraging user-item interaction data, thus provide a static list of recommendations reflecting the long-term interests of each user. However, the performance of recommender systems degrades sharply in the face of data sparsity and the cold start scenarios. To address these two issues, researchers try to utilized side information, such as social networks and knowledge graphs.

In this section, we summarize the overall framework, introduce the representative methods under each subcategory, and discuss both their advantages and possible limitations.

4.1 Without side information

The key mathematical problem underlying recommender systems is matrix completion. Existing works learn the user representations and item representations by decomposing the interaction matrix and use the learned embeddings to infer the user's preferences for the target item. From the perspective of graphs, the interaction data in collaborative filtering can be represented by a bipartite graph between the user and item nodes. The matrix completion for recommender systems can be considered as the link prediction on graphs [33, 54]. In the user-item bipartite graph, the direct neighbors of users are all items and the direct neighbors of items are all users.

Owing to the superiority of Graph Neural Networks in learning on graph data, there are emerging efforts in recommender systems utilizing GNN architecture to capture higher-order interaction between users and items in the form of graphs. Applying GNN methods over the bipartite graph is essentially using the items clicked by users to enhance user representation and using the users once interacted with items to enrich item representation. Note that using the interacted item embeddings

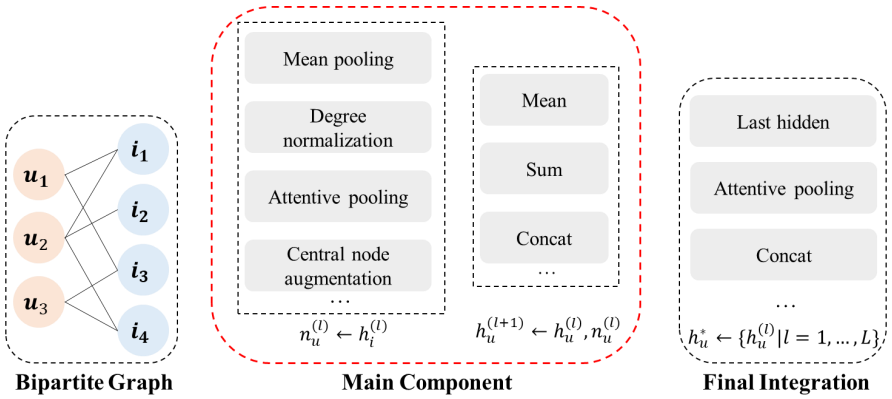


Fig. 6. The overall framework of GNN in general recommendation without side information.

to enhance user representation is not a completely new idea. SVD++ [28] enhances the latent factor based model by leveraging the interacted items of each user. One-layer GNN model on the bipartite graph can be seen as the enhanced version of SVD++, where the user/item representations are enhanced by their interacted items/users respectively. Multi-layer GNN methods enable to simulate information diffusion process more efficiently.

4.1.1 The overall framework. Given the user-item bipartite graph, the key challenge is how to propagate the information of items that the user once interacted with to the user and propagate the information of users that interacted with to the item.

To take the full advantage of GNN methods on the bipartite graph, there are four main issues to deal with:

- **Graph Construction** The set of user/item nodes and the interaction between users and items construct the full graph. Considering computational efficiency, how to sample representative neighbors for graph propagation instead of operating on the full graph?
- **Neighbor Aggregation** How to aggregate the information from neighbor nodes? Specifically, how much information should be propagated?
- **Information Update** How to integrate the central node representation and the aggregated representation of its neighbors?
- **Final Node Representation** Whether to use the node representation in the last layer or the combination of the node representations in all layers as the final node representation?

4.1.2 Representative methods. We briefly introduce some representative methods and illustrate what strategies they adopt to address the four issues respectively.

GC-MC [54][1] aims to tackle the rating prediction problem. The interaction data between users and items are represented by a bipartite user-item graph with labeled edges denoting observed ratings. Without side information, GC-MC models the user(item) node only with its interacted items (users) but ignores the original user (item) representation itself, i.e., users are represented by the aggregated representations of interacted items and items are represented by the aggregated representations of historical users. It adopts the mean-pooling to aggregate the neighbor nodes assuming that different items are equally representative to reflect the user preference and different

[1]- https://github.com/riannevdberg/gc-mc

Algorithm 1 Framework of GNN for User-item Bipartite Graph.

Require: The set of users $\mathcal{U}$; the set of items $\mathcal{I}$; the interactions between users and items $\mathcal{R} : \mathcal{U} \times \mathcal{I}$
Ensure: The overall user embedding $\mathbf{h}_u^*$ and item embedding $\mathbf{h}_i^*$

1: Construct the full graph $\mathcal{G}$ or the subgraph $\mathcal{G}_{u,i}$ by sampling;
2: Iteratively propagate the information of neighbors and update the node embedding;
3: **for** $l = 1$ to L **do** (Take the user node as an example)
4: $\quad \mathbf{n}_u^{(l)} = \text{Aggregator}(\mathbf{h}_u^{(l)}, \mathbf{h}_i^{(l)} | i \in \mathcal{N}_u)$
5: $\quad \mathbf{h}_u^{(l+1)} = \text{Updater}(\mathbf{h}_u^{(l)}, \mathbf{n}_u^{(l)})$
6: **end for**
7: Get the overall user/item representations $\mathbf{h}_u^* = f^*(\mathbf{h}_u^{(0)}, \cdots, \mathbf{h}_u^{(L)})$, $\mathbf{h}_i^* = f^*(\mathbf{h}_i^{(0)}, \cdots, \mathbf{h}_i^{(L)})$

users have equal influence on the interacted item. The propagation formula is as follows:

$$\mathbf{n}_u = \sigma\Big(\sum_{r \in \mathcal{R}_r} \sum_{i \in \mathcal{N}_r(u)} \frac{1}{|\mathcal{N}_v(r)|} \mathbf{W}_r \mathbf{h}_i\Big), \quad \mathbf{h}_u^I = \sigma\left(\mathbf{W}\mathbf{n}_u\right) \tag{1}$$

where $\sigma(\cdot)$ is the activation function, $\mathcal{R}_r$ denotes the set of rating types, $\mathcal{N}_u(r)$ is the degree of the node u for the rating type r and $\mathbf{W}_r$ is the trainable transformation matrix for the rating type r.

GC-MC only takes one-hop neighbors into consideration, which doesn't fully exploit the message passing through the graph structure. Discarding the original information of the user or item node completely might overlook the intrinsic user preference or the intrinsic item property.

STAR-GCN [76][2] is a stack of GCN blocks and the structure of each block is identical, e.g., GC-MC. The motivation of stacking GCN blocks instead of directly stacking multi-layer GCNs is that stacking too many convolution layers might bring about the over-smoothness issue. To bridge the adjacent blocks, STAR-GCN introduces the reconstruction mechanism which aims to recover the initial input node vectors from the aggregated representation and the recovered representations are the inputs of the next block. It worth noting that STAR-GCN pays attention to label leaky issue during training. It proposes to mask some nodes in the training stage and add a reconstruction loss on those masked nodes. The overall loss is the combination of the prediction loss and reconstruction loss.

STAR-GCN utilizes the reconstruction strategy to alleviate the over-smoothness issue and jointly optimizes the prediction task and reconstruction task, which results in better performance compared to GC-MC. The GC-MC and STAR-GCN treat the influence of neighbors equally and the message propagated from neighbors only depends on the neighbor node. Both of them only use the last layer of the node representation for rating prediction.

NGCF [63][3] decides the message propagation on both the graph structure and the affinity between the central node. Besides, it utilizes the representations of different layers to get the final node embedding, which takes the advantage of the residual network. The motivation is that (1) the embeddings of items in line with the user's interests should be passed more to the user (analogously for the items); (2) the representations obtained in different layers emphasize the messages passed over different connections. Specifically, NGCF employs element-wise product to augment the items' features the user cares about or the users' preferences for features the item has. Take the user node as an example,

$$\mathbf{n}_u^{(l)} = \sum_{i \in \mathcal{N}_u} \frac{1}{\sqrt{|\mathcal{N}_u|\,|\mathcal{N}_i|}} \left(\mathbf{W}_1^{(l)} \mathbf{h}_i^{(l)} + \mathbf{W}_2^{(l)} \left(\mathbf{h}_i^{(l)} \odot \mathbf{h}_u^{(l)}\right)\right) \tag{2}$$

[2]https://github.com/jennyzhang0215/STAR-GCN
[3]https://github.com/xiangwang1223/neural_graph_collaborative_filtering

The node embedding is updated by summing up the representations of itself and its neighbors with activation function LeakyReLU.

$$\mathbf{h}_u^{(l+1)} = \text{LeakyReLU}(\mathbf{h}_u^{(l)} + \mathbf{n}_u^{(l)}) \tag{3}$$

The representations in the lower layer reflect the individual feature more while those in the higher layer reflect the neighbor feature more. NGCF adopts the concatenation strategy to take full advantage of representations in different layers.

$$\mathbf{h}_u^I = \mathbf{h}_u^{(0)} \| \cdots \| \mathbf{h}_u^{(L)}, \quad \mathbf{h}_i^U = \mathbf{h}_i^{(0)} \| \cdots \| \mathbf{h}_i^{(L)}, \quad \hat{y}_{\text{NGCF}}(u,i) = \mathbf{h}_u^{I\top} \mathbf{h}_i^U \tag{4}$$

The outperformance of NGCF benefits from the residual strategy for the overall node representation and the affinity mechanism for message propagation.

The above methods apply GNN over the full graph without neighborhood sampling. On the one hand, it preserves the original graph structure; on the other hand, it cannot be applied to large-scale graphs with more node degrees and it relies heavily on the overall graph structure and has low generalization power.

PinSage [73] combines random walk strategy and graph convolutions to learn the embeddings of nodes. Unlike most of the existing works performing propagation over the full graph, PinSage designs a random-walk based sampling method to sample the fixed size of neighborhoods. Specifically, it simulates the process of random walks starting from the specific node, and compute the L_1 normalized visit counts of nodes. The neighborhood of the node u is the top T nodes with the highest normalized visit counts. In this way, those nodes that are not directly adjacent to the node u may also become its neighbors. Moreover, the normalized visit counts are used as the importance of neighbors when aggregating the vector representations of neighbors. Inspired by the GraphSage method, PinSage concatenates the aggregated neighbor vector $\mathbf{n}_u^{(l)}$ with u's current representation $\mathbf{h}_u^{(l)}$. The concatenation operation combined with nonlinear transformation allows more flexibility in feature combination. The normalization operation controls the scale of the node embedding and makes training more stable

$$\mathbf{h}_u^{(l+1)} = \text{ReLU}\left(\mathbf{W}^{(l)} \cdot (\mathbf{h}_u^{(l)} || \mathbf{n}_u^{(l)}) + \mathbf{b}^{(l)}\right), \quad \mathbf{h}_u^{(l+1)} = \mathbf{h}_u^{(l+1)} / \left\|\mathbf{h}_u^{(l+1)}\right\|_2 \tag{5}$$

The final node embeddings are the non-linear transformation of the representations in the last layer, i.e., $\mathbf{h}_u^I \leftarrow \mathbf{G}_2 \cdot \text{ReLU}\left(\mathbf{G}_1 \mathbf{h}_u^{(L)} + \mathbf{g}\right)$.

The experimental results show that the importance-pooling strategy performs better than mean-pooling. It means that the neighbors might not be equally representative for the central node. Owing to the random-walk sampling strategy, PinSage is scalable to web-scale recommendation tasks with millions of users and items. The sampling method gains efficiency at the cost of losing the original graph structure. The randomness of the neighborhood sampling strategy could influence the performance of the model.

IG-MC [77][4] focuses on the inductive matrix completion method without using side information while achieving similar or better performances than state-of-the-art transductive methods. It firstly constructs the one-hop subgraphs based on the user-item pairs rather than operating on the full graph. Specifically, given the target user and the target item, the item set is the combination of the target item and the items which the target user once interacted with, and the user set is the combination of the target user and the users who once interacted with the target item. The entities in the user and item sets are the nodes in the subgraph. The enclosing subgraph extraction design reduces the dependence on the original graph structure, which alleviates the performance degradation in case of sparsity and enhances its generalization for transferring the

[4]https://github.com/muhanzhang/IGMC

model to another dataset. It employs similar propagation strategies as GC-MC and the difference is that it maintains some information of the central node embedding, i.e., $\mathbf{h}_u^{(l+1)} = \mathbf{W}_0^{(l)}\mathbf{h}_i^{(l)} + \sum_{r\in\mathcal{R}_r}\sum_{i\in\mathcal{N}_u(r)}\frac{1}{|\mathcal{N}_r(u)|}\mathbf{W}_r^{(l)}\mathbf{h}_i^{(l)}$. IG-MC also employs concatenation operation to aggregate the information of different layers as the final user/item representations.

IG-MC proposes a novel subgraph construction strategy, which enables the inductive matrix completion and increases its generalization. In the case of data sparsity and cold start problems, IG-MC has a significant advantage over baselines, such as GC-MC; otherwise, it even gets poorer performance.

4.1.3 Summary. We briefly summarize the existing works on general recommendation without side information from the four perspectives.

Graph Construction GNN iteratively propagates the message from the one-hop neighbors layer-by-layer. The size of nodes exponentially increases with the number of layers. To apply GNN to large-scale graphs, it is necessary to sample the neighborhood. For example, PinSage adopts the random-walk strategy to sample the fixed-size neighborhood. IG-MC constructs the subgraph with the one-hop neighbors of the target user/item. Sampling is a trade-off between the original graph information and computational efficiency. IG-MC sacrifices more graph information while PinSage includes more randomness. In terms of transferring, the sampling method of IG-MC is preferable; otherwise, the strategy of PinSage might be better. The performance of the model depends on the sampling strategy and the more efficient sampling strategy for neighborhood construction deserves further studying.

Neighbor Aggregation As shown in Figure 6, the aggregation function can be categorized into four groups: "mean pooling" treats the neighbors indifferently; "degree normalization" assigns weights to nodes based on the graph structure; "attentive pooling" differentiates the importance of neighbors with attention mechanism and "central node augmentation" considers the affinity between nodes and uses the central node to filter the neighbors' message. Differentiating the influence of neighbors tend to improve the performance compared to mean-pooling or degree normalization. The reason would be that the interacted items are not equally representative to reflect user preferences.

Information Update Some works use the aggregated representation of neighbors as the new central node representation. Most of the works combine the representations of the central node and its neighbor. The common ways for integration are mean-pooling, sum-pooling, concatenation with transformation. The third one allows more feature interactions. When it is unnecessary for feature crossing, mean-pooling and sum-pooling are good enough.

Final Node Representation Some works use the node vector in the last layer of GNN as the final representation while some integrate the representations of all layers with weighted-pooling or concatenation operation. Since the output of different layers expresses different connections, utilizing the representations of all layers would be more likely to perform better.

4.2 Social network enhanced

With the emergence of online social networks, social recommender systems have been proposed to utilize each user's local neighbors' preferences to alleviate the data sparsity for better user embedding modeling [67]. The social network is a homogeneous graph, where each user is a node and edges exist between the users with social relationships. Previously, some works design an additional social regularization term to constrain that the users with social relationship have similar representations [23, 38, 39, 53]; some combine the user embedding with the average representation of her friends as the overall user representation [12, 37].

In practice, social influence recursively propagates and diffuses in the social network, i.e., a user might be influenced by her friends' friends. A major feature of graph neural networks is iterative propagation, which is consistent with the information diffusion process among friends. Therefore, researchers start to utilize GNNs to simulate how users are influenced by the recursive social diffusion process.

4.2.1 The overall framework. Existing studies incorporate social influence into recommender systems to leverage interaction behavior and relationships among users, as well as to improve their performance. Social graph has two special characteristics: (1) The edges between users represent social relationships, some of which are strong and some are weak. However, the strengths of social relationships are always unknown. (2) For recommender systems, it supplements the user-item interactions that the knowledge learnt from the social graph helps to enhance the user representations.

Corresponding to the characteristics of the social graph, there are two main issues to deal with:

- **Influence of Friends** Do friends have equal influence? If not, how to distinguish the influence of different friends?
- **Preference Integration** Users have social relationship with their friends and they also have interaction with items. How to integrate the user representations from the social influence perspective and interaction behavior? Figure 7 and 8 depict the two strategies respectively.

Algorithm 2 Framework of GNN for Social-enhanced Recommendation.

Require: The set of users $\mathcal{U}$; the set of items $\mathcal{I}$; the interactions between users and items $\mathcal{R} : \mathcal{U} \times \mathcal{I}$; the social relationship between users $\mathcal{G}_S$

Ensure: The overall user embedding $\mathbf{h}_u^*$ and item embedding $\mathbf{h}_i^*$

1: Iteratively propagate the information of friends and update the embedding of the target user;
2: **for** $l = 1$ to L **do**
3: $\alpha_{uv}^* = s(\mathbf{h}_u^{(l)}, \mathbf{h}_i^{(l)}, \mathbf{e}_{uv}), \quad \alpha_{uv} = \frac{\exp(\alpha_{uv}^*)}{\sum_{v \in \mathcal{N}_u} \exp(\alpha_{uv}^*)}$, where $s(\cdot)$ is the attention function
4: $\mathbf{n}_u^{(l)} = \sum_{v \in \mathcal{N}_u} \alpha_{uv} \mathbf{h}_v^{(l)}$
5: $\mathbf{h}_u^{(l+1)} = \text{Updater}(\mathbf{h}_u^{(l)}, \mathbf{n}_u^{(l)})$
6: **end for**
7: Get the user representation in the social space $\mathbf{h}_u^S = \mathbf{h}_u^{(L)}$
8: Get the user representation in the item space $\mathbf{h}_u^I$
9: Integrate the user representations in social space and item space to get the overall user representation $\mathbf{h}_u^*$

4.2.2 Representative methods. We introduce four representative methods in the social enhanced general recommendation, each of which adopts different strategies for either influence modeling or preference integration.

DiffNet [67][5] models users' preferences from her social relationships and historical behaviors. It adopts the graphSage framework to simulate how users are influenced by the recursive social diffusion process. Based on the assumption that friends have equal influence power, DiffNet uses the mean-pooling function to aggregate the representation of friends. Inspired by SVD++ [28] framework, DiffNet applies mean-pooling over the historical items to get the user preference in

[5]https://github.com/PeiJieSun/diffnet

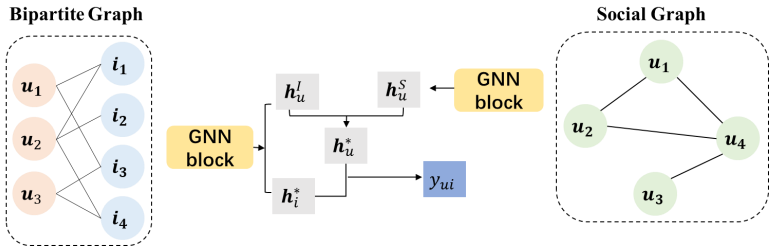


Fig. 7. The overall framework of GNN on two graphs in social-enhanced general recommendation.

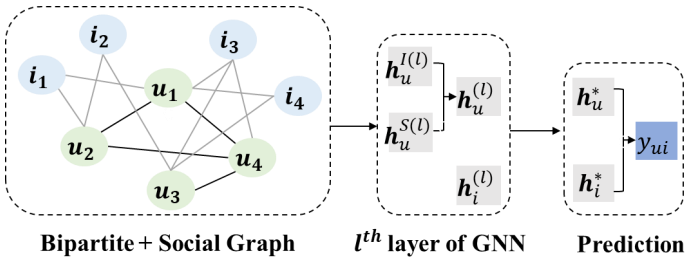


Fig. 8. The overall framework of GNN on the unified graph in social-enhanced general recommendation.

the item space and uses the user representation after the iterative diffusion process to replace the original user embedding.

$$\hat{r}_{ui} = \mathbf{h}_i^\top \left(\mathbf{h}_u^S + \sum_{j \in R_u} \frac{\mathbf{h}_j}{|R_u|} \right), \tag{6}$$

where the right term represents the overall user preference, and $\mathbf{R}_u$ denotes the set of historical items of user u. Note that taking the average representation of interacted item embeddings as the user preference in item space is equivalent to aggregate one-hop neighbors for the user vector. On the two experimental datasets, the optimal depth of diffusion is two layer.

Compared with TrustSVD, DiffNet captures a deeper social diffusion process by leveraging the GNN mechanism. The limitation might be that the assumption of equal influence is not in accordance with the actual situation. Users are more likely to be influenced by the friends with strong social ties or similar preferences. Besides, the representations of items could be enhanced by the interacted users as well.

GraphRec [10] considers users are involved in a social network graph and user-item bipartite graph and learns the user/item embeddings with graph attention network in these two graphs separately. The attention mechanism with concatenation operation is leveraged to differentiate the influence of neighbors. In the social graph, the influence of the friends depends on the similarity between their latent vectors.

$$\alpha_{uv}^* = \boldsymbol{v}_a^\top \text{ReLU}\left(\boldsymbol{W}_a \left[\mathbf{h}_v^{(l)} || \mathbf{h}_u^{(l)} \right] \right) \tag{7}$$

In the user-item bipartite graph, the representativeness of items for the user depends on not only the user and item embeddings but also the linked rating type. To get the overall user representation, it concatenates item-space representation and social-space representation and applies multi-layer

MLPs.

$$\mathbf{c}_1 = \left[\mathbf{h}_u^I \oplus \mathbf{h}_u^S\right] \quad \mathbf{c}_2 = \sigma\left(\mathbf{W}_2 \cdot \mathbf{c}_1 + \mathbf{b}_2\right) \quad \ldots \quad \mathbf{h}_u = \sigma\left(\mathbf{W}_l \cdot \mathbf{c}_{l-1} + \mathbf{b}_l\right) \tag{8}$$

The attention mechanism boosts the overall performance by differentiating the influence of friends and importance of items. However, the social influence might not be fully exploited since GraphRec only uses one-layer GAT to model social influence in the social graph.

DANSER [68][6] considers user-to-user social relationship and item-to-item relationship. Unlike most existing works assume that social effects from friends are static, DANSER leverages graph attention network to collaboratively learn representations for two-fold social effects, where one is modeled by a user-specific attention weight and the other is modeled by a dynamic and context-aware attention weight. The static influence of friends (social homophily) is captured by propagating inherent user preference vectors. The dynamic influence of friends (social influence) is captured by propagating context-aware user vectors, which depend on the target items. The context-aware user vectors are calculated as follows:

$$\mathbf{m}_u^{i^+} = \text{max-pooling}\left\{\mathbf{y}_j \otimes \mathbf{y}_{i^+} | j \in R_u\right\}, \tag{9}$$

where R_u is the set of items clicked by the user and $\mathbf{y}_j$ is the embedding of item j. Analogously, each item also has two attribute representations which are iteratively updated with graph attention networks. Furthermore, it proposes a policy-based fusion algorithm to dynamically allocate weights to the four interacted features according to specific user-item pairs. Overall, this work integrates static and dynamic user preferences under the influence of friends whereas it allows too much freedom for item-to-item relationship.

The above works either only modeling the social influence based on the social graph or firstly learn the user vectors in the social space and item space respectively and then integrate them into the overall user representations. Besides, none of the works take full advantage of the node embeddings in the different layers.

DiffNet++ [66] models the influence diffusion process and the interest diffusion in a unified framework. For the user nodes, it firstly aggregates the information of neighbors in the bipartite graph and social network by utilizing the GAT mechanism respectively. The attention mechanism is leveraged to fuse the two hidden states of neighbors. The representations of user nodes are further updated by adding the fused vectors. For the item nodes, the information of interacted users is propagated with the GAT mechanism as well. To capture the different depths of social relationships, DiffNet++ concatenates the hidden states of different GNN layers to be the overall node representations.

The outperformance of DiffNet++ mainly results from the two-depth diffusion process and the concatenation operation. The attention mechanism to integrate the bipartite graph and social graph performs better than average operation.

4.2.3 Summary. We briefly summarize the existing works on social enhanced general recommendation from two perspectives.

Influence Modeling Most of the existing works utilize the attention mechanism to differentiate the influence of friends. And the results demonstrate its better performance compared to the mean-pooling operation. The rationale is that users tend to have different social ties with different friends. Considering that the influence of friends might vary with items, DANSER also models dynamic user representation corresponding to the specific item as well. The dynamic friends' representation might help a lot when the items are diverse.

Preference Integration The users are involved in two types of networks, one is user-item bipartite graph and the other is social graph. Existing works combine the information from these two

[6]https://github.com/echo740/DANSER-WWW-19

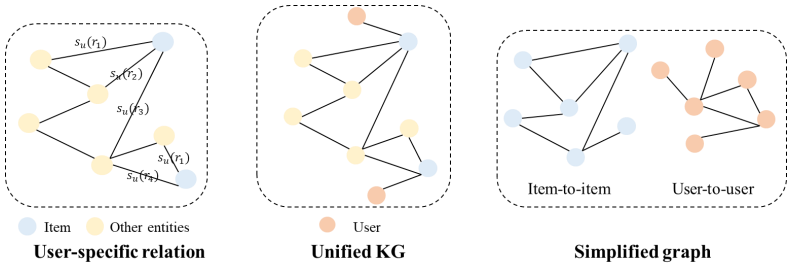


Fig. 9. The overall framework of GNN in knowledge graph enhanced general recommendation.

networks to enhance the user preference representation. Some works learn the user representation from these two networks respectively and then integrate them into the final preference vector. The advantage of this strategy is that the neighbors are homogeneous. Some works combine the two networks into one unified network and learn the preference representation with GNN. The advantage of this strategy is that the user representations are updated by two kinds of information simultaneously at each layer. Up till now, there is no evidence to show which strategy always has better performance.

4.3 Knowledge graph enhanced

Compared with KG-free methods, incorporating Knowledge Graph (KG) into recommendation benefits the results in three ways: (1) The rich semantic relatedness among items in a KG can help explore their latent connections and improve the precision of results; (2) The various types of relations in a KG are helpful for extending a user's interests reasonably and increasing the diversity of recommended items; (3) KG connects a user's historically-liked and recommended items, thereby bringing explainability to recommender systems.

Despite the rich information in KG, utilizing KG in RS is rather challenging due to its complex graph structure, i.e., multi-type entities and multi-type relations [13]. Previous works preprocess KG by knowledge graph embedding methods to learn the embeddings of entities and relations, such as [8, 59, 75, 79] or design meta-paths to aggregate the neighbor information [47, 50, 64, 74]. Inspired by the GNN in learning on graph data, researchers try to utilize GNN to capture the item-item relatedness.

4.3.1 The overall framework. Knowledge graph $\mathcal{G}$ consists of entity-relation-entity triples (h, r, t). Here $h \in \mathcal{E}, r \in \mathcal{R}$, and $t \in \mathcal{E}$ denote the head, relation, and tail of a knowledge triple respectively, where $\mathcal{E}$ and $\mathcal{R}$ are the set of entities and relations. Given the user-item interaction information as well as the knowledge graph, the knowledge graph enhanced recommendation is to take full advantage of the rich information in the knowledge graph to capture connectivity between items and further model the users' preferences for items. Considering the complex structure of knowledge graph, there are three main issues to deal with:

- **Graph Simplification** Considering the high complexity of KG structure, how to simplify the graph structure for efficient propagation?
- **Multi-relation Propagation** To propagate information from linked entities, both the entities and the relations should be taken into consideration.
- **User Integration** To utilize KG in recommendation, it is essential to fuse the role of users into the knowledge graph structure, e.g., user-specific item-item relatedness.

4.3.2 Representative methods. In the following part, we introduce four representative works in the knowledge graph enhanced general recommendation, each of which proposes corresponding strategies to address the main issues in this field.

KGCN [60][7] utilizes user-specific relation-aware graph neural network to aggregate the information of entities in the neighborhood. This work takes advantage of the knowledge graph to get semantic-aware item representations. For computational efficiency, it uniformly samples a fixed size of neighbors for each entity as their receptive field instead of applying any simplification to the knowledge graph structure. Users' preferences are assumed to be static, represented by a set of latent vectors, which are also used to measure the users' interests in relations. Neighbors are weighted by scores dependent on the connecting relation and specific user, which characterizes both the semantic information of KG and users' personalized interests in relation (Equation 10). The motivation is that different users would attach different importance to different relations.

$$a^u_{r_{e_i,e_j}} = \mathbf{u}^\top \mathbf{r}_{\mathbf{e_i},\mathbf{e_j}} \quad \tilde{a}^u_{r_{e_i,e_j}} = \frac{\exp\left(a^u_{r_{e_i,e_j}}\right)}{\sum_{k\in N(e_i)} \exp\left(a^u_{r_{e_i,e_k}}\right)} \tag{10}$$

The propagation algorithm belongs to graph attention network. Unlike the traditional GAT proposed by [56], the importance of neighbors is static given the specific user and the relation type. Note that the overall representations of the items are different for different users, which integrates the semantic information in the knowledge graph from the perspective of the users. The final prediction is the function of the user preference vector and the overall item representation.

KGNN-LS [58][8] aims to learn user-specific item embeddings by identifying important knowledge graph relationships for a given user as well. The novelty of this work is the label smoothness regularization, which posits that adjacent items in the knowledge graph are likely to have similar user relevance labels. KGNN-LS uses a user-specific relation scoring function that provides the importance of relation r for user u. Given the user-specific relation scoring function, the multiple relationships in the knowledge graph can be represented by a user-specific adjacent matrix. The construction of the adjacent matrix transforms the complex knowledge graph structure into a traditional graph, which makes the graph convolutional network applicable. Note that the user-personalized weighted graph makes the classical GCN similar to the GAT.

$$\begin{aligned} A^{ij}_u &= s_u\left(r_{e_i,e_j}\right) = g(\mathbf{u}, \mathbf{r}_{e_i,e_j}) \\ \mathbf{H}_{l+1} &= \sigma\left(\mathbf{D}_u^{-1/2}\mathbf{A}_u\mathbf{D}_u^{-1/2}\mathbf{H}_l\mathbf{W}_l\right), l = 0, 1, \cdots, L-1 \end{aligned} \tag{11}$$

where $g(\cdot)$ is a differentiable function such as inner product. Beyond that, KGNN-LS proposes a label smoothness regularization to help generalize to unobserved interactions more efficiently.

$$E\left(l_u, \mathbf{A}_u\right) = \frac{1}{2} \sum_{e_i\in\mathcal{E}, e_j\in\mathcal{E}} A^{ij}_u \left(l_u\left(e_i\right) - l_u\left(e_j\right)\right)^2, \tag{12}$$

where $l(\cdot)$ is the label function of the entity. Similar to the feature propagation among nodes, label smoothness is equivalent to a label propagation scheme on a graph. The ablation study demonstrates the efficacy of constraining the adjacent nodes with the same labels.

KGAT [62][9] considers user nodes as one type of entities in the knowledge graph and the interaction between users and items as one type of relation. In other words, KGAT integrates

[7]https://github.com/hwwang55/KGCN
[8]https://github.com/hwwang55/KGNN-LS
[9]https://github.com/xiangwang1223/knowledge_graph_attention_network

the two graphs into one graph. On the one hand, it unifies the propagation operations; on the other hand, a unified graph might introduce some noise. To learn the embeddings of nodes, it adopts the GAT mechanism to fully exploit the relationship between entities. Specifically, each node's embedding is refined by recursively propagating the embeddings of its neighbors and the importance of the neighbors is discriminated by the attention mechanism.

$$\begin{aligned} a(h,r,t) &= (\mathbf{W}_r\mathbf{e}_t)^\top \tanh\left((\mathbf{W}_r\mathbf{e}_h + \mathbf{e}_r)\right) \\ \tilde{a}(h,r,t) &= \frac{\exp(a(h,r,t))}{\sum_{(h,r',t')\in\mathcal{N}_h}\exp\left(a\left(h,r',t'\right)\right)} \end{aligned} \tag{13}$$

Note that the importance of neighbors depends on both the linked entities and their relationship. The representations of user preferences are also iteratively updated by the interacted items. Unlike previous works get user-specific item embeddings, KGCN gets the same item representations for all the users. Moreover, it employs TransR, a widely used method, to regularize the embeddings of each entity and relation with translation principle.

$$g(h,r,t) = \|\mathbf{W}_r\mathbf{e}_h + \mathbf{e}_r - \mathbf{W}_r\mathbf{e}_t\|_2^2 \quad \mathcal{L}_{\text{KG}} = \sum_{(h,r,t,t')\in\mathcal{T}} -\ln\sigma\left(g\left(h,r,t'\right) - g(h,r,t)\right) \tag{14}$$

It alternatively optimizes the knowledge graph regularization term and paired-wise score prediction term.

IntentGC [80] reconstructs user-to-user relationship and item-to-item relationship based on the multi-entity knowledge graph. For example, if users u_1 and u_2 are both connected by an auxiliary node, the type of the auxiliary node is denoted as the relationship between these two users. The advantage of this transformation is that it greatly simplifies the graph structure, i.e., it turns the multi-relation graph into two homogeneous graphs. The simplified graph structure further simplifies the propagation operations at the cost of losing partial information. The user embeddings and item embeddings are learned from two graphs separately. Considering that it is not necessary to learn all feature interactions between every pair of features in the concatenated vector, it proposes a more efficient vector-wise convolution operation.

$$\mathrm{g}_u^{k-1}(i) = \sigma\left(w_u^{k-1}(i,1)\cdot\mathbf{h}_u^{k-1} + \sum_{r=1}^{R-2} w_u^{k-1}(i,r+1)\cdot\mathbf{h}_{\mathcal{N}^{(r)}(u)}^{k-1}\right) \tag{15}$$

Note that each type of relation has its corresponding convolutional kernel. The experimental results show that IntentGC is much more efficient than GraphSage, i.e., the vector-wise convolution operation outperforms the concatenation operation.

AKGE [46] automatically extracts high-order subgraphs that link user-item pairs with rich semantics at the first step. The subgraph construction relies on the distance between paired nodes, which is calculated based on their embeddings, and the shortest path algorithm. For the iterative propagation, AKGE adopts an attention mechanism to aggregate the node embeddings of neighbors. The importance of neighbors depends on the linked nodes and their relation. For the update step, AGNN employs the GRU mechanism to integrate the neighbor representation and the central node representation.

$$\begin{aligned} \mathbf{a}_l^t &= \left(\mathbf{A}_{s_l}\odot\mathbf{Q}_l^t\right)\left[\hat{\mathbf{h}}_1^{t-1},\cdots,\hat{\mathbf{h}}_{|\mathcal{E}_s|}^{t-1}\right]^\top + \mathbf{b} \qquad \hat{\mathbf{h}}_k^t = g\left(\mathbf{h}_k^t \oplus \mathbf{r}_{l,k}\right) \\ \alpha_{l,k}^t &= \mathbf{w}_2^{\mathrm{T}}\cdot\left(\mathbf{W}_1\cdot\left[\hat{\mathbf{h}}_k^{t-1}\oplus\mathbf{h}_l^{t-1}\right] + \mathbf{b}_1\right) + b_2 \qquad q_{l,k}^t = \frac{\exp\left(\alpha_{l,k}^t\right)}{\sum_{e_j\in\mathcal{N}_l}\exp\left(\alpha_{l,j}^t\right)} \end{aligned} \tag{16}$$

The ablation study demonstrates the efficacy of shortest path based subgraph construction compared to the random sampling method and meta-path based method. It also illustrates the advantage of enhancing the node embedding with the category it belongs to and the attention design. The limitation of this work is that the subgraph construction depends on the pre-trained entity embeddings by the TransR model and the definition of distance measurement.

4.3.3 Summary. We briefly summarize the existing works on the knowledge graph enhanced general recommendation from the following aspects.

Graph Simplification The knowledge graph contains multiple entities and relationships, which increases the challenge of applying GNN on the graph for representation learning. To address this challenge, some works simplify the graph structure at the first stage. For example, AKGE reconstructs the subgraph based on the shortest path algorithm, and IntentGC only keeps the user-to-user relationship and item-to-item relationship that only one node apart. Graph simplification can improve computational efficiency at the cost of losing some graph information. Hence, an efficient simplification strategy deserves further investigation.

Multi-relation Propagation One characteristic of KG is multi-relation. To differentiate the importance of different relationships, an attention mechanism is widely adopted to aggregate the neighbors' information. The attention function greatly affects the performance of the methodology since it determines the information propagation. To distinguish the influence of neighbors, some works (e.g., KGCN and KGNN-LS) considers the user and the relation between nodes, and some (e.g., KGAT and AKGE) considers the connected nodes and their relation.

User Integration Knowledge graph is a type of side information in addition to the user-item bipartite graph. However, the scale of the KG is always larger than that of the bipartite graph. Some works adopt GNN to learn the representation of items and assume the users have static representations, which are used to measure the users' interests in relations. Some works integrate two graphs into a unified one and consider the users as a type of entities in the KG.

5 SEQUENTIAL RECOMMENDATION

Sequential recommendation predicts users' next preferences based on their most recent activities, which seeks to model sequential patterns among successive items, and generate well-timed recommendations for users [43]. Most the existing works focus on inferring temporal preference only from the sequence. Specifically, the sequence data is transformed into the sequence graph at the first stage and the sequential knowledge is captured by the GNN method. Side information can be utilized to enhance the sequential information as well, although there is little work in this field.

5.1 Without side information

Most the existing works employ Recurrent neural networks, such as LSTM [22] and GRU [7], to capture the transition of item choices. Inspired by the superiority of transformer [55] in modeling sequential patterns, some works utilize a self-attention mechanism to capture users' dynamic preferences. However, the users' preferences are much more complicated than a solely consecutive time pattern, and the inherent order is not the complete randomness by the self-attention [42]. Recent advances in graph neural networks (GNN) further boost the performance of sequential behavior prediction by modeling each behavior sequence as a graph to achieve the state-of-the-art performance [36].

5.1.1 The overall framework. From the perspective of adjacency between items, sequences of items can be modeled as graph-structured data. Based on the session graph, GNN can capture complex transitions of items, which are difficult to be revealed by previous conventional sequential methods [69]. To utilize GNN in the sequential recommendation, there are main issues to deal with:

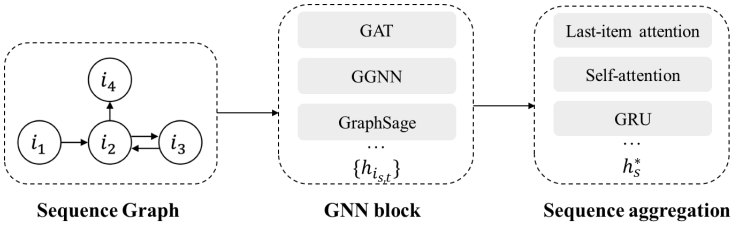


Fig. 10. The overall framework of GNN in sequential recommendation.

- **Graph Construction** To apply GNN in the sequential recommendation, the sequential data should be transformed into a sequence graph. The performance of the GNN model depends on the constructed sequence graph.
- **Information Propagation** Given the sequence graph, it is essential to design an efficient propagation mechanism to capture transition patterns.
- **Sequential Preference** To get the user's temporal preference, the representations in a sequence should be integrated.

Figure 10 illustrates the overall framework of GNN in sequential recommendation.

Algorithm 3 Framework of GNN for Sequential Recommendation.

Require: The set of items $\mathcal{I} = \{i_1, i_2, \cdots, i_m\}$, The set of sequences $\mathcal{S}$, each of which consists of a sequence of items, a sequence s can be represented by a list $s = [i_{s,1}, \cdots, i_{s,n}]$ ordered by timestamps.

Ensure: The overall sequence embedding $\mathbf{h}_s^*$

1: For each sequence s, construct the sequence graph by adding edges between the items interacted at t and $t + k$ ($k = 1, 2, \cdots, K$)
2: Iteratively propagate the information of the connected items and update the embeddings of items in the sequence;
3: Integrate the item representations in a sequence $\{\mathbf{h}_{i_{s,t}} | i_{s,t} \in s\}$ to get the overall sequence representation $\mathbf{h}_s^*$.

5.1.2 Representative methods. We briefly introduce the representative methods in sequential recommendation without side information in this subsection.

SR-GNN [69][10] constructs a directed session graph for each session sequence. Specifically, a directed edge exists between two consecutively clicked items, e.g., there is an edge from item $i_{s,t-1}$ to $i_{s,t}$. For the directed graph, SR-GNN defines two adjacent matrices that represent the weighted connections of outgoing $\mathbf{A}_s^{out}$ and incoming $\mathbf{A}_s^{in}$ edges in the session graph respectively. During the propagation process, the item embeddings are mapped into outgoing representations and incoming representations by $\mathbf{H}_1$, $\mathbf{H}_2$ separately. The information of previously clicked item and the next clicked item is propagated to the current item as follows:

$$\mathbf{a}_{s,j}^{(l+1)} = \left[\mathbf{A}_{s,j:}^{in}\mathbf{I}_s^{(l)}\mathbf{H}_1, \mathbf{A}_{s,j:}^{out}\mathbf{I}_s^{(l)}\mathbf{H}_2\right]^T + \mathbf{b}^{(l)}, \tag{17}$$

where $\mathbf{I}_s^{(l)}$ is the item embeddings of the session s at the l^{th} layer. Note that it assumes the items in the neighborhood are equally important since the normalized adjacent matrix assigns equal

[10]https:// github.com/CRIPAC-DIG/SR-GNN

weights to neighbors. To update the current item embedding, SR-GNN utilizes GRU mechanism to decide what information from neighbors to be propagated and the information from the current node to be maintained.

$$\mathbf{i}_{s,j}^{(l+1)} = \mathbf{GRU}(\mathbf{i}_{s,j}^{(l)}, \mathbf{a}_{s,j}^{(l+1)}) \tag{18}$$

To better predict the user's next click, it is essential to learn the session embeddings reflecting user's recent preference. SR-GNN combines the current interests and the general interests of the session. Specifically, it takes the embedding of the last clicked item in the L^{th} layer of GNN as the local embedding s_l of session s. The global embedding s_g aggregates the embeddings of the items in session s with soft-attention mechanism.

$$\alpha_j = \mathbf{q}^\top \sigma\left(\mathbf{W}_1 \mathbf{i}_n^{(L)} + \mathbf{W}_2 \mathbf{i}_j^{(L)} + \mathbf{c}\right) \quad \mathbf{s}_g = \sum_{j=1}^{n} \alpha_j \mathbf{i}_j^{(L)} \quad \mathbf{h}_s^* = \mathbf{W}_3[\mathbf{s}_l; \mathbf{h}_g], \tag{19}$$

where $\mathbf{q} \in \mathbb{R}^d$ and $\mathbf{W}_1, \mathbf{W}_2 \in \mathbb{R}^{d \times d}$ control the weights of item embedding vectors, and $\mathbf{W}_3 \in \mathbb{R}^{d \times 2d}$ compresses two combined embedding vectors into the overall session embedding.

GC-SAN [72] and [14] adopt the same graph construction and information propagation strategy as SR-GNN. GC-SAN adopts a self-attention mechanism to generate the session representation, which captures more interaction between items in the sequence. The recommendation results imply that GNN-based recommendation models are biased towards recommending popular items, and fail to recommend relevant long-tail items (less popular or less frequent items). To handle the popularity bias issue, NISER applies l2 normalization on both the item and session embeddings. The ablation study shows that the normalization operation efficiently alleviates the popularity bias. Moreover, to enhance the sequential information of item interactions, the positional embeddings are added to the item embeddings.

A-PGNN[70] pays attention to the role of users in the sequence and explicitly model the effect of historical sessions on the current session by attention mechanism. Most of the session-based recommendation deals with the occasion when users are unknown. Therefore, many session-based recommendation methods only consider all sessions of the user as a single sequence, ignoring the relationship among sessions. A-PGNN constructs the user behavior graph with her all session behaviors, which enriches the item-to-item connections. To integrate the influence of users, each item embedding is concatenated with the user embedding. The ablation study shows that incorporating user vectors makes little improvements in performance. The reason might be that the sequential behaviors are sufficient to reflect the users' temporal preferences. The edge weight is linearly proportional to the number of occurrences of the item and its next clicked item instead of treating neighbors equally. A-PGNN adopts the GGNN to capture the information transition among items as well. The novelty is that the current session representation is augmented by the integrated information contained in historical sessions. The effect of the historical sessions on the current session is leveraged by the attention mechanism using the current session embedding as the query. The ablation study demonstrates the effect of augmenting current session embedding with historical sessions to represent the user's preference.

FGNN [42] utilizes graph attention network (GAT) to capture the item transitions in the sequence graph. Specifically, FGNN assigns different weights to different neighbors, which depends on the linked edge and nodes rather than only depends on the adjacent matrix, and uses weighted sum to update the central node instead of a complex gated mechanism. There is no evidence showing that GAT is superior to the GGNN or not and their performance depends on the specific task.

$$\alpha_{ij} = \frac{\exp\left(\text{ LeakyRelu } \left(W_{att}\left[Wx_i || Wx_j || w_{ij}\right]\right)\right)}{\sum_{k \in N(i)} \exp\left(\text{ LeakyRelu } \left(W_{att}\left[Wx_i \| Wx_k \| w_{ik}\right]\right)\right)} \tag{20}$$

Based on all node embeddings for a session graph, FGNN learns a query vector indicating the order of reading from the memory for a graph with GRU and attention mechanism. This design emphasizes the order of items in the sequence, and the ablation study demonstrates its efficacy.

$$\begin{aligned} \boldsymbol{q}_t &= \mathrm{GRU}\left(\boldsymbol{q}^*_{t-1}\right) \quad e_{i,t} = f\left(\boldsymbol{x}_i, \boldsymbol{q}_t\right) \quad a_{i,t} = \frac{\exp\left(e_{i,t}\right)}{\sum_j \exp\left(e_{j,t}\right)} \\ \boldsymbol{r}_t &= \sum_i a_{i,t}\boldsymbol{x}_i \quad \boldsymbol{q}^*_t = \boldsymbol{q}_t \| \boldsymbol{r}_t \end{aligned} \tag{21}$$

MA-GNN integrates the static and dynamic user preferences with item co-occurrence patterns for sequential recommendation. MA-GNN measures the user's preference towards the target item from three aspects: user's static preference, short-term and long-term interests, and item co-occurrence patterns. Specifically, it leverages GNN to model the item contextual information within a short-term period. Note that the sequence graph construction is to extract three subsequent items and add edges between them, which treats the next three items indifferently.

$$\mathbf{h}_i = \tanh\left(\mathbf{W}^{(1)} \cdot \left[\sum_{k \in N_i} \mathbf{e}_k A_{i,k}; \mathbf{e}_i\right]\right), \forall i \in L_{u,l} \tag{22}$$

A shared memory network is utilized to model the long-term user interests. To balance the short-term and long-term interests, MA-GNN adopts a learnable gate to control how much the recent user interest representation and the long-term user interest representation can contribute to the dynamic user interests.

$$\begin{aligned} \mathbf{g}_{u,l} &= \sigma\left(\mathbf{W}_g^{(1)} \cdot \frac{1}{|L|}\sum_{i \in L_{u,l}} \mathbf{h}_i + \mathbf{W}_g^{(2)} \cdot \mathbf{p}^H_{u,l} + \mathbf{W}_g^{(3)} \cdot \mathbf{p}_u\right) \\ \mathbf{p}^C_{u,l} &= \mathbf{g}_{u,l} \odot \frac{1}{|L|}\sum_{i \in L_{u,l}} \mathbf{h}_i + \left(\mathbf{1}_d - \mathbf{g}_{u,l}\right) \odot \mathbf{p}^H_{u,l} \end{aligned} \tag{23}$$

5.1.3 Summary. We summarize the works in sequential recommendation without side information from the following perspectives.

Graph Construction To apply GNN in sequential recommendation, the first step is to construct the sequential graph based on users' sequential behaviors. Most works, such as SR-GNN, construct edges between two consecutively clicked items, i.e., the previous items will only affect the current item through the most recent item. MG-GNN extracts three subsequent items and adds edges between them, which equally treats the last three items. The sequential modeling depends on the constructed graph. Up till now, there is no evidence to suggest which strategy is better. In addition, existing works ignore the time interval between the consecutive items, which might further improve the performance.

Information Propagation Some studies employ mean-pooling to aggregate the neighbors while some utilize attention mechanism to differentiate the influence of neighbors. Since GRU has been demonstrated effective in sequential modeling, most the works adopt the GRU to update the central node representation.integrate the information of neighbors and the central node.

Sequential Preference The output of the GNN component is the node representations. The representations of a sequence of nodes need to be integrated to be the sequential representation. MA-GNN uses mean-pooling for aggregation, SR-GNN, GC-SAN and A-PGNN adopt attention mechanism, and FGNN combines GRU and attention. The motivation of using attention is that the items in the sequence are not equally important to reflect the sequential preference. FGNN uses

GRU to enhance the sequential relationship among items since it uses the attention mechanism in the information propagation stage.

5.2 Social network enhanced

Social network enhanced general recommendation assumes that users' preferences will be influenced by their friends and each users' interests are static. In sequential recommendation, users' interests are dynamic and users would be influenced by their friends as well. For social network enhanced general recommendation, some works only consider the social network as graph data, while others consider both the social information and user-item interaction in the perspective of the graph structure. Analogously, there are two strategies in social network enhanced sequential recommendation. There is little attention to the session-based recommendation with social network information. We will briefly introduce one representative work, named DGRec [48].

DGRec [48][11] only consider the social network as the graph data. Each user's dynamic interest is extracted from their most recent session behavior with the LSTM mechanism. The users' representations are dynamic in the social network. Considering that the social influence may vary with contexts, DGRec employs GAT to differentiate the influence of friends. The ablation study demonstrates the outstanding performance of DGRec compared to the general recommendation with the social network and the sequential recommendation without side information models. The result indicates the rationality of considering dynamic social influence in sequential recommendation.

6 FUTURE RESEARCH DIRECTIONS

Whilst GNNs have achieved great success in recommender systems, this section outlines several promising prospective research directions.

6.1 Efficient GNNs for Heterogeneous Graphs

The recent state-of-art works on graph neural networks (such as GCN, GAT and GGNN) are mainly for homogeneous graphs. However, most of the graph data in recommender systems is heterogeneous, such as the user-item bipartite graph and knowledge graph. The graph neural networks proposed in existing works of utilizing GNNs in recommendation can be considered as variants of basic GNN structures, as summarized in Table 2. For example, PinSage [73], as a variant of GraphSage, employs an importance sampling strategy for subgraph construction and an importance pooling method for neighborhood aggregation. LightGCN [19] simplifies the GCN structure by removing the feature transformation and nonlinear activation since the authors find out that these two design contributes little to the performance of collaborative filtering.

Considering the characteristics of graph data in recommendation, the more efficient GNN structures should be further investigated. Especially for heterogeneous graph, would a specific propagation and update mechanism, different from the existing strategies for homogeneous graph, further boost the performance of GNN in recommendation?

6.2 Multi-graph Information Integration

Side information has been demonstrated a high degree of effectiveness in improving recommender systems especially for the data sparsity and cold start scenarios. To fully use the side information, it is essential to efficiently integrate the side information and user-item interaction. Some works learn representations from different graphs separately and combine the vectors from different sources. For example, GraphRec [10] learns user embeddings from the social network and user-item bipartite graph respectively and applies concatenation and MLPs to integrate the embeddings from

[11]https://github.com/DeepGraphLearning/RecommenderSystems

these two graphs. Some works combine different graphs into one graph at the first step and apply GNN on the unified graph. For example, KGAT [62] combines the user-item bipartite graph and knowledge graph by considering users as one type of entities.

These two strategies have their own advantages and there is no evidence show that which one is better. Therefore, how to effectively integrate the information from different graphs need further studying.

6.3 Scalability of GNNs in Recommendation

Scalability is essential to the applicable of recommendation models in real-world systems and time complexity will also be a principal consideration. To deal with large scale graphs, existing works adopt the sampling methods to construct the subgraph. Most of them follow the sampling strategy proposed in GraphSage [16], some use a random walk strategy to get the neighborhood and some adopt the shortest path algorithm for subgraph construction. The sampling strategy for neighborhood construction does affect the efficiency of the GNN method.

The advantage of the subgraph is that it makes GNN applicable whatever the scale of the full graph. But its shortcoming is that the representation of the node should be recalculated for each propagation layer. Therefore, more future works should be studied on sampling strategy for neighborhood construction and batch generation. It can greatly improve time complexity if the node representations could be partially reused.

6.4 Sequence Graph Construction for Sequential Recommendation

Some information, such as social network and knowledge graph, already has been expressed in the graph structure. Recent works apply GNN in sequential recommendation by transforming sequence data into a sequence graph. and achieve great success. However, most studies connect two consecutive items and MA-GNN extracts three subsequent items and adds edges between them. Whether to connect the current item with just the next one or with its k subsequent items should be further studied. Besides, existing works ignore the item interval between the items. The items with different time intervals might have different influential power in transition. For example, Time-LSTM [83] extends the original LSTM structure with time gates and it greatly outperforms the original LSTM. Since the performance of GNN definitely relies on the sequence graph, the sequence graph construction should be further studied.

7 CONCLUSION

Utilizing GNNs in recommender systems has gain increasing interests in academia and industry. In this article, we provide a comprehensive review of the most recent works on GNN-based recommender systems. We proposed a classification scheme for organizing existing works. For each category, 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. Furthermore, we discuss some potential directions for future research. We hope this survey can provide readers with a general understanding of the recent progress in this field.

REFERENCES

[1] Gediminas Adomavicius and Alexander Tuzhilin. 2005. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE transactions on knowledge and data engineering* 17, 6 (2005), 734–749.

[2] Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, et al. 2016. Interaction networks for learning about objects, relations and physics. In *Advances in neural information processing systems*. 4502–4510.

[3] Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. 2018. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261* (2018).

[4] Daniel Beck, Gholamreza Haffari, and Trevor Cohn. 2018. Graph-to-Sequence Learning using Gated Graph Neural Networks. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 273–283.

[5] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. 2013. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203* (2013).

[6] Lei Chen, Le Wu, Richang Hong, Kun Zhang, and Meng Wang. 2020. Revisiting Graph based Collaborative Filtering: A Linear Residual Graph Convolutional Network Approach. *arXiv preprint arXiv:2001.10167* (2020).

[7] Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS 2014 Workshop on Deep Learning, December 2014*.

[8] Amine Dadoun, Raphaël Troncy, Olivier Ratier, and Riccardo Petitti. 2019. Location Embeddings for Next Trip Recommendation. In *Companion Proceedings of The 2019 World Wide Web Conference*. 896–903.

[9] Travis Ebesu, Bin Shen, and Yi Fang. 2018. Collaborative memory network for recommendation systems. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. 515–524.

[10] 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.

[11] Alex Fout, Jonathon Byrd, Basir Shariat, and Asa Ben-Hur. 2017. Protein interface prediction using graph convolutional networks. In *Advances in neural information processing systems*. 6530–6539.

[12] Guibing Guo, Jie Zhang, and Neil Yorke-Smith. 2015. Trustsvd: Collaborative filtering with both the explicit and implicit influence of user trust and of item ratings. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*.

[13] Qingyu Guo, Fuzhen Zhuang, Chuan Qin, Hengshu Zhu, Xing Xie, Hui Xiong, and Qing He. 2020. A Survey on Knowledge Graph-Based Recommender Systems. *arXiv preprint arXiv:2003.00911* (2020).

[14] Priyanka Gupta, Diksha Garg, Pankaj Malhotra, Lovekesh Vig, and Gautam Shroff. 2019. NISER: Normalized Item and Session Representations with Graph Neural Networks. *arXiv preprint arXiv:1909.04276* (2019).

[15] Takuo Hamaguchi, Hidekazu Oiwa, Masashi Shimbo, and Yuji Matsumoto. 2017. Knowledge transfer for out-of-knowledge-base entities: a graph neural network approach. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*. 1802–1808.

[16] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Advances in neural information processing systems*. 1024–1034.

[17] David K Hammond, Pierre Vandergheynst, and Rémi Gribonval. 2011. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis* 30, 2 (2011), 129–150.

[18] Ruining He and Julian McAuley. 2016. Fusing similarity models with markov chains for sparse sequential recommendation. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*. IEEE, 191–200.

[19] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation. *arXiv preprint arXiv:2002.02126* (2020).

[20] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*. 173–182.

[21] Balázs Hidasi and Alexandros Karatzoglou. 2018. Recurrent neural networks with top-k gains for session-based recommendations. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. 843–852.

[22] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[23] Mohsen Jamali and Martin Ester. 2010. A matrix factorization technique with trust propagation for recommendation in social networks. In *Proceedings of the fourth ACM conference on Recommender systems*. 135–142.

[24] Santosh Kabbur, Xia Ning, and George Karypis. 2013. Fism: factored item similarity models for top-n recommender systems. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. 659–667.

[25] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *2018 IEEE International Conference on Data Mining (ICDM)*. IEEE, 197–206.

[26] Elias Khalil, Hanjun Dai, Yuyu Zhang, Bistra Dilkina, and Le Song. 2017. Learning combinatorial optimization algorithms over graphs. In *Advances in Neural Information Processing Systems*. 6348–6358.

[27] Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *Proceedings of the 5th International Conference on Learning Representations*. 2873–2879.

[28] Yehuda Koren. 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. 426–434.

[29] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009), 30–37.
[30] Daniel D Lee and H Sebastian Seung. 2001. Algorithms for non-negative matrix factorization. In *Advances in neural information processing systems*. 556–562.
[31] Chong Li, Kunyang Jia, Dan Shen, CJ Shi, and Hongxia Yang. 2019. Hierarchical representation learning for bipartite graphs. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*. AAAI Press, 2873–2879.
[32] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. 2017. Neural attentive session-based recommendation. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. 1419–1428.
[33] Xin Li and Hsinchun Chen. 2013. Recommendation as link prediction in bipartite graphs: A graph kernel-based machine learning approach. *Decision Support Systems* 54, 2 (2013), 880–890.
[34] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. 2015. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493* (2015).
[35] Qiao Liu, Yifu Zeng, Refuoe Mokhosi, and Haibin Zhang. 2018. STAMP: short-term attention/memory priority model for session-based recommendation. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1831–1839.
[36] Chen Ma, Liheng Ma, Yingxue Zhang, Jianing Sun, Xue Liu, and Mark Coates. 2019. Memory Augmented Graph Neural Networks for Sequential Recommendation. *arXiv preprint arXiv:1912.11730* (2019).
[37] Hao Ma, Irwin King, and Michael R Lyu. 2009. Learning to recommend with social trust ensemble. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*. 203–210.
[38] Hao Ma, Haixuan Yang, Michael R Lyu, and Irwin King. 2008. Sorec: social recommendation using probabilistic matrix factorization. In *Proceedings of the 17th ACM conference on Information and knowledge management*. 931–940.
[39] Hao Ma, Dengyong Zhou, Chao Liu, Michael R Lyu, and Irwin King. 2011. Recommender systems with social regularization. In *Proceedings of the fourth ACM international conference on Web search and data mining*. 287–296.
[40] Andriy Mnih and Russ R Salakhutdinov. 2008. Probabilistic matrix factorization. In *Advances in neural information processing systems*. 1257–1264.
[41] Federico Monti, Michael Bronstein, and Xavier Bresson. 2017. Geometric matrix completion with recurrent multi-graph neural networks. In *Advances in Neural Information Processing Systems*. 3697–3707.
[42] Ruihong Qiu, Jingjing Li, Zi Huang, and Hongzhi Yin. 2019. Rethinking the Item Order in Session-based Recommendation with Graph Neural Networks. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 579–588.
[43] Massimo Quadrana, Paolo Cremonesi, and Dietmar Jannach. 2018. Sequence-aware recommender systems. *ACM Computing Surveys (CSUR)* 51, 4 (2018), 1–36.
[44] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing personalized markov chains for next-basket recommendation. In *Proceedings of the 19th international conference on World wide web*. 811–820.
[45] Alvaro Sanchez-Gonzalez, Nicolas Heess, Jost Tobias Springenberg, Josh Merel, Martin Riedmiller, Raia Hadsell, and Peter Battaglia. 2018. Graph networks as learnable physics engines for inference and control. *arXiv preprint arXiv:1806.01242* (2018).
[46] Xiao Sha, Zhu Sun, and Jie Zhang. 2019. Attentive Knowledge Graph Embedding for Personalized Recommendation. *arXiv preprint arXiv:1910.08288* (2019).
[47] Chuan Shi, Binbin Hu, Wayne Xin Zhao, and S Yu Philip. 2018. Heterogeneous information network embedding for recommendation. *IEEE Transactions on Knowledge and Data Engineering* 31, 2 (2018), 357–370.
[48] Weiping Song, Zhiping Xiao, Yifan Wang, Laurent Charlin, Ming Zhang, and Jian Tang. 2019. Session-based social recommendation via dynamic graph attention networks. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*. 555–563.
[49] Zhu Sun, Qing Guo, Jie Yang, Hui Fang, Guibing Guo, Jie Zhang, and Robin Burke. 2019. Research commentary on recommendations with side information: A survey and research directions. *Electronic Commerce Research and Applications* 37 (2019), 100879.
[50] Zhu Sun, Jie Yang, Jie Zhang, Alessandro Bozzon, Long-Kai Huang, and Chi Xu. 2018. Recurrent knowledge graph embedding for effective recommendation. In *Proceedings of the 12th ACM Conference on Recommender Systems*. 297–305.
[51] Qiaoyu Tan, Ninghao Liu, Xing Zhao, Hongxia Yang, Jingren Zhou, and Xia Hu. 2020. Learning to Hash with Graph Neural Networks for Recommender Systems. In *Proceedings of The Web Conference 2020*. 1988–1998.
[52] Yong Kiam Tan, Xinxing Xu, and Yong Liu. 2016. Improved recurrent neural networks for session-based recommendations. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*. 17–22.
[53] Jiliang Tang, Xia Hu, Huiji Gao, and Huan Liu. 2013. Exploiting local and global social context for recommendation. In *Twenty-Third International Joint Conference on Artificial Intelligence*.
[54] Rianne van den Berg, Thomas N Kipf, and Max Welling. 2018. Graph Convolutional Matrix Completion. (2018).

[55] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*. 5998–6008.

[56] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).

[57] Haoyu Wang, Defu Lian, and Yong Ge. 2019. Binarized collaborative filtering with distilling graph convolutional networks. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*. AAAI Press, 4802–4808.

[58] Hongwei Wang, Fuzheng Zhang, Mengdi Zhang, Jure Leskovec, Miao Zhao, Wenjie Li, and Zhongyuan Wang. 2019. Knowledge-aware graph neural networks with label smoothness regularization for recommender systems. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 968–977.

[59] Hongwei Wang, Fuzheng Zhang, Miao Zhao, Wenjie Li, Xing Xie, and Minyi Guo. 2019. Multi-task feature learning for knowledge graph enhanced recommendation. In *The World Wide Web Conference*. 2000–2010.

[60] Hongwei Wang Wang, Miao Zhao, Xing Xie, Wenjie Li, and Minyi Guo. 2019. Knowledge Graph Convolutional Networks for Recommender Systems. In *The World Wide Web Conference*. 1210–1221.

[61] Wen Wang, Wei Zhang, Shukai Liu, Qi Liu, Bo Zhang, Leyu Lin, and Hongyuan Zha. 2020. Beyond clicks: Modeling multi-relational item graph for session-based target behavior prediction. In *Proceedings of The Web Conference 2020*. 3056–3062.

[62] 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.

[63] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural graph collaborative filtering. In *Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval*. 165–174.

[64] Xiang Wang, Dingxian Wang, Canran Xu, Xiangnan He, Yixin Cao, and Tat-Seng Chua. 2019. Explainable reasoning over knowledge graphs for recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 5329–5336.

[65] Xiao Wang, Ruijia Wang, Chuan Shi, Guojie Song, and Qingyong Li. 2019. Multi-Component Graph Convolutional Collaborative Filtering. *arXiv preprint arXiv:1911.10699* (2019).

[66] Le Wu, Junwei Li, Peijie Sun, Yong Ge, and Meng Wang. 2020. DiffNet++: A Neural Influence and Interest Diffusion Network for Social Recommendation. *arXiv preprint arXiv:2002.00844* (2020).

[67] Le Wu, Peijie Sun, Yanjie Fu, Richang Hong, Xiting Wang, and Meng Wang. 2019. A neural influence diffusion model for social recommendation. In *Proceedings of the 42nd international ACM SIGIR conference on research and development in information retrieval*. 235–244.

[68] 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.

[69] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. 2019. Session-based recommendation with graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 346–353.

[70] Shu Wu, Mengqi Zhang, Xin Jiang, Xu Ke, and Liang Wang. 2019. Personalizing Graph Neural Networks with Attention Mechanism for Session-based Recommendation. *arXiv preprint arXiv:1910.08887* (2019).

[71] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. 2020. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems* (2020).

[72] 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 28th International Joint Conference on Artificial Intelligence*. AAAI Press, 3940–3946.

[73] 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 *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 974–983.

[74] Xiao Yu, Xiang Ren, Yizhou Sun, Bradley Sturt, Urvashi Khandelwal, Quanquan Gu, Brandon Norick, and Jiawei Han. 2013. Recommendation in heterogeneous information networks with implicit user feedback. In *Proceedings of the 7th ACM conference on Recommender systems*. 347–350.

[75] Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. 2016. Collaborative knowledge base embedding for recommender systems. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. 353–362.

[76] Jiani Zhang, Xingjian Shi, Shenglin Zhao, and Irwin King. 2019. STAR-GCN: stacked and reconstructed graph convolutional networks for recommender systems. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*. AAAI Press, 4264–4270.

[77] Muhan Zhang and Yixin Chen. 2019. Inductive matrix completion based on graph neural networks. *arXiv preprint arXiv:1904.12058* (2019).
[78] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. 2019. Deep learning based recommender system: A survey and new perspectives. *ACM Computing Surveys (CSUR)* 52, 1 (2019), 1–38.
[79] Yongfeng Zhang, Qingyao Ai, Xu Chen, and Pengfei Wang. 2018. Learning over knowledge-base embeddings for recommendation. *arXiv preprint arXiv:1803.06540* (2018).
[80] 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.
[81] Lei Zheng, Chun-Ta Lu, Fei Jiang, Jiawei Zhang, and Philip S Yu. 2018. Spectral collaborative filtering. In *Proceedings of the 12th ACM Conference on Recommender Systems*. 311–319.
[82] 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).
[83] Yu Zhu, Hao Li, Yikang Liao, Beidou Wang, Ziyu Guan, Haifeng Liu, and Deng Cai. 2017. What to Do Next: Modeling User Behaviors by Time-LSTM.. In *IJCAI*, Vol. 17. 3602–3608.