Graph Neural Networks in Recommender Systems: A Survey

SHIWEN WU, Peking University, China
FEI SUN, Alibaba Group, China
WENTAO ZHANG, XU XIE, and BIN CUI, Peking University, China

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 systems, there have always been emerging works in this field. In recommender systems, the main challenge is to learn the effective user/item representations from their interactions and side information (if any). Recently, graph neural network (GNN) techniques have been widely utilized in recommender systems since most of the information in recommender systems essentially has graph structure and GNN has superiority in graph representation learning. This article aims to provide a comprehensive review of recent research efforts on GNN-based recommender systems. Specifically, we provide a taxonomy of GNN-based recommendation models according to the types of information used and recommendation tasks. Moreover, we systematically analyze the challenges of applying GNN on different types of data and discuss how existing works in this field address these challenges. Furthermore, we state new perspectives pertaining to the development of this field. We collect the representative papers along with their open-source implementations in https://github.com/wusw14/GNN-in-RS.

CCS Concepts: • **Information Systems** → **Recommender systems;**

Additional Key Words and Phrases: Recommender system, graph neural network, survey

ACM Reference format:
Shiwen Wu, Fei Sun, Wentao Zhang, Xu Xie, and Bin Cui. 2022. Graph Neural Networks in Recommender Systems: A Survey. *ACM Comput. Surv.* 55, 5, Article 97 (December 2022), 37 pages.
https://doi.org/10.1145/3535101

1 INTRODUCTION

With the rapid development of e-commerce and social media platforms, recommender systems have become indispensable tools for many businesses [15, 25, 84, 183, 190, 200]. They can be recognized as various forms depending on industries, like product suggestions on online e-commerce websites (e.g., Amazon and Taobao) or playlist generators for video and music services (e.g.,

Shiwen Wu and Fei Sun contributed equally to this research.
This work is supported by NSFC (No. 61832001), Beijing Academy of Artificial Intelligence (BAAI), and PKU-Tencent Joint Research Lab.
Authors' addresses: S. Wu, School of CS and Key Lab of High Confidence Software Technologies (MOE), Peking University, Beijing, 100871, China; email: wushw.18@pku.edu.cn; F. Sun, Alibaba Group, Beijing, 100102, China; email: ofey.sf@alibaba-inc.com; W. Zhang (corresponding author), X. Xie, and B. Cui (corresponding author), School of CS and Key Lab of High Confidence Software Technologies (MOE), Institute of Computational Social Science, Peking University (Qingdao), Peking University, Beijing, 100871, China; emails: {wentao_zhang, xu.xie, bin.cui}@pku.edu.cn.
Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2022 Association for Computing Machinery.
0360-0300/2022/12-ART97 $15.00
https://doi.org/10.1145/3535101

YouTube, Netflix, and Spotify). Users rely on recommender systems to alleviate the information overload problem and explore what they are interested in from the vast sea of items (e.g., products, movies, news, or restaurants). Therefore, accurately modeling users' preferences from their historical interactions (e.g., click, watch, read, and purchase) lives at the heart of an effective recommender system.

Broadly speaking, in the past decades, the mainstream modeling paradigm in recommender systems has evolved from neighborhood methods [6, 60, 95, 123] to representation-learning-based frameworks [25, 77, 78, 125, 143]. Item-based neighborhood methods [6, 95, 123] directly recommend items to users that are similar to the historical items they have interacted with. In a sense, they represent users' preferences by directly using their historical interacted items. Early item-based neighborhood approaches have achieved great success in real-world applications because of their simplicity, efficiency, and effectiveness.

An alternative approach is representation-learning-based methods that try to encode both users and items as continuous vectors (i.e., embeddings) in a shared space, thus making them directly comparable. Representation-based models have sparked a surge of interest since the Netflix Prize competition [7] demonstrated that matrix factorization models are superior to classic neighborhood methods for recommendations. After that, various methods have been proposed to learn the representations of users and items, from matrix factorization [77, 78] to deep learning models [25, 58, 125, 200]. Nowadays, deep learning models have been a dominant methodology for recommender systems in both academic research and industrial applications due to the ability in effectively capturing the non-linear and non-trivial user-item relationships and easily incorporating abundant data sources, e.g., contextual, textual, and visual information.

Among all those deep learning algorithms, one line is graph-learning-based methods, which consider the information in recommender systems from the perspective of graphs [151]. Most of the data in recommender systems have a graph structure essentially [8, 190]. For example, the interaction data in a recommendation application can be represented by a bipartite graph between user and item nodes, with observed interactions represented by links. Even the item transitions in users' behavior sequences can also be constructed as graphs. The benefit of formulating recommendation as a task on graphs becomes especially evident when incorporating structured external information, e.g., the social relationship among users [33, 172] and the knowledge graph related to items [146, 196]. In this way, graph learning provides a unified perspective to model the abundant heterogeneous data in recommender systems. Early efforts in graph-learning-based recommender systems utilize graph embedding techniques to model the relations between nodes, which can be further divided into factorization-based methods, distributed-representation-based methods, and neural-embedding-based methods [151]. Inspired by the superior ability of GNN in learning on graph-structured data, a great number of GNN-based recommendation models have emerged recently.

Nevertheless, providing a unified framework to model the abundant data in recommendation applications is only part of the reason for the widespread adoption of GNN in recommender systems. Another reason is that, different from traditional methods that only implicitly capture the collaborative signals (i.e., using user-item interactions as the supervised signals), GNN can naturally and explicitly encode the crucial collaborative signal (i.e., topological structure) to improve the user and item representations. In fact, using collaborative signals to improve representation learning in recommender systems is not a new idea that originated from GNN [41, 69, 76, 184, 203]. Early efforts, such as SVD++ [76] and FISM [69], have already demonstrated the effectiveness of the interacted items in user representation learning. In view of the user-item interaction graph, these previous works can be seen as using one-hop neighbors to improve user representation learning.

The advantage of GNN is that it provides powerful and systematic tools to explore multi-hop relationships that have been proven to be beneficial to the recommender systems [55, 155, 190].

With these advantages, GNN has achieved remarkable success in recommender systems in the past few years. In academic research, a lot of works demonstrate that GNN-based models outperform previous methods and achieve new state-of-the-art results on the public benchmark datasets [55, 155, 210]. Meanwhile, plenty of their variants are proposed and applied to various recommendation tasks, e.g., session-based recommendation [115, 175], **point-of-interest (POI)** recommendation [10, 92, 177], group recommendation [59, 153], multimedia recommendation [164, 165], and bundle recommendation [11]. In industry, GNN has also been deployed in web-scale recommender systems to produce high-quality recommendation results [32, 114, 190]. For example, Pinterest developed and deployed a random-walk-based **Graph Convolutional Network (GCN)** algorithm model named PinSage on a graph with 3 billion nodes and 18 billion edges, and gained substantial improvements in user engagement in online A/B test.

Differences between this survey and existing ones. There exist surveys focusing on different perspectives of recommender systems [4, 16, 22, 28, 45, 117, 200]. However, there are very few comprehensive reviews that position existing works and current progress of applying GNN in recommender systems. For example, Zhang et al. [200] and Batmaz et al. [4] focus on most of the deep-learning techniques in recommender systems while ignoring GNN. Chen et al. [16] summarize the studies on the bias issue in recommender systems. Guo et al. [45] review knowledge-graph-based recommendations, and Wang et al. [150] propose a comprehensive survey in the session-based recommendations. These two works only include some of the GNN methods applied in the corresponding sub-fields and examine a limited number of works. To the extent of our knowledge, the most relevant survey published formally is a short paper [151], which presents a review of graph-learning-based systems and briefly discusses the application of GNN in recommendation. One recent survey under review [40] classifies the existing works in GNN-based recommender systems from four perspectives of recommender systems, i.e., stage, scenario, objective, and application. Such taxonomy emphasizes recommender systems but pays insufficient attention to applying GNN techniques in recommender systems. Besides, this survey [40] provides few discussions on the advantages and limitations of existing methods. There are some comprehensive surveys on the GNN techniques [179, 208], but they only roughly discuss recommender systems as one of the applications.

Given the impressive pace at which the GNN-based recommendation models are growing, we believe it is important to summarize and describe all the representative methods in one unified and comprehensible framework. This survey summarizes the literature on the advances of GNN-based recommendation and discusses open issues or future directions in this field To this end, more than 100 studies were shortlisted and classified in this survey.

Contribution of this survey. The goal of this survey is to thoroughly review the literature on the advances of GNN-based recommender systems and discuss further directions. The researchers and practitioners who are interested in recommender systems could have a general understanding of the latest developments in the field of GNN-based recommendation. The key contributions of this survey are summarized as follows:

• **New taxonomy.** We propose a systematic classification schema to organize the existing GNN-based recommendation models. Specifically, we categorize the existing works based on the type of information used and recommendation tasks into five categories: user-item collaborative filtering, sequential recommendation, social recommendation, knowledge-graph-based recommendation, and other tasks (including POI recommendation, multimedia recommendation, etc.).

Table 1. Key Notations Used in This Article

Notations	Descriptions
$\mathcal{U}/\mathcal{I}$	The set of users/items
$\mathbf{R} = \{r_{u,i}\}$	Interaction between users and items
$\mathcal{G}_{\text{S}}$	Social relationship between users
$\mathcal{G}_{\text{KG}}$	Knowledge graph
$\mathcal{E}_{\text{KG}} = \{e_i\}$	The set of entities in Knowledge graph
$\mathcal{R}_{\text{KG}} = \{r_{e_i,e_j}\}$	The set of relations in Knowledge graph
$\mathbf{A}$	Adjacency matrix of graph
$\mathbf{A}^{\text{in}}/\mathbf{A}^{\text{out}}$	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 representation 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
$\oplus$	Vector concatenation
$\odot$	Element-wise multiplication operation

• **Comprehensive review.** For each category, we demonstrate the main issues to deal with. Moreover, we introduce the representative models and illustrate how they address these issues.

• **Future research.** We discuss the limitations of current methods and propose nine potential future directions.

The remainder of this article is organized as follows: Section 2 introduces the preliminaries for recommender systems and graph neural networks. Then, it discusses the motivations of applying GNNs in recommender systems and categorizes the existing GNN-based recommendation models. Section 3 through 7 summarize the main issues of models in each category and how existing works tackle these challenges, and analyze their advantages and limitations. Section 8 gives a summary of the mainstream benchmark datasets, widely adopted evaluation metrics, and real-world applications. Section 9 discusses the challenges and points out nine future directions in this field. Finally, we conclude the survey in Section 10.

2 BACKGROUNDS AND CATEGORIZATION

Before diving into the details of this survey, we give a brief introduction to recommender systems and GNN techniques. We also discuss the motivation of utilizing GNN techniques in recommender systems. Furthermore, we propose a new taxonomy to classify the existing GNN-based models. Throughout this article, we use bold uppercase characters to denote matrices, bold lowercase characters to denote vectors, italic bold uppercase characters to denote sets, and calligraphic fonts to denote graphs. For easy reading, we summarize the notations that will be used throughout the article in Table 1.

2.1 Recommender Systems

Recommender systems infer users' preferences from user-item interactions or static features and further recommend items that users might be interested in [1]. It 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. Formally, the task is to estimate her/his preference for any item $i \in \mathcal{I}$ by the learned user representation h_u^* and item representation h_i^*, i.e.,

$$y_{u,i} = f(h_u^*, h_i^*), \tag{1}$$

where score function $f(\cdot)$ can be dot product, cosine, multi-layer perceptions, and so forth, and $y_{u,i}$ denotes the preference score for user u on item i, which is usually presented in probability.

According to the types of information used to learn user/item representations, the research of recommender systems can usually be classified into specific types of tasks. The *user-item collaborative filtering recommendation* aims to capture the collaborative signal by leveraging only the user-item interactions; i.e., the user/item representations are jointly learned from pairwise data [58, 78, 80, 121, 125, 178]. When the timestamps of the user's historical behavior are known or the historical behavior is organized in chronological order, the user representations can be enhanced via exploring the sequential patterns in her/his historical interactions [53, 61, 70, 85, 97, 119, 131, 136, 150]. According to whether the users are anonymous or not and whether the behaviors are segmented into sessions, works in this field can be further divided into *sequential recommendation* and *session-based recommendation*. The session-based recommendation can be viewed as a sub-type of sequential recommendation with anonymous and session assumptions [117]. In this survey, we do not distinguish them and refer to them collectively as the much broader term "sequential recommendation" for simplicity since our main focus is the contribution of GNN to recommendation, and the differences between them are negligible for the application of GNN. In addition to sequential information, another line of research exploits the social relationship to enhance the user representations, which is classified as *social recommendation* [43, 65, 103–105, 138]. The social recommendation assumes that the users with social relationships tend to have similar user representations based on the social influence theory that connected people would influence each other. Besides the user representation enhancement, a lot of efforts try to enhance the item representations by leveraging a knowledge graph, which expresses relationships between items through attributes. These works are always categorized as *knowledge-graph-based recommender systems*, which incorporate the semantic relations among items into collaborative signals.

2.2 Graph Neural Network Techniques

Recently, systems based on variants of GNN have demonstrated ground-breaking performances on many tasks related to graph data, such as physical systems [5, 122], protein structure [37], and knowledge graph [49]. In this part, we first introduce the definition of graphs, and then give a brief summary of the existing GNN techniques.

A graph is represented as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of nodes and $\mathcal{E}$ is the set of edges. Let $v_i \in \mathcal{V}$ be a node and $e_{ij} = (v_i, v_j) \in \mathcal{E}$ be an edge pointing from v_j to v_i. The neighborhood of a node v is denoted as $\mathcal{N}(v) = \{u \in \mathcal{V} | (v, u) \in \mathcal{E}\}$. Generally, graphs can be categorized as:

• **Directed/Undirected Graph.** A directed graph is a graph with all edges directed from one node to another. An undirected graph is considered as a special case of directed graphs where there is a pair of edges with inverse directions if two nodes are connected.

• **Homogeneous/Heterogeneous Graph.** A homogeneous graph consists of one type of nodes and edges, and a heterogeneous graph has multiple types of nodes or edges.

• **Hypergraph.** A hypergraph is a generalization of a graph in which an edge can join any number of vertices.

Given the graph data, the main idea of GNN is to iteratively aggregate feature information from neighbors and integrate the aggregated information with the current central node representation during the propagation process [179, 208]. From the perspective of network architecture, GNN stacks multiple propagation layers, which consist of the aggregation and update operations. The formulation of propagation is

$$\begin{aligned} \text{Aggregation:} \quad & \mathbf{n}_v^{(l)} = \text{Aggregator}_l\left(\left\{\mathbf{h}_u^l, \forall u \in \mathcal{N}_v\right\}\right), \\ \text{Update:} \quad & \mathbf{h}_v^{(l+1)} = \text{Updater}_l\left(\mathbf{h}_v^{(l)}, \mathbf{n}_v^{(l)}\right), \end{aligned} \tag{2}$$

where $\mathbf{h}_u^{(l)}$ denotes the representation of node u at the lth layer, and Aggregator_l and Updater_l represent the function of aggregation operation and update operation at the lth layer, respectively. In the aggregation step, existing works either treat each neighbor equally with the mean-pooling operation [50, 89] or differentiate the importance of neighbors with the attention mechanism [140]. In the update step, the representation of the central node and the aggregated neighborhood will be integrated into the updated representation of the central node. In order to adapt to different scenarios, various strategies are proposed to better integrate the two representations, such as GRU mechanism [89], concatenation with nonlinear transformation [50] and sum operation [140]. To learn more about GNN techniques, we refer the readers to the surveys [179, 208].

Here, we briefly summarize the aggregation and update operations of five typical GNN frameworks that are widely adopted in the field of recommendation.

• **GCN** [73] approximates the first-order eigendecomposition of the graph Laplacian to iteratively aggregate information from neighbors. Concretely, it updates the embedding by

$$\text{Aggregation:} \quad \mathbf{n}_v^{(l)} = \sum_{j \in \mathcal{N}_v} d_{vv}^{-\frac{1}{2}} \tilde{a_{vj}} d_{jj}^{-\frac{1}{2}} \mathbf{h}_j^{(l)}, \quad \text{Update:} \quad \mathbf{h}_v^{(l+1)} = \delta\left(\mathbf{W}^{(l)} \mathbf{n}_v^{(l)}\right), \tag{3}$$

where $\delta(\cdot)$ is the nonlinear activation function, like ReLU; $\mathbf{W}^{(l)}$ is the learnable transformation matrix for layer l; $\tilde{a_{vj}}$ is the adjacency weight ($\tilde{a_{vv}} = 1$); and $d_{jj} = \Sigma_k \tilde{a_{jk}}$.

• **GraphSAGE** [50] samples a fixed size of neighborhood for each node, proposes mean/sum/max-pooling aggregator, and adopts concatenation operation for update:

$$\begin{aligned} \text{Aggregation:} \quad & \mathbf{n}_v^{(l)} = \text{Aggregator}_l\left(\left\{\mathbf{h}_u^l, \forall u \in \mathcal{N}_v\right\}\right), \\ \text{Update:} \quad & \mathbf{h}_v^{(l+1)} = \delta\left(\mathbf{W}^{(l)} \cdot \left[\mathbf{h}_v^{(l)} \oplus \mathbf{n}_v^{(l)}\right]\right), \end{aligned} \tag{4}$$

where Aggregator_l denotes the aggregation function at the lth layer, $\delta(\cdot)$ is the nonlinear activation function, and $\mathbf{W}^{(l)}$ is the learnable transformation matrix.

• **GAT** [140] assumes that the influence of neighbors is neither identical nor pre-determined by the graph structure, and thus it differentiates the contributions of neighbors by leveraging the attention mechanism and updates the vector of each node by attending over its neighbors:

$$\begin{aligned} \text{Aggregation:} \quad & \mathbf{n}_v^{(l)} = \sum_{j \in \mathcal{N}_v} \alpha_{vj} \mathbf{h}_j^{(l)}, \alpha_{vj} = \frac{\exp\left(\text{Att}(h_v^{(l)}, \mathbf{h}_j^{(l)})\right)}{\sum_{k \in \mathcal{N}_v} \exp\left(\text{Att}(h_v^{(l)}, \mathbf{h}_k^{(l)})\right)}, \\ \text{Update:} \quad & \mathbf{h}_v^{(l+1)} = \delta\left(\mathbf{W}^{(l)} \mathbf{n}_v^{(l)}\right), \end{aligned} \tag{5}$$

where Att(·) is an attention function and a typical Att(·) is LeakyReLU($\mathbf{a}^T[\mathbf{W}^{(l)}\mathbf{h}_v^{(l)} \oplus \mathbf{W}^{(l)}\mathbf{h}_j^{(l)}]$), $\mathbf{W}^{(l)}$ is responsible for transforming the node representations at the lth propagation, and $\mathbf{a}$ is the learnable parameter.

• **GGNN** [89] adopts a gated **recurrent unit (GRU)** [89] in the update step:

$$\text{Aggregation:} \quad \mathbf{n}_v^{(l)} = \frac{1}{|\mathcal{N}_v|} \sum_{j \in \mathcal{N}_v} \mathbf{h}_j^{(l)}, \quad \text{Update:} \quad \mathbf{h}_v^{(l+1)} = \text{GRU}\left(\mathbf{h}_v^{(l)}, \mathbf{n}_v^{(l)}\right). \tag{6}$$

GGNN executes the recurrent function several times over all nodes [179], which might face the scalability issue when it is applied in large graphs.

• **HGNN** [36] is a typical hypergraph neural network, which encodes high-order data correlation in a hypergraph structure. The hyperedge convolutional layer is in the following formulation:

$$\text{Aggregation:} \quad \mathbf{N}^{(l)} = \mathbf{D}_v^{-\frac{1}{2}}\mathbf{E}\mathbf{W}^0\tilde{\mathbf{D}}_\mathbf{e}^{-1}\mathbf{E}^T\mathbf{D}_v^{-\frac{1}{2}}\mathbf{H}^{(l)}, \quad \text{Update:} \quad \mathbf{H}^{(l+1)} = \delta\left(\mathbf{W}^{(l)}\mathbf{N}^{(l)}\right), \tag{7}$$

where $\delta(\cdot)$ is the nonlinear activation function, like ReLU; $\mathbf{W}^{(l)}$ is the learnable transformation matrix for layer l; $\mathbf{E}$ is the hypergraph adjacent matrix; and $\mathbf{D}_e$ and $\mathbf{D}_v$ denote the diagonal matrices of the edge degrees and the vertex degrees, respectively.

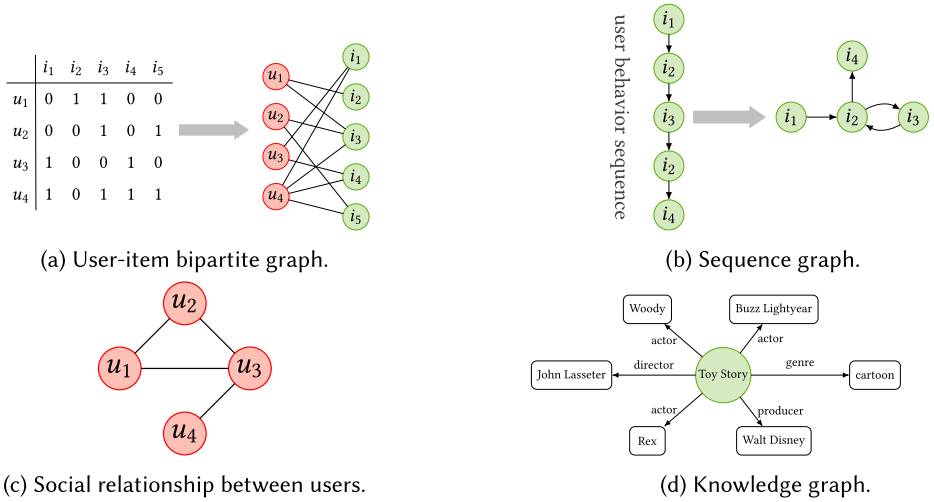
2.3 Why Graph Neural Network for Recommendation

In the past few years, many works on GNN-based recommendation have been proposed. Before diving into the details of the latest developments, it is beneficial to understand the motivations of applying GNN to recommender systems.

The most intuitive reason is that GNN techniques have been demonstrated to be powerful in representation learning for graph data in various domains [44, 208], and most of the data in recommendation has essentially a graph structure as shown in Figure 1. For instance, the user-item interaction data can be represented by a bipartite graph (as shown in Figure 1(a)) between the user and item nodes, where the link represents the interaction between the corresponding user and item. Besides, a sequence of items can be transformed into the sequence graph, where each item can be connected with one or more subsequent items. Figure 1(b) shows an example of a sequence graph where there is an edge between consecutive items. Compared to the original sequence data, a sequence graph allows more flexibility to item-to-item relationships. Beyond that, some side information also naturally has a graph structure, such as a social relationship and knowledge graph, as shown in Figures 1(c) and 1(d).

Due to the specific characteristic of different types of data in recommendation, a variety of models have been proposed to effectively learn their pattern for better recommendation results, which is a big challenge for the model design. Considering the information in recommendation from the perspective of the graph, a unified GNN framework can be utilized to address all these tasks. For example, the task of non-sequential recommendation is to learn the effective node representations, i.e., user/item representations, and to further predict user preferences. The task of sequential recommendation is to learn the informative graph representation, i.e., sequence representation. Both node representation and graph representation can be learned through GNN. Besides, it is more convenient and flexible to incorporate additional information (if available) compared to the non-graph perspective. For instance, the social network can be integrated into the user-item bipartite relationship as a unified graph. Both the social influence and collaborative signal can be captured during the iterative propagation.

Moreover, GNN can explicitly encode the crucial collaborative signal of user-item interactions to enhance the user/item representations through the propagation process. Utilizing collaborative signals for better representation learning is not a completely new idea. For instance, SVD++ [76]



(a) User-item bipartite graph.

(b) Sequence graph.

(c) Social relationship between users.

(d) Knowledge graph.

Fig. 1. Representative graph structures in recommender systems.

incorporates the representations of interacted items to enrich the user representations. ItemRank [41] constructs the item-item graph from interactions and adopts the random-walk algorithm to rank items according to user preferences. Note that SVD++ can be seen as using one-hop neighbors (i.e., items) to improve user representations, while ItemRank utilizes two-hop neighbors to improve item representations. Compared with the non-graph model, GNN is more flexible and convenient to model multi-hop connectivity from user-item interactions, and the captured CF signals in high-hop neighbors have been demonstrated to be effective for recommendation.

2.4 Categories of Graph-neural-network-based Recommendation

In this survey, we propose a new taxonomy to classify the existing GNN-based models. Based on the types of information used and recommendation tasks, the existing works are categorized into user-item collaborative filtering, sequential recommendation, social recommendation, knowledge-graph-based recommendation, and other tasks. In addition to the former four types of tasks, there are other recommendation tasks, such as POI recommendation, multimedia recommendation, and bundle recommendation. Since the studies utilizing GNN in these tasks are not that abundant, we group them into one category and discuss their current developments, respectively.

The rationale of classification is as follows: The graph structure depends to a large extent on the type of information. For example, a social network is naturally a homogeneous graph, and user-item interaction can be considered either a bipartite graph or two homogeneous graphs (i.e., user-user and item-item graphs). Besides, the information type also plays a key role in designing an efficient GNN architecture, such as aggregation and update operations and network depth. For instance, a knowledge graph has multi-type entities and relations, which requires considering such heterogeneity during propagation. Moreover, recommendation tasks are highly related to the type of information used. For example, the social recommendation is to make a recommendation by utilizing the social network information, and the knowledge-graph-based recommendation is to enhance the item representation by leveraging semantic relations among items in the knowledge graph. This survey is mainly for the readers interested in the development of GNN in recommender systems. Thus, our taxonomy is primarily from the perspective of recommender systems but also takes the GNN into account.

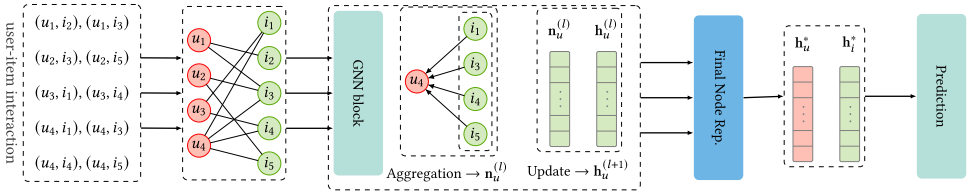


Fig. 2. The overall framework of GNN in user-item collaborative filtering.

3 USER-ITEM COLLABORATIVE FILTERING

Given the user-item interaction data, the basic idea of user-item collaborative filtering is essentially using the items interacted with by users to enhance user representations and using the users' once-interacted-with items to enrich item representations. Inspired by the advantage of GNN techniques in simulating the information diffusion process, recent efforts have studied the design of GNN methods, in order to exploit high-order connectivity from user-item interactions more efficiently. Figure 2 illustrates the pipeline of applying GNN to user-item interaction information.

To take full advantage of GNN methods on capturing collaborative signals from user-item interactions, there are four main issues to deal with:

• **Graph Construction.** Graph structure is essential for the scope and type of information to propagate. The original bipartite graph consists of a set of user/item nodes and the interactions between them. Should GNN be applied over the heterogeneous bipartite graph or should the homogeneous graph be constructed based on two-hop neighbors? Considering computational efficiency, how should representative neighbors be sampled for graph propagation instead of operating on the full graph?

• **Neighbor Aggregation.** How the information be aggregated from neighbor nodes–specifically, whether to differentiate the importance of neighbors, model the affinity between the central node and neighbors, or model the interactions among neighbors?

• **Information Update.** How the central node representation and the aggregated representation of its neighbors be integrated?

• **Final Node Representation.** Predicting the user's preference for the items requires the overall user/item representation. Should the node representation in the last layer or the combination of the node representations in all layers be used as the final node representation?

3.1 Graph Construction

Most works [8, 18, 55, 82, 132, 135, 142, 155, 173, 197, 205] apply the GNN on the original user-item bipartite graph directly. There are two issues in directly applying GNN on the original graph: one is effectiveness, that the original graph structure might not be sufficient enough for learning user/item representations; another one is efficiency, that aggregating the information of the full neighborhoods of nodes requires high computation cost especially for the large-scale graph [190].

One strategy to address the first issue is to enrich the original graph structure by adding edges, such as links between two-hop neighbors and hyperedges. For instance, Multi-GCCF [133] and DGCF [101] add edges between two-hop neighbors on the original graph to obtain the user-user and item-item graph. In this way, the proximity information among users and items can be explicitly incorporated into user-item interactions. DHCF [66] introduces the hyperedges and constructs the user/item hypergraphs in order to capture explicit hybrid high-order correlations. Another strategy is to introduce virtual nodes for enriching the user-item interactions. For example, DGCF [156] introduces virtual intent nodes and decomposes the original graph into a corresponding subgraph for each intent, which represents the node from different aspects and

has better expressive power. HiGNN [91] creates new coarsened user-item graphs by clustering similar users/items and taking the clustered centers as new nodes in order to explicitly capture hierarchical relationships among users and items.

In terms of the second issue, sampling strategies are proposed to make GNN efficient and scalable to large-scale graph-based recommendation tasks. PinSage [190] designs a random-walk-based sampling method to obtain the fixed size of neighborhoods with the highest visit counts. In this way, those nodes that are not directly adjacent to the central node may also become its neighbors. Multi-GCCF [133] and NIA-GCN [132] randomly sample a fixed size of neighbors. Sampling is a tradeoff between the original graph information and computational efficiency. The performance of the model depends on the sampling strategy, and the more efficient sampling strategy for neighborhood construction deserves further studying.

3.2 Neighbor Aggregation

The aggregation step is of the vital importance for information propagation for the graph structure, which decides how much neighbors' information should be propagated. Mean-pooling is one of the most straightforward aggregation operations [8, 133, 135, 197, 198], which treats neighbors equally:

$$\mathbf{n}_u^{(l)} = \frac{1}{|\mathcal{N}_u|}\mathbf{W}^{(l)}\mathbf{h}_i^{(l)}. \tag{8}$$

Mean-pooling is easy for implementation but might be inappropriate when the importance of neighbors is significantly different. Following the traditional GCN, some works employ "degree normalization" [18, 55, 173], which assigns weights to nodes based on the graph structure:

$$\mathbf{n}_u^{(l)} = \sum_{i \in \mathcal{N}_u} \frac{1}{\sqrt{|\mathcal{N}_u|\,|\mathcal{N}_i|}}\mathbf{W}^{(l)}\mathbf{h}_i^{(l)}. \tag{9}$$

Owing to the random-walk sampling strategy, PinSage [190] adopts the normalized visit counts as the importance of neighbors when aggregating the vector representations of neighbors. However, these aggregation functions determine the importance of neighbors according to the graph structure but ignore the relationships between the connected nodes.

Motivated by common sense that the embeddings of items in line with the user's interests should be passed more to the user (analogously for the items), MCCF [158] and DisenHAN [107] leverage the attention mechanism to learn the weights of neighbors [107, 146]. NGCF [155] 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; the aggregated neighbor representation is calculated as follows:

$$\mathbf{n}_u^{(l)} = \sum_{i \in \mathcal{N}_u} \frac{1}{\sqrt{|\mathcal{N}_u|\,|\mathcal{N}_i|}} \left(\mathbf{W}_1^{(l)}\mathbf{h}_i^{(l)} + \mathbf{W}_2^{(l)}\left(\mathbf{h}_i^{(l)} \odot \mathbf{h}_u^{(l)}\right)\right). \tag{10}$$

NIA-GCN [132] argues that existing aggregation functions fail to preserve the relational information within the neighborhood, and thus proposes the pairwise neighborhood aggregation approach to explicitly capture the interactions among neighbors. Concretely, it applies element-wise multiplication between every two neighbors to model the user-user/item-item relationships.

3.3 Information Update

Given the information aggregated from its neighbors, how to update the representation of the node is essential for iterative information propagation. According to whether to retain the information of the node itself, the existing methods can be divided into two directions. One is to discard the original information of the user or item node completely and use the aggregated representation

of neighbors as the new central node representation [8, 55, 156, 197], which might overlook the intrinsic user preference or the intrinsic item property.

Another is to take both the node itself ($\mathbf{h}_u^{(l)}$) and its neighborhood message ($\mathbf{n}_u^{(l)}$) into consideration to update node representations. The most straightforward way is to combine these two representations linearly with sum-pooling or mean-pooling operation [132, 155, 173, 198]. Inspired by the GraphSAGE [50], some works [82, 133, 190] adopt concatenation function with non-linear transformation to integrate these two representations as follows:

$$\mathbf{h}_u^{(l+1)} = \sigma\left(\mathbf{W}^{(l)} \cdot (\mathbf{h}_u^{(l)} \oplus \mathbf{n}_u^{(l)}) + \mathbf{b}^{(l)}\right), \tag{11}$$

where σ denotes the activation function, e.g., ReLU, LeakyReLU, and sigmoid. Compared to linear combination, the concatenation operation with feature transformation allows more complex feature interaction. LightGCN [55] and LR-GCCF [18] observe that nonlinear activation contributes little to the overall performance, and they simplify the update operation by removing the non-linearities, thereby retaining or even improving performance and increasing computational efficiency.

3.4 Final Node Representation

Applying the aggregation and update operations layer by layer generates the representations of nodes for each depth of GNN. The overall representations of users and items are required for the final prediction task.

A mainstream approach is to use the node vector in the last layer as the final representation, i.e., $\mathbf{h}_u^* = \mathbf{h}_u^{(L)}$ [8, 82, 135, 161, 190, 197]. However, the representations obtained in different layers emphasize the messages passed over different connections [155]. Specifically, the representations in the lower layer reflect the individual feature more, while those in the higher layer reflect the neighbor feature more. To take advantage of the connections expressed by the output of different layers, recent studies employ different methods to integrate the messages from different layers:

$$\begin{aligned}
\text{Mean-pooling:} \quad & \mathbf{h}_u^* = \frac{1}{L+1}\sum_{l=0}^{L}\mathbf{h}_u^{(l)}, \\
\text{Sum-pooling:} \quad & \mathbf{h}_u^* = \sum_{l=0}^{L}\mathbf{h}_u^{(l)}, \\
\text{Weighted-pooling:} \quad & \mathbf{h}_u^* = \frac{1}{L+1}\sum_{l=0}^{L}\alpha^{(l)}\mathbf{h}_u^{(l)}, \\
\text{Concatenation:} \quad & \mathbf{h}_u^* = \mathbf{h}_u^{(0)} \oplus \mathbf{h}_u^{(1)} \oplus \cdots \oplus \mathbf{h}_u^{(L)},
\end{aligned} \tag{12}$$

where $\alpha^{(l)}$ is a learnable parameter. Note that mean-pooling and sum-pooling can be seen as two special cases of weighted pooling. Compared to mean-pooling and sum-pooling, weighted pooling allows more flexibility to differentiate the contribution of different layers. Among these four methods, the former three all belong to the linear operation, and only the concatenation operation preserves information from all layers.

3.5 Summary

Corresponding to the discussion at the beginning of this section, we briefly summarize the existing works from four issues:

• **Graph Construction.** The most straightforward way is to directly use the original user-item bipartite graph. If some nodes have few neighbors in the original graph, it would be beneficial to

enrich the graph structure by adding either edges or nodes. When dealing with large-scale graphs, it is necessary to sample the neighborhood for computational efficiency. Sampling is a tradeoff between effectiveness and efficiency, and a more effective sampling strategy deserves further study.

• **Neighbor Aggregation.** When neighbors are more heterogeneous, aggregating neighbors with attentive weights would be preferable to equal weights and degree normalization; otherwise, the latter two are preferable for easier calculation. Explicitly modeling the influence among neighbors or the affinity between the central node and neighbors might bring additional benefits but needs to be verified on more datasets.

• **Information Update.** Compared to discarding the original node, updating the node with its original representation and the aggregated neighbor representation would be preferable. Recent works show that simplifying the traditional GCN by removing the transformation and nonlinearity operation can achieve better performance than the original ones.

• **Final Node Representation.** To obtain overall user/item representation, utilizing the representations from all layers is preferable to directly using the last layer representation. In terms of the function of integrating the representations from all layers, weighted-pooling allows more flexibility, and concatenation preserves information from all layers.

Figure 3 summarizes the typical strategies for each of the main issues and lists the representative works accordingly.

4 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 accurate recommendations for users [117]. From the perspective of adjacency between items, sequences of items can be modeled as graph-structured data. Inspired by the advantage of GNN, it is becoming popular to utilize GNN to capture the transition pattern from users' sequential behaviors by transforming them into the sequence graph.

Figure 4 illustrates the overall framework of GNN in sequential recommendation. To fully utilize GNN in the sequential recommendation, there are three main issues to deal with:

• **Graph Construction.** To apply GNN in the sequential recommendation, the sequence data should be transformed into a sequence graph. Is it sufficient to construct a subgraph for each sequence independently? Would it be better to add edges among several consecutive items than only between the two consecutive items?

• **Information Propagation.** To capture the transition patterns, which propagation mechanism is more appropriate? Is it necessary to distinguish the sequential order of the linked items?

• **Sequential Preference.** To get the user's temporal preference, the item representations in a sequence should be integrated. Should one simply apply attentive pooling or leverage RNN structure to enhance consecutive time patterns?

4.1 Graph Construction

Unlike the user-item interactions, which have essentially a bipartite graph structure, the sequential behaviors are naturally expressed in the order of time, i.e., sequences, instead of sequence graphs. Constructing a graph based on the original bipartite graph is optional and mainly driven by the scalability or heterogeneity issue, whereas the construction of a sequence graph based on users' sequential behaviors is a necessity for applying GNN in sequential recommendation. Figure 5 shows the representative graph construction strategies for sequential behaviors.

Constructing the directed graph for each sequence by treating each item in the sequence as a node and adding edges between two consecutively clicked items is the most straightforward way [48, 115, 116, 175, 185]. However, in most scenarios, the length of the user sequence is short;

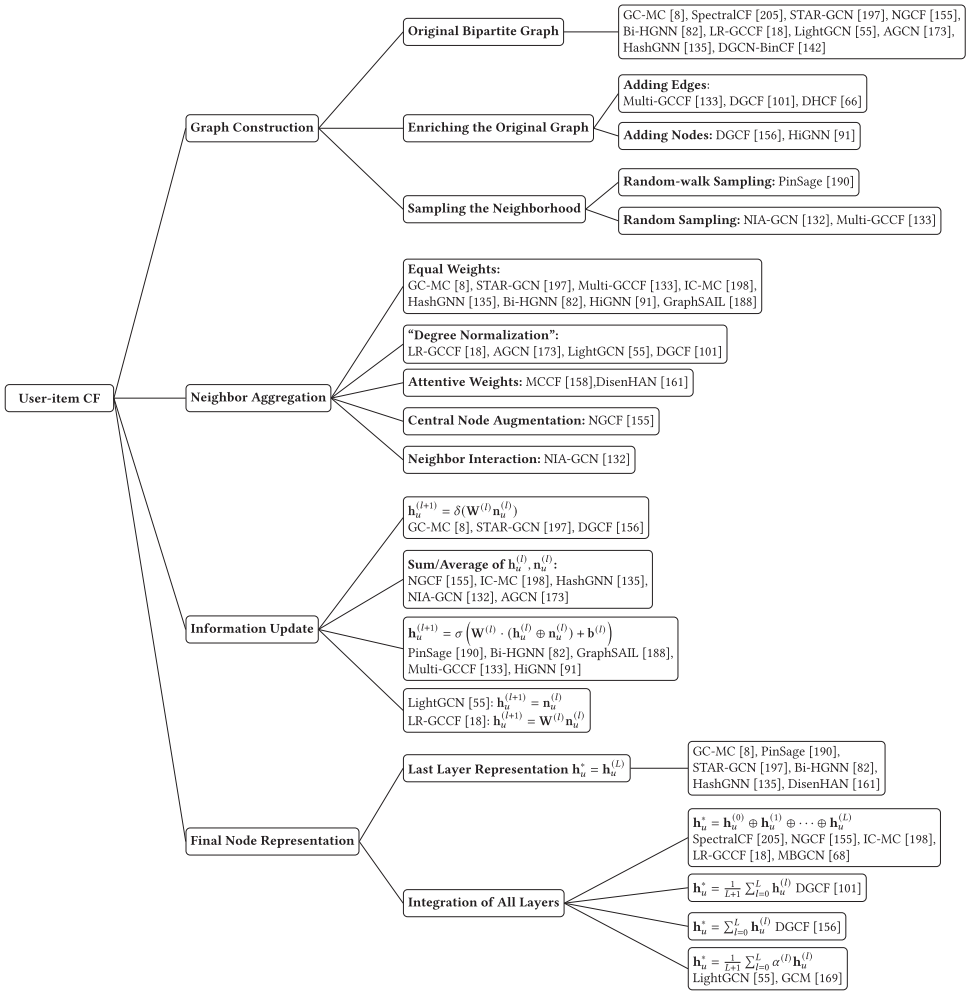


Fig. 3. Summary of representative works in user-item collaborative filtering.

e.g., the average length on the preprocessed Yoochoose1/4[1] dataset is 5.71 [175]. A sequence graph constructed from a single and short sequence consists of a small number of nodes and connections, and some nodes might even have only one neighbor, which contains too limited knowledge to reflect users' dynamic preferences and cannot take full advantage of GNN in graph learning. To tackle this challenge, recent works propose several strategies to enrich the original sequence graph structure, which can be divided into two mainstreams.

One mainstream is to utilize additional sequences to enrich the item-item transitions. The additional sequences can be other types of behavior sequences [152], the historical sequences of the same user [176], or part/all of the sequences in the whole dataset [14, 163, 206, 207]. For instance, HetGNN [152] utilizes all behavior sequences and constructs edges between two consecutive items in the same sequence with their behavior types as the edge types. A-PGNN [176] deals with the

[1]The dataset is available at http://2015.recsyschallenge.com/challege.html. Note that this work preprocesses the dataset by filtering out the sequences of length 1 and items appearing less than five times.

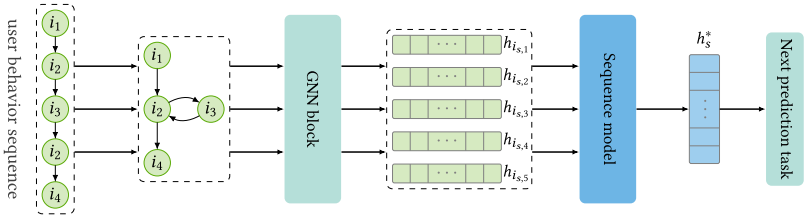


Fig. 4. The overall framework of GNN in sequential recommendation.

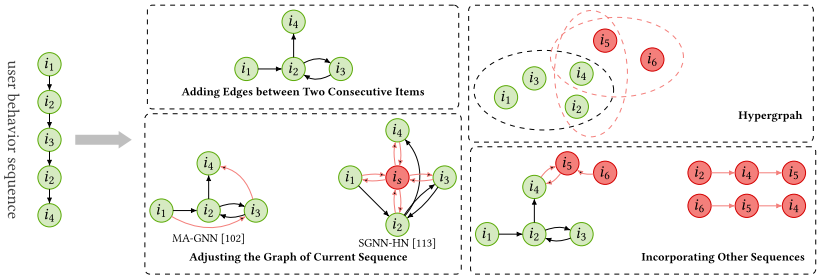


Fig. 5. Representative graph construction methods in the sequential recommendation, where the original nodes and edges are in green, and the additional ones are in red.

occasion when users are known, thus incorporating the user's historical sequences with the current sequence to enrich the item-item connections. GCE-GNN [163] and DAT-MDI [14] exploit the item transitions in all sessions to assist the transition patterns in the current sequence, which leverage the local context and global context. Different from GCE-GNN [163] and DAT-MDI [14] that treats all the transitions equally, TASRec [207] attaches more importance to the recent transitions to augment the more recent transitions. Instead of incorporating all the sessions, DGTN [206] only adds similar sessions to the current session, based on the assumption that similar sequences are more likely to reflect similar transition patterns. All these methods introduce more information into the original graph and improve the performance compared to a single sequence graph.

Another mainstream approach is to adjust the graph structure of the current sequence. For example, assuming the current node has a direct influence on more than one consecutive item, MA-GNN [102] extracts three subsequent items and adds edges between them. Considering that only adding edges between consecutive items might neglect the relationships between distant items, SGNN-HN [113] introduces a virtual "star" node as the center of the sequence, which is linked with all the items in the current sequence. The vector-wise representation of the "star" node reflects the overall characteristics of the whole sequence. Hence, each item can gain some knowledge of the items without direct connections through the "star" node. Chen and Wong [19] point out that existing graph construction methods ignore the sequential information of neighbors and bring about the ineffective long-term capturing problem. Therefore, they propose LESSR, which constructs two graphs from one sequence: one distinguishes the order of neighbors, and the other allows the short-cut path from the item to all the items after it.

In addition to the above two mainstreams, other graph construction methods have emerged recently. Inspired by the advantage of the hypergraph in modeling beyond-pairwise relations, the hypergraph has been leveraged to capture the high-order relations among items and the cross-session information. SHARE [148] constructs a hypergraph for each session, of which the hyperedges are defined by various sizes of sliding windows. DHCN [182] takes each session as one hyperedge and

integrates all the sessions in one hypergraph. To explicitly incorporate cross-session relationships, DHCN [182] and COTREC [181] construct the session-to-session graph, which takes each session as a node and assigns the weights based on the shared items.

4.2 Information Propagation

Given a built sequence graph, it is essential to design an efficient propagation mechanism to capture transition patterns among items. The GGNN framework is widely adopted to propagate information on the directed graph. Specifically, it employs mean-pooling to aggregate the information of the previous items and the next items, respectively; combines the two aggregated representations; and utilizes the GRU [89] component to integrate the information of neighbors and the central node. The propagation functions are given as follows:

$$\begin{aligned} \mathbf{n}_{i_{s,t}}^{\text{in}(l)} &= \frac{1}{|\mathcal{N}_{i_{s,t}}^{\text{in}}|}\Sigma_{j\in\mathcal{N}_{i_{s,t}}^{\text{in}}}\mathbf{h}_j^{(l)}, \quad \mathbf{n}_{i_{s,t}}^{\text{out}(l)} = \frac{1}{|\mathcal{N}_{i_{s,t}}^{\text{out}}|}\Sigma_{j\in\mathcal{N}_{i_{s,t}}^{\text{out}}}\mathbf{h}_j^{(l)}, \\ \mathbf{n}_{i_{s,t}}^{(l)} &= \mathbf{n}_{i_{s,t}}^{\text{in}(l)} \oplus \mathbf{n}_{i_{s,t}}^{\text{out}(l)}, \quad \mathbf{h}_{i_{s,t}}^{(l+1)} = \mathbf{GRU}\left(\mathbf{h}_{i_{s,t}}^{(l)}, \mathbf{n}_{i_{s,t}}^{(l)}\right), \end{aligned} \tag{13}$$

where $\mathcal{N}_{i_{s,t}}^{\text{in}}$, $\mathcal{N}_{i_{s,t}}^{\text{out}}$ denote the neighborhood set of previous items and next items, respectively, and $\mathbf{GRU}(\cdot)$ represents the GRU component. Different from the pooling operation, the gate mechanism in GRU decides what information is to be preserved and discarded. Unlike GGNN, which treats the neighbors equally, the attention mechanism is also utilized to differentiate the importance of neighbors [12, 115, 163]. All the above methods adopt the permutation-invariant aggregation function during the message passing, ignoring the order of items within the neighborhood, which may lead to the loss of information [19]. To address this issue, LESSR [19] preserves the order of items in the graph construction and leverages the GRU component [89] to aggregate the neighbors sequentially, as in the following equation:

$$\mathbf{n}_{i_{s,t},k}^{(l)} = \mathbf{GRU}^{(l)}\left(\mathbf{n}_{i_{s,t},k-1}^{(l)}, \mathbf{h}_{i_{s,t},k}^{(l)}\right), \tag{14}$$

where $\mathbf{h}_{i_{s,t},k}^{(l)}$ represents the kth item in the neighborhood of $i_{s,t}$ ordered by time, and $\mathbf{n}_{i_{s,t},k}^{(l)}$ denotes the neighborhood representation after aggregating k items.

For the sequence graph with hypergraph structure, DHCN [182] adopts the typical hypergraph neural network HGNN [36], which treats the nodes equally during propagation. To differentiate the importance of items within the same hyperedge, SHARE [148] designs two attention mechanisms to propagate the information of item nodes. One is the hyperedges, and the other is the information of the hyperedges to the connected item nodes. For user-aware sequential recommendation, A-PGNN [176] and GAGA [116] implicitly incorporate the user information and augment the representations of items in the neighborhood with user representation.

4.3 Sequential Preference

Due to the limited iteration of propagation, GNN cannot effectively capture long-range dependency among items [19]. Therefore, the representation of the last item (or any item) in the sequence is not sufficient enough to reflect the user's sequential preference. Besides, most of the graph construction methods of transforming sequences into graphs lose part of the sequential information [19]. In order to obtain the effective sequence representation, existing works propose several strategies to integrate the item representations in the sequence.

Considering that the items in a sequence have different levels of priority, the attention mechanism is widely adopted for integration. Some works [113, 116, 175, 206] calculate the attentive weights between the last item and all the items in the sequence, aggregate the item representations

as the global preference, and incorporate it with local preference (i.e., the last item representation) as the overall preference. In this way, the overall preference relies heavily on the relevance of the last item to the user preference. Inspired by the superiority of the multi-layer self-attention strategy in sequence modeling, GC-SAN [185] stacks multiple self-attention layers on top of the item representations generated by GNN to capture long-range dependencies.

In addition to leveraging the attention mechanism for sequence integration, sequential signals are explicitly incorporated into the integration process. For instance, NISER [48] and GCE-GNN [163] add the positional embeddings, which reflect the relative order of the items, to effectively obtain position-aware item representations. To balance the consecutive time and flexible transition pattern, FGNN [115] employs the GRU with the attention mechanism to iteratively update the user preference with item representations in the sequence.

All the above works integrate the item representations within the user's behavior sequence to generate the representation of sequential preference. Apart from these methods, DHCN [182] and COTREC [181] enrich the sequence graph by the session-to-session graph in the graph construction step. Therefore, they combine the sequential representation learned from the session-to-session graph and the one aggregated from items at this step.

4.4 Summary

This part briefly summarizes the reviewed works in terms of the three main issues.

• **Graph Construction.** The most straightforward construction is to add edges between the two consecutive items. When the sequence length is short, utilizing additional sequences can enrich the sequence graph, and it would be preferable if the additional sequences are more similar to the original sequence. Another line is to adjust the graph structure of the behavior sequence. There is no accepted statement on which method is better. Moreover, incorporating the session-to-session graph into the sequence graph is also used to gain further improvements.

• **Information Propagation.** Most of the propagation methods are variants of the propagation methods in traditional GNN frameworks, and there is no consensus on which method is better. Some complex propagation methods, such as LESSR [19], achieve performance gain at the cost of more computation. Whether to adopt complex propagation methods in practice depends on the tradeoff between computation costs and performance gains.

• **Sequential Preference.** To obtain the sequential preference, an attention mechanism is widely adopted to integrate the representations of items in the sequence. Beyond that, adding positional embeddings can enhance the relative order of the items and can bring a few improvements. Whether leveraging RNN structure can boost performance for all the sequential recommendation tasks requires further investigation.

Figure 6 summarizes the typical strategies for each of the main issues and lists the representative works accordingly.

5 SOCIAL RECOMMENDATION

With the emergence of online social networks, social recommender systems have been proposed to utilize each user's local neighbors' preferences to enhance user modeling [43, 65, 103, 104, 172]. All these works assume users with social relationships should have similar representations based on the social influence theory that connected people would influence each other. Some of them directly use such relationship as a regularizer to constrain the final user representations [65, 104, 105, 138], while others leverage such relationship as input to enhance the original user embeddings [43, 103].

From the perspective of graph learning, the early works mentioned above can be seen as modeling the first-order neighbors of each user. However, in practice, a user might be influenced by

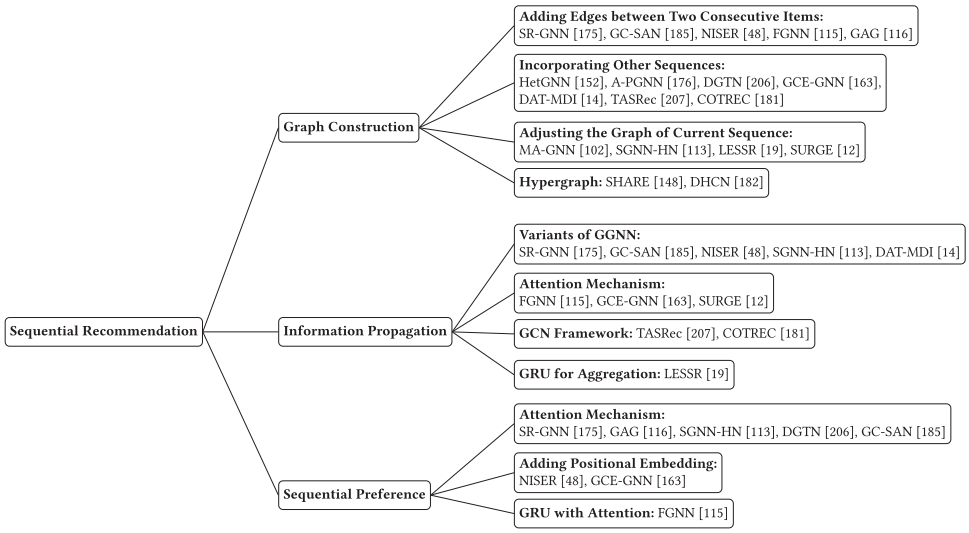


Fig. 6. Summary of representative works in sequential recommendation.

her/his friends' friends. Overlooking the high-order influence diffusion in previous works might lead to the suboptimal recommendation performance [172]. Thanks to the ability to simulate how users are influenced by the recursive social diffusion process, GNN has become a popular choice to model the social information in recommendation.

To incorporate relationships among users into interaction behaviors by leveraging GNN, there are two main issues to deal with:

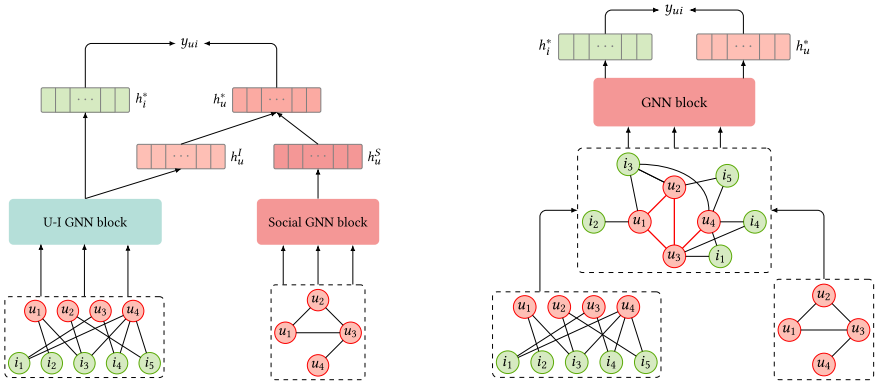
• **Influence of Friends.** Do friends have equal influence? If not, how can one distinguish the influence of different friends?

• **Preference Integration.** Users are involved in two types of relationships, i.e., social relationships with their friends and interactions with items. How can one integrate the user representations from the social influence perspective and interaction behavior?

5.1 Influence of Friends

Generally, a social graph only contains information about whether the users are friends, but the strengths of social ties are usually unknown. To propagate the information of friends, it is essential to decide the influence of friends. DiffNet [172] treats the influence of friends equally by leveraging mean-pooling operation. However, the assumption of equal influence is not in accordance with the actual situation, and the influence of a user is unsuitable to be simply determined by the number of her/his friends. Indeed, users are more likely to be influenced by friends with strong social ties or similar preferences. Therefore, the attention mechanism is widely leveraged to differentiate the influence of neighbors [2, 33, 110, 130, 171, 174]. For example, Song et al. [130] propose DGRec, which dynamically infers the influence of neighbors based on their current interests. It first models dynamic users' behaviors with a recurrent neural network and then acquires the social influence with a graph attention neural network. Compared to the mean-pooling operation, the attention mechanism boosts the overall performance, which further verifies the assumption that different friends have different influence power.

Moreover, a recent work, named ESRF [191], argues that social relations are not always reliable. The unreliability of social information lies in two aspects: on the one hand, the users with explicit social connections might have no influence power; on the other hand, the obtained social



(a) The framework of GNN on the bipartite graph and social network graph separately.

(b) The framework of GNN on the unified graph of user-item interactions and social network.

Fig. 7. Two strategies for social enhanced general recommendation.

relationships might be incomplete. Considering that indiscriminately incorporating unreliable social relationships into recommendation may lead to poor performance, ESRP leverages the autoencoder mechanism to modify the observed social relationships by filtering irrelevant relationships and investigating the new neighbors. Similarly, DiffNetLG [129] involves implicit local influence to predict the unobserved social relationship and then utilizes both explicit and implicit social relations to make recommendations.

5.2 Preference Integration

Users in social recommendation are involved in two types of relationships: one is the user-item interactions and the other is the social graph. To enhance the user preference representation by leveraging social information, there are two strategies for combining the information from these two networks:

- To learn the user representation from these two networks respectively [33, 172, 174] and then integrate them into the final preference vector, as illustrated in Figure 7(a)
- To combine the two networks into one unified network [171] and apply GNN to propagate information, as illustrated in Figure 7(b)

The advantage of the first strategy lies in two aspects: on the one hand, we can differentiate the depth of the diffusion process of two networks since they are treated separately; on the other hand, any advanced method for a user-item bipartite graph can be directly applied, and for social network, a homogeneous graph, GNN techniques are extremely suitable for simulating the influence process since they are originally proposed for homogeneous graphs. As for the integration of the user representations learned from two relationships, there are two main mechanisms, i.e., linearity combination and non-linearity combination. Among the linearity combination, DiffNet [172] treats the user representations from two spaces equally and combines them with a sum-pooling operation. Instead of an equal-weight combination, DANSER [174] dynamically allocates weights according to the user-item paired features. Among the non-linearity combination, multi-layer perceptrons over the concatenated vector are widely adopted to enhance the feature interactions [33, 47, 186].

The advantage of integrating the two graphs into one unified network is that both the higher-order social influence diffusion in the social network and interest diffusion in the user-item

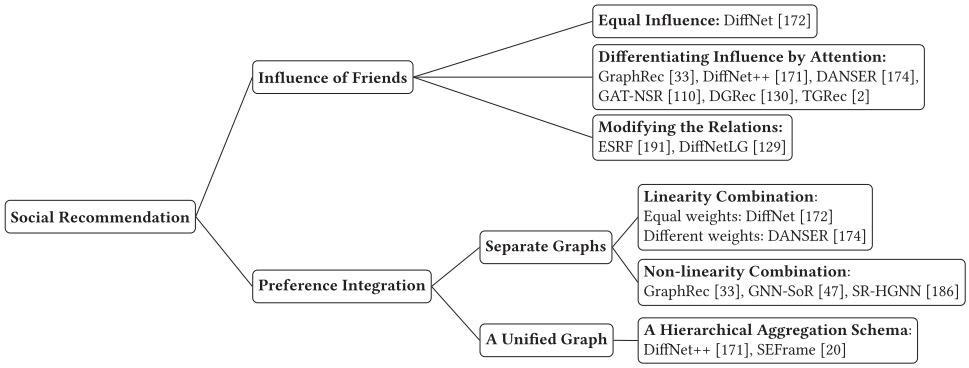


Fig. 8. Summary of representative works in social recommendation.

bipartite graph can be simulated in a unified model, and these two kinds of information simultaneously reflect users' preferences. DiffNet++ [171] designs a two-level attention network to update user nodes at each layer. Specifically, it first aggregates the information of neighbors in the bipartite graph (i.e., interacted items) and social network (i.e., friends) by utilizing the GAT mechanism, respectively. Considering that different users may have different preferences in balancing these two relationships, it further leverages another attention network to fuse the two hidden states of neighbors. Similarly, SEFrame [20] utilizes a heterogeneous graph network to fuse the knowledge from social relationships and user-item interactions and item transitions from the heterogeneous graph, and employs a two-level attention network for propagation. Up till now, there is no evidence to show which strategy always achieves better performance.

5.3 Summary

Corresponding to the discussion at the beginning of this section, we briefly summarize the current works in terms of the two issues:

• **Influence of Friends.** Compared to assigning equal weights to friends, differentiating the influence of different friends is more appropriate. An emerging direction is to automatically modify the social relationship, which can benefit from the presence of noise in social networks.

• **Preference Integration.** The strategies for combining the two sources of information depend on whether to consider the two graphs separately or unify them into one graph. For the separate graphs, user preference is an integration of the overall representations learned from these two graphs. For the unified graph, a commonly adopted strategy is the hierarchical aggregation schema.

Figure 8 summarizes the typical strategies for each of the main issues and lists the representative works accordingly.

6 KNOWLEDGE-GRAPH-BASED RECOMMENDATION

A social network that reflects relationships between users is utilized to enhance user representation, while a knowledge graph that expresses relationships between items through attributes is leveraged to enhance the item representation. Incorporating a knowledge graph into recommendation can bring two-facet benefits [144]: (1) the rich semantic relations among items in a knowledge graph can help explore their connections and improve the item representation, and (2) a knowledge graph connects a user's historically interacted-with items and recommended items, which enhances the interpretability of the results [189].

Despite the above benefits, utilizing a knowledge graph in recommendation is rather challenging due to its complex graph structure, i.e., multi-type entities and multi-type relations. Previous

works preprocess a knowledge graph by **knowledge graph embedding (KGE)** methods to learn the embeddings of entities and relations, such as [26, 146, 196, 202]. The limitation of commonly used KGE methods is that they focus on modeling rigorous semantic relatedness with the transition constraints, which are more suitable for the tasks related to graphs, such as link prediction rather than recommendation [145]. Meta-path-based methods manually define meta-paths that carry the high-order information and feed them into a predictive model, and thus they require domain knowledge and are rather labor-intensive for complicated a knowledge graph [147, 154].

Given the user-item interaction information as well as the knowledge graph, the knowledge-graph-based recommendation seeks to take full advantage of the rich information in the knowledge graph, which can help to estimate the users' preferences for items by explicitly capturing relatedness between items. For the effectiveness of knowledge-graph-based recommendation, there are two main issues to deal with:

• **Graph Construction.** How can one effectively integrate the collaborative signals from user-item interactions and the semantic information from the knowledge graph? Should the user nodes explicitly be incorporated into the knowledge graph or the user nodes be implicitly used to distinguish the importance of different relations?

• **Relation-aware Aggregation.** One characteristic of a knowledge graph is that it has multiple types of relations between entities. How can one design a relation-aware aggregation function to aggregate information from linked entities?

6.1 Graph Construction

For the stage of graph construction, one main concern is how to effectively integrate the collaborative signals and knowledge information.

One direction is to incorporate the user nodes into the knowledge graph. For instance, KGAT [154], MKGAT [134], and CKAN [162] combine the user-item bipartite graph and knowledge graph into one unified graph by taking the user nodes as one type of entity and the relation between users and items as "interaction." Recent efforts focus on the entities and relations relevant to the user-item pair. Therefore, they construct the subgraph that links the user-item pair with the user's historical interacted-with items and the related semantics in the knowledge graph [35, 127]. Based on the assumption that a shorter path between two nodes reflects more reliable connections, AKGE [127] constructs the subgraph by the following steps: pre-train the embeddings of entities in the knowledge graph by TransR [93], calculate the pairwise Euclidean distance between two linked entities, and keep the K paths with the shortest distance between the target user and item node. The potential limitation is that the subgraph structure depends on the pre-trained entity embeddings and the definition of distance measurement. ATBRG [35] exhaustively searches the multi-layer entity neighbors for the target item and the items from the user's historical behaviors and restores the paths connecting the user behaviors and the target item by multiple overlapped entities. In order to emphasize the information-intensive entities, ATBRG further prunes the entities with a single link, which can also help control the scale of the graph. Although these methods can obtain subgraphs more relevant to the user-item pair, it is quite time-consuming to pre-train the entity embedding or search and prune paths exhaustively. An effective and efficient subgraph construction strategy is worthy of further investigation.

Another direction is to implicitly use the user nodes to distinguish the importance of different relations. For instance, KGCN [147] and KGNN-LS [145] take the user nodes as queries to assign weights to different relations. In terms of graph construction, this line of research emphasizes the users' preferences toward relations instead of the collaborative signal in user-item interactions.

6.2 Relation-aware Aggregation

To fully capture the semantic information in a knowledge graph, both the linked entities (i.e., e_i, e_j) and the relations in between (i.e., r_{e_i,e_j}) should be taken into consideration during the propagation process. Besides, from the perspective of recommender systems, the role of users might also have an influence. Owing to the advantage of GAT in adaptively assigning weights based on the connected nodes, most of the existing works apply the variants of the traditional GAT over the knowledge graph; i.e., the central node is updated by the weighted average of the linked entities, and the weights are assigned according to the score function, denoted as $a(e_i, e_j, r_{e_i,e_j}, u)$. The key challenge is to design a reasonable and effective score function.

For the works [35, 134, 154] that regard the user nodes as one type of entity, the users' preferences are expected to be spilled over to the entities in the knowledge graph during the propagation process since the item nodes would be updated with the information of interacted users and related attributes, and then the other entities would contain users' preferences with iterative diffusion. Therefore, these works do not explicitly model users' interests in relations but differentiate the influence of entities by the connected nodes and their relations. For instance, inspired by the transition relationship in a knowledge graph, KGAT [154] assigns the weight according to the distance between the linked entities in the relation space:

$$a(e_i, e_j, r_{e_i,e_j}, u) = \left(\mathbf{W}_r \mathbf{e}_j\right)^\top \tanh\left(\left(\mathbf{W}_r \mathbf{e}_i + \mathbf{e}_{r_{e_i,e_j}}\right)\right), \tag{15}$$

where $\mathbf{W}_r$ is the transformation matrix for the relation, which maps the entity into relation space. In this way, the closer entities would pass more information to the central node. These methods are more appropriate for the constructed subgraph containing user nodes, since it is difficult for the users' interests to extend to all the related entities by stacking a limited number of GNN layers.

For the works that do not combine the two sources of graphs, these studies [145, 147] explicitly characterize users' interests in relations by assigning weights according to the connecting relation and specific user. For example, the score function adopted by KGCN [147] is the dot product of the user embedding and the relation embedding, i.e.,:

$$a(e_i, e_j, r_{e_i,e_j}, u) = \mathbf{u}^\top \mathbf{r}_{\mathbf{e_i},\mathbf{e_j}}. \tag{16}$$

In this way, the entities whose relations are more consistent with users' interests will spread more information to the central node.

6.3 Summary

Corresponding to the discussion at the beginning of this section, we briefly summarize the current works in terms of the two issues:

• **Graph Construction.** Existing works either consider the user nodes as one type of entity or implicitly use the user nodes to differentiate the relations. The first direction can be further divided into the overall unified graph or the specific subgraph for the user-item pair. Compared to the overall unified graph, the user-item subgraph has the advantage of focusing on more related entities and relations, but it requires more computation time and the performance depends on the construction of the subgraph, which still requires further investigation.

• **Relation-aware Aggregation.** The variants of GAT are widely leveraged to aggregate information from linked entities, taking into account the relations. For the graphs that do not explicitly incorporate user nodes, user representations are utilized to assign weights to the relations.

Figure 9 summarizes the typical strategies for each of the main issues and lists the representative works accordingly.

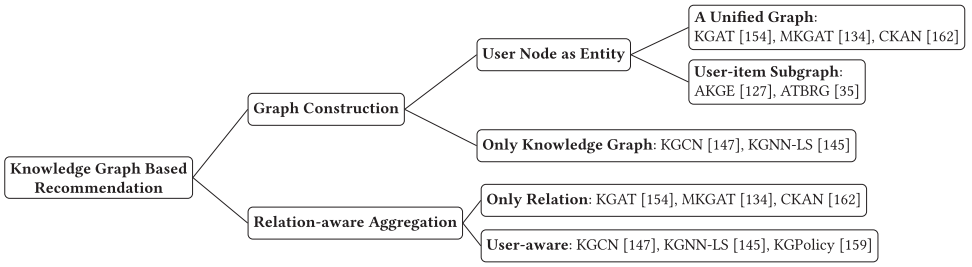


Fig. 9. Summary of representative works in knowledge-graph-based recommendation.

7 OTHER TASKS

In addition to these four types of tasks, researchers have started to utilize GNN for improving the performance of other recommendation tasks, such as POI recommendation and multimedia recommendation. In this section, we will summarize the recent developments for each task respectively.

POI recommendation plays a key role in location-based service, which utilizes the geographical information to capture geographical influence among POIs and users' historical check-ins to model the transition patterns. In the field of POI recommendation, there are several kinds of graph data, such as the user-POI bipartite graph, the sequence graph based on check-ins, and the geographical graph; i.e., the POIs within a certain distance are connected and the edge weights depend on the distance between POIs [10, 92]. SGRec [88] enriches the check-in sequence with the correlated POIs belonging to other check-ins, which allows collaborative signals to be propagated across sequences. Chang et al. [10] believe that the more often users consecutively visited the two POIs, the greater the geographical influence between these two POIs. Hence, the check-ins not only reflect users' dynamic preferences but also indicate the geographical influence among POIs. To explicitly incorporate the information of geographical distribution among POIs, the edge weights in the sequence graph depend on the distance between POIs [10].

Group recommendation aims to suggest items to a group of users instead of an individual one [59] based on their historical behaviors. There exist three types of relationships: user-item, where each user interacts with several items; user-group, where a group consists of several users; and group-item, where a group of users all choose the same item. "Group" can be regarded as a bridge connecting the users and the items in the group recommendation, which can either be treated as a part of the graph or not. Here are two representative works corresponding to these two strategies respectively. GAME [59] introduces the "group node" in the graph and applies the GAT to assign appropriate weights to each interacted neighbor. With the propagation diffusion, group representation can be iteratively updated with interacted items and users. However, this approach cannot be directly applied to the task where groups are changed dynamically and new groups are constantly formed. Different from the former transductive method, GLS-GRL [153] learns the group representative in an inductive way, which constructs the corresponding graph for each group specifically. The group representation is generated by integrating the user representations involved in the group, which can address the new group problem.

Bundle recommendation aims to recommend a set of items as a whole for a user. There are three types of relationships: user-item, where each user interacts with several items; user-bundle, where users choose the bundles; and bundle-item, where a bundle consists of several items. For group recommendation, "group" is made up of users; for bundle recommendation, "group" means a set of items. Analogously, the key challenge is to obtain the bundle representation. BGCN [11] unifies the three relationships into one graph and designs the item level and bundle level propagation from the users' perspective. HFGN [87] considers the bundle as the bridge through which

users interact with the items through bundles. Correspondingly, it constructs a hierarchical structure upon user-bundle interactions and bundle-item mappings and further captures the item-item interactions within a bundle.

Click-through rate (CTR) prediction is an essential task for recommender systems in large-scale industrial applications, which predicts the click rate based on the multi-type features. The key challenges of CTR are to model feature interactions and capture user interests. Inspired by the information diffusion process of GNN, a recent work, Fi-GNN [90], employs GNN to capture the high-order interactions among features. Specifically, it constructs a feature graph, where each node corresponds to a feature field and different fields are connected with each other through edges. Hence, the task of feature interactions is converted to propagate node information across the graph. Despite its considerable performance, Fi-GNN ignores the collaborative signals implicit in user behaviors. DG-ENN [46] designs both the attribute graph and user-item collaborative graph and utilizes GNN techniques to capture the high-order feature interactions and collaborative signals. To further alleviate the sparsity issue of the user-item interactions, DG-ENN enriches the original user-item interaction relationships with user-user similarity relationships and item-item transitions.

Multimedia recommendation has been a core service to help users identify multimedia contents of interest. The main characteristic is that the contents are in multi-modality, e.g., text, images, and videos. Recently, researchers have started to adopt GNN to capture the collaborative signals from users' interactions with multi-modal contents. For instance, MMGCN [165] constructs a user-item bipartite graph for each modality and applies GNN to propagate information for each graph, respectively. The overall user/item representations are the sum of the user/item representations of different modalities. GRCN [164] utilizes the multi-modal contents to refine the connectivity of user-item interactions. For each propagation layer, GRCN takes the maximum value of the user-item similarities in different modalities as the weight of the user-item interaction edges and uses the corresponding weights to aggregate neighbors. MKGAT [134] unifies the user nodes and multi-modal knowledge graph into one graph and employs a relation-aware graph attention network to propagate information. Considering the multi-modal characteristic of entities, MKGAT designs the entity encoder to map each specific data type into a condensed vector.

8 DATASETS, EVALUATION METRICS, AND APPLICATIONS

In this section, we introduce the commonly adopted datasets and evaluation metrics for different recommendation tasks and summarize the real-world applications of GNN-based recommendation. This section can help researchers find suitable datasets and evaluation metrics to test their methods and get an overview of the practical applications of GNN-based recommendation.

8.1 Datasets

This part introduces the public commonly adopted datasets for GNN-based recommendation systems, as summarized in Table 2. Due to the page limit, we do not list the datasets used by other recommender tasks, and we refer readers to the published works.

MovieLens[2] datasets are collected from the MovieLens website, among which three stable benchmark datasets with different scales of rating pairs, i.e., MovieLens-100K, MovieLens-1M, and MovieLens-20M, are most commonly used [51]. Each dataset contains user-item rating pairs with timestamps, the movie's attributes and tags, and user demographic features. The ratings range from 1 to 5, with a minimum interval of 1. The MovieLens datasets are widely adopted as benchmark datasets in user-item collaborative filtering tasks and knowledge-graph-based recommendation.

[2]https://grouplens.org/datasets/movielens/.

Table 2. Datasets of GNN-based Recommendation Tasks

Task	Dataset	Paper
User-Item CF	MovieLens-100K	GC-MC [8], STAR-GCN [197], DHCF [66]
	MovieLens-1M	GC-MC [8], SpectralCF [205], STAR-GCN [197], Bi-HGNN [82], DGCN-BinCF [142]
	MovieLens-10M	GC-MC [8], STAR-GCN [197], DGCN-BinCF [142]
	Amazon	SpectralCF [205], NGCF [155], LR-GCCF [18], LightGCN [55], DGCF [156], AGCN [173], NIA-GCN [132], Multi-GCCF [133]
	Gowalla	NGCF [155], LR-GCCF [18], LightGCN [55], HashGNN [135], DGCF [156], NIA-GCN [132], Multi-GCCF [133]
	Yelp	NGCF [155], LightGCN [55], DGCN-BinCF [142], DGCF [156], Multi-GCCF [133]
Sequential Recommendation	Yoochoose	SR-GNN [175], NISER [48], FGNN [115], HetGNN [152], DGTN [206], DAT-MDI [14], SGNN-HN [113], SHARE [148], DHCN [182],
	Diginetica	SR-GNN [175], GC-SAN [185], NISER [48], FGNN [115], DGTN [206], GCE-GNN [163], DAT-MDI[14], TASRec [207], COTREC [181], SGNN-HN [113], LESSR [19], SHARE [148], DHCN [182]
	Retailrocket	GC-SAN [185], NISER [48], TASRec [207], COTREC [181]
	Amazon	MA-GNN [102], TGSRec [34]
	LastFM	GAG [116], LESSR [19]
	Gowalla	GAG [116], LESSR [19], SERec [20], DGSR [199]
Social Recommendation	Yelp	DiffNet [172], DiffNet++ [171], DGRec [130], GNN-SoR [47], KCGN [62]
	Flickr	DiffNet [172], DiffNet++ [171], MHCN [192]
	Ciao	GraphRec [33], GNN-SoR [47], SR-HGNN [186]
	Epinions	GraphRec [33], DANSER [174], KCGN [62], SR-HGNN [186]
	LastFM	ESRF [191], GNN-SoR [47], MHCN [192]
	Douban	ESRF [191], DGRec [130], SR-HGNN [186], MHCN [192]
KG-based Recommendation	MovieLens-1M	AKGE [127], TGCN [13], KHGT [180]
	MovieLens-10M	MKGAT [134]
	MovieLens-20M	KGCN [147], KGNN-LS [145], CKAN [162]
	Book-Crossing	KGCN [147], KGNN-LS [145], CKAN [162]
	LastFM	KGCN [147], KGNN-LS [145], KGAT [154], AKGE [127], TGCN [13], CKAN [162]
	Dianping	KGNN-LS [145], MKGAT [134], CKAN [162]
	Yelp	KGAT [154], AKGE [127], KHGT [180]
	Amazon	KGAT [154], IntentGC [204], ATBRG [35]
	Taobao	IntentGC [204], ATBRG [35]

The **Amazon**[3] dataset includes reviews (ratings, text, helpfulness votes), product metadata (descriptions, category information, price, brand, and image features), and links (also viewed/also bought graphs) [52]. The full dataset is split into sub-datasets by categories, e.g., Amazon-Books, Amazon-Instant Video, and Amazon-Electronics. The sub-datasets in Amazon are usually adopted to test the performance in user-item collaborative filtering and sequential recommendation.

The **Yelp**[4] dataset contains user check-ins and is still being updated. The Yelp dataset is widely adopted in user-item collaborative filtering and POI recommendation tasks. Existing works usually select 1 year of data for experiments; e.g., NGCF [155] uses the 2018 edition of the Yelp dataset.

The **Gowalla**[5] dataset is the check-in dataset obtained from Gowalla, where users share their locations by checking in [23]. In addition to the check-in information, the Gowalla dataset also contains the social relationship among users. Gowalla is a classical dataset for POI recommendation and adopted in user-item collaborative filtering and sequential recommendation as well.

The **Yoochoose**[6] dataset is obtained from the RecSys Challenge 2015, which contains a stream of user clicks on an e-commerce website within 6 months. Instead of the entire dataset, most of the recent studies use the most recent fractions 1/64 and 1/4 of the sequences as the experimental datasets, named Yoochoose1/64 and Yoochoose1/4, respectively.

Diginetica[7] is provided by CIKM Cup 2016, which contains the transactional data in chronological order. Diginetica is commonly used in session-based recommendation.

The **RetailRocket**[8] dataset has been collected from a real-world ecommerce website, which contains 6 months of user browsing activities.

[3] http://jmcauley.ucsd.edu/data/amazon/links.html.
[4] https://www.yelp.com/dataset.
[5] http://snap.stanford.edu/data/loc-gowalla.html.
[6] http://2015.recsyschallenge.com/challege.html.
[7] http://cikm2016.cs.iupui.edu/cikm-cup.
[8] https://www.kaggle.com/retailrocket/ecommerce-dataset.

Table 3. Evaluation Metrics of GNN-based Recommendation Tasks

Task	Metric	Paper
User-Item CF	Recall	SpectralCF [205], NGCF [155], LightGCN [55], DGCN-BinCF [142], DHCF [66], DGCF [156], NIA-GCN [132], Multi-GCCF [133]
	MAP	SpectralCF [205], DGCN-BinCF [142]
	NDCG	NGCF [155], LR-GCCF [18], LightGCN [55], AGCN [173], HashGNN [135], DGCN-BinCF [142], DHCF [66], DGCF [156], NIA-GCN [132], Multi-GCCF [133]
	HR	LR-GCCF [18], AGCN [173], HashGNN [135], DHCF [66], PinSage [190]
Sequential Recommendation	Precision	SR-GNN [175], FGNN [115], GCE-GNN [163], COTREC [181], SGNN-HN [113], DHCN [182]
	MRR	SR-GNN [175], GC-SAN [185], NISER [48], FGNN [115], GAG [116], HetGNN [152], A-PGNN [176], DGTN [206], GCE-GNN [163], COTREC [181], SGNN-HN [113], LESSR [19], SURGE [12], SHARE [148], DHCN [182], SERec [20], TGSRec [34]
	HR	GC-SAN [185], HetGNN [152], LESSR [19], SHARE [148], SERec [20], DGSR [199]
	Recall	NISER [48], GAG [116], A-PGNN [176], DGTN [206], DAT-MDI[14], TASRec [207], MA-GNN [102], TGSRec [34]
	NDCG	HetGNN [152], DAT-MDI[14], TASRec [207], MA-GNN [102], SURGE [12], DGSR [199]
Social Recommendation	HR	DiffNet [172], DiffNet++ [171], KCGN [62]
	NDCG	DiffNet [172], DiffNet++ [171], ESRF [191], DGRec [130], GNN-SoR [47], KCGN [62], MHCN [192]
	AUC	DANSER [174], HGP [72]
	Precision	DANSER [174], ESRF [191], MHCN [192]
	Recall	ESRF [191], DGRec [130], MHCN [192]
KG-based Recommendation	AUC	KGCN [147], IntentGC [204], ATBRG [35], CKAN [162]
	F1	KGCN [147], CKAN [162]
	Recall	KGNN-LS [145], KGAT [154], MKGAT [134]
	NDCG	KGAT [154], AKGE [127], MKGAT [134], TGCN [13], KHGT [180]
	HR	AKGE [127], TGCN [13], KHGT [180]

The **LastFM**[9] dataset contains musician listening information from a set of 2,000 users and the attributes of artists from the Last.fm[10] online music system [81]. This dataset is widely adopted by sequential recommendation, social recommendation, and knowledge-graph-based recommendation.

The **Epinions** dataset and **Ciao** dataset are shared by Tang et al. [137]. Each dataset contains the users' ratings (from 1 to 5), reviews toward items, and directed trust relationships between users. These two datasets have been widely used as benchmarks for social recommendation.

The **Book-Crossing**[11] dataset contains 1 million ratings (ranging from 0 to 10) of books and the attributes of books (e.g., title, author) in the Book-Crossing community. This dataset is widely used as a benchmark for knowledge-graph-based recommendation.

8.2 Evaluation Metrics

It is essential to select adequate metrics to evaluate the performance of compared methods. Table 3 summarizes the evaluation metrics adopted by different recommendation tasks.

HR measures the proportion of users who have at least one click on the recommended items, i.e.,

$$\mathbf{HR}@K = \frac{1}{|\mathcal{U}|}\Sigma_{u\in\mathcal{U}} I(|R^K(u) \cap T(u)| > 0), \tag{17}$$

where $T(u)$ denotes the ground-truth item set, $R^K(u)$ denotes the top-K recommended item set, and $I(\cdot)$ is the indicator function.

Precision, **Recall,** and **F1** are widely adopted to evaluate the accuracy of top-K recommendation. Precision@K measures the fraction of the items the user will click among the recommended K items. Recall@K measures the proportion of the number of user clicks in the recommended K items to the entire click set. F1@K is the combination of Precision@K and Recall@K:

$$\begin{aligned} \mathbf{Precision}@K(u) &= \frac{|R^K(u) \cap T(u)|}{K}, \quad \mathbf{Recall}@K(u) = \frac{|R^K(u) \cap T(u)|}{|T(u)|}, \\ \mathbf{F1}@K(u) &= \frac{2 \times \mathbf{Precision}@K(u) \times \mathbf{Recall}@K(u)}{\mathbf{Precision}@K(u) + \mathbf{Recall}@K(u)}. \end{aligned} \tag{18}$$

[9] http://mtg.upf.edu/static/datasets/last.fm/lastfm-dataset-1K.tar.gz.
[10] https://www.last.fm/.
[11] http://www2.informatik.uni-freiburg.de/~cziegler/BX/.

The overall metric is the average over all the users, e.g., $\textbf{Precision@}K = \frac{1}{|\mathcal{U}|}\Sigma_{u\in\mathcal{U}}\textbf{Precision@}K(u)$.

NDCG differentiates the contributions of the accurately recommended items based on their ranking positions:

$$\textbf{NDCG@}K = \frac{1}{|\mathcal{U}|}\Sigma_{u\in\mathcal{U}}\frac{\Sigma_{k=1}^{K}\frac{I(R_k^K(u)\in T(u))}{\log(k+1)}}{\Sigma_{k=1}^{K}\frac{1}{\log(k+1)}}, \tag{19}$$

where $R_k^K(u)$ denotes the kth item in the recommended list $R^K(u)$.

MAP is a widely adopted ranking metric, which measures the average precision over users:

$$\textbf{MAP@}K = \frac{1}{|\mathcal{U}|}\Sigma_{u\in\mathcal{U}}\Sigma_{k=1}^{K}\frac{I(R_k^K(u)\in T(u))\textbf{Precision@}k(u)}{K}. \tag{20}$$

AUC is the probability that the model ranks a clicked item more highly than a non-clicked item. When the implicit feedback estimation is considered a binary classification problem, AUC is widely adopted to evaluate the performance:

$$\textbf{AUC}(u) = \frac{\Sigma_{i\in T(u)}\Sigma_{j\in\mathcal{I}\setminus T(u)}I(\hat{r}_i > \hat{r}_j)}{|T(u)||\mathcal{I}\setminus T(u)|}. \tag{21}$$

The overall AUC is the average over all the users.

8.3 Applications

In this part, we summarize the real-world applications of GNN-based recommendation models according to the existing works published by the industry.

Product(Advertisement) recommendation on e-commerce platforms is one of the most common application scenarios [35, 82, 87, 135, 180, 204]. For instance, IntentGC [204] leverages both explicit preferences in user-item interactions and heterogeneous relationships in a knowledge graph by graph convolutional networks, and is deployed at the Alibaba platform for recommending products to users. Another application is the content recommendation, which recommends news articles to users. For example, Wu et al. [174] deploy DANSER on a real-world article recommender system, WeChat Top Story, by exploiting the user-article interactions and social relationships. App recommendation has also attempted to utilize GNN-based models; e.g., GraphSAIL is deployed in the recommender system of a mainstream app store, which selects several hundred apps from the universal set for each user [188]. Besides, Ying et al. [190] deploy PinSage at Pinterest, which can generate higher-quality recommendations of images than comparable deep learning and graph-based alternatives.

9 FUTURE RESEARCH DIRECTIONS AND OPEN ISSUES

While GNN has achieved great success in recommender systems, this section outlines several promising prospective research directions.

9.1 Diverse and Uncertain Representation

In addition to heterogeneity in data types (e.g., node types like user and item and edge types like different behavior types), users in the graph usually also have diverse and uncertain interests [21, 79]. Representing each user as a onefold vector (a point in the low-dimensional vector space) as in previous works makes it hard to capture such characteristics in users' interests. Thus, how to represent users' multiple and uncertain interests is a direction worth exploring.

A natural choice is to extend such onefold vector to multiple vectors with various methods [99, 100, 166], e.g., disentangled representation learning [106, 107] or capsule networks [83]. Some works on GNN-based recommendation also have begun to represent users with multiple vectors. For instance, DGCF [156] explicitly adds orthogonal constraints for multi-aspect representations and iteratively updates the adjacent relationships between the linked nodes for each aspect, respectively. The research of multiple vector representation for recommendation, especially for the GNN-based recommendation model, is still in the preliminary stage, and many issues need to be studied in the future, e.g., how to disentangle the embedding pertinent to users' intents, how to set the different interest number for each user in an adaptive way, how to design an efficient and effective propagation schema for multiple vector representations.

Another feasible solution is to represent each user as a density instead of a vector. Representing data as a density (usually a multi-dimensional Gaussian distribution) provides many advantages, e.g., better encoding uncertainty for a representation and its relationships and expressing asymmetries more naturally than dot product, cosine similarity, or Euclidean distance. Specifically, Gaussian embedding has been widely used to model the data uncertainty in various domains, e.g., word embedding [141], document embedding [42, 112], and network/graph embedding [9, 54, 160]. For recommendation, Dos Santos et al. [31] and Jiang et al. [67] also deploy Gaussian embedding to capture users' uncertain preferences for improving user representations and recommendation performance. Density-based representation, e.g., Gaussian embedding, is an interesting direction that is worth exploring but has not been well studied in the GNN-based recommendation models.

9.2 Scalability of GNN in Recommendation

In industrial recommendation scenarios where the datasets include billions of nodes and edges while each node contains millions of features, it is challenging to directly apply the traditional GNN due to the large memory usage and long training time. To deal with the large-scale graphs, there are two mainstreams: one is to reduce the size of the graph by sampling to make existing GNN applicable; another is to design a scalable and efficient GNN by changing the model architecture. Sampling is a natural and widely adopted strategy for training large graphs. For example, GraphSAGE [50] randomly samples a fixed number of neighbors, and PinSage [190] employs the random walk strategy for sampling. Besides, some works [35, 127] reconstruct the small-scale subgraph from the original graph for each user-item pair. However, sampling will lose more or less part of the information, and few studies focus on how to design an effective sampling strategy to balance the effectiveness and scalability.

Another mainstream to solve this problem is to decouple the operations of non-linearities and collapsing weight matrices between consecutive layers [38, 55, 168]. As the neighbor-averaged features need to be precomputed only once, they are more scalable without the communication cost in the model training. However, these models are limited by their choice of aggregators and updaters, as compared to traditional GNN with higher flexibility in learning [17]. Therefore, more future works should be studied in the face of the large-scale graphs.

9.3 Dynamic Graphs in Recommendation

In real-world recommender systems, not only the objects such as users and items but also the relationships between them are changing over time. To maintain the up-to-date recommendation, the systems should be iteratively updated with the new coming information. From the perspective of graphs, the constantly updated information brings about dynamic graphs instead of static ones. Static graphs are stable so they can be modeled feasibly, while dynamic graphs introduce changing structures. An interesting prospective research problem is how to design the corresponding GNN framework in response to the dynamic graphs in practice. Existing studies in recommendation pay

little attention to the dynamic graphs. As far as we know, GraphSAIL [188] is the first attempt to address the incremental learning on GNN for recommender systems, which deals with the changing of interactions, i.e., the edges between nodes. To balance the update and preservation, it constrains the embedding similarity between the central node and its neighborhood in successively learned models and controls the incrementally learned embedding close to its previous version. Dynamic graphs in recommendation are a largely under-explored area that deserves further study.

9.4 Reception Field of GNN in Recommendation

The reception field of a node refers to a set of nodes including the node itself and its neighbors reachable within K-hops [118], where K is the number of propagation iterations. Generally, the aggregation step K is the same as the number of GNN layers in coupled GNNs (e.g., GCN and GraphSAGE). In addition, some recent graph diffusion-based works [74, 75, 109] decouple the operations of aggregation and update and embrace a larger reception field with a larger aggregation step. For nodes with low degree, they need a deep GNN architecture to enlarge their reception field for sufficient neighborhood information. However, by increasing the propagation steps, the reception field of nodes with high degree will expand too much and may introduce noise, which could lead to the over-smoothing problem [86] and a consequent drop in performance.

For the graph data in recommendation, the degree of nodes exhibits a long-tail distribution; i.e., active users have lots of interactions with items, while cold users have few interactions, and similar to the popular items and cold items. Therefore, applying the same propagation step on all the nodes may be suboptimal. There are only a few emerging works to adaptively decide the propagation step for each node in order to obtain a reasonable reception field [71, 96, 98]. As a result, how to adaptively select a suitable reception field for each user or item in GNN-based recommendation is still an issue worthy of research.

9.5 Self-supervised Learning

Self-supervised learning (SSL) is an emerging paradigm for improving the utilization of data, which can help alleviate the sparsity issue. Inspired by the success of SSL in other areas, recent efforts have leveraged SSL to recommender systems and made remarkable achievements [170, 209]. In the field of GNN-based recommender systems, there exist few attempts to employ SSL as well. For instance, COTREC [181] designs a contrastive learning task by maximizing the agreement between the representations of the last-clicked item and the predicted item samples, accompanied with the given session representation. DHCN [182] maximizes the mutual information between the session representations learned via the session-to-session graph and item-session hypergraph. The key challenge is how to design an effective supervised signal corresponding to the main task. Considering the prevalence of sparsity issue in recommender systems, we believe self-supervised learning in GNN-based recommender systems is a promising direction.

9.6 Robustness in GNN-based Recommendation

Recent studies show that GNN can be easily fooled by small perturbations on the input [187]; i.e., the performance of GNN will be greatly reduced if the graph structure contains noise. In real-world recommendation scenarios, it is a common phenomenon that the relationships between nodes are not always reliable. For instance, users may accidentally click the items, and part of social relationships cannot be captured. In addition, the attacker may also inject fake data into the recommender systems. Due to the vulnerability of GNN to noisy data, it is of great practical significance to construct a robust recommender system that is able to generate stable recommendations even in the presence of shilling attacks [201]. In the field of GNN, there are emerging efforts in graph adversarial learning to enhance the robustness [56, 187, 211]. Few attempts in GNN-based

recommendation have started to pay attention to robustness. For instance, GraphRf [201] jointly learns the rating prediction and fraudster detection, where the probability of being a fraudster determines the reliability of the user's rating in the rating prediction component. How to build a more robust recommender system is worth exploring but has not been well studied in the GNN-based recommender systems.

9.7 Privacy Preservation

Due to the strict privacy protection under the General Data Protection Regulation,[12] the privacy preservation in recommender systems has aroused lots of concern in academia and industry since most of the data may be deemed confidential/private, e.g., social network and historical behavior [57]. An emerging paradigm is to use federated learning to train recommender systems without uploading users' data to the central server [94, 111, 149]. However, the local user data only contains first-order user-item interactions, which makes it challenging to capture high-order connectivity without privacy linkage [167]. Another line is to employ differential privacy to guarantee the user privacy in the procedure of recommender systems [39, 128]. One limitation of differential privacy is that it usually brings a decrease in performance [29].

Some efforts have focused on privacy preservation in GNN-based recommendation. For instance, FedGNN [167] trains the GNN model locally based on the local user-item graph. The pseudo interacted items and a user-item graph expansion method are proposed to protect the items and exploit the high-order interactions, respectively. Based on **local differential privacy (LDP)** [24], FedGNN may add noise to the local gradient of each model and thus decrease the model accuracy. To reduce the amount of noise while maintaining privacy protection, PPGRec [120] converts the LDP model into the central differential privacy model and only adds noise to the aggregated global gradient. With society's increasing emphasis on privacy protection, privacy preservation in GNN-based recommendations should be an attractive direction due to its practical value.

9.8 Fairness in GNN-based Recommender System

Recent years have seen a surge of research efforts on recommendation biases to achieve fairness [16]. For instance, the recommendation performance for users of different demographic groups should be close, and each item should have an equal probability of overall exposure. With the widespread nature of GNN, there is an increasing societal concern that GNN could make discriminatory decisions [30]. Some explorations have been made toward alleviating biases in GNN-based recommender systems. For instance, NISER [48] applies a normalization operation over the representations to handle popularity bias. FairGNN [27] employs an adversarial learning paradigm to eliminate the bias of GNN by leveraging graph structures and limited sensitive information. Due to the prevalence of biases in recommender systems and society's growing focus on fairness, ensuring fairness while maintaining comparable performance in GNN-based recommender systems deserves further studying.

9.9 Explainability

Explainability is beneficial for recommender systems: on the one hand, explainable recommendations to users allow them to know why the items are recommended and could be persuasive; on the other hand, the practitioners can know why the model works, which could help further improvements [200]. Due to the significance of explainability, many interests have focused on designing explainable recommendation models or conducting post hoc interpretations [63, 126, 139].

[12] https://gdpr-info.eu/.

With the proliferation of GNN, recent efforts have investigated GNN explainability methods [194]. The methods can be divided into two categories: the instance-level methods provide example-specific explanations by identifying important input features for its prediction [3, 124, 195]; the model-level methods provide high-level interpretations and a generic understanding of how deep graph models work [193]. There are also some attempts on explainability on GNN-based recommendation [64, 108, 157]. Most of them utilize semantic information in knowledge graph and conduct post hoc interpretations. Up till now, the explainable GNN-based recommender systems still have not been fully explored, which should be an interesting and beneficial direction.

10 CONCLUSION

Owing to the superiority of GNN in learning on graph data, utilizing GNN techniques in recommender systems has gained increasing interest in academia and industry. In this survey, we provided 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 clarified the main issues, detailed the corresponding strategies adopted by the representative models, and discussed their advantages and limitations. Furthermore, we suggested several promising directions for future research. We hope this survey can provide readers with a general understanding of the recent progress in this field and shed some light on future developments.

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. *TKDE* 17, 6 (2005), 734–749.

[2] Ting Bai, Youjie Zhang, Bin Wu, and Jian-Yun Nie. 2020. Temporal graph neural networks for social recommendation. In *ICBD*. 898–903.

[3] Federico Baldassarre and Hossein Azizpour. 2019. Explainability techniques for graph convolutional networks. *arXiv preprint arXiv:1905.13686* (2019).

[4] Zeynep Batmaz, Ali Yurekli, Alper Bilge, and Cihan Kaleli. 2019. A review on deep learning for recommender systems: Challenges and remedies. *Artificial Intelligence Review* 52, 1 (2019), 1–37.

[5] Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, and Koray Kavukcuoglu. 2016. Interaction networks for learning about objects, relations and physics. In *NeurIPS*. 4509–4517.

[6] Robert M. Bell and Yehuda Koren. 2007. Scalable collaborative filtering with jointly derived neighborhood interpolation weights. In *ICDM*. 43–52.

[7] James Bennett and Stan Lanning. 2007. The Netflix prize. In *KDD Cup Workshop 2007*. 3–6.

[8] Rianne van den Berg, Thomas N. Kipf, and Max Welling. 2017. Graph convolutional matrix completion. *arXiv preprint arXiv:1706.02263* (2017).

[9] Aleksandar Bojchevski and Stephan Günnemann. 2018. Deep Gaussian embedding of graphs: Unsupervised inductive learning via ranking. In *ICLR*.

[10] Buru Chang, Gwanghoon Jang, Seoyoon Kim, and Jaewoo Kang. 2020. Learning graph-based geographical latent representation for point-of-interest recommendation. In *CIKM*. 135–144.

[11] Jianxin Chang, Chen Gao, Xiangnan He, Depeng Jin, and Yong Li. 2020. Bundle recommendation with graph convolutional networks. In *SIGIR*. 1673–1676.

[12] Jianxin Chang, Chen Gao, Yu Zheng, Yiqun Hui, Yanan Niu, Yang Song, Depeng Jin, and Yong Li. 2021. Sequential recommendation with graph neural networks. In *SIGIR*. 378–387.

[13] Bo Chen, Wei Guo, Ruiming Tang, Xin Xin, Yue Ding, Xiuqiang He, and Dong Wang. 2020. TGCN: Tag graph convolutional network for tag-aware recommendation. In *CIKM*. 155–164.

[14] Chen Chen, Jie Guo, and Bin Song. 2021. Dual attention transfer in session-based recommendation with multi-dimensional integration. In *SIGIR*. 869–878.

[15] Jihong Chen, Wei Chen, Jinjing Huang, Jinhua Fang, Zhixu Li, An Liu, and Lei Zhao. 2020. Co-purchaser recommendation for online group buying. *DSE* 5, 3 (2020), 280–292.

[16] Jiawei Chen, Hande Dong, Xiang Wang, Fuli Feng, Meng Wang, and Xiangnan He. 2020. Bias and debias in recommender system: A survey and future directions. *arXiv preprint arXiv:2010.03240* (2020).

[17] Lei Chen, Zhengdao Chen, and Joan Bruna. 2020. On graph neural networks versus graph-augmented MLPs. In *ICLR*.

[18] Lei Chen, Le Wu, Richang Hong, Kun Zhang, and Meng Wang. 2020. Revisiting graph based collaborative filtering: A linear residual graph convolutional network approach. In *AAAI*. 27–34.
[19] Tianwen Chen and Raymond Chi-Wing Wong. 2020. Handling information loss of graph neural networks for session-based recommendation. In *SIGKDD*. 1172–1180.
[20] Tianwen Chen and Raymond Chi-Wing Wong. 2021. An efficient and effective framework for session-based social recommendation. In *WSDM*. 400–408.
[21] Wanyu Chen, Pengjie Ren, Fei Cai, Fei Sun, and Maarten de Rijke. 2020. Improving end-to-end sequential recommendations with intent-aware diversification. In *CIKM*. 175–184.
[22] Janneth Chicaiza and Priscila Valdiviezo-Diaz. 2021. A comprehensive survey of knowledge graph-based recommender systems: Technologies, development, and contributions. *Information* 12, 6 (2021), 232.
[23] Eunjoon Cho, Seth A. Myers, and Jure Leskovec. 2011. Friendship and mobility: User movement in location-based social networks. In *SIGKDD*. 1082–1090.
[24] Graham Cormode, Somesh Jha, Tejas Kulkarni, Ninghui Li, Divesh Srivastava, and Tianhao Wang. 2018. Privacy at scale: Local differential privacy in practice. In *SIGMOD*. 1655–1658.
[25] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for Youtube recommendations. In *RecSys*. 191–198.
[26] Amine Dadoun, Raphaël Troncy, Olivier Ratier, and Riccardo Petitti. 2019. Location embeddings for next trip recommendation. In *WWW*. 896–903.
[27] Enyan Dai and Suhang Wang. 2021. Say no to the discrimination: Learning fair graph neural networks with limited sensitive attribute information. In *WSDM*. 680–688.
[28] Aminu Da'u and Naomie Salim. 2020. Recommendation system based on deep learning methods: A systematic review and new directions. *Artificial Intelligence Review* 53, 4 (2020), 2709–2748.
[29] Josep Domingo-Ferrer, David Sánchez, and Alberto Blanco-Justicia. 2021. The limits of differential privacy (and its misuse in data release and machine learning). *Communications of the ACM* 64, 7 (2021), 33–35.
[30] Yushun Dong, Ninghao Liu, Brian Jalaian, and Jundong Li. 2021. EDITS: Modeling and mitigating data bias for graph neural networks. *arXiv preprint arXiv:2108.05233* (2021).
[31] Ludovic Dos Santos, Benjamin Piwowarski, and Patrick Gallinari. 2017. Gaussian embeddings for collaborative filtering. In *SIGIR*. 1065–1068.
[32] Chantat Eksombatchai, Pranav Jindal, Jerry Zitao Liu, Yuchen Liu, Rahul Sharma, Charles Sugnet, Mark Ulrich, and Jure Leskovec. 2018. Pixie: A system for recommending 3+ billion items to 200+ million users in real-time. In *WWW*. 1775–1784.
[33] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. 2019. Graph neural networks for social recommendation. In *WWW*. 417–426.
[34] Ziwei Fan, Zhiwei Liu, Jiawei Zhang, Yun Xiong, Lei Zheng, and Philip S. Yu. 2021. Continuous-time sequential recommendation with temporal graph collaborative transformer. In *CIKM*. 433–442.
[35] Yufei Feng, Binbin Hu, Fuyu Lv, Qingwen Liu, Zhiqiang Zhang, and Wenwu Ou. 2020. ATBRG: Adaptive target-behavior relational graph network for effective recommendation. In *SIGIR*. 2231–2240.
[36] Yifan Feng, Haoxuan You, Zizhao Zhang, Rongrong Ji, and Yue Gao. 2019. Hypergraph neural networks. In *AAAI*. 3558–3565.
[37] Alex Fout, Jonathon Byrd, Basir Shariat, and Asa Ben-Hur. 2017. Protein interface prediction using graph convolutional networks. In *NeurIPS*. 6533–6542.
[38] Fabrizio Frasca, Emanuele Rossi, Davide Eynard, Benjamin Chamberlain, Michael Bronstein, and Federico Monti. 2020. SIGN: Scalable inception graph neural networks. In *ICML 2020 Workshop on Graph Representation Learning and Beyond*.
[39] Chen Gao, Chao Huang, Dongsheng Lin, Depeng Jin, and Yong Li. 2020. DPLCF: Differentially private local collaborative filtering. In *SIGIR*. 961–970.
[40] Chen Gao, Yu Zheng, Nian Li, Yinfeng Li, Yingrong Qin, Jinghua Piao, Yuhan Quan, Jianxin Chang, Depeng Jin, Xiangnan He, et al. 2021. Graph neural networks for recommender systems: Challenges, methods, and directions. *arXiv preprint arXiv:2109.12843* (2021).
[41] Marco Gori, Augusto Pucci, V. Roma, and I. Siena. 2007. Itemrank: A random-walk based scoring algorithm for recommender engines. In *IJCAI*. 2766–2771.
[42] Antoine Gourru, Julien Velcin, and Julien Jacques. 2020. Gaussian embedding of linked documents from a pretrained semantic space. In *IJCAI*. 3912–3918.
[43] 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 *AAAI*. 123–129.
[44] Qipeng Guo, Xipeng Qiu, XiangYang Xue, and Zheng Zhang. 2021. Syntax-guided text generation via graph neural network. *Science China Information Sciences* 64, 5 (2021), 1–10.

[45] Qingyu Guo, Fuzhen Zhuang, Chuan Qin, Hengshu Zhu, Xing Xie, Hui Xiong, and Qing He. 2020. A survey on knowledge graph-based recommender systems. *TKDE* (2020), 1–1.
[46] Wei Guo, Rong Su, Renhao Tan, Huifeng Guo, Yingxue Zhang, Zhirong Liu, Ruiming Tang, and Xiuqiang He. 2021. Dual graph enhanced embedding neural network for CTR prediction. In *SIGKDD*. 496–504.
[47] Zhiwei Guo and Heng Wang. 2020. A deep graph neural network-based mechanism for social recommendations. *IEEE Transactions on Industrial Informatics* 17, 4 (2020), 2776–2783.
[48] Priyanka Gupta, Diksha Garg, Pankaj Malhotra, Lovekesh Vig, and Gautam M. Shroff. 2019. NISER: Normalized item and session representations with graph neural networks. *arXiv preprint arXiv:1909.04276* (2019).
[49] Takuo Hamaguchi, Hidekazu Oiwa, Masashi Shimbo, and Yuji Matsumoto. 2017. Knowledge transfer for out-of-knowledge-base entities: A graph neural network approach. In *IJCAI*. 1802–1808.
[50] William L. Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *NeurIPS*. 1025–1035.
[51] F. Maxwell Harper and Joseph A. Konstan. 2015. The Movielens datasets: History and context. *TIIS* 5, 4 (2015), 1–19.
[52] Ruining He and Julian McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *WWW*. 507–517.
[53] Ruining He and Julian J. McAuley. 2016. Fusing similarity models with Markov chains for sparse sequential recommendation. In *ICDM*. 191–200.
[54] Shizhu He, Kang Liu, Guoliang Ji, and Jun Zhao. 2015. Learning to represent knowledge graphs with Gaussian embedding. In *CIKM*. 623–632.
[55] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, YongDong Zhang, and Meng Wang. 2020. LightGCN: Simplifying and powering graph convolution network for recommendation. In *SIGIR*. 639–648.
[56] Xiangnan He, Zhankui He, Xiaoyu Du, and Tat-Seng Chua. 2018. Adversarial personalized ranking for recommendation. In *SIGIR*. 355–364.
[57] Xinlei He, Jinyuan Jia, Michael Backes, Neil Zhenqiang Gong, and Yang Zhang. 2021. Stealing links from graph neural networks. In *USENIX Security*. 2669–2686.
[58] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *WWW*. 173–182.
[59] Zhixiang He, Chi-Yin Chow, and Jia-Dong Zhang. 2020. GAME: Learning graphical and attentive multi-view embeddings for occasional group recommendation. In *SIGIR*. 649–658.
[60] Jonathan L. Herlocker, Joseph A. Konstan, Al Borchers, and John Riedl. 1999. An algorithmic framework for performing collaborative filtering. In *SIGIR*. 230–237.
[61] Balázs Hidasi and Alexandros Karatzoglou. 2018. Recurrent neural networks with top-k gains for session-based recommendations. In *CIKM*. 843–852.
[62] Chao Huang, Huance Xu, Yong Xu, Peng Dai, Lianghao Xia, Mengyin Lu, Liefeng Bo, Hao Xing, Xiaoping Lai, and Yanfang Ye. 2021. Knowledge-aware coupled graph neural network for social recommendation. In *AAAI*. 4115–4122.
[63] Jin Huang, Wayne Xin Zhao, Hongjian Dou, Ji-Rong Wen, and Edward Y. Chang. 2018. Improving sequential recommendation with knowledge-enhanced memory networks. In *SIGIR*. 505–514.
[64] Xiaowen Huang, Quan Fang, Shengsheng Qian, Jitao Sang, Yan Li, and Changsheng Xu. 2019. Explainable interaction-driven user modeling over knowledge graph for sequential recommendation. In *ACM MM*. 548–556.
[65] Mohsen Jamali and Martin Ester. 2010. A matrix factorization technique with trust propagation for recommendation in social networks. In *RecSys*. 135–142.
[66] Shuyi Ji, Yifan Feng, Rongrong Ji, Xibin Zhao, Wanwan Tang, and Yue Gao. 2020. Dual channel hypergraph collaborative filtering. In *SIGKDD*. 2020–2029.
[67] Junyang Jiang, Deqing Yang, Yanghua Xiao, and Chenlu Shen. 2019. Convolutional Gaussian embeddings for personalized recommendation with uncertainty. In *IJCAI*. 2642–2648.
[68] Bowen Jin, Chen Gao, Xiangnan He, Depeng Jin, and Yong Li. 2020. Multi-behavior recommendation with graph convolutional networks. In *SIGIR*. 659–668.
[69] Santosh Kabbur, Xia Ning, and George Karypis. 2013. FISM: Factored item similarity models for top-n recommender systems. In *SIGKDD*. 659–667.
[70] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *ICDM*. 197–206.
[71] Anees Kazi, Shayan Shekarforoush, S. Arvind Krishna, Hendrik Burwinkel, Gerome Vivar, Karsten Kortüm, Seyed-Ahmad Ahmadi, Shadi Albarqouni, and Nassir Navab. 2019. InceptionGCN: Receptive field aware graph convolutional network for disease prediction. In *IPMI*. 73–85.
[72] Kyung-Min Kim, Dong-Hyun Kwak, Hanock Kwak, Young-Jin Park, Sangkwon Sim, Jae-Han Cho, Minkyu Kim, Jihun Kwon, Nako Sung, and Jung-Woo Ha. 2019. Tripartite heterogeneous graph propagation for large-scale social recommendation. In *ACM RecSys 2019 Late-Breaking Results Track*. 56–60.
[73] Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *ICLR*.

[74] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. 2018. Predict then propagate: Graph neural networks meet personalized Pagerank. In *ICLR*.
[75] Johannes Klicpera, Stefan Weißenberger, and Stephan Günnemann. 2019. Diffusion improves graph learning. In *NeurIPS*. 13366–13378.
[76] Yehuda Koren. 2008. Factorization meets the neighborhood: A multifaceted collaborative filtering model. In *SIGKDD*. 426–434.
[77] Yehuda Koren and Robert Bell. 2015. Advances in collaborative filtering. In *Recommender Systems Handbook*, 77–118.
[78] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009), 30–37.
[79] Matevž Kunaver and Tomaž Požrl. 2017. Diversity in recommender systems - A survey. *KBS* 123 (2017), 154–162.
[80] Daniel D. Lee and H. Sebastian Seung. 2000. Algorithms for non-negative matrix factorization. In *NeurIPS*. 535–541.
[81] Mark Levy and Klaas Bosteels. 2010. Music recommendation and the long tail. In *1st Workshop on Music Recommendation And Discovery*.
[82] Chong Li, Kunyang Jia, Dan Shen, C. J. Shi, and Hongxia Yang. 2019. Hierarchical representation learning for bipartite graphs. In *IJCAI*. 2873–2879.
[83] Chao Li, Zhiyuan Liu, Mengmeng Wu, Yuchi Xu, Huan Zhao, Pipei Huang, Guoliang Kang, Qiwei Chen, Wei Li, and Dik Lun Lee. 2019. Multi-interest network with dynamic routing for recommendation at Tmall. In *CIKM*. 2615–2623.
[84] Guangjie Li, Hui Liu, Ge Li, Sijie Shen, and Hanlin Tang. 2020. LSTM-based argument recommendation for non-API methods. *Science China Information Sciences* 63, 9 (2020), 1–22.
[85] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. 2017. Neural attentive session-based recommendation. In *CIKM*. 1419–1428.
[86] Qimai Li, Zhichao Han, and Xiao-Ming Wu. 2018. Deeper insights into graph convolutional networks for semi-supervised learning. In *AAAI*. 3538–3545.
[87] Xingchen Li, Xiang Wang, Xiangnan He, Long Chen, Jun Xiao, and Tat-Seng Chua. 2020. Hierarchical fashion graph network for personalized outfit recommendation. In *SIGIR*. 159–168.
[88] Yang Li, Tong Chen, Yadan Luo, Hongzhi Yin, and Zi Huang. 2021. Discovering collaborative signals for next POI recommendation with iterative seq2graph augmentation. In *IJCAI*. 1491–1497.
[89] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. 2016. Gated graph sequence neural networks. In *ICLR*.
[90] Zekun Li, Zeyu Cui, Shu Wu, Xiaoyu Zhang, and Liang Wang. 2019. Fi-GNN: Modeling feature interactions via graph neural networks for CTR prediction. In *CIKM*. 539–548.
[91] Zhao Li, Xin Shen, Yuhang Jiao, Xuming Pan, Pengcheng Zou, Xianling Meng, Chengwei Yao, and Jiajun Bu. 2020. Hierarchical bipartite graph neural networks: Towards large-scale e-commerce applications. In *ICDE*. 1677–1688.
[92] Nicholas Lim, Bryan Hooi, See-Kiong Ng, Xueou Wang, Yong Liang Goh, Renrong Weng, and Jagannadan Varadarajan. 2020. STP-UDGAT: Spatial-temporal-preference user dimensional graph attention network for next POI recommendation. In *CIKM*. 845–854.
[93] Hailun Lin, Yong Liu, Weiping Wang, Yinliang Yue, and Zheng Lin. 2017. Learning entity and relation embeddings for knowledge resolution. *Procedia Computer Science* 108 (2017), 345–354.
[94] Yujie Lin, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Dongxiao Yu, Jun Ma, Maarten de Rijke, and Xiuzhen Cheng. 2020. Meta matrix factorization for federated rating predictions. In *SIGIR*. 981–990.
[95] Greg Linden, Brent Smith, and Jeremy York. 2003. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing* 7, 1 (2003), 76–80.
[96] Meng Liu, Hongyang Gao, and Shuiwang Ji. 2020. Towards deeper graph neural networks. In *SIGKDD*. 338–348.
[97] Qiao Liu, Yifu Zeng, Refuoe Mokhosi, and Haibin Zhang. 2018. STAMP: Short-term attention/memory priority model for session-based recommendation. In *SIGKDD*. 1831–1839.
[98] Ziqi Liu, Chaochao Chen, Longfei Li, Jun Zhou, Xiaolong Li, Le Song, and Yuan Qi. 2019. Geniepath: Graph neural networks with adaptive receptive paths. In *AAAI*. 4424–4431.
[99] Zhaoyang Liu, Haokun Chen, Fei Sun, Xu Xie, Jinyang Gao, Bolin Ding, and Yanyan Shen. 2020. Intent preference decoupling for user representation on online recommender system. In *IJCAI*. 2575–2582.
[100] Zheng Liu, Jianxun Lian, Junhan Yang, Defu Lian, and Xing Xie. 2020. Octopus: Comprehensive and elastic user representation for the generation of recommendation candidates. In *SIGIR*. 289–298.
[101] Zhiwei Liu, Lin Meng, Jiawei Zhang, and Philip S. Yu. 2020. Deoscillated graph collaborative filtering. *arXiv preprint arXiv:2011.02100* (2020).
[102] Chen Ma, Liheng Ma, Yingxue Zhang, Jianing Sun, Xue Liu, and Mark Coates. 2020. Memory augmented graph neural networks for sequential recommendation. In *AAAI*. 5045–5052.
[103] Hao Ma, Irwin King, and Michael R. Lyu. 2009. Learning to recommend with social trust ensemble. In *SIGIR*. 203–210.

[104] Hao Ma, Haixuan Yang, Michael R. Lyu, and Irwin King. 2008. SoRec: Social recommendation using probabilistic matrix factorization. In *CIKM*. 931–940.
[105] Hao Ma, Dengyong Zhou, Chao Liu, Michael R. Lyu, and Irwin King. 2011. Recommender systems with social regularization. In *WSDM*. 287–296.
[106] Jianxin Ma, Chang Zhou, Peng Cui, Hongxia Yang, and Wenwu Zhu. 2019. Learning disentangled representations for recommendation. In *NeurIPS*. 5712–5723.
[107] Jianxin Ma, Chang Zhou, Hongxia Yang, Peng Cui, Xin Wang, and Wenwu Zhu. 2020. Disentangled self-supervision in sequential recommenders. In *SIGKDD*. 483–491.
[108] Weizhi Ma, Min Zhang, Yue Cao, Woojeong Jin, Chenyang Wang, Yiqun Liu, Shaoping Ma, and Xiang Ren. 2019. Jointly learning explainable rules for recommendation with knowledge graph. In *WWW*. 1210–1221.
[109] Xupeng Miao, Nezihe Merve Gürel, Wentao Zhang, Zhichao Han, Bo Li, Wei Min, Susie Xi Rao, Hansheng Ren, Yinan Shan, Yingxia Shao, et al. 2021. Degnn: Improving graph neural networks with graph decomposition. In *SIGKDD*. 1223–1233.
[110] Nan Mu, Daren Zha, Yuanye He, and Zhihao Tang. 2019. Graph attention networks for neural social recommendation. In *ICTAI*. 1320–1327.
[111] Khalil Muhammad, Qinqin Wang, Diarmuid O'Reilly-Morgan, Elias Tragos, Barry Smyth, Neil Hurley, James Geraci, and Aonghus Lawlor. 2020. FedFast: Going beyond average for faster training of federated recommender systems. In *SIGKDD*. 1234–1242.
[112] Giannis Nikolentzos, Polykarpos Meladianos, François Rousseau, Yannis Stavrakas, and Michalis Vazirgiannis. 2017. Multivariate Gaussian document representation from word embeddings for text categorization. In *EACL*. 450–455.
[113] Zhiqiang Pan, Fei Cai, Wanyu Chen, Honghui Chen, and Maarten de Rijke. 2020. Star graph neural networks for session-based recommendation. In *CIKM*. 1195–1204.
[114] Andreas Pfadler, Huan Zhao, Jizhe Wang, Lifeng Wang, Pipei Huang, and Dik Lun Lee. 2020. Billion-scale recommendation with heterogeneous side information at Taobao. In *ICDE*. 1667–1676.
[115] Ruihong Qiu, Jingjing Li, Zi Huang, and Hongzhi YIn. 2019. Rethinking the item order in session-based recommendation with graph neural networks. In *CIKM*. 579–588.
[116] Ruihong Qiu, Hongzhi Yin, Zi Huang, and Tong Chen. 2020. GAG: Global attributed graph neural network for streaming session-based recommendation. In *SIGIR*. 669–678.
[117] Massimo Quadrana, Paolo Cremonesi, and Dietmar Jannach. 2018. Sequence-aware recommender systems. *ACM Computing Surveys* 51, 4 (2018), 36.
[118] Pei Quan, Yong Shi, Minglong Lei, Jiaxu Leng, Tianlin Zhang, and Lingfeng Niu. 2019. A brief review of receptive fields in graph convolutional networks. In *WI*. 106–110.
[119] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing personalized Markov chains for next-basket recommendation. In *WWW*. 811–820.
[120] Sisong Ru, Bingbing Zhang, Yixin Jie, Chi Zhang, Lingbo Wei, and Chengjie Gu. 2021. Graph neural networks for privacy-preserving recommendation with secure hardware. In *NaNA*. 395–400.
[121] Ruslan Salakhutdinov and Andriy Mnih. 2007. Probabilistic matrix factorization. In *NeurIPS*. 1257–1264.
[122] 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. In *ICML*. 4470–4479.
[123] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *WWW*. 285–295.
[124] Thomas Schnake, Oliver Eberle, Jonas Lederer, Shinichi Nakajima, Kristof T. Schütt, Klaus-Robert Müller, and Grégoire Montavon. 2021. Higher-order explanations of graph neural networks via relevant walks. *IEEE PAMI* (2021).
[125] Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. 2015. AutoRec: Autoencoders meet collaborative filtering. In *WWW*. 111–112.
[126] Sungyong Seo, Jing Huang, Hao Yang, and Yan Liu. 2017. Interpretable convolutional neural networks with dual local and global attention for review rating prediction. In *RecSys*. 297–305.
[127] Xiao Sha, Zhu Sun, and Jie Zhang. 2019. Attentive knowledge graph embedding for personalized recommendation. *arXiv preprint arXiv:1910.08288* (2019).
[128] Hyejin Shin, Sungwook Kim, Junbum Shin, and Xiaokui Xiao. 2018. Privacy enhanced matrix factorization for recommendation with local differential privacy. *TKDE* 30, 9 (2018), 1770–1782.
[129] Changhao Song, Bo Wang, Qinxue Jiang, Yehua Zhang, Ruifang He, and Yuexian Hou. 2021. Social recommendation with implicit social influence. In *SIGIR*. 1788–1792.
[130] Weiping Song, Zhiping Xiao, Yifan Wang, Laurent Charlin, Ming Zhang, and Jian Tang. 2019. Session-based social recommendation via dynamic graph attention networks. In *WSDM*. 555–563.
[131] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer. In *CIKM*. 1441–1450.

[132] Jianing Sun, Yingxue Zhang, Wei Guo, Huifeng Guo, Ruiming Tang, Xiuqiang He, Chen Ma, and Mark Coates. 2020. Neighbor interaction aware graph convolution networks for recommendation. In *SIGIR*. 1289–1298.
[133] Jianing Sun, Yingxue Zhang, Chen Ma, Mark Coates, Huifeng Guo, Ruiming Tang, and Xiuqiang He. 2019. Multi-graph convolution collaborative filtering. In *ICDM*. 1306–1311.
[134] Rui Sun, Xuezhi Cao, Yan Zhao, Junchen Wan, Kun Zhou, Fuzheng Zhang, Zhongyuan Wang, and Kai Zheng. 2020. Multi-modal knowledge graphs for recommender systems. In *CIKM*. 1405–1414.
[135] 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 *WWW*. 1988–1998.
[136] Yong Kiam Tan, Xinxing Xu, and Yong Liu. 2016. Improved recurrent neural networks for session-based recommendations. In *1st Workshop on Deep Learning for Recommender Systems*. 17–22.
[137] Jiliang Tang, Huiji Gao, Huan Liu, and Atish Das Sarma. 2012. ETrust: Understanding trust evolution in an online world. In *SIGKDD*. 253–261.
[138] Jiliang Tang, Xia Hu, Huiji Gao, and Huan Liu. 2013. Exploiting local and global social context for recommendation. In *IJCAI*. 2712–2718.
[139] Yi Tay, Anh Tuan Luu, and Siu Cheung Hui. 2018. Multi-pointer co-attention networks for recommendation. In *SIGKDD*. 2309–2318.
[140] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. In *ICLR*.
[141] Luke Vilnis and Andrew McCallum. 2015. Word representations via gaussian embedding. In *ICLR*.
[142] Haoyu Wang, Defu Lian, and Yong Ge. 2019. Binarized collaborative filtering with distilling graph convolutional networks. In *IJCAI*. 4802–4808.
[143] Hao Wang, Naiyan Wang, and Dit-Yan Yeung. 2015. Collaborative deep learning for recommender systems. In *SIGKDD*. 1235–1244.
[144] Hongwei Wang, Fuzheng Zhang, Jialin Wang, Miao Zhao, Wenjie Li, Xing Xie, and Minyi Guo. 2018. RippleNet: Propagating user preferences on the knowledge graph for recommender systems. In *CIKM*. 417–426.
[145] 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 *SIGKDD*. 968–977.
[146] Hongwei Wang, Fuzheng Zhang, Miao Zhao, Wenjie Li, Xing Xie, and Minyi Guo. 2019. Multi-task feature learning for knowledge graph enhanced recommendation. In *WWW*. 2000–2010.
[147] Hongwei Wang, Miao Zhao, Xing Xie, Wenjie Li, and Minyi Guo. 2019. Knowledge graph convolutional networks for recommender systems. In *WWW*. 3307–3313.
[148] Jianling Wang, Kaize Ding, Ziwei Zhu, and James Caverlee. 2021. Session-based recommendation with hypergraph attention networks. In *SDM*. 82–90.
[149] Qinyong Wang, Hongzhi Yin, Tong Chen, Junliang Yu, Alexander Zhou, and Xiangliang Zhang. 2021. Fast-adapting and privacy-preserving federated recommender system. *VLDBJ* (2021), 1–20.
[150] Shoujin Wang, Longbing Cao, Yan Wang, Quan Z. Sheng, Mehmet A. Orgun, and Defu Lian. 2021. A survey on session-based recommender systems. *ACM Computing Surveys* 54, 7 (2021), 38.
[151] Shoujin Wang, Liang Hu, Yan Wang, Xiangnan He, Quan Z. Sheng, Mehmet Orgun, Longbing Cao, Nan Wang, Francesco Ricci, and Philip S. Yu. 2020. Graph learning approaches to recommender systems: A review. *arXiv preprint arXiv:2004.11718* (2020).
[152] 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 *WWW*. 3056–3062.
[153] Wen Wang, Wei Zhang, Jun Rao, Zhijie Qiu, Bo Zhang, Leyu Lin, and Hongyuan Zha. 2020. Group-aware long- and short-term graph representation learning for sequential group recommendation. In *SIGIR*. 1449–1458.
[154] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. 2019. KGAT: Knowledge graph attention network for recommendation. In *SIGKDD*. 950–958.
[155] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural graph collaborative filtering. In *SIGIR*. 165–174.
[156] Xiang Wang, Hongye Jin, An Zhang, Xiangnan He, Tong Xu, and Tat-Seng Chua. 2020. Disentangled graph collaborative filtering. In *SIGIR*. 1001–1010.
[157] Xiang Wang, Dingxian Wang, Canran Xu, Xiangnan He, Yixin Cao, and Tat-Seng Chua. 2019. Explainable reasoning over knowledge graphs for recommendation. In *AAAI*. 5329–5336.
[158] Xiao Wang, Ruijia Wang, Chuan Shi, Guojie Song, and Qingyong Li. 2020. Multi-component graph convolutional collaborative filtering. In *AAAI*. 6267–6274.
[159] Xiang Wang, Yaokun Xu, Xiangnan He, Yixin Cao, Meng Wang, and Tat-Seng Chua. 2020. Reinforced negative sampling over knowledge graph for recommendation. In *WWW*. 99–109.

[160] Yun Wang, Lun Du, Guojie Song, Xiaojun Ma, Lichen Jin, Wei Lin, and Fei Sun. 2019. Tag2Gauss: Learning tag representations via Gaussian distribution in tagged networks. In *IJCAI*. 3799–3805.
[161] Yifan Wang, Suyao Tang, Yuntong Lei, Weiping Song, Sheng Wang, and Ming Zhang. 2020. DisenHAN: Disentangled heterogeneous graph attention network for recommendation. In *CIKM*. 1605–1614.
[162] Ze Wang, Guangyan Lin, Huobin Tan, Qinghong Chen, and Xiyang Liu. 2020. CKAN: Collaborative knowledge-aware attentive network for recommender systems. In *SIGIR*. 219–228.
[163] Ziyang Wang, Wei Wei, Gao Cong, Xiao-Li Li, Xian-Ling Mao, and Minghui Qiu. 2020. Global context enhanced graph neural networks for session-based recommendation. In *SIGIR*. 169–178.
[164] Yinwei Wei, Xiang Wang, Liqiang Nie, Xiangnan He, and Tat-Seng Chua. 2020. Graph-refined convolutional network for multimedia recommendation with implicit feedback. In *ACM MM*. 3541–3549.
[165] Yinwei Wei, Xiang Wang, Liqiang Nie, Xiangnan He, Richang Hong, and Tat-Seng Chua. 2019. MMGCN: Multi-modal graph convolution network for personalized recommendation of micro-video. In *ACM MM*. 1437–1445.
[166] Jason Weston, Ron J. Weiss, and Hector Yee. 2013. Nonlinear latent factorization by embedding multiple user interests. In *RecSys*. 65–68.
[167] Chuhan Wu, Fangzhao Wu, Yang Cao, Yongfeng Huang, and Xing Xie. 2021. FedGNN: Federated graph neural network for privacy-preserving recommendation. *arXiv preprint arXiv:2102.04925* (2021).
[168] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. 2019. Simplifying graph convolutional networks. In *ICML*. 6861–6871.
[169] Jiancan Wu, Xiangnan He, Xiang Wang, Qifan Wang, Weijian Chen, Jianxun Lian, and Xing Xie. 2022. Graph convolution machine for context-aware recommender system. *FCS* 16, 6 (2022), 1–12.
[170] Jiancan Wu, Xiang Wang, Fuli Feng, Xiangnan He, Liang Chen, Jianxun Lian, and Xing Xie. 2021. Self-supervised graph learning for recommendation. In *SIGIR*. 726–735.
[171] Le Wu, Junwei Li, Peijie Sun, Richang Hong, Yong Ge, and Meng Wang. 2020. Diffnet++: A neural influence and interest diffusion network for social recommendation. *TKDE* (2020), 1–1.
[172] Le Wu, Peijie Sun, Yanjie Fu, Richang Hong, Xiting Wang, and Meng Wang. 2019. A neural influence diffusion model for social recommendation. In *SIGIR*. 235–244.
[173] Le Wu, Yonghui Yang, Kun Zhang, Richang Hong, Yanjie Fu, and Meng Wang. 2020. Joint item recommendation and attribute inference: An adaptive graph convolutional network approach. In *SIGIR*. 679–688.
[174] 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 *WWW*. 2091–2102.
[175] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. 2019. Session-based recommendation with graph neural networks. In *AAAI*. 346–353.
[176] 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).
[177] Shiwen Wu, Yuanxing Zhang, Chengliang Gao, Kaigui Bian, and Bin Cui. 2020. Garg: Anonymous recommendation of point-of-interest in mobile networks by graph convolution network. *DSE* 5, 4 (2020), 433–447.
[178] Yao Wu, Christopher DuBois, Alice X. Zheng, and Martin Ester. 2016. Collaborative denoising auto-encoders for top-n recommender systems. In *WSDM*. 153–162.
[179] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S. Yu Philip. 2020. A comprehensive survey on graph neural networks. *TNNLS* 32, 1 (2020).
[180] Lianghao Xia, Chao Huang, Yong Xu, Peng Dai, Xiyue Zhang, Hongsheng Yang, Jian Pei, and Liefeng Bo. 2021. Knowledge-enhanced hierarchical graph transformer network for multi-behavior recommendation. In *AAAI*. 4486–4493.
[181] Xin Xia, Hongzhi Yin, Junliang Yu, Yingxia Shao, and Lizhen Cui. 2021. Self-supervised graph co-training for session-based recommendation. In *CIKM*. 2180–2190.
[182] Xin Xia, Hongzhi Yin, Junliang Yu, Qinyong Wang, Lizhen Cui, and Xiangliang Zhang. 2021. Self-supervised hypergraph convolutional networks for session-based recommendation. In *AAAI*. 4503–4511.
[183] Xu Xie, Zhaoyang Liu, Shiwen Wu, Fei Sun, Cihang Liu, Jiawei Chen, Jinyang Gao, Bin Cui, and Bolin Ding. 2021. CausCF: Causal collaborative filtering for recommendation effect estimation. In *CIKM*. 4253–4263.
[184] Xu Xie, Fei Sun, Xiaoyong Yang, Zhao Yang, Jinyang Gao, Wenwu Ou, and Bin Cui. 2021. Explore user neighborhood for real-time e-commerce recommendation. In *ICDE*. 2464–2475.
[185] 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 *IJCAI*. 3940–3946.
[186] Huance Xu, Chao Huang, Yong Xu, Lianghao Xia, Hao Xing, and Dawei Yin. 2020. Global context enhanced social recommendation with hierarchical graph neural networks. In *ICDM*. 701–710.

[187] Han Xu, Yaxin Li, Wei Jin, and Jiliang Tang. 2020. Adversarial attacks and defenses: Frontiers, advances and practice. In *SIGKDD*. 3541–3542.
[188] Yishi Xu, Yingxue Zhang, Wei Guo, Huifeng Guo, Ruiming Tang, and Mark Coates. 2020. GraphSAIL: Graph structure aware incremental learning for recommender systems. In *CIKM*. 2861–2868.
[189] Zuoxi Yang and Shoubin Dong. 2020. HAGERec: Hierarchical attention graph convolutional network incorporating knowledge graph for explainable recommendation. *KBS* 204 (2020), 106194.
[190] 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 *SIGKDD*. 974–983.
[191] Junliang Yu, Hongzhi Yin, Jundong Li, Min Gao, Zi Huang, and Lizhen Cui. 2020. Enhance social recommendation with adversarial graph convolutional networks. *TKDE* (2020), 1–1.
[192] Junliang Yu, Hongzhi Yin, Jundong Li, Qinyong Wang, Nguyen Quoc Viet Hung, and Xiangliang Zhang. 2021. Self-supervised multi-channel hypergraph convolutional network for social recommendation. In *WWW*. 413–424.
[193] Hao Yuan, Jiliang Tang, Xia Hu, and Shuiwang Ji. 2020. XGNN: Towards model-level explanations of graph neural networks. In *SIGKDD*. 430–438.
[194] Hao Yuan, Haiyang Yu, Shurui Gui, and Shuiwang Ji. 2020. Explainability in graph neural networks: A taxonomic survey. *arXiv preprint arXiv:2012.15445* (2020).
[195] Hao Yuan, Haiyang Yu, Jie Wang, Kang Li, and Shuiwang Ji. 2021. On explainability of graph neural networks via subgraph explorations. In *ICML*. 12241–12252.
[196] Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. 2016. Collaborative knowledge base embedding for recommender systems. In *SIGKDD*. 353–362.
[197] Jiani Zhang, Xingjian Shi, Shenglin Zhao, and Irwin King. 2019. STAR-GCN: Stacked and reconstructed graph convolutional networks for recommender systems. In *IJCAI*. 4264–4270.
[198] Muhan Zhang and Yixin Chen. 2020. Inductive matrix completion based on graph neural networks. In *ICLR*.
[199] Mengqi Zhang, Shu Wu, Xueli Yu, Qiang Liu, and Liang Wang. 2022. Dynamic graph neural networks for sequential recommendation. *TKDE* (2022), 1–1.
[200] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. 2019. Deep learning based recommender system: A survey and new perspectives. *ACM Computing Surveys* 52, 1 (2019), 38.
[201] Shijie Zhang, Hongzhi Yin, Tong Chen, Quoc Viet Nguyen Hung, Zi Huang, and Lizhen Cui. 2020. GCN-based user representation learning for unifying robust recommendation and fraudster detection. In *SIGIR*. 689–698.
[202] Yongfeng Zhang, Qingyao Ai, Xu Chen, and Pengfei Wang. 2018. Learning over knowledge-base embeddings for recommendation. *arXiv preprint arXiv:1803.06540* (2018).
[203] Yuan Zhang, Fei Sun, Xiaoyong Yang, Chen Xu, Wenwu Ou, and Yan Zhang. 2020. Graph-based regularization on embedding layers for recommendation. *TOIS* 39, 1 (2020), 1–27.
[204] 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 *SIGKDD*. 2347–2357.
[205] Lei Zheng, Chun-Ta Lu, Fei Jiang, Jiawei Zhang, and Philip S. Yu. 2018. Spectral collaborative filtering. In *RecSys*. 311–319.
[206] Yujia Zheng, Siyi Liu, Zekun Li, and Shu Wu. 2020. DGTN: Dual-channel graph transition network for session-based recommendation. In *ICDMW*. 236–242.
[207] Huachi Zhou, Qiaoyu Tan, Xiao Huang, Kaixiong Zhou, and Xiaoling Wang. 2021. Temporal augmented graph neural networks for session-based recommendations. In *SIGIR*. 1798–1802.
[208] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. 2020. Graph neural networks: A review of methods and applications. *AI Open* 1 (2020), 57–81.
[209] Kun Zhou, Hui Wang, Wayne Xin Zhao, Yutao Zhu, Sirui Wang, Fuzheng Zhang, Zhongyuan Wang, and Ji-Rong Wen. 2020. S3-rec: Self-supervised learning for sequential recommendation with mutual information maximization. In *CIKM*. 1893–1902.
[210] Qi Zhou, Yizhi Ren, Tianyu Xia, Lifeng Yuan, and Linqiang Chen. 2019. Data poisoning attacks on graph convolutional matrix completion. In *ICA3PP*. 427–439.
[211] Dingyuan Zhu, Ziwei Zhang, Peng Cui, and Wenwu Zhu. 2019. Robust graph convolutional networks against adversarial attacks. In *SIGKDD*. 1399–1407.

Received 19 April 2021; revised 27 January 2022; accepted 20 March 2022