Survey on GNN in KG-based Recommender System

Jingyue Shen
University of California, Los Angeles
brianshen@ucla.edu

Boyuan He
University of California, Los Angeles
boyuan_heboyuan@live.com

Haochen Li
University of California, Los Angeles
joshuali1997@yahoo.com

ABSTRACT

We present a survey in regards to the state of art application of Graph Neural Network in knowledge graphs based recommender system and compare their efficiency and accuracy. Our survey is influenced by works from Guo et al. [9] and Yang et al. [19]

1 INTRODUCTION

Today, recommender systems are used in almost every applications. Traditionally, there are two general categories of methods to do recommendation: collaborative filtering based methods and content-based methods. Collaborative filtering based methods focus on finding similar users and do recommendation based on those users. One issue with this type of method is that, when user-item iterations is sparse, their performance suffers. Content-based methods incorporate user/item's side information as additional features. However, they have the assumption that each feature is independent, thus failing to capture high-order relations between users/items.

With the exponential growth of heterogenous data, the relations among different items and their attributes can be very complex, and a graph representation would be a more ideal way to model these relations. A recent trend is to use GNN to do recommendation on graph data. And we would like to explore the usage of GNN in knowledge graph based recommender system in particular. In general, knowledge graph based recommender system can use the knowledge graph to perform more accurate recommendation when user-item interactions are sparse. And with the help of GNN, those systems are able to aggregate different relation/entity information effectively, so that they can discover hidden connections among items in heterogenous knowledge graphs.

2 RELATED WORK

We found two survey papers related to our topic. Guo et al. [9] conducted a survey on knowledge-graph based recommender systems. But their work does not include some categories of GNN-based method. Specifically, some work classified as "relation-unaware" and "relation-aware node categorization" method in our work are not shown in their paper.

Yang et al. [19] did a survey on GNN-based knowledge-aware deep recommender systems. But that paper does not include new works emerged in 2020 and 2021. Our work makes a comprehensive summary of the usage of GNN in knowledge-graph based recommender system update to date. And we provide a different perspective on classifying those systems.

3 RELATION-UNAWARE METHOD

Relation unaware recommendation system doesn't distinguish between different relations (edges) in a KG. The papers that we surveyed mainly focused on "how to pick nodes for aggregation" and "how to aggregate the nodes picked".

Fan et al. [11]. argue that existing intent recommendations, which are widely used in e-commerce, rely too heavily on laboring features and fail to utilize rich interaction information. Thus, he proposed a novel Metapath-guided Embedding method for Intent Recommendation (MEIRec) to model complex objects and the interactions in a recommendation. Specifically, MEIRec would first embed user, item, and user past queries into vectors. Then a user or query is represented by its neighbors along different metapath on heterogeneous GNN. For example, one metapath can be user-item-query (figure 1 below). In this case, MEIRec would aggregate 2nd level neighbor query q1 to get 1st level neighbor item i1 using the average function, then aggregate item i1 to get embedding for the user u1 using LSTM. The reason for the different aggregation method is that users' interaction query can item have timestamps, and use LSTM would help capture that information. Repeating similar processes on different metapaths will produce multiple embeddings for a user, and aggregating all those different embeddings together to get final user representation (query embedding is done similarly). After getting all the embeddings using GNN metapah info, MEIRec feed user, query, and other static information to an MLP layer to predict how likely user will search a query.

The metapath approach helps to choose good node candidates improves performance and reduces computation requirement comparing to aggregate every node. The use of LSTM for user node and average function for other node improves performance comparing to simply take the average for every node. The improvement is about 5-10 percent better AUC on Taobao's mobile application dataset.

However, this metapath based approach also means that domain-specific knowledge is required to implement correct metapaths, and low-quality metapaths would almost definitely hurt performance.

Instead of pre-configured metapah Ying et al.[10] propose to use random walk to assign importance scores to nodes and aggregate the neighbor node with high scores. Specifically, to generate an embedding for a node n in the graph, they would first use a random walk to visit its neighbors. How many times a neighbor node gets visited is used as the importance score. Scores and features of neighbors with high scores are thrown into a convolutional layer to generate an intermediate representation vector. The vector is then concatenated with the current node n's other attribute and through into another dense neural network to get partial representations of n. The vector is then concatenated with the current node n's other attribute and through into another dense neural network to get partial representations of n. Figure 2 below is a simplified example

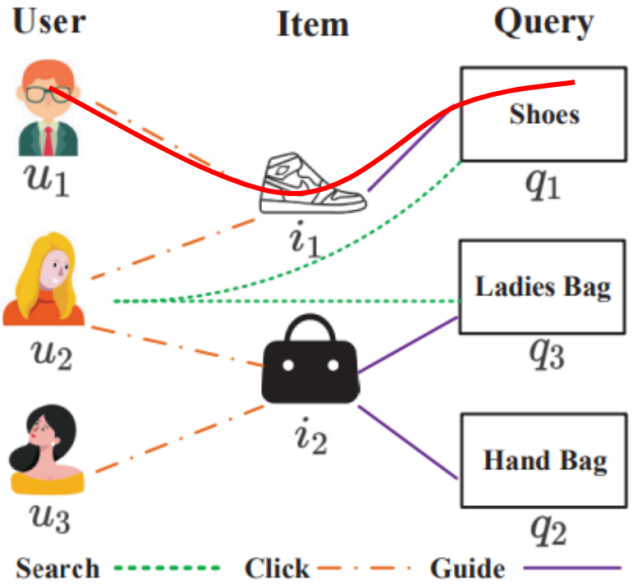

Figure 1: example of a metapath of user-item-query, marked in red.

of random walk, the first random walk A -> B ->C, second random walk A -> C ->E, and the third random walk A -> B -> C.

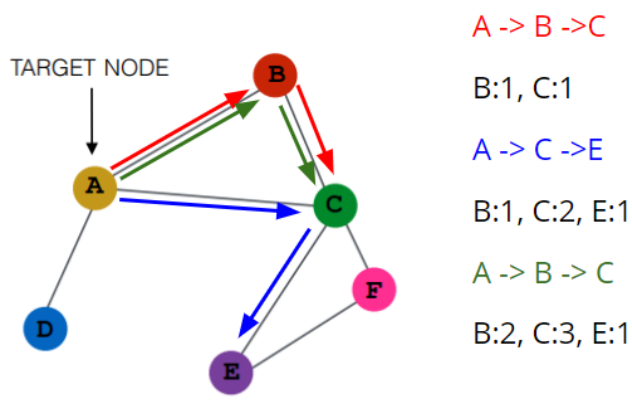

Figure 2: simplified example of random walk

The random walk based method doesn't require pre-defined paths, and it allows use programmer to specify the number of nodes to aggregate not matter what kind of graph they are working which (e.g. only pick 4 nodes in terms of score). However, a study[1] has shown that random walk based sampling on HIN might lead to low-quality samples due to the significant bias to the dominant node types and highly visible nodes.

4 RELATION-AWARE METHOD

Relation-aware methods distinguish between different types of edges. We further categorize the relation-aware method into three different categories: node categorization method, and attentive aggregation method and explicit relation aggregation method. The node categorization method categorizes an edge by the types of nodes it is connecting to. So for instance, the E-commerce graph, user-item, and user-query would be two different relations. The subgraph categorization method split the neighborhood graph into multiple subgraphs, and all edges of a subgraph belong to only one specific categorization. The attentive aggregation method assigns each relation with different weights and covert the KG to a weighted graph and aggregate all neighbor nodes into a context vector using weighted sum. The explicit relation aggregation method directly incorporate relation embedding into the node embedding, in order to preserve semantic meaning of relations in node embedding. Some of the papers we are going to introduce can be categorized into multiple categories.

4.1 Relation-aware node categorization method

Xu et al.[3] argue that the existing recommendation system doesn't take into account different types of relations when doing aggregation for a node. For instance, the edge between user and item and user and selling agent have vastly different implications. Thus, they proposed to use different aggregators for different edges. Specifically, if a user node u has two edges to a selling agent and an item, then it would need 2 different aggregators for each edge. Each aggregator taking in u's embedding vector and selling agent/item's embedding vector, feed it into the neural network to get an intermediate vector. We would get multiple intermediate vectors along a metapath. They assign a weight to (described below) each intermediate vector, concatenate it with original embedding and then feed it into a fully connected layer and a nonlinear activation function. Figure 3 below is an example KG and different aggregators.

To calculate the weight of an intermediate embedding from a particular metapath, they feed v's neighbor's embedding and v into two learnable matrices that output two vectors, dot product these two vectors together, and pass to a nonlinear activation function to get weight.

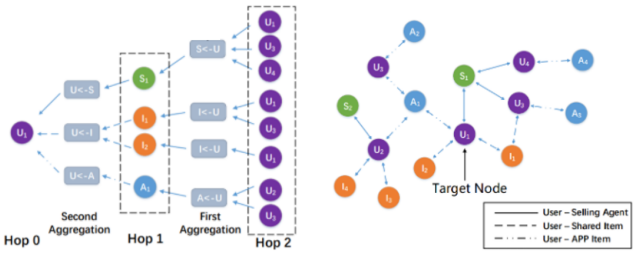

Figure 3: example of different aggregator for different type of edge

Besides the pros and cons that come with metapath based method, this relational aware method helps to capture different implication that comes with different types of edges, which can be very significant depending on what kind of KG is applied. The assigned weight also provides a potential future work: use assigned weight as an indicator to learn which metapath is important, with a deeper

understanding of the metapath importance, we might be able to build a system to automatically pick effective metapath.

Zhang et al.[4] propose STAR-GCN, which uses graph convolutional encoders and neural network decoders to learn the representation of a node in GNN. They argue that STAR-GCN can boost recommendation system performance, especially in cold start cases. Given a node n, an encoder, which is specific for each edge type, takes in the vector embedding of n's neighbor v and uses a link-specific weight matrix to map the vector to lower dimension vector representation. The low dimensional representations of all n's neighbors are summed together and fed into a decoder with two-layer feedforward neural network to generate the updated embedding that has the same dimension as the original embedding. If external features are available, they would first be processed by another two-layer feedforward neural network and then concatenated with the previous embedding. This encoding and decoding process can be performed multiple times to gather attributes from multi-hop neighbors for a node's embedding. Figure 4 below shows an example encode and decode process, encoders are marked with different colors to indicate different types.

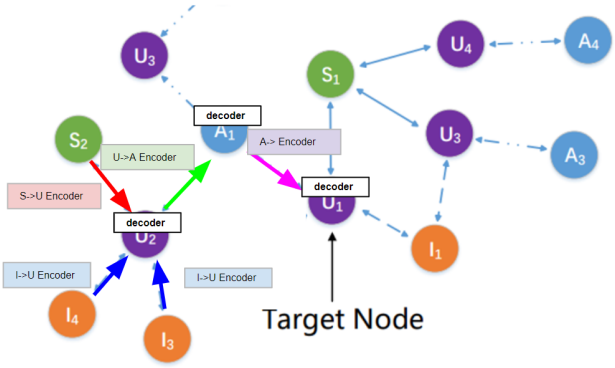

Figure 4: example of encoder and decoder process, encoder are marked different color based on edge type

To tackle the cold start problem, the authors mask a portion of nodes during training. Masked nodes have embedding set to 0, which makes the encoder learn to deal with nodes that don't have embedding. This helps the model do better in cold-start scenarios where a new node that doesn't have embedding can be set to 0.

The encoder-decoder approach helps to limit the model space complexity by mapping the embedding to lower dimensions and solve the cold start problem by train the encoder to learn 0 vector input. However, the lower dimension size needs to be picked carefully, since too few dimensions might cause underfitting problem.

Zhao et al.[6] proposed the IntentGC framework that utilizes auxiliary information to boost recommendation system performance and uses vector-wise convolutions to reduce unnecessary computation waste.

Instead of only using first-order proximity from auxiliary information, they utilize second-order proximity. The similarity between users/items is measured by the number of common auxiliary neighbors of the same type. Using this information, they added edges between users/items with similarity count as edge weight. Moreover, they also consider different kinds of auxiliary information and assign different types to the edges accordingly. Figure 5 below shows an example of new connections added.

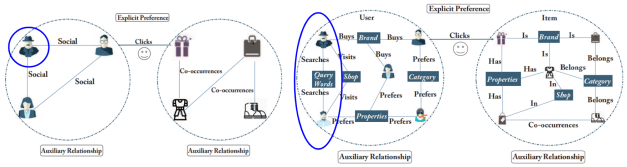

Figure 5: left is traditional KG, right is with auxiliary relation. blue circle is one example of new connection added

Convolute nodes on a graph are extremely expensive. The traditional conventional approach would just concatenate the node and neighbor embedding, and learn a matrix that's multiplied to the on the concatenated vector. However, IntentGC would use two separate filter weights, one for node embedding and one for neighbor embedding. Then the result would add together and feed into a vector-wise layer to get the intermediate representation. This representation for the neighbor is then fed into a convolutional layer and eventually a 3 level dense layer to get the new embedding for a node.

The paper presented a new approach to fully utilize the auxiliary information provided to the recommendation system. Although adding new edges to the graph would require a lot more computations, they were able to use vector-wise convolution to combat it. Moreover, vector-wise convolution can even produce better results since it eliminates useless feature interactions. I feel the vector-wise convolution approach can be generalized to many other models since almost all the models mentioned in this survey uses some sort of matrix-wise convolution. This improvement might also enable other models to convolute more nodes and achieve even better performance.

4.2 Relation-aware attentive aggregation method

The relation-aware attentive aggregation method treats different relations in knowledge graphs differently. Specifically, for each entity, this method assigns a weight to those edges connecting its neighbors. Different types of relations will have different weights. Then the context embedding is obtained by weighted sum of the embedding of neighbor nodes.

In this section, we will first introduce some general recommender systems that use this method. Then we will introduce some recommender systems that are built for specific domains like social networks.

Wang et al. proposes an end-to-end model called KGCN [14]. This system's workflow is shown in Fig 6.

For each user, it defines a scoring function $\pi_r^u = g(u, r)$ between each user and relation in the knowledge graph, where function $g()$

is inner product. The function has mapping $R^d x R^d \rightarrow R$, where d is the dimension of user embedding and relation embedding. So this function defines the user's preference towards different relations in the Knowledge Graph. For example, a specific user might pay more attentions on books' genre, rather than books' authors. Then the scores are converted to normalized weights across an entity's direct neighbors: $\tilde{\pi}^u_{r\,v,e} = \frac{exp(\pi^u_{r_{v,e}})}{\sum_{e \in N(v) exp(\pi^u_{r\,v,e})}}$, where v represents current node, e represents an entity, and $N(v)$ represents v's direct neighbors. And then neighbors' embeddings are aggregated into a single context vector using weighted sum: $v^u_{N(v)} = \sum_{e \in N(v)} \tilde{\pi}^u_{r\,v,e}\mathbf{e}$.

To combine the context embedding with current node's embedding, the paper explores three types of aggregators: sum aggregator, concat aggregator and context-only aggregator:

$$agg_{sum} = \sigma(\mathbf{W} \cdot (v + v^u_{S(v)}) + b)$$

$$agg_{concat} = \sigma(\mathbf{W} \cdot concat(v, v^u_{S(v)}) + b)$$

$$agg_{context-only} = \sigma(\mathbf{W}^u_{S(v)} + b)$$

where σ is a nonlinear function such as ReLU. And it will take the embedding after L-layers of aggregation as the final node embedding.

This paper extends the traditional GCN approaches to the knowledge graph by aggregating weighted neighborhood information. In this way, it is able to capture user's preference on different relations and do recommendation accordingly.

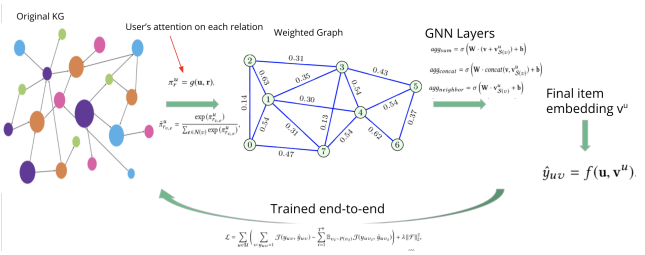

Figure 6: KGCN's workflow

Later, Wang et al. build on top of the KGCN, and propose the KGNN-LS model[13]. It is basically the same as KGCN, but with an additional regularization term in its loss function to regularize the edge weights. This regularization term tries to capture the KG structure to regularize the recommender system's model learning. Specifically, this term based on the label smoothness assumption: adjacent items tend to have similar user relevance labels.

The authors further investigate the model performance in cold-start scenarios. Compared to other baseline models with or without KG, KGNN-LS's performance does not degrade much, even when training set is 20% of original size.

While KGCN/KGNN-LS only focus on item-side knowledge graph, KGAT[15] try to incorporate user-item interactions into the item-side knowledge graph and call it "Collaborative Knowledge Graph". Specifically, it adds two additional relations: *Interact* and *InteractedBy* between each user-item interaction. The workflow of KGAT is shown in Fig.7.

By incorporate users into the knowledge graph, the GNN layers are able to aggregate information through high-order relation paths that connect different users. To some extent, by this way, this approach incorporates the advantages of collaborative-filtering based recommendation methods.

Similar to KGNN-LS, this work also adds a regularization term into loss function to preserve KG structure information. It uses TransR[8], a well-known knowledge graph embedding method for knowledge graph completion, as the initial entity embedding before doing aggregation by GNN layers. Given a triplet (h,r,t), TransR embeds each entity and relation by preserving the translation principle $e^r_h + e_r \approx e^r_t$. In order to preserve the invariant, it defines an energy score for each triplet (h,r,t) and tries to minimize it:

$$g(h, r, t) = \|W_r e_h + e_r - W_r e_t\|^2_2,$$

where W_r is the transformation matrix that project head and tail entities from entity space to relation space.

The general architecture of KGAT is similar to KGCN/KGNN-LS, as we can see from Fig. 6 and Fig. 7. However, the design of attention weight, how to aggregate context embedding with current node's embedding, and how to represent the final node embedding are different. The differences are as follows:

For attention weight, it uses "relational attention mechanism" defined as:

$$\pi(h, r, t) = (\mathbf{W}_r \mathbf{e}_t)^T tanh(\mathbf{W}_r \mathbf{e}_h + \mathbf{e}_r),$$

so that more information will be propagated for closer entities.

For aggregating context embedding with current node's embedding, KGAT explores sum aggregator, concat aggregator, and a self-defined bi-interation aggregator, which tries to capture different aspect of feature interactions:

$$f_{Bi-Interaction} = LeakyReLU(W_1(e_h + e_{N_h})) + LeakyReLU(W_1(e_h \odot e_{N_h})),$$

where e_{N_h} is the aggregated context vector.

For representing the final user/item embedding, the authors concatenate all the L-layer embeddings, in order to capture all the aggregated information with different path length.

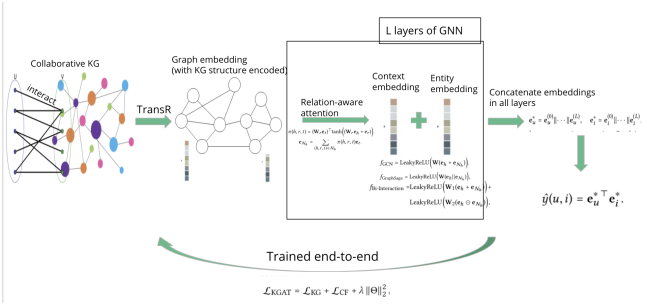

Figure 7: KGAT's workflow

Relation-aware attentive aggregation method could also be applied with random path approach to capture more features. The Yang et al[18] proposed CGAT to capture both local graph context and nonlocal graph context. The model is targeting in modeling user-specific preferences on entitled and capturing explicitly the

nonlocal graph context. To capture local graph context along with taking user's personalized preferences into consideration, the model uses users-specific graph attention mechanisms to aggregate the neighboring information of an entity in the knowledge graph. It firstly integrates neighbor nodes and uses linear transformation to embed the relation. It applied a weight matrix to the concatenation of neighbor embedding. Then it calculates the user-specific attention score for the importance of entity to entity h for target u through using an operation π performed by a single layer feed neural network. Then with all these parameters, the model can obtain the local neighborhood embedding and finally aggregate the embedding of entity h to get the local contextual embedding. For non-local graph context, it uses the biased random walk based on GRU module for aggregation. To encourage a wider search, the model sets the probability to travel to an unexplored node to be high. These random walks would output a set of paths. Then the model sorts the entities according to their frequency appearing in these paths. All these entities are denoted as nonlocal graph context. Then it aggregates the entities to form a non-local contextual embedding. Finally, it uses gate mechanism to integrate the local and non-local embedding.

Besides the above general model with relation-aware attentive aggregation, some other models with attentive mechanism that applies to specific domain such as social network or video has also proved to be effective.

Fan et al [2] propose a novel graph neural network GraphRec that intends to (1) aggregate/combine the social graph and user-item graph, (2) capture interaction and opinions between users and items jointly, and (3) distinguish social relations with heterogeneous strength. The entire architecture, as shown in Figure 8, contains three components: user modeling, item modeling and rating prediction. User modeling is trying to learn the latent factors of users. To learn this, people need to address the challenge of combining the user-item graph and social graph. Therefore, the paper presents item aggregation and social aggregation to combine user latent factors from both item-space and social-space. Item aggregation is implemented to learn item-space user latent factors. Given the rating of items, GraphRec tried to represent user's opinion in an embedded opinion vector. It then used concatenation of item embedding and MLP to embed a list of vectors containing the opinion-aware representations x_ia between a user i and an item a. So with this new representation, the item space latent factor could be computed through a function that uses an arbitrary aggregation function to aggregate all the opinion-aware representations across all items for a particular target user in equation (1).

$$h = \sigma(W \cdot Aggre(x_ia, \forall a \in C(i)) + b) \tag{1}$$

One of the aggregation function is the mean-based aggregation function. However, there is also some limitation when using the mean-based aggregation function since influence on different users could be vary dramatically while mean based uses fixed weights for all items. To address the problem, the paper uses attention network by the inspiration with attention mechanism to parameterize the item attention to replace the fixed weights in mean-based aggregation function. With this same reasoning as item aggregation, both social aggregation and Social aggregation is both used to compute the user latent factor under social space and item latent factor under item space respectively. With a concatenation function, all the latent factor would be aggregate together for model prediction.

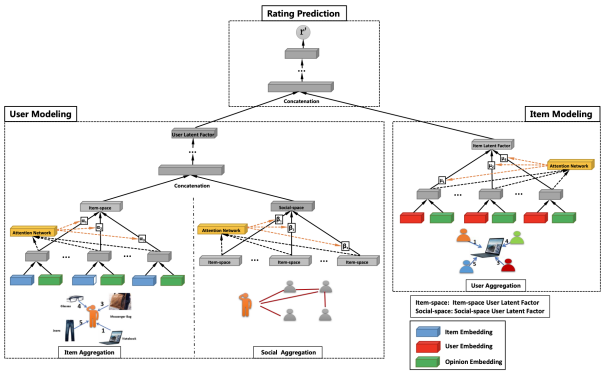

Figure 8: The overall architecture of GraphRec

Besides, Song et al[12] claim that most recommendation system failed to address users dynamic interest and their social influence jointly. In their paper, they propose DGRec, a attention based graph neural network to build a recommendation system to address both users' dynamic interests and social influence. This model contains four modules as shown in figure 9. The first and second modules are using RNN to model a user's sequence of the user's current session to grasp his/her dynamic/short term individual interests and his/her friends' dynamic preference since these interests are time sensitive. It will also grasp his/her friends' interest by representing them as a single vector since they are static and not time sensitive. Then it is using a Graph attention network to obtain a combination representation of the target user and his/her friends. It encodes the relationship graph in a graph where nodes represent users' dynamic interest and edges represent their friendship and then propagate features along edges using a message passing algorithm where an attention mechanism will weigh the features during the traveling. This is achieved by firstly calculating the similarity between the user's representation and her friends' representation at layer l where layer l means iteration that message passing has been performed. And then use a linear combination between the (1) and their friends' interest to model the mixture of friends' interest to complete the aggregation. This process will be repeated in a fixed number of iteration to achieve the optimal mixture representation. When performing a prediction/recommendation, the user's representation will be passed in a fully-connected layer to output a value.

Wu et al[16] also propose a different model, DANSER, as a Dual Graph Attention Network to address the issue that most recommendation system assume the social effects are static. The model's layout, in figure 10, is shown following: first the raw data inputs are one user social network and one user-item interaction graph. Then this raw data will be passed into an embedding layer to create a lower dimension of representation of the data. This representation will be forward into the Dual GCN/GAT layer. The two dual graph

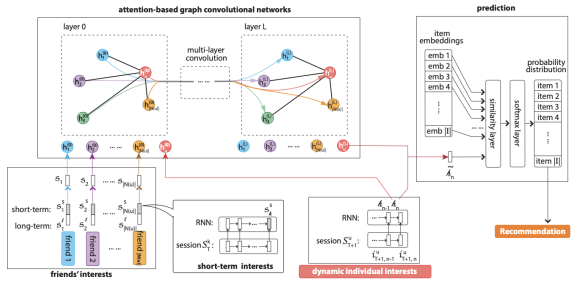

Figure 9: The overall architecture of DGRec

attention network to embed and capture the user static/dynamic preference factor and item static/dynamic attribute factor. This will aggregate the the embedding of the neighboring nodes to propagate the friends or related items' social effects to adjacent node by using four different aggregation operation. The difference between the GCN/GAT operation to capture dynamic and static factors is that models compute different attention weights for dynamic factors by max pooling to choose the dominates feature in euqation(3) while it compute average attention weights for static factors in equation(2).

$$\alpha_{uv}^{P} = \frac{attn_U(\mathbf{W}_P\mathbf{p}_u, \mathbf{W}_P\mathbf{p}_v, \mathbf{W}_E\mathbf{e}_{uv})}{\sum_{w\in\Gamma_U(u)} attn_U(\mathbf{W}_P\mathbf{p}_u, \mathbf{W}_P\mathbf{p}_w, \mathbf{W}_E\mathbf{e}_{uv})}, v \in \Gamma_U(u) \quad (2)$$

$$m_{ud}^{i^+} = \max_{j\in R_I(u)} \{y_{jd} \cdot y_{i^+d}\}, \forall d = 1, \ldots, D \quad (3)$$

After obtaining the representation of the above four factors, they will be passed into the pairwise interaction layers to let the two preference factors and two attribute factors pair-wisely interact with each other. Then the Fusion layer will fuse fuse the four factors into a synthetic one. The output layer will just be a fully connected layer with sigmoid activation function for prediction.

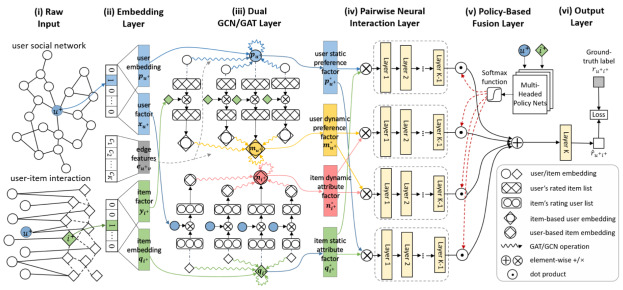

Figure 10: The overall architecture of DANSER

In addition to the domain of social network, attention based aggregation in GNN has also been used in the recommendation system for assigning video hashtag. By observation, the hashtags distribution tends to be a long tail distribution which means that more frequently hashtags tends to have higher probability to occur within the dataset and rare hashtag does not have lower probability so that few common hashtags are dominating the error rate and it is hard to predict the rare hashtags. Li et al[7] proposes a graph neural network model with attentive mechanism to address the issue of long tail hashtag distribution. The model could solve the above issue in following methods shown in figure 11. First, it will constructs a graph by exploring hashtag correlations. Then it uses external knowledge to define the relation property of the edges. In the paper, they concluded four different category of the properties: composition relation, super-subordinate relation, positive relation and co-occurance relation. Then the initial weights will be assigned to edges according to their relations properties and be preserved in a correlation matrix. Correlation matrix would be re-weighted to obtain each node's own feature. After that the information from the correlated matrix will be propagate through the GCN in multiple iterations to obtain the optimal hashtag embedding representation.

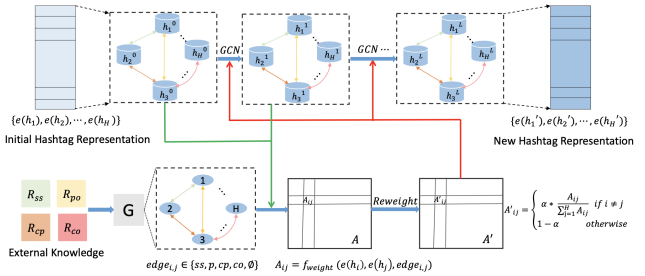

Figure 11: The overall pipeline to obtain optimal hashtag representation through GCN

5 RELATION-AWARE EXPLICIT RELATION AGGREGATION METHOD

In this aggregation method, relations no longer simply serve as attentive weights, but the relation embeddings are explicitly incorporated into item embeddings.

This category is pretty new, and we found only one such paper that is published several months ago, KGIN [17]. The workflow of KGIN is shown in Fig. 12. It has two novel points:

(1) It models an addition embedding called "user intent", which is a distribution over all the relations in the knowledge graph. It use attentive weights on relations to model the user intent.
(2) It explicitly incorporates relation embedding into item embedding, by the formula:

$$e_i^{(1)} = \frac{1}{|N_i|} \sum_{(r,v)\in N_i} e_r \odot e_v^{(0)},$$

where $e_i^{(1)}$ is the item embedding after aggregating direct neighbors of item i. And N_i is the neighbor nodes of item i. So if we expand the formula after L-layer, the equation would be:

$$e_i^{(l)} = \sum_{s\in N_i^l} \frac{e_{r1}}{|N_{s1}|} \odot \frac{e_{r2}}{|N_{s2}|} \odot \ldots \odot \frac{e_{rl}}{|N_{sl}|} \odot e_{sl}^{(0)},$$

where N_i^l is the set of all item i's l-hop paths. We can see that item embedding of l-th layer will incorporate all the relation embeddings from the 1^{st}-hop to l^{th}-hop.

Personally, I think the second point is an innovation, but I doubt the efficacy of modeling user intent. Their ablation studies show the performance after removing user intent and after removing both user intents and relational modeling. But I am curious about what is the performance of KGAT, with relational modeling but without user-intent. If would be great if the paper can show this the comparison between KGIN and KGAT with relational modeling, to prove the usefulness of user intent modeling.

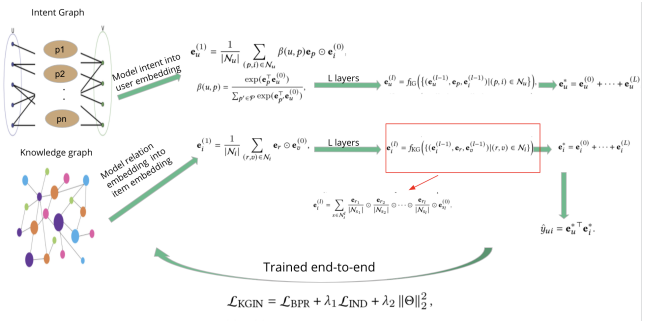

Figure 12: KGIN's workflow

6 OTHER METHODS

In this section, we introduce a method that is so different that we feel it deserve its own section.

Jin et al.[5] argue that existing methods are hard to use since KG connections are sparse or noisy, and compressing nodes together with neighborhood information into a single embedding overlooks the rich interactions among nodes. Thus, they proposed the Neighborhood-based Interaction Model for Recommendation (NIRec). NIRec still uses the meta-path method repeatedly to get desired nodes for consideration. A user node and an item node each would have multiple meta paths. To get the prediction for a connection between a particular user and an item, they first group metapath by length, then for each of the groups they would calculate the similarity in a shift fashion. For instance, if one metapath pair for user u1 and item i1 is u1, i1, a2, i3 (a is extra attributes node) and i1,u3, i4, u4, they would first reverse the item metapath (u4, i4, u3, i1) and shift the metapath to create overlaps (shown at (a) below). The first overlap would be u1, i1, the second would be u1, u3 and i1, i1.

They perform element-wise AND operation for each overlap and add them together if there are multiple overlaps. Keep doing this and we will get multiple ANDed vectors. Concatenate these vectors together and we get a longer vector. Repeating these operations for different metapath combinations we get multiple vectors, which are then stacked together to form a matrix with each cell representing the interaction between different nodes. Figure 13 below is an example of how an interaction matrix is constructed.

For interaction in the interaction matrix, they measure its importance with trainable matrices. The importance value along with

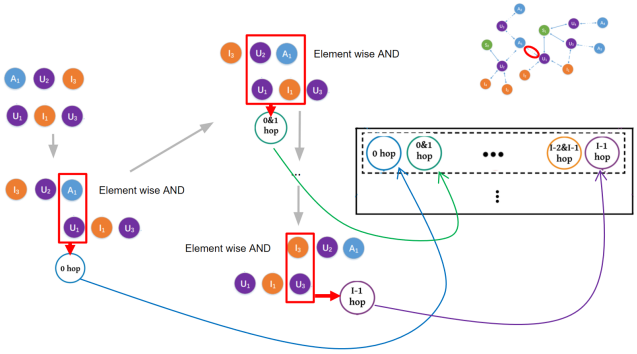

Figure 13: example interaction matrix construction. The two metapath nodes are turned into a row in interaction matrix

the vector representation of interactions are used to train a matrix that gives the embedding of the interaction. Multiple of these embedding vectors can be derived by repeating previous steps for different length metapath, and adding all those vectors together (with trained importance for each) will produce the final embedding for the interaction between a user and an item.

Instead of embedding each node and perform analysis, this paper embeds each edge by evaluating interactions from nodes on metapath. This solves the problem of sparse connection in KG and captures the rich interactions among nodes. However, we also need to note that this potentially adds much more computation burden, thus it is not likely to run well on a dense KG. Moreover, it adds interactions between different nodes on different metapath by considering all combinations, which might introduce interactions between nodes that don't have meaningful interactions. Consider the case where metapaths have 10 nodes, then the interaction matrix would consider the interaction of node that have 18 nodes in between, which might not be that meaningful.

7 COMMON DATASET

In this section we present commonly used datasets, their characteristics and papers that uses them. Dataset details are in the table 1, some specialized datasets have more than one type of item, we choose to sum all those into one item count, and when there is inconsistency in node and edge count between dataset and paper, we choose the official discretion from the dataset.

8 METRICS AND PERFORMANCE

The metrics that the papers have been using are including AUC (Area under the ROC curve), MRR(mean reciprocal rank), recall@N, discounted cumulative Gain, F1, RMSE(Root Mean Square Error) and MAE(Mean Absoluted Error). To explain in detail, recall@N means the proportion of the top-K recommended items that are in the evaluation set and NDGA means normalized distributed cumulative gain for a top-K recommended items where k is normally 20.

In respect to the performance, a sample of performance is listed in Figure 14. We categorize the performance based on the dataset and showed the comparison of performance for different model.

Dataset	Users	Items	edges	Used by
Beidian	87105	77982		RecoGC
Flixster	2341	2956	26173	STAR-GCN
Douban	2999	3000	136891	STAR-GCN,DGRec
ML-100K	943	1682	100000	STAR-GCN
ML-1M	6040	3706	1M	STAR-GCN,CGAT
ML-10M	69878	10677	10M	STAR-GCN
ML-20M	138493	27278	20M	STAR-GCN,NIRec,KGCN,KGNN-LS
LastFM	92800	1892	105551	NIRec, KGCN , KGNN-LS, KGAT, KGIN,CGAT
AMiner	1.7M	2.1M	1M	NIRec
Taobao	278M	250M	1.8B	IntentGC
Amazon	14M	6M	12M	IntentGC, NIRec, KGAT, KGIN
Epinions	18088	261649	508837	GraphRec, DANSER
Ciao	7317	104975	11781	GraphRec
YFCC200M	8126	23054		
Book-Crossing	278858	271379	1.1M	KGCN, KGNN-LS,CGAT
Dianping-Food	2.3M	1,362	23,4M	KGNN-LS, DGRec
Yelp	1.2M	156,639	4.7M	KGAT
Alibaba-iFashion	114,737	30,040	1.8M	KGIN
Delicious	1650	4282		DGRec
Wechat	200000	100000	2M	DANSER

Table 1: Dataset details

Dataset	Model	Performance							
		AUC	MRR	Recall@10	Recall@20	NDCG	F1	RMSE	MAE
Amazon Books	IG(Single) with GraphSage	0.808	1.721						
	IntentGC(Single) with IntentNet	0.826	2.225						
	NIRec	0.749							
	KGAT				0.149	0.101			
Book-Crossing	KGNN-LS	0.744		0.082					
	KGCN-Sum	0.738					0.688		
	KGCN-concat	0.734					0.681		
	KGIN-3				0.169	0.092			
	KGCN-neighbor	0.728					0.679		
douban	DG-Rec(social + temporal)				0.186	0.195			
	Star-GCN							0.727±0.0006	
Flixster	Star-GCN							0.879±0.0030	
MovieLens	NIRec	0.847							
	KGNN-LS	0.979 (20M)		0.155 (20M)					
	Star-GCN							0.895±0.0009(10M)	
Last.FM	NIRec	0.940							
	KGNN-LS	0.803		0.122					
	KGAT				0.087	0.133			
Delicious	DGRec				0.407	0.294			
Yelp	DGRec				0.084	0.143			
Dianping	KGNN-LS	0.850			0.170				
Epinions	GraphRec (80%)							1.063	0.817
	Danser							1.027	0.778
Beidian	RecoGCN		0.263			0.409			

Table 2: Sample models' performance on each dataset

9 FUTURE DIRECTION

One potential future direction of KG-based recommender system is to handle dynamic graphs. Many GNN-based approach in KG-based recommender system requires long training time. If new user/item is added, it needs to train from scratch. So a potential research direction can be, how to efficiently incrementally update the node representations in such systems.

10 CONCLUSION

In this survey paper, we surveyed various usage of GNN in knowledge graph based recommender system. In general, they utilize GNN to aggregate neighbor nodes information and they have various aggregation mechanisms. We categorized them into several categories based on how they model the relations in knowledge graph, and summarized the datasets they used and their performance.

REFERENCES

[1] Wayne Xin Zhao Binbin Hu, Chuan Shi and Philip S Yu. 2020. Leveraging Meta-path based Context for Top- N Recommendation with A Neural Co-Attention Model. *KDD '18: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery Data Mining* (2020), 1531–1540. https://doi.org/10.1145/3219819.3219965

[2] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. 2019. Graph Neural Networks for Social Recommendation. In *The World Wide Web Conference* (San Francisco, CA, USA) *(WWW '19)*. Association for Computing Machinery, New York, NY, USA, 417–426. https://doi.org/10.1145/3308558.3313488

[3] Zhenyu Han Yong Li Yujian Xu Xing Xie Fengli Xu, Jianxun Lian. 2019. Graph Convolutional Neural Networks for Web-Scale Recommender Systems. *CIKM '19: Proceedings of the 28th ACM International Conference on Information and Knowledge Management* (2019), 529–538. https://doi.org/10.1145/3357384.3357924

[4] Shenglin Zhao Irwin King Jiani Zhang, Xingjian Shi. 2019. STAR-GCN: Stacked and Reconstructed Graph Convolutional Networks for Recommender Systems. arXiv:1905.13129 [cs.IR]

[5] Yuchen Fang Kounianhua Du Weinan Zhang Yong Yu Zheng Zhang Alexander J. Smola Jiarui Jin, Jiarui Qin. 2020. An Efficient Neighborhood-based Interaction Model for Recommendation on Heterogeneous Graph. *KDD '20: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery Data Mining* (2020), 75–84. https://doi.org/10.1145/3394486.3403050

[6] Ziyu Guan Wei Zhao Wei Ning Guang Qiu Xiaofei He Jun Zhao, Zhou Zhou. 2019. IntentGC: A Scalable Graph Convolution Framework Fusing Heterogeneous Information for Recommendation. *KDD '19: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery Data Mining* (2019), 2347–2357. https://doi.org/10.1145/3292500.3330686

[7] Mengmeng Li, Tian Gan, Meng Liu, Zhiyong Cheng, Jianhua Yin, and Liqiang Nie. 2019. Long-Tail Hashtag Recommendation for Micro-Videos with Graph Convolutional Network *(CIKM '19)*. Association for Computing Machinery, New York, NY, USA, 509–518. https://doi.org/10.1145/3357384.3357912

[8] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning Entity and Relation Embeddings for Knowledge Graph Completion. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence* (Austin, Texas) *(AAAI'15)*. AAAI Press, 2181–2187.

[9] Chuan Qin Hengshu Zhu Xing Xie Hui Xiong Qing He Qingyu Guo, Fuzhen Zhuang. 2020. A Survey on Knowledge Graph-Based Recommender Systems. arXiv:2003.00911 [cs.IR]

[10] Kaifeng Chen Pong Eksombatchai William L. Hamilton Jure Leskovec Rex Ying, Ruining He. 2018. Graph Convolutional Neural Networks for Web-Scale Recommender Systems. *KDD '18: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery Data Mining* (July 2018), 974–983. https://doi.org/10.1145/3219819.3219890

[11] Xiaotian Han Chuan Shi Linmei Hu Biyu Ma Shaohua Fan, Junxiong Zhu and Yongliang Li. 2019. Metapath Guided Heterogeneous Graph Neural Network for Intent Recommendation. *KDD '19: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery Data Mining* (July 2019), 2478–2486. https://doi.org/10.1145/3292500.3330673

[12] Weiping Song, Zhiping Xiao, Yifan Wang, Laurent Charlin, Ming Zhang, and Jian Tang. 2019. Session-Based Social Recommendation via Dynamic Graph Attention Networks. *WSDM '19: Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, 555–563. https://doi.org/10.1145/3289600.3290989

[13] Hongwei Wang, Fuzheng Zhang, Mengdi Zhang, Jure Leskovec, Miao Zhao, Wenjie Li, and Zhongyuan Wang. 2019. Knowledge-Aware Graph Neural Networks with Label Smoothness Regularization for Recommender Systems. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery amp; Data Mining* (Anchorage, AK, USA) *(KDD '19)*. Association for Computing Machinery, New York, NY, USA, 968–977. https://doi.org/10.1145/3292500.3330836

[14] Hongwei Wang, Miao Zhao, Xing Xie, Wenjie Li, and Minyi Guo. 2019. Knowledge Graph Convolutional Networks for Recommender Systems. In *The World Wide Web Conference* (San Francisco, CA, USA) *(WWW '19)*. Association for Computing Machinery, New York, NY, USA, 3307–3313. https://doi.org/10.1145/3308558.3313417

[15] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. 2019. KGAT: Knowledge Graph Attention Network for Recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery amp; Data Mining* (Anchorage, AK, USA) *(KDD '19)*. Association for Computing Machinery, New York, NY, USA, 950–958. https://doi.org/10.1145/3292500.3330989

[16] 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. https://doi.org/10.1145/3308558.3313442

[17] Dingxian Wang Yancheng Yuan Zhenguang Liu Xiangnan He Tat-Seng Chua Xiang Wang, Tinglin Huang. 2021. Learning Intents behind Interactions with Knowledge Graph for Recommendation. arXiv:2102.07057 [cs.IR]

[18] Susen Yang, Yong Liu, Yonghui Xu, Chunyan Miao, Min Wu, and Juyong Zhang. 2020. Contextualized Graph Attention Network for Recommendation with Item Knowledge Graph.

[19] Yu Lin Hang Gao Latifur Khan Yang Gao, Yi-Fan Li. 2020. Deep Learning on Knowledge Graph for Recommender System: A Survey. arXiv:2004.00387 [cs.IR]