Table 1: Statistics of the datasets.

		Amazon-book	Last-FM	Yelp2018
User-Item Interaction	#Users	70,679	23,566	45,919
	#Items	24,915	48,123	45,538
	#Interactions	847,733	3,034,796	1,185,068
Knowledge Graph	#Entities	88,572	58,266	90,961
	#Relations	39	9	42
	#Triplets	2,557,746	464,567	1,853,704

collection. To ensure the quality of the dataset, we use the 10-core setting, *i.e.,* retaining users and items with at least ten interactions.

Last-FM[3]**:** This is the music listening dataset collected from Last.fm online music systems. Wherein, the tracks are viewed as the items. In particular, we take the subset of the dataset where the timestamp is from Jan, 2015 to June, 2015. We use the same 10-core setting in order to ensure data quality.

Yelp2018[4]**:** This dataset is adopted from the 2018 edition of the Yelp challenge. Here we view the local businesses like restaurants and bars as the items. Similarly, we use the 10-core setting to ensure that each user and item have at least ten interactions.

Besides the user-item interactions, we need to construct item knowledge for each dataset. For Amazon-book and Last-FM, we follow the way in [40] to map items into Freebase entities via title matching if there is a mapping available. In particular, we consider the triplets that are directly related to the entities aligned with items, no matter which role (*i.e.,* subject or object) it serves as. Distinct from existing knowledge-aware datasets that provide only one-hop entities of items, we also take the triplets that involve two-hop neighbor entities of items into consideration. For Yelp2018, we extract item knowledge from the local business information network (*e.g.,* category, location, and attribute) as KG data. To ensure the KG quality, we then preprocess the three KG parts by filtering out infrequent entities (*i.e.,* lowever than 10 in both datasets) and retaining the relations appearing in at least 50 triplets. We summarize the statistics of three datasets in Table 1.

For each dataset, we randomly select 80% of interaction history of each user to constitute the training set, and treat the remaining as the test set. From the training set, we randomly select 10% of interactions as validation set to tune hyper-parameters. For each observed user-item interaction, we treat it as a positive instance, and then conduct the negative sampling strategy to pair it with one negative item that the user did not consume before.

4.2 Experimental Settings

4.2.1 ***Evaluation Metrics.*** For each user in the test set, we treat all the items that the user has not interacted with as the negative items. Then each method outputs the user's preference scores over all the items, except the positive ones in the training set. To evaluate the effectiveness of top-K recommendation and preference ranking, we adopt two widely-used evaluation protocols [13, 35]: recall@K and ndcg@K. By default, we set $K = 20$. We report the average metrics for all users in the test set.

4.2.2 ***Baselines.*** To demonstrate the effectiveness, we compare our proposed KGAT with SL (FM and NFM), regularization-based (CFKG and CKE), path-based (MCRec and RippleNet), and graph neural network-based (GC-MC) methods, as follows:

- **FM** [23]: This is a bechmark factorization model, where considers the second-order feature interactions between inputs. Here we treat IDs of a user, an item, and its knowledge (*i.e.,* entities connected to it) as input features.
- **NFM** [11]: The method is a state-of-the-art factorization model, which subsumes FM under neural network. Specially, we employed one hidden layer on input features as suggested in [11].
- **CKE** [38]: This is a representative regularization-based method, which exploits semantic embeddings derived from TransR [19] to enhance matrix factorization [22].
- **CFKG** [1]: The model applies TransE [2] on the unified graph including users, items, entities, and relations, casting the recommendation task as the plausibility prediction of $(u, Interact, i)$ triplets.
- **MCRec** [14]: This is a path-based model, which extracts qualified meta-paths as connectivity between a user and an item.
- **RippleNet** [29]: Such model combines regularization- and path-based methods, which enrich user representations by adding that of items within paths rooted at each user.
- **GC-MC** [26]: Such model is designed to employ GCN [17] encoder on graph-structured data, especially for the user-item bipartite graph. Here we apply it on the user-item knowledge graph. Especially, we employ one graph convolution layers as suggested in [26], where the hidden dimension is set equal to the embedding size.

4.2.3 ***Parameter Settings.*** We implement our KGAT model in Tensorflow. The embedding size is fixed to 64 for all models, except RippleNet 16 due to its high computational cost. We optimize all models with Adam optimizer, where the batch size is fixed at 1024. The default Xavier initializer [8] to initialize the model parameters. We apply a grid search for hyper-parameters: the learning rate is tuned amongst $\{0.05, 0.01, 0.005, 0.001\}$, the coefficient of L_2 normalization is searched in $\{10^{-5}, 10^{-4}, \cdots, 10^{1}, 10^{2}\}$, and the dropout ratio is tuned in $\{0.0, 0.1, \cdots, 0.8\}$ for NFM, GC-MC, and KGAT. Besides, we employ the node dropout technique for GC-MC and KGAT, where the ratio is searched in $\{0.0, 0.1, \cdots, 0.8\}$. For MCRec, we manually define several types of user-item-attribute-item meta-paths, such as *user-book-author-user* and *user-book-genre-user* for Amazon-book dataset; we set the hidden layers as suggested in [14], which is a tower structure with 512, 256, 128, 64 dimensions. For RippleNet, we set the number of hops and the memory size as 2 and 8, respectively. Moreover, early stopping strategy is performed, *i.e.,* premature stopping if recall@20 on the validation set does not increase for 50 successive epochs. To model the third-order connectivity, we set the depth of KGAT L as three with hidden dimension 64, 32, and 16, respectively; we also report the effect of layer depth in Section 4.4.1. For each layer, we conduct the Bi-Interaction aggregator.

4.3 Performance Comparison (RQ1)

We first report the performance of all the methods, and then investigate how the modeling of high-order connectivity alleviate the sparsity issues.

[3]https://grouplens.org/datasets/hetrec-2011/.
[4]https://www.yelp.com/dataset/challenge.