1 Model details

We introduce the key symbols used in this article, and summarize the notations in Table 1.

Table 1: Notations and Descriptions

Notations	Descriptions
$\mathcal{U}$	the set of users
$\mathcal{V}$	the set of items
$\mathcal{G}_b$	the user-item interaction graph
$\mathcal{G}_k$	the knowledge graph
$\mathcal{I}$	the set of entities, including items $\mathcal{V}$ and non-item entities $\mathcal{I}\backslash\mathcal{V}$
$\mathcal{R}$	the set of relations
$\mathcal{N}_i$	the neighbors of user/item/entity i
$\mathcal{E}_b$	the edge set of interaction graph $\mathcal{G}_b$
$\mathcal{E}_k$	the edge set of knowledge graph $\mathcal{G}_k$
$\hat{\mathcal{G}}_b$	the edge subset of interaction view after adaptive data augmentation
$\hat{\mathcal{G}}_k$	the edge subset of knowledge view after adaptive data augmentation
$\mathbb{N}$	the set of negative samples for contrastive learning
GNN_{rec}	the GAT encoder for interaction graph
GNN_{v1}	the GAT encoder for interaction view
GNN_{v2}	the Relation-aware GAT encoder for knowledge view
$\mathbf{h}_u, \mathbf{h}_v$	the representation of user u/item v after GNN_{rec}
$\hat{\mathbf{h}}_u, \hat{\mathbf{h}}_v$	the representation of user u/item v after GNN_{v1}
$\hat{\mathbf{e}}_v$	the representation of item v after GNN_{v2}
$y(u, v)$	the prediction score of user u on item v

To optimize the multi-task objective in Eq.(17), we decouple the training process into three iterative stages: knowledge graph regularization, contrastive learning and recommendation task. We iteratively update the corresponding parameters to minimize the losses until we reach the best performance on the validation set. In particular, we update the parameters of two view generators in the recommendation loss, and then freeze the parameters in the contrastive learning to adaptively filter out unimportant and recommendation-irrelevant edges. All the randomness in data augmentation will be re-sampled in every epoch. For negative samples, we will use all other items in the same batch. We employ Adam optimizer for parameter training. We summarize the running process of KACL in Alg. 1

2 Dataset Description

- **Amazon-Book**: Amazon-Book is a popular dataset for book recommendation, which is a subset of Amazon-review[1] (including product reviews, product meta-

[1] http://jmcauley.ucsd.edu/data/amazon/

Algorithm 1 Algorithm KACL

Input: The user-item interaction graph $\mathcal{G}_b$ and the knowledge graph $\mathcal{G}_k$.

Output: The user representations, item representations and entity representations in KACL.

Initialize model parameters θ with an Xavier initialization;

for $iter$ = 1, ... ,N_{iter} **do**

Sample a mini-batch of positive and negative interactions from $\mathcal{G}_b$;

Sample a mini-batch of entities and relations from $\mathcal{G}_k$;

Sample a mini-batch of positive and negative item pairs from $\mathcal{I}$;

Compute the representation h_u for users and the representation h_v for items based on $\mathcal{G}_b$ and encoder $\mathrm{GNN}_{\mathrm{rec}}$;

Perform graph corruption on original interaction graph $\mathcal{G}_b$ and knowledge graph $\mathcal{G}_k$, and then derive two augmented graphs $\hat{\mathcal{G}}_b$ and $\hat{\mathcal{G}}_k$ based on two view generators, respectively;

Compute the node representation $\hat{\mathbf{h}}_n$ of interaction view based on $\hat{\mathcal{G}}_b$ and encoder $\mathrm{GNN}_{\mathrm{v1}}$;

Compute the node representation $\hat{\mathbf{e}}_n$ of knowledge view based on $\hat{\mathcal{G}}_k$ and encoder $\mathrm{GNN}_{\mathrm{v2}}$ with relation-aware attention according to Equation (11);

Calculate the contrastive learning loss $\mathcal{L}_{CL}$ according to Equation (13);

Calculate the bayesian personalized ranking loss $\mathcal{L}_{CF}$ according to Equation (6) and Equation (14);

Calculate the knowledge graph auxiliary loss $\mathcal{L}_{KG}$ according to Equation (16);

Calculate the joint loss according to Equation (17);

Optimize the parameters according to Adam optimizer on these mini-batches by back-propagation.

end for

return User, item and entity representations for recommendation.

data and behavior links). Amazon-Book contains binary implicit feedback between users and books. If a user interacts with an item (a book), the interaction between them is 1; otherwise, the interaction is 0.

- **LastFM**: LastFM is a recommendation dataset about music, which contains musician listening information extracted from an online music system last.fm. We use the release[2] by KGAT. LastFM also contains binary implicit feedback between users and music. If a user interacts with a piece of music, the interaction between them is 1; otherwise, the interaction is 0.

- **Movielens**: Movielens dataset is collected from Movielens-20M[3], which is a widely used benchmark dataset in movie recommendation and consists of approximately 539 thousand explicit ratings (ranging from 1 to 5) of 48 thousand movies on the MovieLens website. We set the threshold of positive rating as 4.

[2]https://grouplens.org/datasets/hetrec-2011/

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

If a user rates an item (a movie) at 4 or 5, the interaction between them is 1; otherwise, the interaction is 0.

All the above datasets adopt the 10-core setting (*i.e.* filtering out the low-frequency users and items which appear less than ten times) to ensure the quality of interaction data. Also, we need to construct item-side knowledge graph for each dataset. For Amazon-Book and LastFM, we adopt Freebase to construct the KGs. Specifically, we first align items from interaction data to Freebase entities via title matching if there is mapping available. Then we use the aligned items as seeds, and construct the KG subgraph by extracting the triplets that are directly related to the entities aligned with items. Meanwhile, we also extract the triplets that involve both one-hop neighbors and two-hop neighbors of items to enrich the KG's relations and information. For Movielens, we construct KG from Microsoft Satori, which employ a similar alignment for KG subgraph extraction. To ensure the KG quality, we preprocess the three KG by adopting the same procedure as to filter out infrequent entities.

3 Evaluation Metrics

To evaluate the performance of top-K recommendation, we adopt two widely used metrics, including Recall@K and NDCG@K:

- Recall@K (Recall calculated over top-K items). The Recall@K score indicates the percentage of one's rated items that emerge in the top K recommended items.

$$\text{Recall@K} = \frac{1}{|\mathcal{U}_{test}|} \sum_{u \in \mathcal{U}_{test}} \frac{1}{|\mathcal{V}_u|} \sum_{k=1}^{K} \mathbb{I}\left[\omega(k) \in \mathcal{V}_u\right]$$

  where $\mathcal{U}_{\text{test}}$ is user set of test set, $\mathcal{V}_u$ is the ground truth interacted item set of user u in test set, $\omega(k)$ denotes the predicted item at position k, and $\mathbb{I}[\cdot]$ is an indicator function that returns 1 if the condition is true.

- NDCG@K (normalized discounted cumulative gain over top-K items). The NDCG@K score is the metric for measuring the ranking quality, which takes the position of correctly recommended items into account.

$$\text{NDCG@}K = \frac{1}{|\mathcal{U}_{\text{test}}|} \sum_{u \in \mathcal{U}_{\text{test}}} \frac{1}{IDCG_u} \sum_{k=1}^{K} \frac{2^{\mathbb{I}[\omega(k) \in \mathcal{V}_u]} - 1}{\log_2(k+1)}$$

  where $IDCG_u$ is the ideal discounted cumulative gain (DCG), computed by the real item ranking list of user u.

We set K to 20. For each user in the test sets, we treat all the items that this user has not interacted with as negative items. We report the averaged results over all the users in the testing set. Larger values indicate better performance.

Table 2: Comparison of KACL and its variants with different backbones. The improvements of KACL over all variants are significant (0.01 level paired t-test).

Model	Amazon-Book		LastFM		Movielens	
	Recall@20	NDCG@20	Recall@20	NDCG@20	Recall@20	NDCG@20
KACL (GCN)	**0.1631**	**0.0894**	**0.1059**	**0.0935**	**0.4704**	**0.3192**
KACL (GAT)	**0.1625**	**0.0899**	**0.1104**	**0.0973**	**0.4711**	**0.3197**
KACL (RGCN)	**0.1612**	**0.0881**	**0.1062**	**0.0937**	**0.4692**	**0.3184**
KACL	**0.1657***	**0.0915***	**0.1133***	**0.0989***	**0.4752***	**0.3278***

4 Ablation Analysis

Since our proposed KACL is not limited to the architectures of GNNs, we further explore the effects of different GNN backbones for studying the universality of KACL. Specifically, KACL (GCN) is a variant of KACL, which employs GCN to replace all GNN modules. KACL (GAT) is the variant that ignores relation-modeling in knowledge view and utilizes classical GAT as GNN_{v2}. KACL (RGCN) uses GCN for interaction graph and employs RGCN to encode knowledge view.

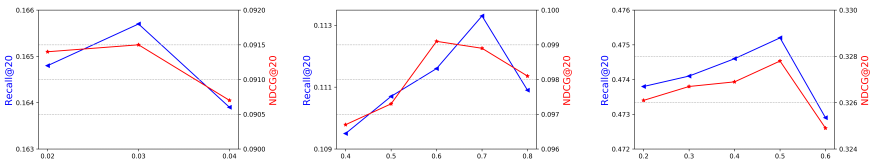
As presented in Table 2, we can observe that KACL-based framework is capable of enhancing the recommendation task with different GNN backbones, and performs better than most baselines. Besides, our proposed KACL consistently achieves the best performance gain among all the variants. The reason is that KACL with relation-aware graph attention can effectively encode KG information into representations. We also observe that the KACL (GCN) and KACL (RGCN) perform worse than KACL (GAT), and the reason may be that KG includes a lot of recommendation-irrelevant information, while GCN-based methods cannot alleviate the impact of irrelevant information as attention-base models.

5 Influence of Hyper-parameters

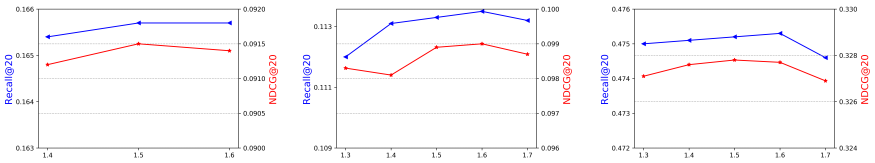
We investigate the influence of hyper-parameters on the recommendation performance and present the experimental results on three datasets.

- **Effect of temperatures τ_k and τ_b** Figure 1 shows the experimental results of different temperatures on three datasets, which are evaluated by both Recall@20 and NDCG@20 scores. For τ_k, we can find that the model has the best performance in 0.03, 0.7, 0.5 on Amazon-Book, LastFM and Movielens, respectively. For τ_b, the change under different values is even less than that of τ_k, the performances on all datasets are stable within a certain range. For convenience, we hold $\tau_b = 1.5$ for our KACL on all datasets.

- **Effect of temperature τ_{cl}** To analyze the influence of temperature τ_{cl}, we vary it from 0.5 to 0.9 in this experiment and illustrate the performance comparisons on three datasets in Figure 2(a)-(c). we report both Recall@20 and NDCG@20 scores to show the effect of τ_{cl}. For Amazon-Book and LastFM, the performance grows when the value increases from 0.5 to 0.7 and then decreases slightly as



(a) Different temperature τ_k on Amazon-Book (b) Different temperature τ_k on LastFM (c) Different temperature τ_k on Movielens



(d) Different temperature τ_b on Amazon-Book (e) Different temperature τ_b on LastFM (f) Different temperature τ_b on Movielens

Figure 1: Performance comparisons of different temperature coefficients of view generator(*i.e.* τ_k and τ_b) on three datasets.

the value increases. For Movielens dataset, the model has stable performance between 0.5 and 0.7. We hold $\tau_{cl} = 0.7$ for all datasets.

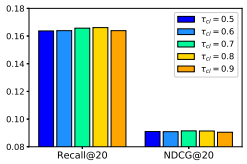
- **Effect of edge dropout rate** ρ To explore the impact of edge dropout rate, we experiment KACL under different rates while keeping other hyper-parameters unchanged. The comparison results are shown in Figure 2(d)-(f). For LastFM, we can observe that the performance of KACL improves when the rate increases from 0.1 to 0.2, and then drops slightly when continues to increase. Compared with LastFM, the performances of Amazon-Book and Movielens remain stable as the rate change.

- **Effect of layer number** L **of both** $\mathrm{GNN}_{\mathrm{v}_1}$ **and** $\mathrm{GNN}_{\mathrm{v}_2}$ Figure 2(d)-(f) show the Recall@20 and NDCG@20 results by stacking GNN layer from 1 to 4. For all datasets, we can observe that the worst performances are got in $L = 1$, due to the poor ability of graph structure extraction. The best performances are achieved when L in $\{2,3\}$ and the results start to decline when we further increase the number of layers because of over-smoothing.

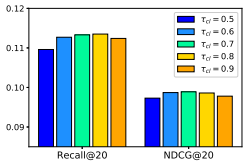
6 Discussion on Amazon-Book dataset

6.1 Effectiveness of alleviating interaction domination.

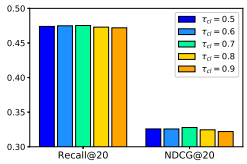
To demonstrate the effectiveness of KACL on alleviating interaction domination, we conduct performance comparisons of complete model and its model variant without KG information (*i.e.*, model w/o KG) for KACL and baselines, and then compute the



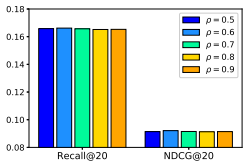
(a) Different temperature τ_{cl} on Amazon-Book



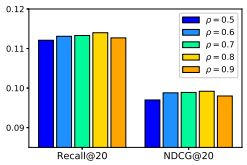
(b) Different temperature τ_{cl} on LastFM



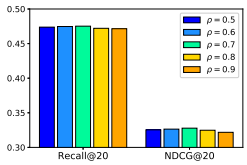
(c) Different temperature τ_{cl} on Movielens



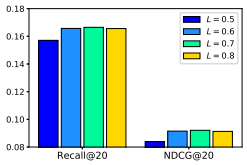
(d) Different dropout rate ρ on Amazon-Book



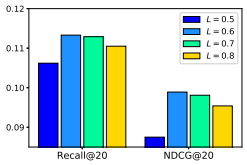
(e) Different dropout rate ρ on LastFM



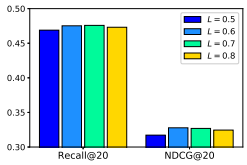
(f) Different dropout rate ρ on Movielens



(g) Different GNN layer size on Amazon-Book



(h) Different GNN layer size on LastFM



(i) Different GNN layer size on Movielens

Figure 2: Performance comparisons of different CL temperature τ_{cl}, dropout rate ρ and layer size on three datasets.

performance improvements of different models by introducing KG information. Higher improvement means that more KG information can be utilized and thus interaction domination problem gets alleviated. As shown in Table 3, we report the CKG-based baselines since other baselines mostly rely on KG for node presentation learning and cannot obtain the variants without KG information. Moreover, other baselines achieve worse results than CKG-based models. The improvement of KACL is consistently better than the baselines, indicating the effectiveness of alleviating interaction domination.

6.2 Inference efficiency analysis

As online services usually require real-time recommendation, inference time efficiency is very important for recommender models. For all KG-based recommender models and our KACL, we compare the inference time taken on Amazon-Book dataset, which is shown in Figure 3. All the experiments are conducted on the same GPU of an NVIDIA Tesla M40. As we can see, KACL has comparable computation complexity to

Table 3: Interaction domination analysis of KACL and baselines on Amazon-Book dataset.

Model	Model w/o KG		Complete model		Improvement %	
	Recall@20	NDCG@20	Recall@20	NDCG@20	Recall@20	NDCG@20
KGAT	0.1518	0.0807	0.1507	0.0802	-0.72%	-0.62%
Simple-HGN	0.1593	0.0858	0.1587	0.0854	-0.38%	-0.47%
DSKReG	0.1543	0.0827	0.1551	0.0863	0.52%	4.35%
KGIN	0.1586	0.0857	0.1631	0.0881	2.84%	2.80%
KACL	0.1577	0.0852	0.1657	0.0915	5.07%	7.39%

embedding-based methods (*i.e.,* CKE), being more efficient than GNN-based methods (*e.g.,* KGAT, KGCN, KGNNLS), which makes KACL more practical in real-world systems.

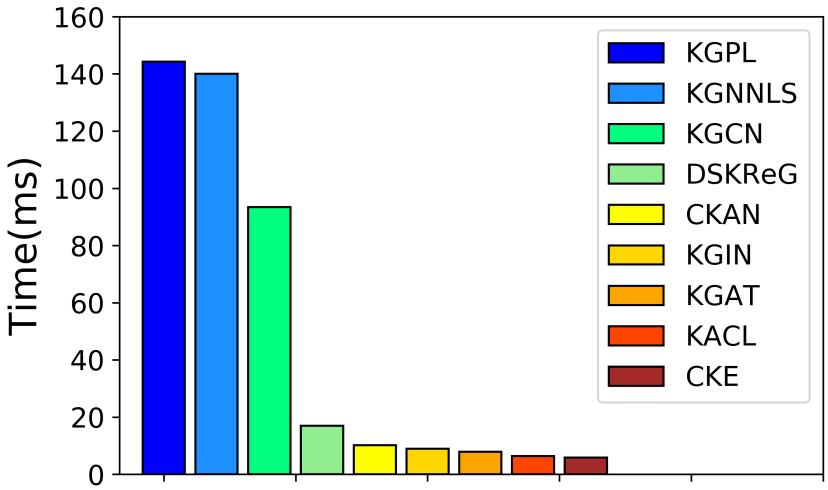


Figure 3: Inference time per user on Amazon-Book dataset.