Knowledge Representation Learning with Contrastive Completion Coding

Bo Ouyang[1], **Wenbing Huang**[1], **Runfa Chen**[1], **Zhixing Tan**[1], **Yang Liu**[1], **Maosong Sun**[1] **and Jihong Zhu**[2*]
[1]Department of Computer Science and Technology,
[2]Department of Precision Instrument,
Tsinghua University, Beijing, P.R.China
{oyb19,crf21}@mails.tsinghua.edu.cn
{zxtan,liuyang2011,sms,jhzhu}@tsinghua.edu.cn
hwenbing@126.com

Abstract

Knowledge representation learning (KRL) has been used in plenty of knowledge-driven tasks. Despite fruitfully progress, existing methods still suffer from the immaturity on tackling potentially-imperfect knowledge graphs and highly-imbalanced positive-negative instances during training, both of which would hinder the performance of KRL. In this paper, we propose Contrastive Completion Coding (C^3), a novel KRL framework that is composed of two functional components: **1.** Hierarchical Architecture, which integrates both low-level standalone features and high-level topology-aware features to yield robust embedding for each entity/relation. **2.** Normalized Contrasitive Training, which conducts normalized one-to-many contrasitive learning to emphasize different negatives with different weights, delivering better convergence compared to conventional training losses. Extensive experiments on several benchmarks verify the efficacy of the two proposed techniques and combing them together generally achieves superior performance against state-of-the-art approaches.

1 Introduction

Knowledge graph (KG), as a well-structured representation of knowledge, plays an important role in a variety of knowledge-driven applications. Upon KG, knowledge representation learning (KRL) (Lin et al., 2018) aims to embed the high-dimension and usually discrete features of entities/relations into a low-dimension vector space. These learned representations, by encoding the underlying semantic relationships among entities/relations, are able to facilitate various downstream tasks, such as question answering (Bordes et al., 2014), recommendation (Wang et al., 2019b) and relation extraction (Bastos et al., 2021) to name some. As a basic research topic, KRL has always attracted many attentions of researchers in relevant domains.

Previous KRL methods generally consider KG completion (*a.k.a* link prediction) as the learning goal. In particular, they define certain score or energy function to accomplish the training by pushing up the score with respect to the observed positive triplets while simultaneously pushing down the score in terms of those negative ones (Ahrabian et al., 2020). To further take the KG connectivity into account, recent works propose to take advantage of graph neural network (GNN) (Vashishth et al., 2019; Ye et al., 2019) to exploit graph topology in KRL (Dettmers et al., 2018). The GNN-based approaches have dominated the state-of-the-art performance in popular benchmarks.

Despite fruitful progress they have achieved, existing methods still suffer from the immature ability on tacking incomplete/noisy KG and imbalanced positive-negative pairs. Regarding the first issue, it is hard to construct perfect KG in practice owing to the expensive annotation effort, let alone that the information in KG is dynamically updating and it is difficult to detect the change at any time. In this situation, using GNN to aggregate information among noisy instances will increase the spread of noise and cause detriment to knowledge representation learning.

In terms of the second issue, it is common that the number of negative instances is much greater than that of positive instances, and the importance of different negative instances differs greatly. Recalling the training loss in previous KRL methods (such as the margin-based (Chechik et al., 2009) and logistic-based (Gutmann and Hyvärinen, 2010) loss), it coequally compares each positive instance with **only one** negative instance at each training iteration. In this way, it not only restrains the interaction between positive-negative instances, but also overlooks the different weights of different negative samples to each positive instance, which, in general, would lead to bias and slow training convergence. Taking the triple *(Kobe Bryant, na-*

[*]Corresponding author.

tionality, United States) for example, we replace the tail entity with others to generate negative triple set, including *(Kobe Bryant, nationality, Italy)* and *(Kobe Bryant, nationality, Michael Jordan)*. In fact, for the second triple, Michael Jordan is not even a nation name and such negative fact should be weighted less compared to others, such as the first triple.

To address the both issues as mentioned above, this paper proposes Contrastive Completion Coding (C^3), a novel framework to allow robust and efficient KRL. C^3 is mainly composed of two functional parts: **1.** Hierarchical Architecture, which is designed to preserve mixed information from both low-level (embedding net) and high-level (GNN) features of each instance. By ensembling different levels of features, we can make full use of topology structure by GNN while effectively suppressing the dispersion of noise over imperfect KG. **2.** Normalized Contrasitive Training, which maximizes the normalized probability of the positive instance over all potential candidates that includes more than one negative sample. In this manner, the importance of different negative triples will be automatically reflected with regard to the positive instance during training. Indeed, this objective is also known as InfoNCE, a kind of mutual information loss that has been applied widely in machine learning and computer vision (van den Oord et al., 2018; Hjelm et al., 2019; Chen et al., 2020).

We summarize our contributions as follows:

- We propose hierarchical KRL to deal with representation learning on imperfect KG. By integrating both low-level (embedding net) and high-level (GNN) features of each instance, C^3 can exploit the topology-aware message passing while suppressing the noisy and invalid propagation by GNN.
- We develop a Normalised One-to-Many Contrastive Objective to train the model on imbalanced positive-negative pairs. To be specific, we adopt InfoNCE, a kind of mutual information loss to attend the different importance of different negative sample, giving rise to more effective learning.
- Extensive experimental evaluations on two link prediction benchmarks, FB15k-237 and WN18RR, reveal that the two proposed techniques are effective and compatible with each other, and the proposed C^3 generally outperforms various state-of-the-art counterparts.

2 Related Work

Our work is closely related to two main branches of study in knowledge representation learning and contrastive loss.

2.1 Knowledge Representation Learning

Knowledge Representation Learning (KRL) is a widely studied field (Xie et al., 2018) with pretext tasks like KG completion. Traditionally, one line of research focuses on designing score or energy functions in margin-based models (Bordes et al., 2013; Wang et al., 2014; Xie et al., 2018; Ahrabian et al., 2020) or un-normalized probability models (Dettmers et al., 2018; Jiang et al., 2019; Balažević et al., 2019b). However, all of the above works adopt margin-based losses or logistic-based losses, which overlook the importance of different negative samples. In contrast, our C^3 uses a new training strategy, InfoNCE for training, which incorporates negative samples with a multiclass classification problem with a Soft-Max and cross-entropy loss.

For KG is a special graph-structured data, some works use a graph neural network (Schlichtkrull et al., 2018; Wang et al., 2019a; Ye et al., 2019; Vashishth et al., 2019) to extract the semantic structure information of KG. In this work, we use an embedding network and a GNN to learn different levels' features of instances in KG and preserve mutual information between context and both them.

2.2 Contrastive Loss

Contrastive losses measure the distance, or similarity, between representations in the latent space, which is one of the key differences between contrastive learning methods and other representation learning approaches (Le-Khac et al., 2020). Motivated from energy-based models (LeCun and Huang, 2005), Chopra et al. (2005) first introduce and then reformulate in (Hadsell et al., 2006) the original margin-based loss and its generalised version (Chechik et al., 2010; Collobert and Weston, 2008; Weinberger and Saul, 2009). Another form of contrastive loss is the logistic-based loss (Gutmann and Hyvärinen, 2010) , which is an estimation method for an un-normalised probabilistic model that avoids the need to evaluate the partition function through a proxy binary classification task. Instead of this form, Józefowicz et al. (2016) extend the un-normalized probability loss to a normalized probability loss. van den Oord et al. (2018) first

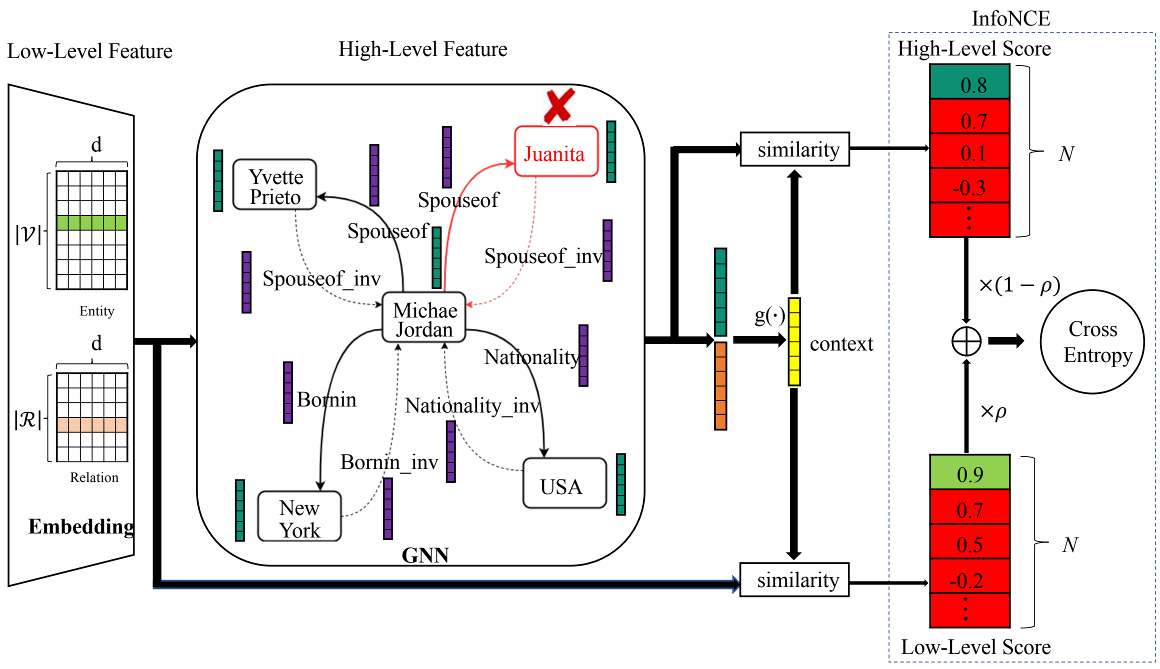

Figure 1: Overview of Contrastive Completion Coding framework. C^3 is mainly composed of two functional parts: **1.** Hierarchical Architecture, which is designed to preserve mixed information from both low-level (embedding net) and high-level (GNN) features of each instance. **2.** Normalized Contrasitive Training, which maximizes the normalized probability of the positive instance over all potential candidates that includes more than one negative sample. $|\mathcal{V}|$: the number of entities, $|\mathcal{R}|$: the number of relations, d: the dimension of representations. Light green and light blue denote low-level features. Dark green and dark blue represent high-level features. The yellow vector is the representation of context. In the score table, green denotes positive score and red represents negative score. We can see a noise triple *(Juanita is Michael Jordan's ex-wife, not the current wife)* in the upper right corner of the KG, which will be learned by the GNN and affect the quality of knowledge representation.

prove that minimising this loss based on NCE is equivalent to maximising a lower bound on the mutual information. Chen et al. (2020) further elaborate on it advantages over other losses. For addressing the imbalance between positive triples and negative triples during KRL training, this normalized one-to-many training objective is also used in our model.

3 Contrastive Completion Coding

In this section, we first present the problem definition in our task, and then follow it up by providing the details of our architecture framework and the training strategy in Figure 1.

3.1 Problem Definition

Knowledge Graph is defined as $\mathcal{G}$ =($\mathcal{V}, \mathcal{R}, \mathcal{T}$), where $\mathcal{V}$, $\mathcal{R}$, $\mathcal{T}$ represent the set of entities, relations and triples, respectively. Each triple (h, r, t) $\in \mathcal{T}$ indicates the relation $r \in \mathcal{R}$ between the head entity $h \in \mathcal{V}$ and the tail entity $t \in \mathcal{V}$. We usually assume that information can flow along both directions of every edge. So for each triple $(h, r, t) \in \mathcal{T}$, its inverse triple (t, r^{-1}, h) is also included in $\mathcal{G}$.

KRL aims to represent entities of KG in a low-dimensional vector space $\{e^v(v) \in \mathbb{R}^n | v \in \mathcal{V}\}$ and relations $\{e^r(r) \in \mathbb{R}^n | r \in \mathcal{R}\}$, where n denotes representation dimension. $e^v(v)$ and $e^r(r)$ represents the embedding of entity and relation, respectively. To do so, KRL usually conducts the KG completion task (*a.k.a* link prediction) as the pretext task. For example, in the case of tail entity inference, common KRL methods contend that the positive embedding should achieve the larger score than all other negative embeddings, *w.r.t.* to context consisting of the head entity and relation. In form, KRL objective:

$$S(e^v(t^+), g(e^v(h), e^r(r))) \gg S(e^v(t^-), g(e^v(h), e^r(r))), \tag{1}$$

where $g(\cdot)$ represents the completion function that returns the representation of context (h, r), t^+ and t^- denote a positive instance and a negative instance, respectively, and $(h, r, t^-) \notin \mathcal{G}$. $S(\cdot)$ denotes the scoring function, which will be discussed in Section 3.2.

3.2 Hierarchical Architecture

As introduced before, embedding each entity and relation with *i.i.d.* function will omit the graph structure that is capable of characterizing high-order interactions. On the contrary, employing GNN alone for embedding learning will be vulnerable to imperfect KG. For the sake of robust embedding, this work combines both the low-level standalone features and high-level topology-aware features. Specifically, we propose a GNN based hierarchical encoding method. Intuitively, not theoretically, different levels of features can be regarded as different views of context-instance. We imply that the low-level representation is to capture the feature view of each node instance, and the high-level representation is to characterize the topology view of the whole KG.

Encoding Function. First, we define the low-level instance feature $\boldsymbol{z}_L$ obtained from the *i.i.d.* embedding network $e_L(\cdot)$ as follows:

$$\boldsymbol{z}_L^v = e_L^v(v); \boldsymbol{z}_L^r = e_L^r(r), \tag{2}$$

where the superscripts v and r denote an entity and a relation, respectively.

Then, the high-level instance feature $\boldsymbol{z}_H$ obtained from the graph-aware encoding function $e_H(\cdot; \mathcal{G})$ is defined below:

$$\boldsymbol{z}_H^v = e_H^v(v; \mathcal{G}); \boldsymbol{z}_H^r = e_H^r(r; \mathcal{G}), \tag{3}$$

where $e_H(\cdot; \mathcal{G})$ is implemented by a specific GNN (Vashishth et al., 2019).

Completion Function. For inferring the missing part of the triple, the completion function is proposed to encode the context representation $\boldsymbol{c}$.

$$\boldsymbol{c} = g(\boldsymbol{z}^v, \boldsymbol{z}^r), \tag{4}$$

where the completion function $g(\cdot)$ can be implemented as any type of Addition, Multiplication, Decomposition, MLP, Convolution, etc. We also define c_L and c_H are the context representation vectors, which are generated by the compeltion function using $(\boldsymbol{z}_L^v, \boldsymbol{z}_L^r)$ and $(\boldsymbol{z}_H^v, \boldsymbol{z}_H^r)$ respectively.

Scoring Function. The scoring function $S(\cdot)$ measures the similarity or distance between two inputs. A trivial form of $S(\cdot)$ is given by a inner/dot product between two vectors $S(\boldsymbol{z}, \boldsymbol{c}) = \boldsymbol{z}^\top \boldsymbol{c}$. This is a most commonly used measurement in literature (Dettmers et al., 2018; Vashishth et al., 2019). Another popular option is utilizing the cosine similarity, $S(\boldsymbol{z}, \boldsymbol{c}) = \frac{\boldsymbol{z}^\top \boldsymbol{c}}{\|\boldsymbol{z}\| \|\boldsymbol{c}\|}$, whose value is bounded between -1 and 1, and equal to 0 for orthogonal vectors. Unless otherwise specified, we adopt the cosine similarity in our method.

To allow hierarchical scoring, we contrast the context vector $\boldsymbol{c}$ with both low-level feature $\boldsymbol{z}_L$ and high-level feature $\boldsymbol{z}_H$ as a weighted combination:

$$\begin{aligned} S(\boldsymbol{z}, \boldsymbol{c}) &= \rho S_L(\boldsymbol{z}_L, \boldsymbol{c}_H) + (1-\rho) S_H(\boldsymbol{z}_H, \boldsymbol{c}_H) \\ &= \rho \frac{\boldsymbol{z}_L^\top \boldsymbol{c}_H}{\|\boldsymbol{z}_L\| \|\boldsymbol{c}_H\|} + (1-\rho) \frac{\boldsymbol{z}_H^\top \boldsymbol{c}_H}{\|\boldsymbol{z}_H\| \|\boldsymbol{c}_H\|}, \end{aligned} \tag{5}$$

where $0 \le \rho \le 1$ is a hyper-parameter that controls trade-off of both levels. We will discuss this hyper-parameter in Section 4.5. The reason why we choose the combination in Eq. 5 to calculate the hierarchical score is mainly for the consideration of calculation efficiency and experimental effect. We will discuss it in detail in Section 4.3. Albeit its simplicity, our experiments support that such simple linear combination is sufficient to provide desired performance.

3.3 Normalized Contrasitive Training

With the scoring function at hand, the last step is how to formulate a training objective to fulfil the ranking in Eq. 1. There exist two typical training losses including the margin-based method (Chechik et al., 2009) and the logistic-based method (Lin et al., 2018).

Specifically, the margin-based objective is given by

$$\mathcal{L}_{margin} = \max\left(0, \gamma + S(\boldsymbol{z}^+, \boldsymbol{c}) - S(\boldsymbol{z}^-, \boldsymbol{c})\right), \tag{6}$$

as well as the gradient w.r.t. $\boldsymbol{c}$:

$$\nabla_{\boldsymbol{c}} \mathcal{L}_{\text{margin}} = \begin{cases} \boldsymbol{z}^+ - \boldsymbol{z}^-, S(\boldsymbol{z}^+, \boldsymbol{c}) - S(\boldsymbol{z}^-, \boldsymbol{c}) < \gamma; \\ 0, \text{otherwise}. \end{cases} \tag{7}$$

As for the logistic-based method, it considers a surrogate binary classification task using the logistic-based loss function. To be specific, it computes

$$\mathcal{L}_{logistic} = -\mathbb{E}_{p^+}[\log \sigma(S(\boldsymbol{z}^+, \boldsymbol{c}))] \\ -\mathbb{E}_{p^-}[\log(1-\sigma(S(\boldsymbol{z}^-, \boldsymbol{c}))], \tag{8}$$

along with the gradient w.r.t. $\boldsymbol{c}$ as follow:

$$\nabla_{\boldsymbol{c}}\mathcal{L}_{\text{logistic}} = \sigma(-\boldsymbol{c}^\top \boldsymbol{z}^+)/\boldsymbol{z}^+ - \sigma(\boldsymbol{c}^\top \boldsymbol{z}^-)/\boldsymbol{z}^-, \tag{9}$$

where $\sigma(\cdot)$ is the Sigmoid function.

Although these two kinds of losses have been applied widely in KRL, they contrast the one-to-one difference between the positive and negative instances, which is unable to handle the imbalance between positive triples and negative triples during training, provided that the number of negatives are usually far greater than that of positives. In addition, by checking their gradients, the update directions by these two objectives are distributively related with each instance without further drawing the different importance of different negative. Inspired by (Ahrabian et al., 2020; Chen et al., 2020), it is essential to mine "hard" negative samples to avoid easy pairs that provide no substantial learning signal in any learning system.

In order to overcome this limitation, we propose to apply a normalized one-to-many training objective (one positive and many negatives at a time). In particular, we sample a candidate set as $\mathcal{Z} = \{\boldsymbol{z}^+, \boldsymbol{z}_1^-, \ldots, \boldsymbol{z}_{N-1}^-\}$ with one positive instance but apply all possible negative samples in the objective function, leading to a total sample number as N. Different entity could have different number of negative samples, hence N varies. We then compute the score between each candidate and the context $\boldsymbol{c}$. By applying a Soft-Max (Bishop, 2006; Goodfellow et al., 2016) on all scores, the training target is to maximize the normalized score of the positive instance, leading to

$$\mathcal{L}_{\text{N}} = -\mathop{\mathbb{E}}_{\mathcal{G}}\left[\log \frac{\exp(S(\boldsymbol{z}^+, \boldsymbol{c}))}{\sum_{\boldsymbol{z}_j \in \mathcal{Z}} \exp(S(\boldsymbol{z}_j, \boldsymbol{c}))}\right], \tag{10}$$

with the gradient given by

$$\nabla_{\boldsymbol{c}}\mathcal{L}_{\text{N}} = (1 - \frac{\exp(S(\boldsymbol{z}^+, \boldsymbol{c}))}{\sum_{\boldsymbol{z}_j \in \mathcal{Z}} \exp(S(\boldsymbol{z}_j, \boldsymbol{c}))})\boldsymbol{z}^+ \\ - \sum_{\boldsymbol{z}^-} \frac{\exp(S(\boldsymbol{z}^-, \boldsymbol{c}))}{\sum_{\boldsymbol{z}_j \in \mathcal{Z}} \exp(S(\boldsymbol{z}_j, \boldsymbol{c}))}\boldsymbol{z}^-. \tag{11}$$

From Eq. 11, we can see that the gradients of the negatives are no longer treated equally and are weighted by the relativity to the sum of the exponentiate scores of all samples. If this term is large, then the corresponding negative sample will greatly influence the gradient and the training process. This property is clearly different from the conventional gradient in Eq. 7 where the weights of all negative samples are the same (i.e. 1). In this way, the training will focus more on the crucial negative sample with large relativity, yielding better convergence. We will compare its effectiveness with other training losses in the experiments Section 4.5.

Note that Eq. 10 is also known as InfoNCE loss that is initially proposed in CPC (van den Oord et al., 2018). It is proved that InfoNCE is actually a lower bound of mutual information, in other words,

$$I(\boldsymbol{z}, \boldsymbol{c}) \geq \log(N) - \mathcal{L}_{\text{N}}. \tag{12}$$

Following *normalised-temperature cross-entropy (NT-Xent)* loss (Chen et al., 2020), we also use a temperature parameter τ to control the sensitivity of the scoring function. Note that the temperature τ determines the attraction-repulsion radius around the context, and thus acts similarly as the margin γ in the margin-based loss.

In summary, the objective of C^3 is derived as

$$\mathcal{L}_{\text{N}} = -\mathop{\mathbb{E}}_{\mathcal{G}}\left[\log \frac{\exp(S(\boldsymbol{z}^+, \boldsymbol{c})/\tau)}{\sum_{x_j \in \mathcal{G}} \exp(S(\boldsymbol{z}_j, \boldsymbol{c})/\tau)}\right]. \tag{13}$$

For better readability, we illustrate the flowchart of our method in Algorithm 1.

4 Experiments

4.1 Setup

Datasets. We evaluate our C^3 models on two standard link prediction datasets: **FB15k-237** (Toutanova and Chen, 2015) that is created from FB15K (Bordes et al., 2013) and **WN18RR** (Dettmers et al., 2018) which is a subset of WN18 (Miller, 1995).

Baselines. We compare our C^3 with the following previous state-of-the-art KRL methods: TransE (Bordes et al., 2013), DistMult (Yang et al., 2014), ComplEx (Trouillon et al., 2016), R-GCN (Schlichtkrull et al., 2018), KBGAN (Cai and Wang, 2018), ConvE (Dettmers et al., 2018), SACN (Shang et al., 2019), HypER (Balažević et al., 2019a), RotatE (Sun

	WN18RR				FB15k-237			
	MRR	H@10	H@3	H@1	MRR	H@10	H@3	H@1
TransE (Bordes et al., 2013)	.226	.501	-	-	.294	.465	-	-
DistMult (Yang et al., 2014)	.43	.49	.44	.39	.241	.419	.263	.155
ComplEx (Trouillon et al., 2016)	.44	.51	.46	.41	.247	.428	.275	.158
R-GCN (Schlichtkrull et al., 2018)	-	-	-	-	.248	.417		.151
KBGAN (Cai and Wang, 2018)	.214	.472	-	-	.278	.458		-
ConvE (Dettmers et al., 2018)	.43	.52	.44	.40	.325	.501	.356	.237
SACN (Shang et al., 2019)	.47	.54	.48	.43	.35	.54	.39	.26
HypER (Balažević et al., 2019a)	.465	.522	.477	.436	.341	.520	.376	.252
RotatE (Sun et al., 2019)	.476	.571	.492	.428	.338	.533	.375	.241
ConvR (Jiang et al., 2019)	.475	.537	.489	.443	.350	.528	.385	.261
VR-GCN (Ye et al., 2019)	-	-	-	-	.248	.432	.272	.159
TuckER (Balažević et al., 2019b)	.470	.526	.482	.443	.358	.544	.394	**.266**
COMPGCN (Vashishth et al., 2019)	.479	.546	.494	.443	.355	.535	.390	.264
SANS (Ahrabian et al., 2020)	.480	.571	-	-	.336	.531	-	-
Our C^3	**.492**	**.572**	**.508**	**.451**	**.360**	**.549**	**.397**	**.266**

Table 1: KG completion performance of our C^3 and several recent models on FB15k-237 and WN18RR datasets. The results of all the baseline methods are taken directly from precious papers ('-' indicates missing values). We find that C^3 outperforms all the existing methods on both WN18RR and FB15k-237 datasets. We achieve state-of-the-art results.

et al., 2019), ConvR (Jiang et al., 2019), VR-GCN (Ye et al., 2019), TuckER (Balažević et al., 2019b), COMPGCN (Vashishth et al., 2019) and SANS (Ahrabian et al., 2020).

Implementation Details. For all experiments, we adopt SGD with momentum as the optimizer to train our models. We used a cosine decay schedule (Loshchilov and Hutter, 2016; Chen et al., 2020) with the initial learning rate set as 1e-4, the momentum as 0.9, and the temperature τ as 0.07 (He et al., 2020). Unless otherwise specified, the trade-off hyper-parameter ρ is set to 0.5. The embedding net $e_L(\cdot)$ that we use is a one-layer learnable embedding network, the GNN $e_H(\cdot)$ is the GCN model used in COMPGCN (Vashishth et al., 2019), and the completion function follows the implementation in ConvE (Dettmers et al., 2018).

Evaluation Metrics. We use the following two measurements as our evaluation metrics: (1) Mean Reciprocal Rank; (2) Hits@10, Hits@3 and Hits@1 that indicate the proportion of correct answers ranked in top 10, 3, 1, respectively.

4.2 Comparisons with state-of-the-arts

Table 1 shows the performance comparisons between our C^3 models and SOTA models on WN18RR and FB15k-237 datasets. The results of all SOTA methods are taken directly from the previous papers (Vashishth et al., 2019; Balažević et al., 2019b; Ahrabian et al., 2020). Clearly, in terms of the most two crucial metrics MRR and H@10, our method improves the baseline COMPGCN (that shares the same backbone with our method but is free of hierarchical embeddings and contrastive training) from 0.479 to 0.492 in MRR and 0.546 to 0.572 in H@10, which validates the effectiveness of our two proposed contributions. Overall, to the best of our knowledge, C^3 outperforms all existing methods on both datasets and achieves superior performance against state-of-the-art approaches.

4.3 Analysis of Hierarchical Structure

Different Context-Instance Training Strategies. We try all context-instance combinations to study all kinds of context-instance relationships in Table 2 [1]. We find that the method using both low-level features e_L and high-level features e_H of instances is better than the variant that using either low-level or high-level features. In particular, the (MRR, H@10) of our results is (0.492, 0.572), while the counterparts with only high-level or low-level features achieve (0.478, 0.565) and (0.458, 0.517), respectively under the same context $\boldsymbol{c}_H$. Interestingly, for the method of using only low-level features e_L, we find context representation $\boldsymbol{c}_L$ performs better than $\boldsymbol{c}_H$. When we use both features of instances, $\boldsymbol{c}_H$ outperforms $\boldsymbol{c}_L$, which implies that deeper $\boldsymbol{c}_H$ has more expressive capacity. The results support the hypothesis that leveraging low-level and high-level features is able to capture different levels of contextual information, thus more

[1]The experimental results on FB15k-237 dataset are similar, which are shown in Table 10.

Q (Context)		K (Instance)		WN18RR	
c_L	c_H	e_L	e_H	MRR	H@10
✓		✓		.470	.538
✓			✓	.468	.538
✓		✓	✓	.484	.557
	✓	✓		.458	.517
	✓		✓	.478	.565
	✓	✓	✓	**.492**	**.572**

Table 2: Results of different context-instance relationships. Experiments settings: representation dimension = 500, batch size = 128.

capable of knowledge representation learning.

Hierarchical Scoring Function. We have chosen the hierarchical scoring function shown in Eq. 5 as discussed preciously. In fact, it is possible to leverage both c_H and c_L under any combination as shown in Table 3. We choose the combination (c_H for z_L, c_H for z_H) in Eq. 5 due to the following two reasons: 1. From the perspective of computational efficiency, the first two combinations will double the FLOPs by computing two-level contexts. 2. As reported by Table 3, it achieves the best result, which explains that the score between the high-level context and the features of both levels is sufficient to capture the multi-view patterns in KG.

Quantitative Statistics on Different Levels Features. We conduct experiments on the test set to verify the importance of low-level and high-level features, respectively. Table 4 shows that the number of $S_H \leq S_L$ is much more than that of $S_H > S_L$ when predicting entities on FB15k-237, while low-level features and high-level features of instances have almost equal effects on the prediction results on WN18RR. It may imply that the degree of incomplete/noise varies greatly in different datasets. The above two experimental results show that we can make full use of topology structure by GNN while effectively suppressing the dispersion of noise over imperfect KG by ensembling different levels of features.

4.4 Hierarchical Structure for Noise Suppression

To validate the assumption that different levels of features help in proportion to the degree of incomplete/noisy information present on different datasets, we introduce 10% and 20% noise to the datasets according to the principle in the

	WN18RR			
Combinations	MRR	H@10	H@3	H@1
(c_L for z_L, c_H for z_H)	.398	.504	.436	.335
(c_H for z_L, c_L for z_H)	.461	.534	.477	.422
(c_L for z_L, c_L for z_H)	.484	.557	.497	.446
(c_H for z_L, c_H for z_H)	**.492**	**.572**	**.508**	**.451**

Table 3: Performance of different combinations on link prediction task evaluated on WN18RR dataset.

	# Predict Tail Entity		# Predict Head Entity	
Datasets	$S_H > S_L$	$S_H \leq S_L$	$S_H > S_L$	$S_H \leq S_L$
WN18RR	1,359	1,775	1,597	1,537
FB15k-237	3,927	16,539	5,034	15,432

Table 4: Quantitative statistics dominated by scores at different levels on the test set. S_L and S_H are the scores computed by using the high-level context vector c_H to calculate the similarity scores with low-level features' e_L and high-level features' instances e_H, respectively. "#" denotes the number of entities.

CKRL (Xie et al., 2018). The results on WN18RR are shown in Table 9 [2]. According to the results, we can see that the last column result is better than the other columns. Hence, we further confirm the method of using both low-level features e_L and high-level features e_H of instances is better than using only the low-level features or high-level features. In particular, with the increase of noise, the gap between our method and other methods increases, which supports the robustness of our method on preventing noise. For example, the improvement regarding MRR between our C3 and the high-level baseline (COMPGCN) is increased from 0.018 to 0.043 when the noise is from 10% to 20%.

4.5 Contrasitive Training

Loss Function. We evaluate the effects of C^3 using different loss functions: margin-based loss, un-normalized logistic-based loss, and normalized probability-based InfoNCE loss as what we have done above. The experimental results are shown in Figure 2. By observing the best MRR recorded on the validation set during the training process, we can find that 1) using margin-based loss converges slowly and has poor performance; 2) using logistic-based loss converges slowly at first, but after a certain period of warming up, it exhibits a faster convergence speed; 3) using InfoNCE, both

[2]The experimental results on FB15k-237 dataset are similar, which are shown in Table 9.

	COMPGCN		C^3(low-level)		C^3(high-level)		C^3(both-level)	
Noise	MRR	H@10	MRR	H@10	MRR	H@10	MRR	H@10
10%	.396	.472	.390	.473	.391	.470	**.414**	**.498**
20%	.309	.400	.319	.389	.323	.403	**.352**	**.433**

Table 5: The performance of our model using different levels of features and COMPGCN on the WN18RR dataset with different scales of noise.

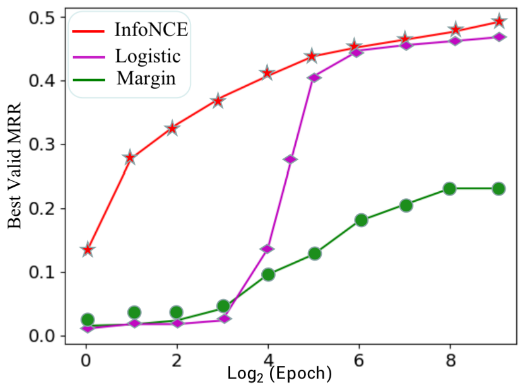

Figure 2: The effect of loss function. Experiments settings: representation dimension = 500, batch size = 128. It shows the best MRR on the validation set.

Datasets	ρ=0.0	ρ=0.2	ρ=0.5	ρ=0.8	ρ=1.0
WN18RR	.478	.490	.492	.472	.458
FB15k-237	.356	.358	.360	.355	.351

Table 6: The MRR in the experiment when the hyper-parameter ρ in Eq. 5 takes different values.

the convergence rate and eventual performance far exceed the other two counterparts, thereby demonstrating the rationality of our choice.

Analysis of Key Hyper-parameters. Table 6 shows the results when the hyper-parameter ρ in Eq. 5 takes different values. This hyper-parameter controls the trade-off between both levels features. We can see that the best value of ρ lies between 0.2 and 0.8 on both datasets. We empirically set ρ to be 0.5, and find it works promisingly.

Figure 3 records the impact of different representation dimensions. It is observed that, as the dimension increases, the performance of C^3 improves gradually and steadily, while the growth rate decreases gradually. On the contrary, the results of other compared methods such as COMPGCN, almost keep unchanged when the dimension varies. It is supported that, our C^3 model benefits more from a larger representation dimension than its KRL counterparts [3], which may imply that the benefit of our method in better mining the representation capacity within the input graphs.

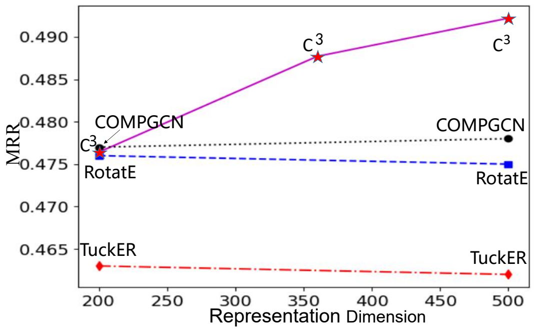

Figure 3: C^3 trained with different dimensions on WN18RR. The green rhombus represents the result of SANS, and its dimension is 1000.

Analysis of the Train Time. For better addressing imbalanced positive-negative pairs to increase the interaction between positive-negative instances and approximating the lower bound of mutual information in Eq. 12, we sample as many negative instances as possible to better normalize the probability of the positive instance over all potential candidates. Nevertheless, sampling all negative instances, as the same procedure applied in both COMPGCN and our method, occupies a very small proportion in total computation. This is because the representations of all entities have already be obtained in memory when calculating InfoNCE, and the main calculations lie in the SoftMax with respect to all negative representations, which counts little compared to the representation computations. For example, the sampling time for COMPGCN and our C3 are close, about 0.16s/iter and 0.21s/iter, respectively.

5 Conclusion

In this paper, we present C^3, a novel knowledge representation learning framework, which is mainly

[3]The results of other methods re-produced on our own according to their papers.

composed of two functional parts: 1) Hierarchical Architecture, which has also exhibited the effectiveness in suppressing the spread of noise. 2) Normalized Contrasitive Training, which can attend the different importance of different negative samples, giving rise to more effective learning. Comparable experimental evaluations reveal that the two proposed techniques are efficient and compatible with each other. The analysis of hierarchical scoring shows that low-level and high-level features are both very necessary for robust KRL, and they complement each other. The contrastive training experiments show that InfoNCE loss is more suitable and efficient for the KG completion task, as our C^3 converges faster and performs better. To the best of our knowledge, we are the first to apply InfoNCE to attend to the different importance of different negative samples in KRL. Our proposed method is simple, yet effective and well-motivated for resolving crucial issues in KRL.

Acknowledgments

This work was jointly sponsored by CAAI-Huawei MindSpore Open Fund and the National Natural Science Foundation of China (Grant No. 62006137). The authors would like to thank anonymous reviewers for their valuable comments.

References

Kian Ahrabian, Aarash Feizi, Yasmin Salehi, William Hamilton, and Avishek Bose. 2020. Structure aware negative sampling in knowledge graphs. pages 6093–6101.

Ivana Balažević, Carl Allen, and Timothy M Hospedales. 2019a. Hypernetwork knowledge graph embeddings. In *International Conference on Artificial Neural Networks*.

Ivana Balažević, Carl Allen, and Timothy M Hospedales. 2019b. Tucker: Tensor factorization for knowledge graph completion. In *Empirical Methods in Natural Language Processing*.

Anson Bastos, Abhishek Nadgeri, Kuldeep Singh, Isaiah Onando Mulang', Saeedeh Shekarpour, Johannes Hoffart, and Manohar Kaul. 2021. Recon: Relation extraction using knowledge graph context in a graph neural network.

Christopher M Bishop. 2006. *Pattern recognition and machine learning*. springer.

Antoine Bordes, Sumit Chopra, and Jason Weston. 2014. Question answering with subgraph embeddings. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 615–620, Doha, Qatar. Association for Computational Linguistics.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 2787–2795. Curran Associates, Inc.

Liwei Cai and William Yang Wang. 2018. KBGAN: Adversarial learning for knowledge graph embeddings. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1470–1480.

Gal Chechik, Varun Sharma, Uri Shalit, and Samy Bengio. 2009. Large scale online learning of image similarity through ranking. In *Iberian Conference on Pattern Recognition and Image Analysis*, pages 11–14. Springer.

Gal Chechik, Varun Sharma, Uri Shalit, and Samy Bengio. 2010. Large scale online learning of image similarity through ranking.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR.

Sumit Chopra, Raia Hadsell, and Yann LeCun. 2005. Learningasimilaritymetricdiscriminatively. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 539–546. IEEE.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167.

Tim Dettmers, Minervini Pasquale, Stenetorp Pontus, and Sebastian Riedel. 2018. Convolutional 2d knowledge graph embeddings. In *Proceedings of the 32th AAAI Conference on Artificial Intelligence*, pages 1811–1818.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`.

Michael Gutmann and Aapo Hyvärinen. 2010. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 297–304.

Raia Hadsell, Sumit Chopra, and Yann LeCun. 2006. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1735–1742. IEEE.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. 2020. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. 2019. Learning deep representations by mutual information estimation and maximization.

Xiaotian Jiang, Quan Wang, and Bin Wang. 2019. Adaptive convolution for multi-relational learning. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Rafal Józefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. *CoRR*, abs/1602.02410.

Phuc H Le-Khac, Graham Healy, and Alan F Smeaton. 2020. Contrastive representation learning: A framework and review. *IEEE Access*.

Yann LeCun and Fu Jie Huang. 2005. Loss functions for discriminative training of energy-based models. In *AIStats*, volume 6, page 34. Citeseer.

Yankai Lin, Xu Han, Ruobing Xie, Zhiyuan Liu, and Maosong Sun. 2018. Knowledge representation learning: A quantitative review. *CoRR*, abs/1812.10901.

Ilya Loshchilov and Frank Hutter. 2016. SGDR: stochastic gradient descent with restarts. *CoRR*, abs/1608.03983.

George A. Miller. 1995. Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39–41.

Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *The Semantic Web*, pages 593–607, Cham. Springer International Publishing.

Chao Shang, Yun Tang, Jing Huang, Jinbo Bi, Xiaodong He, and Bowen Zhou. 2019. End-to-end structure-aware convolutional networks for knowledge base completion. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3060–3067.

Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. *Advances in neural information processing systems*, 26:926–934.

Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. Rotate: Knowledge graph embedding by relational rotation in complex space. In *International Conference on Learning Representations*.

Kristina Toutanova and Danqi Chen. 2015. Observed versus latent features for knowledge base and text inference. In *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*, pages 57–66.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pages 2071–2080. JMLR.org.

Aäron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *CoRR*, abs/1807.03748.

Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, and Partha P. Talukdar. 2019. Composition-based multi-relational graph convolutional networks. *CoRR*, abs/1911.03082.

Rui Wang, Bicheng Li, Shengwei Hu, Wenqian Du, and Min Zhang. 2019a. Knowledge graph embedding via graph attenuated attention networks. *IEEE Access*, 8:5212–5224.

Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. 2019b. Kgat. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery Data Mining*.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, AAAI'14, pages 1112–1119. AAAI Press.

Kilian Q Weinberger and Lawrence K Saul. 2009. Distance metric learning for large margin nearest neighbor classification. *Journal of machine learning research*, 10(2).

Ruobing Xie, Zhiyuan Liu, Fen Lin, and Leyu Lin. 2018. Does william shakespeare really write hamlet? knowledge representation learning with confidence. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and li Deng. 2014. Embedding entities and relations for learning and inference in knowledge bases.

Rui Ye, Xin Li, Yujie Fang, Hongyu Zang, and Mingzhong Wang. 2019. A vectorized relational graph convolutional network for multi-relational network alignment. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 4135–4141. International Joint Conferences on Artificial Intelligence Organization.

A Appendix

A.1 Relation to Previous KRL Models

As is shown in Table 13, several previous methods, TransE (Bordes et al., 2013), DistMult (Yang et al., 2014), ConvE (Dettmers et al., 2018), TuckER (Balažević et al., 2019b) and COMPGCN (Vashishth et al., 2019), can be viewed as a special case in C^3 framework.

A.2 More details about Experiment

A.2.1 Datasets

In this section, we provide the details of the datasets used in the experiments. We use the following two datasets:

FB15k-237 (Toutanova and Chen, 2015) is a pruned version of FB15k (Bordes et al., 2013) dataset with inverse relations removed to prevent direct inference.

WN18RR (Dettmers et al., 2018) is a subset from WN18 (Bordes et al., 2013) dataset which is derived from WordNet (Miller, 1995).

Details of train/validation/test splits is listed in Table 7. The datasets can be download on https://github.com/thunlp/OpenKE. or https://github.com/malllabiisc/CompGCN.

Datasets	#Ent	#Rel	#Train	#Test	#Valid
FB15k-237	14,541	237	272,115	20,466	17,535
WN18RR	40,943	11	86,835	3,134	3,034

Table 7: Statistics of FB15k-237 and WN18RR datasets.

A.2.2 Evaluation Metrics

In this paper, we conduct our experiments on the KG completion task. It concentrates on the quality of knowledge representations (Socher et al., 2013), which aims to complete a triple when head entity or tail entity is missing.

We conduct two measures as our evaluation metrics: (1)Mean Reciprocal Rank, that is a relative score that calculates the average of the inverse of the ranks at which the first relevant entity was retrieved for a set of queries. and (2)Hits@10, Hits@3 and Hits@1 indicate the proportion of correct answers ranked in top 10, 3, 1 respectively.

For COMPGCN which is closely related to our method, we have conducted the comparison in a fair and comprehensive setting to justify the significance of our proposed idea. For other methods (such as SANS) we have tried to reproduce the results for all metrics but fail to obtain the comparable numbers as reported. Hence, a conservative solution is to directly copy the numbers from their papers.

A.2.3 Hyper-parameters

For selecting the best model, we perform a hyperparameter search using the validation data over the values listed in Table 8 through selecting the highest MRR. In our best setting, we use learnable convolution networks ConvE as our completion function $g(\cdot)$. The best learning rate lr = 0.09, the batch size is 128, the representation dimension is 500, the dropout is 0.1 and the composition operators is multiplication for two-layers f_{gnn} for FB15k-237 (600epoch) and circular-correlation in one-layer f_{gnn} for WN18RR (800epoch). Our C^3 model build on PyTorch geometric framework(Compatible with Python 3.x). Total number of parameters of C^3 model is 64.613M, and total number of FLOPs is 9.154G.

Hyperparameters	Values
Number of GNN Layers	{1, 2}
Number of epoch	{200, 400,600, 800}
Number of dim (d)	{100, 200, 500, 1000}
Learning rate	{0.001, 0.015, 0.03, 0.09, 0.1, 0.2}
Batch size	{32, 64, 128, 256, 512, 1024}
Dropout	{0.0, 0.1, 0.2}
temperature τ	{0.01, 0.05, 0.07, 0.1, 0.2}

Table 8: Details of hyperparameters.

A.2.4 Additional results

	COMPGCN		C^3(low-level)		C^3(high-level)		C^3(both-level)	
Noise Ratio	MRR	H@10	MRR	H@10	MRR	H@10	MRR	H@10
10%	.319	.489	.314	.490	.333	.521	**.340**	**.526**
20%	.308	.474	.308	.480	.320	.503	**.329**	**.511**

Table 9: The performance of our model using different levels of features and COMPGCN on the FB15K-237 dataset with different scales of noise.

Q (Context)		**K (Instance)**		**FB15k-237**	
c_L	c_H	e_L	e_H	MRR	H@10
✓		✓		.348	.527
✓			✓	.329	.511
✓		✓	✓	.347	.530
	✓	✓		.351	.536
	✓		✓	.356	.543
	✓	✓	✓	**.360**	**.549**

Table 10: Results of different context-instance relationships. Experiments settings: representation dimension = 500, batch size =128.

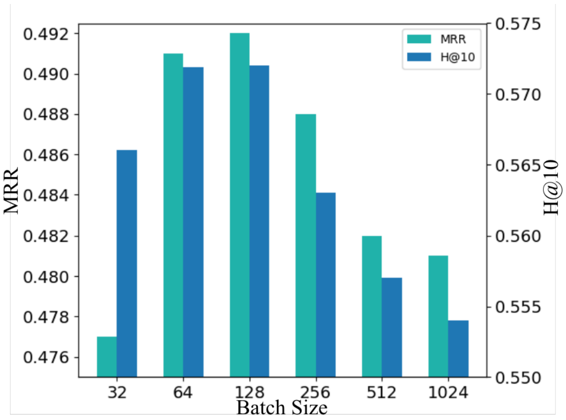

Figure 4: C^3 trained with different batch sizes on WN18RR. Experiments settings: representation dimension = 500, epoch = 800.

Completion Function	**Loss Function**	**WN18RR** MRR	H@10
Addition	InfoNCE	.268	.510
Multiplication	InfoNCE	.444	.517
Convolution	**InfoNCE**	**.492**	**.572**
Convolution	Margin	.231	.416
Convolution	Logistic	.468	.522

Table 11: Effects of completion function and loss function. Experiments settings: representation dimension = 500, batch size =128. Results in the first three rows show that convolution completion function gives a substantial improvement than others. And the last three rows of results show the performance of InfoNCE loss function far exceeds others.

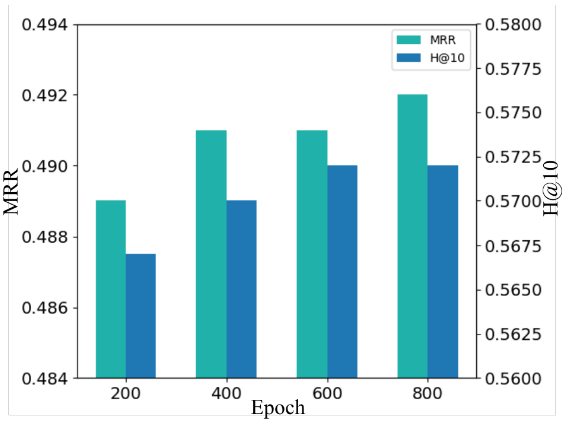

Figure 5: C^3 trained with different epochs on WN18RR. Experiments settings: representation dimension = 500, batch size = 128.

Completion Function	**Loss Function**	**FB15K-237** MRR	H@10
Addition	InfoNCE	.337	.524
Multiplication	InfoNCE	.346	.533
Convolution	**InfoNCE**	**.360**	**.549**
Convolution	**Margin**	.182	.318
Convolution	**Logistic**	.322	.499

Table 12: Effects of completion function and loss function. Experiments settings: representation dimension = 500, batch size =128. Results in the first three rows show that convolution completion function gives a substantial improvement than others. And the last three rows of results show the performance of InfoNCE loss function far exceeds others.

Method	Instances	Encoding Functions	Completion Functions	Scoring Functions	Training Losses
TransE	low-level	Embedding	Addition	Distance	Margin-based
DistMult	low-level	Embedding	Multiplication	Similarity	Margin-based
ConvE	low-level	Embedding	Convolution	Similarity	Logistic-based
TuckER	low-level	Embedding	Decomposition	Similarity	Logistic-based
COMPGCN	high-level	Embedding + GNN	$*$	Similarity	Logistic-based
our C^3	low&high-level	Embedding + GNN	$*$	Cosine Similarity	InfoNCE

Table 13: Relation to previous KRL models. Other methods can be viewed as a case in our C^3 framework. $*$ indicates any completion function.

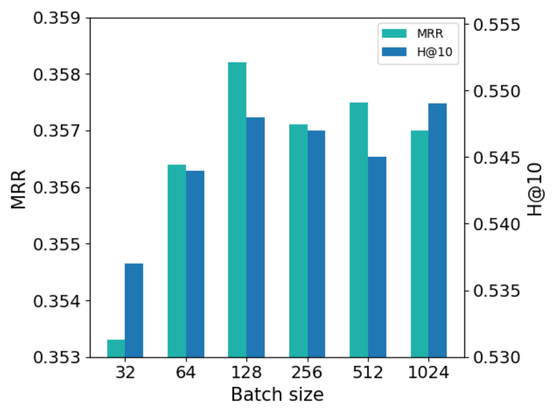

Figure 6: C^3 trained with different batch size on FB15k-237 dataset. Experiments settings: representation dimension = 500, epoch = 200.

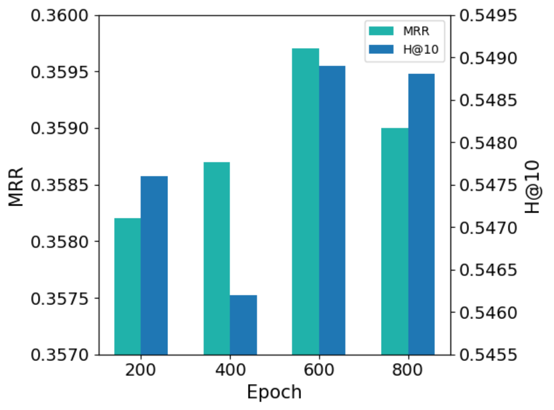

Figure 7: C^3 trained with different epochs on FB15k-237 dataset. Experiments settings: representation dimension = 500, batch size = 128.

Algorithm 1 Implmenting C^3

```
**Input:** The triple set $\mathcal{T}$, the entity set $\mathcal{V}$, the relation set $\mathcal{R}$. Initialize the parameters $\theta$ of the Hierarchical Architecture. Batch size is $m$. The number of instances is $N$. We still use the case of inferring the tail entity as an example.
  **while** $\theta$ has not converged **do**
    Randomly sample $\{h_i, r_i\}_{i=1}^m \sim p(h, r)$;
    Get high-level features:
    $\{\boldsymbol{z}_H^{h_i} \leftarrow e_H^v(h_i); \boldsymbol{z}_H^{r_i} \leftarrow e_H^r(r_i)\}_{i=1}^m$;
    Get high-level context representation:
    $\{\boldsymbol{c}_{Hi} \leftarrow g(\boldsymbol{z}_H^{h_i}, \boldsymbol{z}_H^{r_i}\}_{i=1}^m$;
    Sample one positive instance:
    $\{t_{i+}^0 \sim p^+(\cdot|h_i, r_i), (h_i, r_i, t_i^+) \in \mathcal{T}\}_{i=1}^m$;
    Sample $N-1$ negative instances:
    $\{t_{i-}^k \sim p^-(\cdot|h_i, r_i), (h_i, r_i, t_i^-) \notin \mathcal{T}, k = \{1, 2, \cdots, N-1\}\}_{i=1}^m$;
    Get the instance set:
    $\{\mathcal{X}_i = \{t_{i+}^0, t_{i-}^1, ..., t_{i-}^{N-1}\}\}_{i=1}^m$;
    Get different levels of features:
    $\{\boldsymbol{z}_L^{t_i^j} \leftarrow e_L^v(t_i^j); \boldsymbol{z}_H^{t_i^j} \leftarrow e_H^v(t_i^j), t_i^j \in \mathcal{X}_i\}_{i=1}^m$;
    Calculate the low-level score:
    $\{S_L(\boldsymbol{z}_L^{t_i^j}, \boldsymbol{c}_{Hi}) \leftarrow \frac{\boldsymbol{z}_L^{t_i^j \top} \boldsymbol{c}_{Hi}}{\left\|\boldsymbol{z}_L^{t_i^j}\right\| \|\boldsymbol{c}_{Hi}\|}, t_i^j \in \mathcal{X}_i\}_{i=1}^m$;
    Calculate the high-level score:
    $\{S_H(\boldsymbol{z}_H^{t_i^j}, \boldsymbol{c}_{Hi}) \leftarrow \frac{\boldsymbol{z}_H^{t_i^j \top} \boldsymbol{c}_{Hi}}{\left\|\boldsymbol{z}_H^{t_i^j}\right\| \|\boldsymbol{c}_{Hi}\|}, t_i^j \in \mathcal{X}_i\}_{i=1}^m$;
    Calculate the total score:
    $\{S(\boldsymbol{z}^{t_i^j}, \boldsymbol{c}_i) \leftarrow \rho S_L(\boldsymbol{z}_L^{t_i^j}, \boldsymbol{c}_{Hi}) + (1 - \rho) S_H(\boldsymbol{z}_H^{t_i^j}, \boldsymbol{c}_{Hi}), t_i^j \in \mathcal{X}_i\}_{i=1}^m$;
    Calculate full objective:
    $\mathcal{L}_\mathrm{N} \leftarrow -\frac{1}{m} \sum_{i=1}^{m} \left[ \log \frac{\exp(S(\boldsymbol{z}^{t_{i+}^0}, \boldsymbol{c}_i)/\tau)}{\sum_{t_i^j \in \mathcal{X}_i} \exp(S(\boldsymbol{z}^{t_i^j}, \boldsymbol{c}_i)/\tau)} \right]$;
    Update $\theta \leftarrow \nabla_\theta \mathcal{L}_\mathrm{N}$
  **end while**
**Output:** low-level feature $\boldsymbol{z}_L$, high-level feature $\boldsymbol{z}_H$ and context representation $\boldsymbol{c}$.
```