Contrastive Learning with Generated Representations for Inductive Knowledge Graph Embedding

Qian Li[1,2], **Shafiq Joty**[2,3], **Daling Wang**[1]*, **Shi Feng**[1], **Yifei Zhang**[1] and **Chengwei Qin**[2]
[1] Northeastern University, China
[2] Nanyang Technological University, Singapore
[3]Salesforce Research
feiwangyuzhou@foxmail.com, srjoty@ntu.edu.sg
{wangdaling,fengshi,zhangyifei}@cse.neu.edu.cn
chengwei003@e.ntu.edu.sg

Abstract

With the evolution of Knowledge Graphs (KGs), new entities emerge which are not seen before. Representation learning of KGs in such an *inductive* setting aims to capture and transfer the structural patterns from existing entities to new entities. However, the performance of existing methods in inductive KGs are limited by sparsity and implicit transfer. In this paper, we propose **VMCL**, a **C**ontrastive **L**earning (CL) framework with graph guided **V**ariational autoencoder on **M**eta-KGs in the inductive setting. We first propose representation generation to capture the encoded and generated representations of entities, where the generated variations can augment the representation space with complementary features. Then, we design two CL objectives that work across entities and meta-KGs to simulate the transfer mode. With extensive experiments we demonstrate that our proposed VMCL can significantly outperform previous state-of-the-art baselines.

1 Introduction

Knowledge Graphs (KGs) structure objective knowledge in the form of ("head entity", "relation", "tail entity") triples to express factual connections. Representation learning of KGs aims to learn implicit representations (embeddings) of entities and then applies the learned representations to knowledge-intensive tasks such as link prediction (Li et al., 2021) and question answering (Hao et al., 2017). Conventionally, KG are embedded in a *transductive* setting with a fixed predefined set of entities under the assumption that entities to be tested are seen during training. In this *transductive* setting, the learned entity representations from a source KG can easily be applied to a target KG (Sun et al., 2019).

However, it is well admitted that KGs are not static, rather they evolve over time where new KGs with a novel set of entities emerge. For KGs in such an *inductive* setting, the learned representations of existing (source) entities are not applicable to the new (target) entities (Chen et al., 2022). Thus, the problem of capturing structural patterns from existing source entities and transferring them to a new set of target entities is of practical importance and posits a new set of interesting research challenges to tackle. We provide empirical insights about the representation learning problem in the inductive setting with an example in Fig. 1. Entities in the target KG are new which are different from entities in the source KG. While there are similarities in structural patterns between the two KGs, e.g., entity 1 in the target KG may be a company like Facebook in the source KG. Representation learning of inductive KGs aims to capture the structural patterns from the source KG, then transfer them to help the learning for the target KG. Note that what are transferred are structural patterns, not the learned representations because entities in the target and source KGs are different.

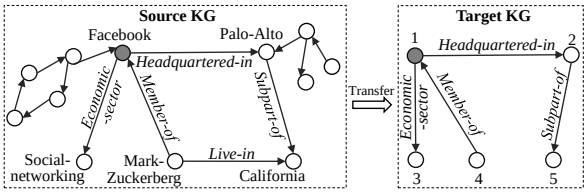


Figure 1: Example of inductive KGs with a source KG and a target KG. Entities in the target KG are different from entities in the source KG. However, there are similarities in relational patterns, e.g., entity Facebook in the source KG has relations: *Headquartered-in*, *Economic-sector*, *Member-of*, and entity 1 in the target KG has similar relation patterns. So it can be deduced that entity 1 may be a company like Facebook.

Several efforts devoted to representation learning of inductive KGs. AMIE (Galárraga et al., 2013), RuleN (Meilicke et al., 2018), Neural-LP (Yang et al., 2017) and DRUM (Sadeghian et al., 2019) learn probabilistic logical and entity-independent

*Corresponding author

Findings of the Association for Computational Linguistics: ACL 2023, pages 14273–14287

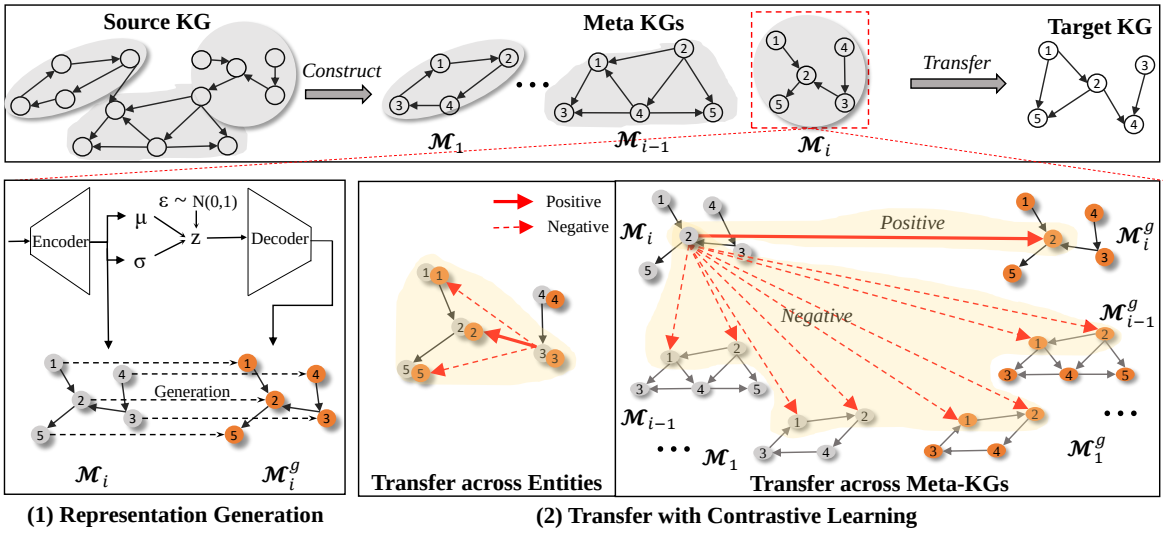


Figure 2: The overall framework of VMCL. (1) For a given meta-KG $\mathcal{M}_i$, **Representation Generation** uses a graph guided variational autoencoder to generate representation variations (denoted by $\mathcal{M}_i^g$) of the entities. (2) **Transfer with Contrastive Learning** simulate explicitly the transfer mode with CL across entities and meta-KGs.

rules from the source KG and apply such rules to the target KG. GraIL (Teru et al., 2020) and CoMPILE (Mai et al., 2021) learn the ability of relation prediction by subgraph extraction and graph neural networks independent of any specific entities, and generalize such ability to the target KG. Recently, MorsE (Chen et al., 2022) learns transferable and entity-independent meta knowledge by meta learning (Qin et al., 2023). Although these methods have shown promising results for representation learning of inductive KGs, their performance is affected by at least two factors. First, their capability is limited by the *sparsity* of KGs caused by missing links. In four inductive KGs: N1-N4 (Teru et al., 2020), the degree of 65%, 47%, 46%, 42% entities is less than 3. Second, effective transfer is often limited by the *implicitness*, e.g., MorsE uses meta learning to learn transfer patterns implicitly between source and target KGs, lacking explicit joint comparison across the entities and KGs.

In this work, we propose **VMCL**, a **C**ontrastive **L**earning (CL) framework with graph guided **V**ariational autoencoder on **M**eta-KGs [1] in the inductive setting. Fig. 2 gives the framework. We first propose representation generation to capture the encoded and generated representations of entities, where the generated variations can augment the representation space with complementary features (Fig. 2(1)). Then, we design two CL objectives that work across entities and meta-KGs to simulate the transfer mode (Fig. 2(2)). We perform extensive evaluation and theoretical derivation to understand the performance of VMCL. Empirical results and analysis show the superiority of VMCL over state-of-the-art baselines.

In summary, our key contributions are:

- We propose a graph guided variational autoencoder to augment entity representations, which provides complementary features to alleviate the sparsity and lay the foundation for transfer.
- We simulate explicitly the transfer mode with CL across entities and meta-KGs.
- Extensive experiments show the superiority of VMCL over state-of-the-art baselines. We release the code and datasets in `https://github.com/feiwangyuzhou/VMCL`.

2 Related Work

2.1 Knowledge Graphs

Knowledge Graphs (KGs) structure objective facts to express potential connections between entities (Li et al., 2022b). In general, KGs are considered in a transductive setting where they remain static. Representation learning in this transductive setting aims to learn informative representations of entities through a source KG and apply the learned representations to a target KG, because the entities in the target KG also appear in the source KG. TransE (Bordes et al., 2013) and RotatE (Sun

[1] Each meta-KG is a subgraph sampled randomly from the source KG and re-labeled to imitate a new KG.

et al., 2019) design complex similarity scoring functions to mine semantic features for entities and relations. GGAE (Li et al., 2021) introduces attention networks (Huang et al., 2021) while R-GCN (Schlichtkrull et al., 2018) adapts GNNs to encode the features of entities and relations with k-hop neighbor structures. Although many methods have been proved effective in handling various transductive KGs, they cannot handle tasks related to entities unseen during training.

As mentioned, KGs evolve over time and new KGs with new entities emerge. This produces inductive KGs where the target KG is inducted by the source KG with predefined relations. In the inductive setting, entities in the target KG could be different from the entities in the source KG. Representation learning of inductive KGs aims to capture structural patterns from the source KG, then transfer them to the target KG. Representation learning of inductive KGs is more difficult than that of transductive KGs, but it is more realistic and practical. Much work devotes to the representation learning of inductive KGs with probabilistic logical rules (Galárraga et al., 2013; Meilicke et al., 2018; Yang et al., 2017; Sadeghian et al., 2019), relation prediction ability (Teru et al., 2020; Mai et al., 2021) and meta-learning framework (Chen et al., 2022).

However, their feature mining and transfer capability is limited by sparsity and implicit transfer. In contrast, we propose representation generation to augment the representation space, and design two CL objectives that work across entities and meta-KGs to simulate the transfer mode.

2.2 Contrastive Learning

CL aims to learn effective representation by pulling semantically close neighbors together and pushing apart non-neighbors, which has achieved great success in vision (He et al., 2020), text (Gao et al., 2021) and graph (Zhu et al., 2021). CL in graphs improves the performance by leveraging a contrastive loss at node (Velickovic et al., 2019) and graph (Sun et al., 2020) levels. (Hassani and Ahmadi, 2020) and (Zhu et al., 2021) attempt to contrasts multiple structural views of graphs. (Ahrabian et al., 2020), (Wang et al., 2022) and (Li et al., 2022a) introduce CL into transductive KGs by designing negative sampling strategies. Although there has been a lot of CL concerned with graphs and transductive KGs, relatively little work focuses on inductive KGs. In contrast, we propose a simple but effective CL framework for inductive KGs to transfer knowledge from source KGs to target KGs.

2.3 Variational AutoEncoder (VAE)

Deep generative models have recently attracted much attention in that they can generate unseen samples which have the same distribution as the original data (Kipf and Welling, 2016; Simonovsky and Komodakis, 2018). The generated data, as a supplement to original data, can help the model better mine potential features and make the model more robust. VAE (Kingma and Welling, 2014) is an unsupervised generative framework and has been extensively studied and applied in various tasks such as question answering (Zhang et al., 2018) and graph autoencoder (Ahn and Kim, 2021; Li et al., 2023). In this paper, we introduce a graph guided VAE to generate similar representations of entities, which broadens the representations with complementary features.

3 Preliminaries

3.1 Inductive KGs

We denote a KG as a graph $\mathcal{G}$=$(\mathcal{E}, \mathcal{R})$, where $\mathcal{E}$ refers to the set of entities and $\mathcal{R}$ refers to the set of relations in the KG. A KG organizes entities and relations as triples of (h, r, t), where $h, t \in \mathcal{E}$ respectively represent the head and tail entities, and $r \in \mathcal{R}$ represents the relation between h and t. A group of inductive KGs includes at least a source KG $\mathcal{G}_S$=$(\mathcal{E}_S, \mathcal{R}_S)$ and a target KG $\mathcal{G}_T$=$(\mathcal{E}_T, \mathcal{R}_T)$ with the condition $\mathcal{R}_T \subseteq \mathcal{R}_S$ and $\mathcal{E}_T \cap \mathcal{E}_S$=$\emptyset$. The goal of representation learning for inductive KGs is to capture the structural patterns f from $\mathcal{G}_S$ and transfer it to $\mathcal{G}_T$. We expect f to be able to map entities in $\mathcal{G}_T$ into representations with the following property: the score of a true triple $(h, r, t) \in \mathcal{G}_T$ should be higher than that of any false triple (h^-, r^-, t^-).

In general, target KGs (1) may retain some old entities, e.g., character relationships, or (2) may not retain any, e.g., new events (however, there are similarities in the propagation modes of different events). To our knowledge, case (2) is more difficult than case (1). Our definition is designed for case (2), and can be easily applied to case (1) by initializing old entities with old embeddings.

3.2 Meta-KGs

Following MorsE (Chen et al., 2022), we use meta-KGs to simulate the transfer learning mode, where each meta-KG $\mathcal{M}_i$ is a subgraph sampled randomly

from the source KG $\mathcal{G}_S$. We re-label the id of all entities in $\mathcal{M}_i$ in order to make the process label insensitive as our main interest is to capture the structural patterns. We extract multiple meta-KGs $\{\mathcal{M}_i\}_{i=0}^{|\mathcal{M}|}$ from $\mathcal{G}_S$ with $|\mathcal{M}|$ being the number of meta-KGs sampled. For any two meta-KGs $\mathcal{M}_i$=$(\mathcal{E}_{\mathcal{M}_i}, \mathcal{R}_{\mathcal{M}_i})$ and $\mathcal{M}_j$=$(\mathcal{E}_{\mathcal{M}_j}, \mathcal{R}_{\mathcal{M}_j})$, we ensure that $\mathcal{E}_{\mathcal{M}_i} \cap \mathcal{E}_{\mathcal{M}_j}$=$\emptyset$, $\mathcal{R}_{\mathcal{M}_i} \subseteq \mathcal{R}_S$, $\mathcal{R}_{\mathcal{M}_j} \subseteq \mathcal{R}_S$.

3.3 Representation Notations

In the remaining, we use the following notations:

- $\mathbf{R} \in \mathbb{R}^{|\mathcal{R}_S| \times D}$ denotes learnable relation representations which reserve internal semantics of relations; $\mathbf{r} \in \mathbf{R}$ is the representation for the relation r. D is the dimension of representations.
- $\mathbf{R}_{head} \in \mathbb{R}^{|\mathcal{R}_S| \times D}$ and $\mathbf{R}_{tail} \in \mathbb{R}^{|\mathcal{R}_S| \times D}$ are learnable relation embeddings to initialize the representation of the respective entities. For a relation r, $\mathbf{r}_{head} \in \mathbf{R}_{head}$ denotes the relation embedding for the head entities connected by r, and $\mathbf{r}_{tail} \in \mathbf{R}_{tail}$ denotes the relation embedding for the tail entities connected by r.
- For a meta-KG $\mathcal{M}_i$=$(\mathcal{E}_{\mathcal{M}_i}, \mathcal{R}_{\mathcal{M}_i})$, we follow (Chen et al., 2022) and initialize the representation of each entity with its relation patterns. Formally, the representation of entity h in a meta-KG $\mathcal{M}_i$ is initialized with features of the involved relations in $\mathcal{M}_i$:

$$\mathbf{h^0} = \frac{\Sigma_{r \in \mathcal{O}(h)} \mathbf{r}_{tail} + \Sigma_{r \in \mathcal{I}(h)} \mathbf{r}_{head}}{|\mathcal{O}(h)| + |\mathcal{I}(h)|} \quad (1)$$

  where we use $\mathcal{O}(h)$=$\{r|\exists x, (h, r, x) \in \mathcal{M}_i\}$, $\mathcal{I}(h)$ = $\{r|\exists x, (x, r, h) \in \mathcal{M}_i\}$ to denote outgoing and incoming relations for entity h.
- For a meta-KG $\mathcal{M}_i$=$(\mathcal{E}_{\mathcal{M}_i}, \mathcal{R}_{\mathcal{M}_i})$, we use $\mathbf{h} \in \mathbb{R}^{|\mathcal{E}_{\mathcal{M}_i}| \times D}$ and $\mathbf{h^g} \in \mathbb{R}^{|\mathcal{E}_{\mathcal{M}_i}| \times D}$ to denote the encoded representation and generated representation of entity h, respectively.

4 Method

Fig. 2 depicts the overall framework of our VMCL. In the following, we first introduce *representation generation* to capture the encoded and generated representations of entities in each meta-KG. Then, we design two CL objectives that work across entities and meta-KGs to simulate the transfer mode. *Training Procedure* is described at the end.

4.1 Representation Generation

With the initialized representations of entities, we augment the representation space with variants that are similar but still different from the initialized ones (Fig. 2(1)). We design a graph guided variational autoencoder to generate such variations because it provides finer-grained control (through the prior) to generate unseen representations with the same distribution as the original data (Kipf and Welling, 2016). Its architecture includes an encoder and a decoder, where the encoder aims to learn the encoded representation $\mathbf{h}$ and latent Gaussian distribution $\mathcal{N}(\mu_h, \sigma_h^2\mathbf{I})$, and the decoder aims to generate representation $\mathbf{h^g}$ by sampling from the above latent Gaussian prior.

The encoder first captures the graph features and then learns a latent Gaussian distribution. We use an graph neural network (GNN) to modulate the representations of entities with their multi-hop neighborhood structures,

$$\mathbf{h}^l = \sigma\Big(\frac{1}{|\mathcal{E}_{\mathcal{M}_i}(h)|} \sum_{(t,r) \in \mathcal{E}_{\mathcal{M}_i}(h)} W_r^l \mathbf{t}^{l-1} + W_0^l \mathbf{h}^{l-1}\Big) \quad (2)$$

where $\mathcal{E}_{\mathcal{M}_i}(h)$=$\{(t, r)|(t, r, h) \in \mathcal{M}_i\}$ denotes the set of (head entity, relation)-pair of immediate incoming neighbor triples of entity h. $W_r^l \in \mathbb{R}^{D^l \times D^{l-1}}$ is the relation-specific transformation matrix for relation r in the l-th layer, $W_0^l \in \mathbb{R}^{D^l \times D^{l-1}}$ is a self-loop transformation matrix for entity h in the l-th layer, D^l and D^{l-1} are the dimension in the l-th and $l-1$-th layers. σ is a ReLU activation function. For simplicity, let Eq. (2) be,

$$\mathbf{h}_e^l = \sigma\big(\text{GNN}(\mathcal{E}_{\mathcal{M}_i}(h), \mathbf{h}_e^{l-1}, D^l, D^{l-1})\big) \quad (3)$$

We use L-layer GNNs to capture the graph features, where $\mathbf{h}_e^0$=$\mathbf{h^0}$ with D^0=D (Eq. (1)), $\{D^0, ..., D^L\}$ are a list of dimensions. Then we use $\mathbf{h}_e^L$ with dimension D^L to learn a latent Gaussian distribution,

$$\mu_h = W_\mu(\mathbf{h}_e^L), \quad \sigma_h^2 = W_\sigma(\mathbf{h}_e^L)) \quad (4)$$

$$q(z|h) = \mathcal{N}(\mu_h, \sigma_h^2\mathbf{I}) \quad (5)$$

where W_μ and W_σ denote the weight matrices corresponding to the mean and variance of the (latent) Guassian distribution, respectively.

The decoder generates representation $\mathbf{h^g}$ for entity h. We use the reparametrization trick to sample $\mathbf{z}$ from the above latent Guassian distribution,

$$\mathbf{z} = \mu_h + \sigma_h \circ \epsilon \quad (6)$$

where $\epsilon \sim \mathcal{N}(0, \mathbf{I})$ (i.e., a standard normal distribution), $\circ$ denotes element-wise multiplication. We (re)construct $\mathbf{h}^g$ from $\mathbf{z}$ using another L-layer GNNs. The GNNs use dimension list $\{D^L, ..., D^0\}$ which is reversed with that in Eq. (3) and thus we denote this process from L-layer to 0-layer,

$$\mathbf{h}_d^l = \sigma\big(\text{GNN}(\mathcal{E}_{\mathcal{M}_i}(h), \mathbf{h}_d^{l+1}, D^l, D^{l+1})\big) \quad (7)$$

where $\mathbf{h}_d^L$=$\mathbf{z}$. Let $\mathbf{h}^g$ = $\mathbf{h}_d^0$ denote the generated representation of entity h.

We optimize the parameters with the combined loss of reconstruction and KL divergence,

$$L^G(h) = ||\mathbf{h} - \mathbf{h}^g||^2 + \mathcal{KL}(\mathcal{N}(\mu_h, \sigma_h^2), \mathcal{N}(0, \mathbf{I})) \quad (8)$$

here, $\mathbf{h}$ = $W_L(\mathbf{h}_e^L)$ [2], W_L is a projection matrix to transform dimension from D^L to D.

During inference, we encode a representation $\mathbf{h}$ and generate a representation $\mathbf{h}^g$ for entity h in $\mathcal{M}_i$. To distinguish, we use $\mathcal{M}_i^g$ to denote the generated meta-KG which has the same structure as $\mathcal{M}_i$ but different representations, i.e., the representation of entity h in $\mathcal{M}_i$ is the encoded representation $\mathbf{h}$ and that of entity h^g in $\mathcal{M}_i^g$ is the corresponding generated representation $\mathbf{h}^g$ [3]. In summary, the generated representation variations can augment the representation space with complementary features, which are later used in the CL objectives.

4.2 Transfer with Contrastive Learning

With the support of encoded representation $\mathbf{h}$ and generated representation $\mathbf{h}^g$, we design two CL objectives to simulate the transfer mode across entities and meta-KGs (Fig. 2(2)).

Transfer across Entities focuses on enhancing transferability across entities. We design a CL objective that works across entities inside a meta-KG,

$$L^I(h, r, t) = -\log \frac{e^{(\beta(\mathbf{h}^I, \mathbf{r}, \mathbf{t}^I)/\tau)}}{\sum_{n \in \{P, \mathcal{N}_I\}} e^{(\beta(n)/\tau)}} \quad (9)$$

where we use a triple P=(h, r, t) in the meta-KG $\mathcal{M}_i$ as a positive sample, and contrast it with negative samples $\mathcal{N}_I$. $\mathbf{h}^I$ = $\mathbf{h}$ + $\mathbf{h}^g$ and $\mathbf{t}^I$ = $\mathbf{t}$ + $\mathbf{t}^g$ are respectively the representations of the head and tail entity computed as the sum of the encoded and generated representations. The representations of the head and tail entities in a negative sample use the same summation format. For relation, we use its initial representation $\mathbf{r}$. τ is the temperature hyperparameter and $\beta(\mathbf{h}^I, \mathbf{r}, \mathbf{t}^I)$ is a similarity score which can be any Knowledge Graph Embedding (KGE) methods, such as TransE ($-||\mathbf{h}^I + \mathbf{r} - \mathbf{t}^I||$) (Bordes et al., 2013) or RotatE ($-||\mathbf{h}^I \circ \mathbf{r} - \mathbf{t}^I||$) (Sun et al., 2019). For a positive sample P=(h, r, t), its negative samples can be generated as,

$$\mathcal{N}_h = \{(h_j^-, r, t)\}_{j=0}^U, \quad \mathcal{N}_t = \{(h, r, t_j^-)\}_{j=0}^U \quad (10)$$

$$\mathcal{N}_r = \{(h, r_j^-, t)\}_{j=0}^U \quad (11)$$

where $\mathcal{N}_h$ is a set of negative samples with number-U generated by replacing the head entity of (h, r, t) with negative "head" entities h_j^-, which are sampled from a candidate entity list $\mathcal{E}_{\mathcal{M}_i} - \mathcal{E}_{\mathcal{M}_i}(r, t)$ with $\mathcal{E}_{\mathcal{M}_i}(r, t)$ being an entity list of true head entities (i.e., $h_j \in \mathcal{E}_{\mathcal{M}_i}(r, t)$ iff $(h_j, r, t) \in \mathcal{M}_i$). Similarly, $\mathcal{N}_t$ is a set of negative samples generated by replacing the tail entity of (h, r, t) with negative "tail" entities t_j^-, which are sampled from an entity list $\mathcal{E}_{\mathcal{M}_i} - \mathcal{E}_{\mathcal{M}_i}(h, r)$ with $\mathcal{E}_{\mathcal{M}_i}(h, r)$ being an entity list of true tail entities (i.e., $t_j \in \mathcal{E}_{\mathcal{M}_i}(h, r)$ iff $(h, r, t_j) \in \mathcal{M}_i$). Likewise, $\mathcal{N}_r$ is a set of negative samples generated by replacing the relation of (h, r, t) with negative relations r_j^-, which are sampled from a candidate relation list $\mathcal{R}_S - \mathcal{R}_{\mathcal{M}_i}(h, t)$ with $\mathcal{R}_{\mathcal{M}_i}(h, t)$ being a relation list that satisfies the condition: $r_j \in \mathcal{R}_{\mathcal{M}_i}(h, t)$ if $(h, r_j, t) \in \mathcal{M}_i$. Let all negative samples be $\mathcal{N}_I$ = $\{\mathcal{N}_h, \mathcal{N}_t, \mathcal{N}_r\}$.

To analyse how this CL loss affects the transfer across entities, we perform gradient analysis. The gradients with respect to the head entities are:

$$-\frac{\partial L^I(h, r, t)}{\partial \mathbf{h}^I} = \frac{1}{\tau\psi}\Big(\big[\sum_{n \in \mathcal{N}_I} e^{\frac{\beta(n)}{\tau}}\big]\beta'(\mathbf{h}^I, \mathbf{r}, \mathbf{t}^I) - \big[\sum_{n \in \{\mathcal{N}_t, \mathcal{N}_r\}} e^{(\frac{\beta(n)}{\tau})}\big]\beta'(n)\Big) \quad (12)$$

where $'$ denotes the partial derivative, ψ is a normalization constant (see Appendix). The gradients of the head entity h are closely related to positive entity t, negative entities $t_j^- \in \mathcal{N}_t$ and relations $r_j^- \in \mathcal{N}_r$. Thus, this CL can transfer the features from positive and negatives samples to head entities. The gradients with respect to the tail entity t are closely related to positive entity h, negative

[2] The encoder representation $\mathbf{h}_e^L$ shows better performance than the initialized representation $\mathbf{h}^0$.

[3] Relations in source KGs and target KGs are the same, so relation representations do not need to be transferred. We have tried to generate relation representations for the generated meta-KG, but the results are not ideal. To ensure completeness, the relation representations of the generated meta-KG are the same as that of the original meta-KG.

entities $h_j^- \in \mathcal{N}_h$ and relations $r_j^- \in \mathcal{N}_r$ (see Appendix), which proves the transferability from positive and negatives samples to tail entities.

Transfer across Meta-KGs aims to learn transferability across meta-KGs. With the support that the generated representation $\mathbf{h^g}$ in $\mathcal{M}_i^g$ has the same distribution with respect to the encoded representation $\mathbf{h}$ in $\mathcal{M}_i$, we add a relation r^g to create the link between the two meta-KGs $\mathcal{M}_i$ and $\mathcal{M}_i^g$. Specifically, for entity h in $\mathcal{M}_i$, we form the h^g in $\mathcal{M}_i^g$ as a positive sample and use the entities in other meta-KGs $\{\mathcal{M}_k\}_{k=0}^{U_a}$ as negative samples. We design a CL objective that works across meta-KGs to simulate the transfer mode,

$$L^A(h, r^g, h^g) = -\log \frac{e^{(\beta(\mathbf{h},\mathbf{r^g},\mathbf{h^g})/\tau)}}{\sum_{n\in\{P,\mathcal{N}_A\}} e^{(\beta(n)/\tau)}} \tag{13}$$

$$\mathcal{N}_A = \{\{(\mathbf{h}, \mathbf{r^g}, \mathbf{t^-_{k,j}}), (\mathbf{h}, \mathbf{r^g}, \{\mathbf{t^-_{k,j}}\}^{\mathbf{g}})\}_{j=0}^{U}\}_{k=0}^{U_a} \tag{14}$$

Similar to the positive sample, we form each negative sample as $(h, r^g, t^-_{k,j})$ by adding r^g between $\mathcal{M}_i$ and $\mathcal{M}_k$, where $t^-_{k,j} \in \mathcal{M}_k$ denotes the j-th negative entity sampled from $\mathcal{M}_k$ and $\mathbf{t^-_{k,j}}$ is its representation; $\{t^-_{k,j}\}^g \in \mathcal{M}_k^g$ denotes the j-th negative entity sampled from $\mathcal{M}_k^g$ and $\{\mathbf{t^-_{k,j}}\}^{\mathbf{g}}$ is its representation. We sample U negative entities randomly from both $\mathcal{M}_k$ and $\mathcal{M}_k^g$, and sample U_a negative meta-KGs from the same batch.

To analyse how this CL loss affects the transfer across meta-KGs, we perform gradient analysis. The gradients $(-\frac{\partial L^A(h,r^g,h^g)}{\partial \mathbf{h}})$ with respect to the head entities are,

$$\frac{1}{\tau\psi^g}\Big(\big[\sum_{n\in\mathcal{N}_A} e^{\frac{\beta(n)}{\tau}}\big]\beta'(\mathbf{h}, \mathbf{r}^g, \mathbf{h}^g) - \big[\sum_{n\in\mathcal{N}_A} e^{\frac{\beta(n)}{\tau}}\big]\beta'(n)\Big) \tag{15}$$

where ψ^g is a normalization constant (see Appendix). According to the gradients, this CL can transfer positive features from $h^g \in \mathcal{M}_i^g$ and negative features from $t^-_{k,j} \in \mathcal{M}_k, \{t^-_{k,j}\}^g \in \mathcal{M}_k^g$ to the object entity $h \in \mathcal{M}_i$.

In summary, these CLs, simulating the transfer mode across entities and multiple KGs, can enhance transferability explicitly to help the model capture transferable structural patterns. And our CL objectives are relatively independent of the representation generation module. Although we use the generated representations as contrastive objects, we can use the random, initial or word embeddings as contrastive objects for other models and tasks.

	Source KG					Target KG				
	Rel	Ent	Train	Valid	Test	Rel	Ent	Train	Valid	Test
F1	180	1,594	4,245	489	492	142	1,093	1,993	206	205
F2	200	2,608	9,739	1,166	1,180	172	1,660	4,145	469	478
F3	215	3,668	17,986	2,194	2,214	183	2,501	7,406	866	865
F4	219	4,707	27,203	3,352	3,361	200	3,051	11,714	1,416	1,424
N1	14	3,103	4,687	414	439	14	225	833	101	100
N2	88	2,564	8,219	922	968	79	2,086	4,586	459	476
N3	142	4,647	16,393	1,851	1,873	122	3,566	8,048	811	809
N4	76	2,092	7,546	876	867	61	2,795	7,073	716	731

Table 1: Dataset statistics. Rel, Ent show the no. of relations, entities in the source/target KG. Train, Valid, Test show the no. of triples in train, valid, test sets of the source/target KG.

4.3 Training Procedure

PreTrain (PT) stage is trained on multiple meta-KGs (extracted from the source KG) with the combined loss of the generation loss (L^G), CL losses (L^I, L^A) and task-specific loss (L^ζ):

$$L = \sum_{\mathcal{M}_i\in\mathcal{M}_S}\sum_{(h,r,t)\in\mathcal{M}_i} \eta_1(L^G + L^I + L^A) + \eta_2 L^\zeta \tag{16}$$

where the weights η_1 and η_2 are used to bring the two kinds of losses to the same order of magnitude. We use *link prediction* as a downstream task to optimize the parameters.

$$L^\zeta(h, r, t) = -\log\vartheta(\lambda + \beta(\mathbf{h}^I, \mathbf{r}, \mathbf{t}^I)) - \textstyle\sum_{n\in\mathcal{N}_I} \zeta(n)\log\vartheta(-\lambda - \beta(n)) \tag{17}$$

$$\zeta(n) = \frac{e^{(\beta(n))}}{\sum_{n\in\mathcal{N}_I} e^{(\beta(n))}} \tag{18}$$

where $\mathbf{h}^I, \mathbf{r}, \mathbf{t}^I$ are the representations of h, r, t (similar to Eq. (9)); $\zeta(n)$ is a self-adversarial negative sampling function which generates different weights for different negative samples according to their importance to the triple (h, r, t). ϑ is the sigmoid function, λ is a fixed margin.

FineTune (FT) With pretrained parameters (by the source KG) as initialization, the FT stage is fine-tuned on the target KG with task-specific loss.

$$L = \Sigma_{(h,r,t)\in\mathcal{G}_T} L^\zeta(h, r, t) \tag{19}$$

5 Experiments

5.1 Datasets

Table 1 shows the statistics of our datasets. We use 8 benchmark inductive KGs: F1, F2, F3, F4 and

Model	F1			F2			F3			F4		
	MRR	H@1	H@10	MRR	H@1	H@10	MRR	H@1	H@10	MRR	H@1	H@10
Neural-LP *	46.13	40.21	52.92	51.85	45.68	58.94	48.70	44.09	52.90	49.54	44.12	55.88
DRUM *	47.55	42.71	52.92	52.78	47.49	58.73	49.64	45.84	52.90	50.43	45.53	55.88
RuleN *	45.97	41.46	49.76	69.08	62.13	77.82	73.68	65.95	87.69	74.19	67.21	85.60
GraIL *	48.56	40.00	64.15	62.54	52.20	81.80	70.35	60.25	82.83	70.60	60.99	89.29
CoMPILE *	-	-	67.64	-	-	82.98	-	-	84.67	-	-	87.44
MorsE	69.61	58.82	90.39	80.38	71.34	96.33	**78.58**	69.73	95.34	**80.74**	**72.02**	96.62
VMCL	**72.68**	**62.61**	**92.47**	**81.66**	**73.05**	**97.29**	78.29	**69.75**	**95.51**	80.42	71.68	**96.72**

Table 2: Results of VMCL over baselines on F1-F4 datasets, where the results with * are taken from (Chen et al., 2022). Bold numbers denote the best results. VMCL is significantly better than MorsE with p-value=.028.

Model	N1			N2			N3			N4		
	MRR	H@1	H@10	MRR	H@1	H@10	MRR	H@1	H@10	MRR	H@1	H@10
Neural-LP *	35.71	29.13	40.78	64.68	55.45	78.73	69.93	60.46	82.71	68.21	59.27	80.58
DRUM *	18.89	14.08	19.42	66.44	58.55	78.55	72.28	64.29	82.71	70.42	63.09	80.58
RuleN *	46.35	39.00	53.50	70.80	63.08	81.75	68.76	61.99	77.26	56.31	51.44	61.35
GraIL *	52.04	46.50	59.50	72.92	63.08	93.25	74.37	64.22	91.41	62.98	57.27	73.19
CoMPILE *	-	-	58.38	-	-	93.87	-	-	92.77	-	-	75.19
MorsE	54.10	46.80	66.68	80.13	70.35	96.65	86.38	78.43	**98.34**	80.11	70.29	96.12
VMCL	**55.06**	**48.16**	**66.86**	**84.70**	**76.32**	**97.61**	**87.17**	**79.65**	98.30	**82.17**	**72.86**	**96.91**

Table 3: Results of VMCL over baselines on N1-N4 datasets, where the results with * are taken from (Chen et al., 2022). Bold numbers denote the best results. VMCL is significantly better than MorsE with p-value=.006.

N1, N2, N3, N4 derived from (Teru et al., 2020). A group of inductive KGs (e.g., F1) has a source KG and a target KG, where the source KG has train, valid, test sets which are different with the train, valid, test sets of the target KG. All experiments are evaluated on the test set of the target KG. The average degree of F1-4 and N1-4 is increasing, e.g., F1-4 (2.7, 3.7, 4.9, 5.8) and N1-4 (1.5, 3.2, 3.5, 3.6). Low average degree represents a sparser KG, while high average degree represents a denser KG.

Obtaining meta-KGs. Our model uses meta-KGs to simulate the inductive setting where a meta-KG $\mathcal{M}_i$ is sampled from a source KG $\mathcal{G}_S$ with the following steps (Fig. 2(1)): (1) Sample an entity from the entity list $\mathcal{E}_{\mathcal{G}_S}$ of the source KG $\mathcal{G}_S$, and put it into set $\mathcal{E}_{\mathcal{M}_i}$. (2) Sample an entity from $\mathcal{E}_{\mathcal{M}_i}$, walk randomly n_1 times with length-n_2, and put the walked entities into set $\mathcal{E}_{\mathcal{M}_i}$. (3) Repeat the above step (2) n_3 times and use entities in $\mathcal{E}_{\mathcal{M}_i}$ to induce a meta-KG $\mathcal{M}_i$. (4) Anonymize the entities in $\mathcal{M}_i$ by re-labeling the id of entities to be $\{1,2,...,|\mathcal{E}_{\mathcal{M}_i}|\}$ in an arbitrary order.

5.2 Evaluation Metrics and Baselines

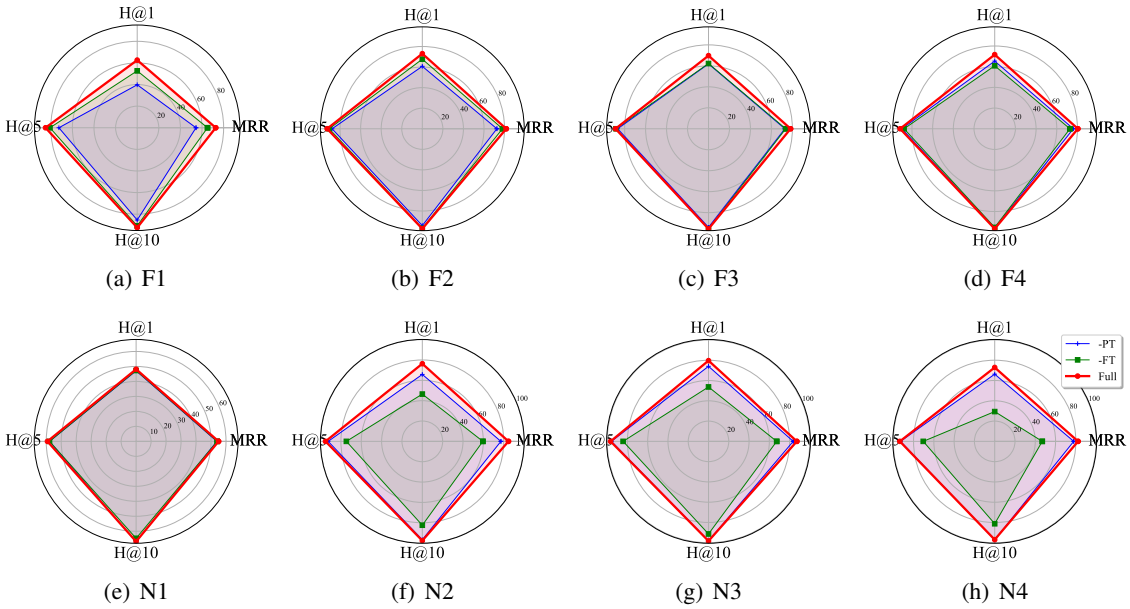
We report MRR, H@N scores to evaluate the performance of link prediction in the test set of the target KG. The results are averaged by head and tail predictions. Following the settings in (Chen et al., 2022), the results of baselines and VMCL are approximated by ranking each test triple among 50 other randomly sampled negative triples five times.

To show the effectiveness of our approach, we compare VMCL against several strong baselines:

- **One Stage** models are directly adapted to the target KG after being trained on the source KG. They do not have pretraining-finetuning steps. This includes: RuleN (Meilicke et al., 2018), Neural-LP (Yang et al., 2017), DRUM (Sadeghian et al., 2019), GraIL (Teru et al., 2020), CoMPILE (Mai et al., 2021).

- **PT+FT** models are trained on the source KG then finetuned on the target KG. This includes variants of MorsE (Chen et al., 2022) with different KGEs (TransE/RotatE). To our best knowledge, these MorsE models are the state-of-the-art baselines.

VMCL is a PT+FT model. We design variants of VMCL with different KGEs (TransE/RotatE) and training settings (Full/-PT/-FT). The Full is finetuned on the target KG with pretrained parameters from the source KG then uses the finetuned param-



(a) F1 (b) F2 (c) F3 (d) F4

(e) N1 (f) N2 (g) N3 (h) N4

Figure 3: Results of VMCL over pretrain/finetune stages on F1-F4 and N1-N4.

eters to test the target KG, -PT variant is trained on the target KG with random parameters, -FT variant is trained on the source KG then uses pretrained parameters to test the target KG. Please refer to Appendix (A.3) for the settings and hyperparameters.

5.3 Results

In this section, we conduct extensive experiments to show the effectiveness of VMCL.

• **VMCL over Baselines** The results of VMCL and baselines are shown in Table 2 (for F1-F4 datasets) and Table 3 (for N1-N4 datasets).[4] We notice that VMCL achieves significant improvements over the baselines. The one stage baselines perform poorly because they only allow training on the source KG and lack adaptive training on the target KG, while MorsE is better than those as it also finetunes the model on the target KG. Our VMCL, which uses generated representations to provide complementary features and uses CL to enhance the transfer across entities and KGs, outperforms MorsE by a large margin. Specifically, the H@1 score of VMCL increases by 1.36, 5.97, 1.22, 2.57 points in N1, N2, N3,and N4 datasets, respectively (Table 3). And we find that VMCL performs better on sparse KGs (F1-F3, N1-N4), and does not have a significant effect on dense KGs (F4), which aligns with our motivation to alleviate sparsity. These significant improvements over the baselines prove the effectiveness of our model.

[4]The default KGE model of MorsE and VMCL is RotatE.

• **VMCL over PT/FT Stages** We report the results of VMCL with Full/-PT/-FT settings in Fig. 3. First, we show the necessity of PT stage by comparing the Full with -PT models. The Full model outperforms -PT with sizeable margins in F1-F4 and N2-N4 and competitive margin in N1. Concretely, H@1 score of Full increases by 22.83 in F1, 12.31 in F2, 7.86 in F3, 6.27 in F4 and 1.06 in N1, 10.52 in N2, 5.60 in N3, and 6.61 in N4. The performance improvement of Full over -PT indicates the transfer capability of PT stage in VMCL. That is, the structural patterns, captured by PT stage from the source KG, can help the representation learning of the target KG achieve better results. Second, we show the necessity of FT stage by comparing the Full with -FT models. The performance of Full is better than that of -FT on all datasets. Notably, H@1 score of Full increases by 9.93 point in F1, 5.35 in F2, 7.53 in F3, 10.79 in F4, and 0.86 in N1, 29.85 in N2, 25.97 in N3, and 43.50 point in N4. The above results of Full over -FT show that although the pretrained parameters by the source KG can bring performance improvements, it is necessary to finetune the parameters on the target KG to adapt to its own characteristics.

• **VMCL over KGE Models** We investigate the impact of different KGE (TransE/RotatE) models, and the results are shown in Table 4. Through observation, we find the following two conclusions: (1) With the same KGE, the proposed VMCL always outperforms the baseline MorsE with sizeable

	Model	F1			F2			F3			F4		
		MRR	H@1	H@10	MRR	H@1	H@10	MRR	H@1	H@10	MRR	H@1	H@10
TransE	MorsE	67.49	55.90	89.73	79.22	69.58	96.37	77.03	67.55	94.77	79.48	70.01	96.66
	VMCL	70.80	59.64	91.81	80.47	71.45	96.00	77.52	68.12	95.41	79.52	70.21	**96.84**
RotatE	MorsE	69.61	58.82	90.39	80.38	71.34	96.33	**78.58**	69.73	95.34	**80.74**	**72.02**	96.62
	VMCL	**72.68**	**62.61**	**92.47**	**81.66**	**73.05**	**97.29**	78.29	**69.75**	**95.51**	80.42	71.68	96.72
	Model	N1			N2			N3			N4		
		MRR	H@1	H@10	MRR	H@1	H@10	MRR	H@1	H@10	MRR	H@1	H@10
TransE	MorsE	47.61	39.92	61.42	76.53	64.63	96.69	81.55	70.20	98.60	74.71	62.04	95.62
	VMCL	49.88	41.50	63.70	78.55	66.97	97.53	83.57	73.24	**98.74**	76.71	64.87	95.82
RotatE	MorsE	54.10	46.80	66.68	80.13	70.35	96.65	86.38	78.43	98.34	80.11	70.29	96.12
	VMCL	**55.06**	**48.16**	**66.86**	**84.70**	**76.32**	**97.61**	**87.17**	**79.65**	98.30	**82.17**	**72.86**	**96.91**

Table 4: Results of different KGE models.

Model	F1	F2	F3	F4
Full	62.61	73.05	69.75	71.68
w/o G	52.05	62.84	58.83	59.47
w/o GFNN	51.96	65.86	60.48	61.25
w/o CL	58.76	72.75	69.69	71.98
w/o CLE	61.19	71.70	70.06	72.79
w/o CLM	58.93	72.93	70.46	71.86

Table 5: Results (H@1%) of ablation study in F1-F4.

margins in all datasets. (2) The performance of the models with RotatE is better than that with TransE.

• **Ablation Study** We first investigate the ablation study (Table 5) of *representation generation* by removing / replacing one component from the Full at a time: (w/o G) removing the generated representations in which the CL across meta-KGs is also removed, (w/o GFNN) replacing the GNN with a fully-connected neural network. Compared to Full, removing the representation generation module (w/o G) decreases the performance in all datasets, especially, 10.56, 10.21, 10.92, 12.21 points in F1-F4. And the performance degradation of GFNN over Full proves the effectiveness of the proposed GNN module. Second, we investigate the ablation study of *Transfer with CL* by: (w/o CL) removing the two CL objectives, (w/o CLE) removing the CL across entities, (w/o CLM) removing the CL across meta-KGs. The results in Table 5 show that these CL modules (across entities and meta-KGs) play a crucial role in improving the performance of F1, F2 datasets and has little effect in F3, F4 datasets. That is, these CL modules are more effective on sparse KGs (e.g., F1, F2) because the introduced negative samples across entities and meta-KGs effectively alleviate the problem of insufficient training caused by sparse data (missing links). Please refer to Appendix for the ablation study (A.1) in N1-N4.

• **Time Complexity** We analyze the time complexity of VMCL for a meta-KG $\mathcal{M}_i$=($\mathcal{E}_{\mathcal{M}_i}, \mathcal{R}_{\mathcal{M}_i}$) with $|\mathcal{M}_i|$ as the number of triples. A GNN (Eq. (2)) requires $\mathcal{O}(|\mathcal{M}_i| \cdot D^2)$, where we use D represent the maximum dimension in $\{D^0, ..., D^L\}$. So the representation generation module requires $\mathcal{O}(2L \cdot |\mathcal{M}_i| \cdot D^2)$. A CL objective (Eq. (9) or Eq. (13)) requires $\mathcal{O}((\mathcal{N}+1) \cdot 2L \cdot |\mathcal{M}_i| \cdot D^2)$ with $\mathcal{N}$ as the maximum of $\mathcal{N}_I$ and $\mathcal{N}_A$. So the two CL objectives for meta-KG $\mathcal{M}_i$ require $\mathcal{O}(2|\mathcal{M}_i| \cdot (\mathcal{N}+1) \cdot 2L \cdot |\mathcal{M}_i| \cdot D^2)$, omitted and abbreviated as $\mathcal{O}(|\mathcal{M}_i|^2 \cdot \mathcal{N} \cdot D^2)$. The time complexity of all meta-KGs ($|\mathcal{M}|$) is $\mathcal{O}(|\mathcal{M}| \cdot |\mathcal{M}_i|^2 \cdot \mathcal{N} \cdot D^2)$. Because the number of triples in meta-KG $|\mathcal{M}_i|$ is small relative to source KG, the calculation speed in fact is very fast.

6 Conclusion

In this work, we have provided empirical insights about inductive KGs, and proposed VMCL to alleviate the sparsity and implicit transfer. VMCL first uses representation generation to capture the encoded and generated representations of entities, where the generated variations can augment the representation space with complementary features. Then, VMCL uses two CL objectives that work across entities and meta-KGs to simulate the transfer mode. Extensive experiments and comprehensive analysis have shown that VMCL outperforms state-of-the-art baselines with sizeable margins.

Limitations

The representation dimension (default is 32) is important but limited by our GPU resources. With the support of large GPU, a large dimension (e.g., 512) may achieve better performance. We also attempt to expand the inductive setting (Relational patterns are same in the source and target KGs) to the independent setting (The source KG is independent from the target KG), but experimental performance is not good. That is, if the two KGs are irrelevant, it may be impossible to transfer information.

Acknowledgements

The work was supported by National Natural Science Foundation of China (62172086, 62272092) and Chinese Scholarship Council.

References

Seong Jin Ahn and MyoungHo Kim. 2021. Variational graph normalized autoencoders. In *Proceedings of the 30th ACM International Conference on Information and Knowledge Management*, page 2827–2831, New York, NY, USA.

Kian Ahrabian, Aarash Feizi, Yasmin Salehi, William L. Hamilton, and Avishek Joey Bose. 2020. Structure aware negative sampling in knowledge graphs. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP, Online, November 16-20, 2020*, pages 6093–6101.

Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *27th Annual Conference on Neural Information Processing Systems. December 5-8, 2013, Lake Tahoe, Nevada, United States*, pages 2787–2795.

Mingyang Chen, Wen Zhang, Yushan Zhu, Hongting Zhou, Zonggang Yuan, Changliang Xu, and Huajun Chen. 2022. Meta-knowledge transfer for inductive knowledge graph embedding. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 927–937.

Luis Antonio Galárraga, Christina Teflioudi, Katja Hose, and Fabian Suchanek. 2013. Amie: association rule mining under incomplete evidence in ontological knowledge bases. In *Proceedings of the 22nd international conference on World Wide Web*.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. Simcse: Simple contrastive learning of sentence embeddings. *CoRR*, abs/2104.08821.

Yanchao Hao, Yuanzhe Zhang, Kang Liu, Shizhu He, Zhanyi Liu, Hua Wu, and Jun Zhao. 2017. An end-to-end model for question answering over knowledge base with cross-attention combining global knowledge. In *55th Annual Meeting of the Association for Computational Linguistics*, Vancouver, Canada.

Kaveh Hassani and Amir Hosein Khas Ahmadi. 2020. Contrastive multi-view representation learning on graphs. In *Proceedings of the 37th International Conference on Machine Learning, ICML, 13-18 July, Virtual Event*, volume 119, pages 4116–4126.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross B. Girshick. 2020. Momentum contrast for unsupervised visual representation learning. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR, Seattle, WA, USA, June 13-19, 2020*, pages 9726–9735.

Faliang Huang, Xuelong Li, Changan Yuan, Shichao Zhang, Jilian Zhang, and Shaojie Qiao. 2021. Attention-emotion-enhanced convolutional lstm for sentiment analysis. *IEEE transactions on neural networks and learning systems*, 33(9):4332–4345.

Diederik P. Kingma and Max Welling. 2014. Auto-encoding variational bayes. In *2nd International Conference on Learning Representations, ICLR, Banff, AB, Canada, April 14-16, 2014*.

Thomas N. Kipf and Max Welling. 2016. Variational graph auto-encoders. *CoRR*, abs/1611.07308.

Qian Li, Shafiq Joty, Daling Wang, Shi Feng, and Yifei Zhang. 2022a. Alleviating sparsity of open knowledge graphs with ternary contrastive learning. *arXiv preprint arXiv:2211.03950*.

Qian Li, Daling Wang, Shi Feng, Cheng Niu, and Yifei Zhang. 2021. Global graph attention embedding network for relation prediction in knowledge graphs. *IEEE Transactions on Neural Networks and Learning Systems*.

Qian Li, Daling Wang, Shi Feng, Kaisong Song, Yifei Zhang, and Ge Yu. 2023. Variational autoencoder densified graph attention for fusing synonymous entities: Model and protocol. *Knowl. Based Syst.*, 259:110061.

Qian Li, Daling Wang, Shi Feng Kaisong Song, Yifei Zhang, and Ge Yu. 2022b. Oerl: Enhanced representation learning via open knowledge graphs. *IEEE Transactions on Knowledge and Data Engineering*.

Sijie Mai, Shuangjia Zheng, Yuedong Yang, and Haifeng Hu. 2021. Communicative message passing for inductive relation reasoning. In *AAAI*, pages 4294–4302.

Christian Meilicke, Manuel Fink, Yanjie Wang, Daniel Ruffinelli, Rainer Gemulla, and Heiner Stuckenschmidt. 2018. Fine-grained evaluation of rule-and embedding-based systems for knowledge graph completion. In *International semantic web conference*.

Chengwei Qin, Shafiq R. Joty, Qian Li, and Ruochen Zhao. 2023. Learning to initialize: Can meta learning improve cross-task generalization in prompt tuning? *CoRR*, abs/2302.08143.

Ali Sadeghian, Mohammadreza Armandpour, Patrick Ding, and Daisy Zhe Wang. 2019. Drum: End-to-end differentiable rule mining on knowledge graphs. *Advances in Neural Information Processing Systems*, 32.

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 *European semantic web conference*, pages 593–607. Springer.

Martin Simonovsky and Nikos Komodakis. 2018. Graphvae: Towards generation of small graphs using variational autoencoders. In *ICANN-27th International Conference on Artificial Neural Networks, Rhodes, Greece, October 4-7, 2018, Proceedings, Part I*, volume 11139, pages 412–422.

Fan-Yun Sun, Jordan Hoffmann, Vikas Verma, and Jian Tang. 2020. Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. In *8th International Conference on Learning Representations, ICLR, Addis Ababa, Ethiopia, April 26-30, 2020*.

Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. Rotate: Knowledge graph embedding by relational rotation in complex space. In *7th International Conference on Learning Representations, ICLR, New Orleans, LA, USA, May 6-9, 2019*.

Komal Teru, Etienne Denis, and Will Hamilton. 2020. Inductive relation prediction by subgraph reasoning. In *International Conference on Machine Learning*, pages 9448–9457. PMLR.

Petar Velickovic, William Fedus, William L. Hamilton, Pietro Liò, Yoshua Bengio, and R. Devon Hjelm. 2019. Deep graph infomax. In *7th International Conference on Learning Representations, ICLR, New Orleans, LA, USA, May 6-9, 2019*.

Liang Wang, Wei Zhao, Zhuoyu Wei, and Jingming Liu. 2022. Simkgc: Simple contrastive knowledge graph completion with pre-trained language models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics, ACL, Dublin, Ireland, May 22-27, 2022*, pages 4281–4294.

Fan Yang, Zhilin Yang, and William W Cohen. 2017. Differentiable learning of logical rules for knowledge base reasoning. *Advances in neural information processing systems*, 30.

Yuyu Zhang, Hanjun Dai, Zornitsa Kozareva, Alexander J. Smola, and Le Song. 2018. Variational reasoning for question answering with knowledge graph. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 6069–6076.

Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. 2021. Graph contrastive learning with adaptive augmentation. In *WWW '21: The Web Conference 2021, Virtual Event / Ljubljana, Slovenia, April 19-23, 2021*, pages 2069–2080.

A Appendix

A.1 Ablation Study in N1-N4

Comp	N1	N2	N3	N4
Full	48.16	76.32	79.65	72.86
w/o G	48.10	55.40	67.35	48.75
w/o GFNN	49.60	63.26	69.88	56.82
w/o CL	47.24	77.36	80.71	73.14
w/o CLE	47.88	75.92	80.20	71.90
w/o CLM	47.46	75.71	79.39	72.80

Table 6: Results (H@1%) of ablation study on N1-N4.

We first investigate the ablation study (Table 6) in N1-N4 datasets. The representation generation module improve the performance in N2-N4 datasets, and these CL modules (across entities and meta-KGs) play a crucial role in improving performance of N1 datasets.

A.2 Gradient Reasoning

$$
\begin{aligned}
&-\frac{\partial L^I(h,r,t)}{\partial h} \\
&= \frac{\partial}{\partial h}\left(\frac{\beta(\mathbf{h}^I,\mathbf{r},\mathbf{t}^I)}{\tau} - \log \sum_{n\in\{p,\mathcal{N}_I\}} e^{\left(\frac{\beta(n)}{\tau}\right)}\right) \\
&= \frac{\partial}{\partial h}\Big(\frac{\beta(\mathbf{h}^I,\mathbf{r},\mathbf{t}^I)}{\tau} - \log\Big(e^{\left(\frac{\beta(\mathbf{h}^I,\mathbf{r},\mathbf{t}^I)}{\tau}\right)} \\
&\quad + \sum_{(h,r,t_j^-)\in\mathcal{N}_t} e^{\left(\frac{\beta(\mathbf{h}^I,\mathbf{r},\mathbf{t}_\mathbf{j}^{-I})}{\tau}\right)} \\
&\quad + \sum_{(h_j^-,r,t)\in\mathcal{N}_h} e^{\left(\frac{\beta(\mathbf{h}_\mathbf{j}^{-I},\mathbf{r},\mathbf{t}^I)}{\tau}\right)} \\
&\quad + \sum_{(h,r_j^-,t)\in\mathcal{N}_r} e^{\left(\frac{\beta(\mathbf{h}^I,\mathbf{r}_\mathbf{j}^-,\mathbf{t}^I)}{\tau}\right)}\Big)\Big) \\
&= \frac{\beta'(\mathbf{h}^I,\mathbf{r},\mathbf{t}^I)}{\tau} - \frac{1}{\psi}\Big(e^{\left(\frac{\beta(\mathbf{h}^I,\mathbf{r},\mathbf{t}^I)}{\tau}\right)}\frac{\beta'(\mathbf{h}^I,\mathbf{r},\mathbf{t}^I)}{\tau} \\
&\quad + \sum_{(h,r,t_j^-)\in\mathcal{N}_t} e^{\left(\frac{\beta(\mathbf{h}^I,\mathbf{r},\mathbf{t}_\mathbf{j}^{-I})}{\tau}\right)}\frac{\beta'(\mathbf{h}^I,\mathbf{r},\mathbf{t}_\mathbf{j}^{-I})}{\tau} \\
&\quad + \sum_{(h,r_j^-,t)\in\mathcal{N}_r} e^{\left(\frac{\beta(\mathbf{h}^I,\mathbf{r}_\mathbf{j}^-,\mathbf{t}^I)}{\tau}\right)}\frac{\beta'(\mathbf{h}^I,\mathbf{r}_\mathbf{j}^-,\mathbf{t}^I)}{\tau}\Big) \\
&= \frac{1}{\tau\psi}\Big(\psi\beta'(\mathbf{h}^I,\mathbf{r},\mathbf{t}^I) - e^{\left(\frac{\beta(\mathbf{h}^I,\mathbf{r},\mathbf{t}^I)}{\tau}\right)}\beta'(\mathbf{h}^I,\mathbf{r},\mathbf{t}^I) \\
&\quad - \sum_{(h,r,t_j^-)\in\mathcal{N}_t} e^{\left(\frac{\beta(\mathbf{h}^I,\mathbf{r},\mathbf{t}_\mathbf{j}^{-I})}{\tau}\right)}\beta'(\mathbf{h}^I,\mathbf{r},\mathbf{t}_\mathbf{j}^{-I}) \\
&\quad - \sum_{(h,r_j^-,t)\in\mathcal{N}_r} e^{\left(\frac{\beta(\mathbf{h}^I,\mathbf{r}_\mathbf{j}^-,\mathbf{t}^I)}{\tau}\right)}\beta'(\mathbf{h}^I,\mathbf{r}_\mathbf{j}^-,\mathbf{t}^I)\Big) \\
&= \frac{1}{\tau\psi}\Big(\sum_{(n)\in\mathcal{N}_I} e^{\left(\frac{\beta(n)}{\tau}\right)}\beta'(\mathbf{h}^I,\mathbf{r},\mathbf{t}^I) \\
&\quad - \sum_{(h,r,t_j^-)\in\mathcal{N}_t} e^{\left(\frac{\beta(\mathbf{h}^I,\mathbf{r},\mathbf{t}_\mathbf{j}^{-I})}{\tau}\right)}\beta'(\mathbf{h}^I,\mathbf{r},\mathbf{t}_\mathbf{j}^{-I}) \\
&\quad - \sum_{(h,r_j^-,t)\in\mathcal{N}_r} e^{\left(\frac{\beta(\mathbf{h}^I,\mathbf{r}_\mathbf{j}^-,\mathbf{t}^I)}{\tau}\right)}\beta'(\mathbf{h}^I,\mathbf{r}_\mathbf{j}^-,\mathbf{t}^I)\Big) \\
&= \frac{1}{\tau\psi}\Big(\Big[\sum_{n\in\mathcal{N}_I} e^{(\beta(n)/\tau)}\Big]\beta'(\mathbf{h}^I,\mathbf{r},\mathbf{t}^I) \\
&\quad - \Big[\sum_{n\in\{\mathcal{N}_t,\mathcal{N}_r\}} e^{\left(\frac{\beta(n)}{\tau}\right)}\Big]\beta'(n)\Big)
\end{aligned}
\tag{20}
$$

$$
\begin{aligned}
&-\frac{\partial L^I(h,r,t)}{\partial t} \\
&= \frac{\partial}{\partial t}\left(\frac{\beta(\mathbf{h}^I,\mathbf{r},\mathbf{t}^I)}{\tau} - \log \sum_{n\in\{p,\mathcal{N}_I\}} e^{\left(\frac{\beta(n)}{\tau}\right)}\right) \\
&= \frac{\partial}{\partial t}\Big(\frac{\beta(\mathbf{h}^I,\mathbf{r},\mathbf{t}^I)}{\tau} - \log\Big(e^{\left(\frac{\beta(\mathbf{h}^I,\mathbf{r},\mathbf{t}^I)}{\tau}\right)} \\
&\quad + \sum_{(h,r,t_j^-)\in\mathcal{N}_t} e^{\left(\frac{\beta(\mathbf{h}^I,\mathbf{r},\mathbf{t}_\mathbf{j}^{-I})}{\tau}\right)} \\
&\quad + \sum_{(h_j^-,r,t)\in\mathcal{N}_h} e^{\left(\frac{\beta(\mathbf{h}_\mathbf{j}^{-I},\mathbf{r},\mathbf{t}^I)}{\tau}\right)} \\
&\quad + \sum_{(h,r_j^-,t)\in\mathcal{N}_r} e^{\left(\frac{\beta(\mathbf{h}^I,\mathbf{r}_\mathbf{j}^-,\mathbf{t}^I)}{\tau}\right)}\Big)\Big) \\
&= \frac{\beta'(\mathbf{h}^I,\mathbf{r},\mathbf{t}^I)}{\tau} - \frac{1}{\psi}\Big(e^{\left(\frac{\beta(\mathbf{h}^I,\mathbf{r},\mathbf{t}^I)}{\tau}\right)}\frac{\beta'(\mathbf{h}^I,\mathbf{r},\mathbf{t}^I)}{\tau} \\
&\quad + \sum_{(h_j^-,r,t)\in\mathcal{N}_h} e^{\left(\frac{\beta(\mathbf{h}_\mathbf{j}^{-I},\mathbf{r},\mathbf{t}^I)}{\tau}\right)}\frac{\beta'(\mathbf{h}_\mathbf{j}^{-I},\mathbf{r},\mathbf{t}^I)}{\tau} \\
&\quad + \sum_{(h,r_j^-,t)\in\mathcal{N}_r} e^{\left(\frac{\beta(\mathbf{h}^I,\mathbf{r}_\mathbf{j}^-,\mathbf{t}^I)}{\tau}\right)}\frac{\beta'(\mathbf{h}^I,\mathbf{r}_\mathbf{j}^-,\mathbf{t}^I)}{\tau}\Big) \\
&= \frac{1}{\tau\psi}\Big(\psi\beta'(\mathbf{h}^I,\mathbf{r},\mathbf{t}^I) - e^{\left(\frac{\beta(\mathbf{h}^I,\mathbf{r},\mathbf{t}^I)}{\tau}\right)}\beta'(\mathbf{h}^I,\mathbf{r},\mathbf{t}^I) \\
&\quad - \sum_{(h_j^-,r,t)\in\mathcal{N}_h} e^{\left(\frac{\beta(\mathbf{h}_\mathbf{j}^{-I},\mathbf{r},\mathbf{t}^I)}{\tau}\right)}\beta'(\mathbf{h}_\mathbf{j}^{-I},\mathbf{r},\mathbf{t}^I) \\
&\quad - \sum_{(h,r_j^-,t)\in\mathcal{N}_r} e^{\left(\frac{\beta(\mathbf{h}^I,\mathbf{r}_\mathbf{j}^-,\mathbf{t}^I)}{\tau}\right)}\beta'(\mathbf{h}^I,\mathbf{r}_\mathbf{j}^-,\mathbf{t}^I)\Big) \\
&= \frac{1}{\tau\psi}\Big(\sum_{(n)\in\mathcal{N}_I} e^{\left(\frac{\beta(n)}{\tau}\right)}\beta'(\mathbf{h}^I,\mathbf{r},\mathbf{t}^I) \\
&\quad - \sum_{(h_j^-,r,t)\in\mathcal{N}_h} e^{\left(\frac{\beta(\mathbf{h}_\mathbf{j}^{-I},\mathbf{r},\mathbf{t}^I)}{\tau}\right)}\beta'(\mathbf{h}_\mathbf{j}^{-I},\mathbf{r},\mathbf{t}^I) \\
&\quad - \sum_{(h,r_j^-,t)\in\mathcal{N}_r} e^{\left(\frac{\beta(\mathbf{h}^I,\mathbf{r}_\mathbf{j}^-,\mathbf{t}^I)}{\tau}\right)}\beta'(\mathbf{h}^I,\mathbf{r}_\mathbf{j}^-,\mathbf{t}^I)\Big) \\
&= \frac{1}{\tau\psi}\Big(\Big[\sum_{n\in\mathcal{N}_I} e^{(\beta(n)/\tau)}\Big]\beta'(\mathbf{h}^I,\mathbf{r},\mathbf{t}^I) \\
&\quad - \Big[\sum_{n\in\{\mathcal{N}_h,\mathcal{N}_r\}} e^{\left(\frac{\beta(n)}{\tau}\right)}\Big]\beta'(n)\Big)
\end{aligned}
\tag{21}
$$

$$
\begin{aligned}
\psi &= e^{\left(\frac{\beta(\mathbf{h}^I,\mathbf{r},\mathbf{t}^I)}{\tau}\right)} \\
&\quad + \sum_{(h,r,t_j^-)\in\mathcal{N}_h} e^{\left(\frac{\beta(\mathbf{h}^I,\mathbf{r},\mathbf{t}_\mathbf{j}^{-I})}{\tau}\right)} \\
&\quad + \sum_{(h_j^-,r,t)\in\mathcal{N}_t} e^{\left(\frac{\beta(\mathbf{h}_\mathbf{j}^{-I},\mathbf{r},\mathbf{t}^I)}{\tau}\right)} \\
&\quad + \sum_{(h,r_j^-,t)\in\mathcal{N}_r} e^{\left(\frac{\beta(\mathbf{h}^I,\mathbf{r}_\mathbf{j}^-,\mathbf{t}^I)}{\tau}\right)}
\end{aligned}
\tag{22}
$$

$$
\begin{aligned}
\psi^g &= e^{\left(\frac{\beta(\mathbf{h},\mathbf{r}^g,\mathbf{h}^g)}{\tau}\right)} \\
&\quad + \sum_{(h,r^g,t_{k,j}^-)\in\mathcal{N}_A} e^{\left(\frac{\beta(\mathbf{h},\mathbf{r}^g,\mathbf{t}_\mathbf{j}^-)}{\tau}\right)} \\
&\quad + \sum_{(h,r^g,\{t_{k,j}^-\}^g)\in\mathcal{N}_A} e^{\left(\frac{\beta(\mathbf{h},\mathbf{r}^g,\{\mathbf{t}_\mathbf{j}^-\}^g)}{\tau}\right)}
\end{aligned}
\tag{23}
$$

$$
\begin{aligned}
&-\frac{\partial L^A(h,r^g,h^g)}{\partial h} \\
&= \frac{\partial}{\partial h}(\log \frac{e^{(\beta(\mathbf{h},\mathbf{r^g},\mathbf{h^g})/\tau)}}{\sum_{n\in\{P,\mathcal{N}_A\}} e^{(\beta(n)/\tau)}}) \\
&= \frac{\partial}{\partial h}(\frac{\beta(\mathbf{h},\mathbf{r}^g,\mathbf{h}^g)}{\tau} - \log \sum_{n\in\{p,\mathcal{N}_A\}} e^{(\frac{\beta(n)}{\tau})}) \\
&= \frac{\partial}{\partial h}(\frac{\beta(\mathbf{h},\mathbf{r}^g,\mathbf{h}^g)}{\tau} - \log(e^{(\frac{\beta(\mathbf{h},\mathbf{r}^g,\mathbf{h}^g)}{\tau})} \\
&\quad + \sum_{(h,r^g,t^-_{k,j})\in\mathcal{N}_A} e^{(\frac{\beta(\mathbf{h},\mathbf{r}^g,\mathbf{t}^-_{\mathbf{j}})}{\tau})} \\
&\quad + \sum_{(h,r^g,\{t^-_{k,j}\}^g)\in\mathcal{N}_A} e^{(\frac{\beta(\mathbf{h},\mathbf{r}^g,\{\mathbf{t}^-_{\mathbf{j}}\}^g)}{\tau})}))) \\
&= \frac{\beta'(\mathbf{h},\mathbf{r}^g,\mathbf{h}^g)}{\tau} - \frac{1}{\psi^g}(e^{(\frac{\beta(\mathbf{h},\mathbf{r}^g,\mathbf{h}^g)}{\tau})}\frac{\beta'(\mathbf{h},\mathbf{r}^g,\mathbf{h}^g)}{\tau} \\
&+ \sum_{(h,r^g,t^-_{k,j})\in\mathcal{N}_A} e^{(\frac{\beta(\mathbf{h},\mathbf{r}^g,\mathbf{t}^-_{\mathbf{k,j}})}{\tau})}\frac{\beta'(\mathbf{h},\mathbf{r}^g,\mathbf{t}^-_{\mathbf{k,j}})}{\tau} \\
&+ \sum_{(h,r^g,\{t^-_{k,j}\}^g)\in\mathcal{N}_A} e^{(\frac{\beta(\mathbf{h},\mathbf{r}^g,\{\mathbf{t}^-_{\mathbf{k,j}}\}^g)}{\tau})}\frac{\beta'(\mathbf{h},\mathbf{r}^g,\{\mathbf{t}^-_{\mathbf{k,j}}\}^g)}{\tau}) \\
&= \frac{1}{\tau\psi^g}(\psi^g\beta'(\mathbf{h},\mathbf{r}^g,\mathbf{h}^g) - e^{(\frac{\beta(\mathbf{h},\mathbf{r}^g,\mathbf{h}^g)}{\tau})}\beta'(\mathbf{h},\mathbf{r}^g,\mathbf{h}^g) \\
&- \sum_{(h,r^g,t^-_{k,j})\in\mathcal{N}_A} e^{(\frac{\beta(\mathbf{h},\mathbf{r}^g,\mathbf{t}^-_{\mathbf{k,j}})}{\tau})}\beta'(\mathbf{h},\mathbf{r}^g,\mathbf{t}^-_{\mathbf{k,j}}) \\
&- \sum_{(h,r^g,\{t^-_{k,j}\}^g)\in\mathcal{N}_A} e^{(\frac{\beta(\mathbf{h},\mathbf{r}^g,\{\mathbf{t}^-_{\mathbf{k,j}}\}^g)}{\tau})} \\
&\qquad\qquad \beta'(\mathbf{h},\mathbf{r}^g,\{\mathbf{t}^-_{\mathbf{k,j}}\}^g)) \\
&= \frac{1}{\tau\psi^g}(\sum_{n\in\mathcal{N}_A} e^{(\frac{\beta(n)}{\tau})}\beta'(\mathbf{h},\mathbf{r}^g,\mathbf{h}^g) \\
&\quad -[\sum_{n\in\mathcal{N}_A} e^{(\frac{\beta(n)}{\tau})}]\beta'(n))
\end{aligned}
\tag{24}
$$

A.3 Settings and Hyperparameters

The results of the baselines with * are taken from (Chen et al., 2022) while the results of MorsE and its variants (TransE/DistMult/ComplEx/RotatE) are reproduced with publicly available code and optimal values of hyperparameters.[5] The reproduced baselines and our VMCL are implemented in PyTorch and DGL with a single GeForce RTX 2080 GPU. With the same setting with baselines, walk times n_1 = 10, walk length n_2 = 5, and repeat times n_3 = 10, the number of negative entities U = 32(PT)/64(FT), the dimension D = 32, the fixed margin λ = 10. The optimizer is set to Adam with learning rate of 0.01/0.001, the epoch is set to 10/100 for pretraining/finetuning stages. For our VMCL, the dimension D^L is set to 128, $D^1 = ... = D^{l-1} = D$. The layers L of the encoder is selected from {1, 3, 5} and the best L is 3. The number U_a of negative meta-KGs is selected from {0, 1, 4, 8} and the best U_a is 4. The temperatures τ, τ_ζ is selected from {0.01, 0.05, 1} and the best τ is 0.05 and the best τ_ζ is 1. The loss weights η_1, η_2 are selected from {.002, .001, .0005} and the best η_1 is .001 and the best η_2 is 1.

[5] https://github.com/zjukg/MorsE

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
7

☒ A2. Did you discuss any potential risks of your work?
no potential risks of your work

☑ A3. Do the abstract and introduction summarize the paper's main claims?
Abstract, 1

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☒ Did you use or create scientific artifacts?

Left blank.

☐ B1. Did you cite the creators of artifacts you used?
No response.

☐ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
No response.

☐ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
No response.

☐ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
No response.

☐ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
No response.

☐ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
No response.

C ☑ Did you run computational experiments?

5

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
A.2

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
A.2

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
5.2

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
A.2

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?

Left blank.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
No response.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
No response.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
No response.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
No response.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
No response.