Contrastive Learning with Generated Representations for Inductive Knowledge Graph Embedding

Qian Li\textsuperscript{1,2}, Shafiq Joty\textsuperscript{2,3}, Daling Wang\textsuperscript{1}, Shi Feng\textsuperscript{1}, Yifei Zhang\textsuperscript{1} and Chengwei Qin\textsuperscript{2}

\textsuperscript{1} Northeastern University, China
\textsuperscript{2} Nanyang Technological University, Singapore
\textsuperscript{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 Contrastive Learning (CL) framework with graph guided Variational autoencoder on Meta-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.

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

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.
rules from the source KG and apply such rules to the target KG. GraIL (Teru et al., 2020) and CoM-PILE (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 Contrastive Learning (CL) framework with graph guided Variational autoencoder on Meta-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 (Veličković et al., 2019) and graph (Sun et al., 2020) levels. (Hassani and Ahmadi, 2020) and (Zhu et al., 2021) attempt to contrast multiple structural views of graphs. (Ahriabian et al., 2020), (Wang et al., 2022) and (Li et al., 2022) 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 \( G=(E, R) \), where \( E \) refers to the set of entities and \( R \) refers to the set of relations in the KG. A KG organizes entities and relations as triples of \((h, r, t)\), where \( h, t \in E \) respectively represent the head and tail entities, and \( r \in R \) represents the relation between \( h \) and \( t \). A group of inductive KGs includes at least a source KG \( G_S=(E_S, R_S) \) and a target KG \( G_T=(E_T, R_T) \) with the condition \( R_T \subseteq R_S \) and \( E_T \cap E_S = \emptyset \). The goal of representation learning for inductive KGs is to capture the structural patterns \( f \) from \( G_S \) and transfer it to \( G_T \). We expect \( f \) to be able to map entities in \( G_T \) into representations with the following property: the score of a true triple \((h, r, t) \in G_T \) 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 \( M_i \) is a subgraph sampled randomly.
from the source KG $G_S$. We re-label the id of all entities in $M_i$ in order to make the process label insensitive as our main interest is to capture the structural patterns. We extract multiple meta-KGs \{$M_j$\}_{j=0}^{[M]}$ from $G_S$ with $[M]$ being the number of meta-KGs sampled. For any two meta-KGs $M_i = (E_{M_i}, R_{M_i})$ and $M_j = (E_{M_j}, R_{M_j})$, we ensure that $E_{M_i} \cap E_{M_j} = \emptyset$, $R_{M_i} \subseteq R_S$, $R_{M_j} \subseteq R_S$.

### 3.3 Representation Notations

In the remaining, we use the following notations:

1. $R \in \mathbb{R}^{[R_S] \times D}$ denotes learnable relation representations which reserve internal semantics of relations; $r \in R$ is the representation for the relation $r$. $D$ is the dimension of representations.

2. $R_{\text{head}} \in \mathbb{R}^{[R_S] \times D}$ and $R_{\text{tail}} \in \mathbb{R}^{[R_S] \times D}$ are learnable relation embeddings to initialize the representation of the respective entities. For a relation $r$, $r_{\text{head}} \in R_{\text{head}}$ denotes the relation embedding for the head entities connected by $r$, and $r_{\text{tail}} \in R_{\text{tail}}$ denotes the relation embedding for the tail entities connected by $r$.

3. For a meta-KG $M_i = (E_{M_i}, R_{M_i})$, 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 $M_i$ is initialized with features of the involved relations in $M_i$:

$$h^0 = \frac{\sum_{r \in \mathcal{O}(h)} r_{\text{tail}} + \sum_{r \in \mathcal{I}(h)} r_{\text{head}}}{|\mathcal{O}(h)| + |\mathcal{I}(h)|}$$

(1)

where we use $\mathcal{O}(h)=\{r|\exists x, (h, r, x) \in M_i\}$, $\mathcal{I}(h) = \{r|\exists x, (x, r, h) \in M_i\}$ to denote outgoing and incoming relations for entity $h$.

4. For a meta-KG $M_i = (E_{M_i}, R_{M_i})$, we use $h \in \mathbb{R}^{[E_{M_i}] \times D}$ and $h^g \in \mathbb{R}^{[E_{M_i}] \times D}$ 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 $h$ and latent Gaussian distribution $\mathcal{N}(\mu_h, \sigma_h^2 \mathbf{I})$, and the decoder aims to generate representation $h^g$ 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 the entities with their multi-hop neighborhood structures,

$$h^l = \sigma\left(\frac{1}{|\mathcal{E}_{M_i}(h)|} \sum_{(t, r) \in \mathcal{E}_{M_i}(h)} W^l_t t^{l-1} + W^l_i h^{l-1}\right)$$

(2)

where $\mathcal{E}_{M_i}(h) = \{(t, r)| (t, r, h) \in M_i\}$ denotes the set of (head entity, relation)-pair of immediate incoming neighbor triples of entity $h$. $W^l_t \in \mathbb{R}^{D^l \times D^{l-1}}$ is the relation-specific transformation matrix for relation $r$ in the $l$-th layer, $W^l_i \in \mathbb{R}^{D^l \times D^{l-1}}$ 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. $\sigma$ is a ReLU activation function. For simplicity, let Eq. (2) be,

$$h^l_e = \sigma(\text{GNN}(\mathcal{E}_{M_i}(h), h^{l-1}_e, D^l, D^{l-1}))$$

(3)

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

$$\mu_h = W_\mu(h^L_e), \quad \sigma^2_h = W_\sigma(h^L_e)$$

(4)

where $W_\mu$ and $W_\sigma$ denote the weight matrices corresponding to the mean and variance of the (latent) Gaussian distribution, respectively.

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

$$z = \mu_h + \sigma_h \circ \epsilon$$

(6)
where $\epsilon \sim \mathcal{N}(0, \mathbf{I})$ (i.e., a standard normal distribution), $\circ$ denotes element-wise multiplication. We (re)construct $h^g$ from $z$ using another L-layer GNNs. The GNNs use dimension list $\{D_L, \ldots, D_0\}$ which is reversed with that in Eq. (3) and thus we denote this process from $L$-layer to 0-layer,
\[
    h^g_d = \sigma(\text{GNN}(\mathcal{E}_{\mathcal{M}_h}(h), h^{d+1}, D^1, D^{L+1}))
\]
where $h^g_d = z$. Let $h^g = h^0_d$ denote the generated representation of entity $h$.

We optimize the parameters with the combined loss of reconstruction and KL divergence,
\[
    L_G(h) = ||h - h^g||^2 + KL(N(\mu_h, \sigma^2_h), N(0, \mathbf{I}))
\]
(8)

here, $h = W_L(h^g)^2$, $W_L$ is a projection matrix to transform dimension from $D_L$ to $D$.

During inference, we encode a representation $h$ and generate a representation $h^g$ for entity $h$ in $\mathcal{M}_t$.

To distinguish, we use $\mathcal{M}_q^h$ to denote the generated meta-KG which has the same structure as $\mathcal{M}_t$ but different representations, i.e., the representation of entity $h$ in $\mathcal{M}_t$ is the encoded representation and that of entity $h^g$ in $\mathcal{M}_q^h$ is the corresponding generated representation $h^g$.

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 $h$ and generated representation $h^g$, 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,
\[
    L^I(h, r, t) = -\log \frac{e^{(\beta(h^I, r, t^I)/\tau)}}{\sum_{n \in \{P, N_t, N_r\}} e^{(\beta(n)/\tau)}}
\]
where we use a triple $P = (h, r, t)$ in the meta-KG $\mathcal{M}_t$ as a positive sample, and contrast it with negative samples $N_t$. $h^I = h + h^g$ and $t^I = t + t^g$ 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 $r$. $\tau$ is the temperature hyperparameter and $\beta(h^I, r, t^I)$ is a similarity score which can be any Knowledge Graph Embedding (KGE) methods, such as TransE ($-|h^I + r - t^I|$) (Bordes et al., 2013) or RotatE ($-|h^I \circ r - t^I|$) (Sun et al., 2019).

For a positive sample $P = (h, r, t)$, its negative samples can be generated as,
\[
    N_h = \{(h^j, r, t)\}_{j=0}^U, \ N_t = \{(h, r, t^j)\}_{j=0}^U\]
\[
    N_r = \{(h, r^j, t)\}_{j=0}^U
\]
(10)

where $N_h$ 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 $\mathcal{E}_{\mathcal{M}_r} - \mathcal{E}_{\mathcal{M}_h}(r, t)$ with $\mathcal{E}_{\mathcal{M}_h}(r, t)$ being an entity list of true head entities (i.e., $h^j \in \mathcal{E}_{\mathcal{M}_h}(r, t)$ iff $(h^j, r, t) \in \mathcal{M}_t$).

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

To analyse how this CL loss affects the transfer across entities, we perform gradient analysis. The gradients with respect to the head entities are:
\[
    -\frac{\partial L^I(h, r, t)}{\partial h^I} = \frac{1}{\tau \psi} \left( \sum_{n \in \{P, N_t, N_r\}} \frac{e^{(\beta(n)/\tau)}}{e^{(\beta(n)/\tau)}} \beta'(h^I, r, t^I) \right)
\]
\[
    -\left[ \sum_{n \in \{N_h, N_r\}} e^{(\beta(n)/\tau)} \right] \beta'(n)
\]
(12)

where $'$ denotes the partial derivative, $\psi$ is a normalization constant (see Appendix). The gradients of the head entity $h$ are closely related to positive entity $t$, negative entities $t^- \in N_t$ and relations $r^- \in N_r$. 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
entities $h^{-}_j \in N_h$ and relations $r^{-}_j \in N_r$ (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 $h^g$ in $M^g_i$ has the same distribution with respect to the encoded representation $h$ in $M_i$, we add a relation $r^g$ to create the link between the two meta-KGs $M_i$ and $M^g_i$. Specifically, for entity $h$ in $M_i$, we form the $h^g$ in $M^g_i$ as a positive sample and use the entities in other meta-KGs $\{M^g_k\}_{k=0}^{U_a}$ as negative samples. We design a CL objective that works across meta-KGs to simulate the transfer mode,

$$L^A(h, r^g, h^g) = -\log \frac{e^{(\beta(h, r^g, h^g)/\tau)}}{\sum_{n \in \{P(N_A)\}} e^{(\beta(n)/\tau)}}$$

$$N_A = \{(h, r^g, t^g_{k,j}), (h, r^g, \{t^g_{k,j}\})\}_{j=1}^{U_a} \bigcup \{(h, r^g, t^g_{k,j})\}_{j=0}^{U_a}$$

Similar to the positive sample, we form each negative sample as $(h, r^g, t^g_{k,j})$ by adding $r^g$ between $M_i$ and $M_k$, where $t^g_{k,j} \in M_k$ denotes the $j$-th negative entity sampled from $M_k$ and $t^g_{k,j}$ is its representation; $\{t^g_{k,j}\}^n \in M^g_k$ denotes the $j$-th negative entity sampled from $M^g_k$; and $\{t^g_{k,j}\}^g$ is its representation. We sample $U$ negative entities randomly from both $M_k$ and $M^g_k$, 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 ($-\frac{\partial L^A(h, r^g, h^g)}{\partial h}$) with respect to the head entities are,

$$\frac{1}{\tau\psi^g} \left( \sum_{n \in N_A} \frac{e^{(\beta(n))}}{\psi^g} \beta'(h, r^g, h^g) \right) - \left[ \sum_{n \in N_A} \frac{e^{(\beta(n))}}{\psi^g} \beta'(n) \right]$$

where $\psi^g$ is a normalization constant (see Appendix). According to the gradients, this CL can transfer positive features from $h^g \in M^g_i$ and negative features from $t^g_{k,j} \in M_k$, $\{t^g_{k,j}\}^g \in M^g_k$ to the object entity $h \in M_i$.

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.

### 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 ($L^\xi$):

$$L = \sum_{M_i \in M_S} \sum_{(h, r, t) \in M_i} \eta_1 (L^G + L^I + L^A) + \eta_2 L^\xi$$

where the weights $\eta_1$ and $\eta_2$ 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.

$$L^\xi(h, r, t) = -\log(\lambda + \beta(h^I, r, t^I)) - \sum_{n \in N_I} \psi^I \log(\vartheta - \beta(n))$$

where $h^I, r, t^I$ are the representations of $h, r, t$ (similar to Eq. (9)); $\psi^I$ is a self-adversarial negative sampling function which generates different weights for different negative samples according to their importance to the triple $(h, r, t)$. $\vartheta$ is the sigmoid function. $\lambda$ 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.

$$L = \sum_{(h, r, t) \in G_T} L^\xi(h, r, t)$$

### 5 Experiments

#### 5.1 Datasets

Table 1 shows the statistics of our datasets. We use 8 benchmark inductive KGs: F1, F2, F3, F4 and
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=0.028.

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=0.006.

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 tested 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 $M_i$ is sampled from a source KG $G_S$ with the following steps (Fig. 2(1)): (1) Sample an entity from the entity list $E_{G_S}$ of the source KG $G_S$, and put it into set $E_{M_i}$. (2) Sample an entity from $E_{M_i}$, walk randomly $n_1$ times with length-$n_2$, and put the walked entities into set $E_{M_i}$. (3) Repeat the above step (2) $n_3$ times and use entities in $E_{M_i}$ to induce a meta-KG $M_i$. (4) Anonymize the entities in $M_i$ by re-labeling the id of entities to be {1,2,...,$|E_{M_i}|$} 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 fine-tuned on the target KG with pretrained parameters from the source KG then uses the finetuned param-

---

<table>
<thead>
<tr>
<th>Model</th>
<th>F1 MRR</th>
<th>F1 H@1</th>
<th>F1 H@10</th>
<th>F2 MRR</th>
<th>F2 H@1</th>
<th>F2 H@10</th>
<th>F3 MRR</th>
<th>F3 H@1</th>
<th>F3 H@10</th>
<th>F4 MRR</th>
<th>F4 H@1</th>
<th>F4 H@10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Neural-LP *</td>
<td>46.13</td>
<td>40.21</td>
<td>52.92</td>
<td>51.85</td>
<td>45.68</td>
<td>58.94</td>
<td>48.70</td>
<td>44.09</td>
<td>52.90</td>
<td>49.54</td>
<td>44.12</td>
<td>55.88</td>
</tr>
<tr>
<td>DRUM *</td>
<td>47.55</td>
<td>42.71</td>
<td>52.92</td>
<td>52.78</td>
<td>47.49</td>
<td>58.73</td>
<td>73.68</td>
<td>65.95</td>
<td>87.69</td>
<td>74.19</td>
<td>67.21</td>
<td>85.60</td>
</tr>
<tr>
<td>RuleN *</td>
<td>45.97</td>
<td>41.46</td>
<td>49.76</td>
<td>69.08</td>
<td>62.13</td>
<td>77.82</td>
<td>70.35</td>
<td>60.25</td>
<td>82.83</td>
<td>70.60</td>
<td>60.99</td>
<td>89.29</td>
</tr>
<tr>
<td>GraIL *</td>
<td>48.56</td>
<td>40.00</td>
<td>64.15</td>
<td>62.54</td>
<td>52.20</td>
<td>81.80</td>
<td>70.35</td>
<td>60.25</td>
<td>82.83</td>
<td>70.60</td>
<td>60.99</td>
<td>89.29</td>
</tr>
<tr>
<td>CoMPILE *</td>
<td>-</td>
<td>-</td>
<td>67.64</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Model</th>
<th>N1 MRR</th>
<th>N1 H@1</th>
<th>N1 H@10</th>
<th>N2 MRR</th>
<th>N2 H@1</th>
<th>N2 H@10</th>
<th>N3 MRR</th>
<th>N3 H@1</th>
<th>N3 H@10</th>
<th>N4 MRR</th>
<th>N4 H@1</th>
<th>N4 H@10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Neural-LP *</td>
<td>35.71</td>
<td>29.13</td>
<td>40.78</td>
<td>64.68</td>
<td>55.45</td>
<td>78.73</td>
<td>69.93</td>
<td>60.46</td>
<td>82.71</td>
<td>68.21</td>
<td>59.27</td>
<td>80.58</td>
</tr>
<tr>
<td>DRUM *</td>
<td>18.89</td>
<td>14.08</td>
<td>19.42</td>
<td>66.44</td>
<td>58.55</td>
<td>78.55</td>
<td>72.28</td>
<td>64.29</td>
<td>82.71</td>
<td>70.42</td>
<td>63.09</td>
<td>80.58</td>
</tr>
<tr>
<td>RuleN *</td>
<td>46.35</td>
<td>39.00</td>
<td>53.50</td>
<td>70.80</td>
<td>63.08</td>
<td>81.75</td>
<td>68.76</td>
<td>61.99</td>
<td>77.26</td>
<td>56.31</td>
<td>51.44</td>
<td>61.35</td>
</tr>
<tr>
<td>GraIL *</td>
<td>52.04</td>
<td>46.50</td>
<td>59.50</td>
<td>72.92</td>
<td>63.08</td>
<td>93.25</td>
<td>74.37</td>
<td>64.22</td>
<td>91.41</td>
<td>62.98</td>
<td>57.27</td>
<td>73.19</td>
</tr>
<tr>
<td>CoMPILE *</td>
<td>-</td>
<td>-</td>
<td>58.38</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Model</th>
<th>N1 MRR</th>
<th>N1 H@1</th>
<th>N1 H@10</th>
<th>N2 MRR</th>
<th>N2 H@1</th>
<th>N2 H@10</th>
<th>N3 MRR</th>
<th>N3 H@1</th>
<th>N3 H@10</th>
<th>N4 MRR</th>
<th>N4 H@1</th>
<th>N4 H@10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Neural-LP *</td>
<td>35.71</td>
<td>29.13</td>
<td>40.78</td>
<td>64.68</td>
<td>55.45</td>
<td>78.73</td>
<td>69.93</td>
<td>60.46</td>
<td>82.71</td>
<td>68.21</td>
<td>59.27</td>
<td>80.58</td>
</tr>
<tr>
<td>DRUM *</td>
<td>18.89</td>
<td>14.08</td>
<td>19.42</td>
<td>66.44</td>
<td>58.55</td>
<td>78.55</td>
<td>72.28</td>
<td>64.29</td>
<td>82.71</td>
<td>70.42</td>
<td>63.09</td>
<td>80.58</td>
</tr>
<tr>
<td>RuleN *</td>
<td>46.35</td>
<td>39.00</td>
<td>53.50</td>
<td>70.80</td>
<td>63.08</td>
<td>81.75</td>
<td>68.76</td>
<td>61.99</td>
<td>77.26</td>
<td>56.31</td>
<td>51.44</td>
<td>61.35</td>
</tr>
<tr>
<td>GraIL *</td>
<td>52.04</td>
<td>46.50</td>
<td>59.50</td>
<td>72.92</td>
<td>63.08</td>
<td>93.25</td>
<td>74.37</td>
<td>64.22</td>
<td>91.41</td>
<td>62.98</td>
<td>57.27</td>
<td>73.19</td>
</tr>
<tr>
<td>CoMPILE *</td>
<td>-</td>
<td>-</td>
<td>58.38</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

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
tery 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). 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.

- 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
Table 4: Results of different KGE models.

<table>
<thead>
<tr>
<th>Model</th>
<th>F1</th>
<th>F2</th>
<th>F3</th>
<th>F4</th>
<th>MRR</th>
<th>H@1</th>
<th>H@10</th>
</tr>
</thead>
<tbody>
<tr>
<td>TransE</td>
<td>MorsE</td>
<td>67.49</td>
<td>55.90</td>
<td>89.73</td>
<td>79.22</td>
<td>69.58</td>
<td>96.37</td>
</tr>
<tr>
<td>VMCL</td>
<td>70.80</td>
<td>59.64</td>
<td>91.81</td>
<td>80.47</td>
<td>71.45</td>
<td>96.00</td>
<td></td>
</tr>
<tr>
<td>RotatE</td>
<td>MorsE</td>
<td>69.61</td>
<td>58.82</td>
<td>90.39</td>
<td>80.38</td>
<td>71.34</td>
<td>96.33</td>
</tr>
<tr>
<td>VMCL</td>
<td>72.68</td>
<td>62.61</td>
<td>92.47</td>
<td>81.66</td>
<td>73.05</td>
<td>97.29</td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Model</th>
<th>F1</th>
<th>F2</th>
<th>F3</th>
<th>F4</th>
<th>MRR</th>
<th>H@1</th>
<th>H@10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full</td>
<td>62.61</td>
<td>73.05</td>
<td>69.75</td>
<td>71.68</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>w/o G</td>
<td>52.05</td>
<td>62.84</td>
<td>58.83</td>
<td>59.47</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>w/o GFNN</td>
<td>51.96</td>
<td>65.86</td>
<td>60.48</td>
<td>61.25</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>w/o CL</td>
<td>58.76</td>
<td>72.75</td>
<td>69.69</td>
<td>71.98</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>w/o CLE</td>
<td>61.19</td>
<td>71.70</td>
<td>70.06</td>
<td>72.79</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>w/o CLM</td>
<td>58.93</td>
<td>72.93</td>
<td>70.46</td>
<td>71.86</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

• Ablation Study We first investigate the ablation study (Table 5) of representation generation by removing or 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 $M_i=(E_{M_i}, R_{M_i})$ with $|M_i|$ as the number of triples. A GNN (Eq. (2)) requires $O(|M_i| \cdot D^2)$, where we use $D$ represent the maximum dimension in $\{D^0, ..., D^L\}$. So the representation generation module requires $O(2^L \cdot |M_i| \cdot D^2)$. A CL objective (Eq. (9) or Eq. (13)) requires $O((N_I + 1) \cdot 2L \cdot |M_i| \cdot D^2)$ with $N$ as the maximum of $N_I$ and $N_A$. So the two CL objectives for meta-KG $M_i$ require $O(2|M_i| \cdot (N_I + 1) \cdot 2L \cdot |M_i| \cdot D^2)$, omitted and abbreviated as $O(|M_i|^2 \cdot N \cdot D^2)$. The time complexity of all meta-KGs ($|M|$) is $O(|M|\cdot |M_i|^2 \cdot N \cdot D^2)$. Because the number of triples in meta-KG $|M_i|$ 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


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.


A Appendix

A.1 Ablation Study in N1-N4

<table>
<thead>
<tr>
<th>Comp</th>
<th>N1</th>
<th>N2</th>
<th>N3</th>
<th>N4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full</td>
<td>48.16</td>
<td>76.32</td>
<td>79.65</td>
<td>72.86</td>
</tr>
<tr>
<td>w/o G</td>
<td>48.10</td>
<td>55.40</td>
<td>67.35</td>
<td>58.62</td>
</tr>
<tr>
<td>w/o CLNN</td>
<td>49.60</td>
<td>63.26</td>
<td>69.88</td>
<td>56.82</td>
</tr>
<tr>
<td>w/o CL</td>
<td>47.24</td>
<td>77.36</td>
<td>80.71</td>
<td>73.14</td>
</tr>
<tr>
<td>w/o CLM</td>
<td>47.88</td>
<td>75.92</td>
<td>80.20</td>
<td>71.90</td>
</tr>
</tbody>
</table>

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

\[
\frac{\partial L^I(h,r,t)}{\partial h} = \beta(h',r,t') - \log \sum_{\beta(h',r,t')/\tau} e^{(\beta(h',r,t')/\tau)} - \log \sum_{\beta(h',r,t')/\tau} e^{(\beta(h',r,t')/\tau)}
\]

\[
\frac{\partial L^I(h,r,t)}{\partial r} = \beta(h',r,t') - \log \sum_{\beta(h',r,t')/\tau} e^{(\beta(h',r,t')/\tau)} - \log \sum_{\beta(h',r,t')/\tau} e^{(\beta(h',r,t')/\tau)} + \sum_{(h,r\tau,j)\in N_h} e^{(\beta(h',r,t')/\tau)}
\]

\[
\frac{\partial L^I(h,r,t)}{\partial t} = \beta(h',r,t') - \log \sum_{\beta(h',r,t')/\tau} e^{(\beta(h',r,t')/\tau)} + \sum_{(h,r\tau,j)\in N_h} e^{(\beta(h',r,t')/\tau)} + \sum_{(h,r\tau,j)\in N_h} e^{(\beta(h',r,t')/\tau)}
\]

\[
\frac{1}{\tau} \psi = e^{(\beta(h',r,t')/\tau)} - \frac{\sum_{\beta(h',r,t')/\tau} e^{(\beta(h',r,t')/\tau)}}{\sum_{\beta(h',r,t')/\tau} e^{(\beta(h',r,t')/\tau)}} - \sum_{(h,r\tau,j)\in N_h} e^{(\beta(h',r,t')/\tau)} - \frac{\sum_{\beta(h',r,t')/\tau} e^{(\beta(h',r,t')/\tau)}}{\sum_{\beta(h',r,t')/\tau} e^{(\beta(h',r,t')/\tau)}}
\]

\[
\frac{1}{\tau} \psi^g = e^{(\beta(h',r,t')/\tau)} - \frac{\sum_{\beta(h',r,t')/\tau} e^{(\beta(h',r,t')/\tau)}}{\sum_{\beta(h',r,t')/\tau} e^{(\beta(h',r,t')/\tau)}} - \sum_{(h,r\tau,j)\in N_h} e^{(\beta(h',r,t')/\tau)} - \frac{\sum_{\beta(h',r,t')/\tau} e^{(\beta(h',r,t')/\tau)}}{\sum_{\beta(h',r,t')/\tau} e^{(\beta(h',r,t')/\tau)}}
\]

\[
\frac{1}{\tau} \psi^g = e^{(\beta(h',r,t')/\tau)} - \frac{\sum_{\beta(h',r,t')/\tau} e^{(\beta(h',r,t')/\tau)}}{\sum_{\beta(h',r,t')/\tau} e^{(\beta(h',r,t')/\tau)}} - \sum_{(h,r\tau,j)\in N_h} e^{(\beta(h',r,t')/\tau)} - \frac{\sum_{\beta(h',r,t')/\tau} e^{(\beta(h',r,t')/\tau)}}{\sum_{\beta(h',r,t')/\tau} e^{(\beta(h',r,t')/\tau)}}
\]

14284
\[ - \frac{\partial L^A(h, r^g, h^g)}{\partial h} \]
\[ = \frac{\partial}{\partial h} \left( \log \sum_{n \in \{ P, N \}} e^{\beta(n) / \tau} \right) \]
\[ = \frac{\partial}{\partial h} \left( \frac{\beta(h, r^g, h^g)}{\tau} \right) - \log \sum_{n \in \{ P, N \}} e^{\beta(n) / \tau} \]
\[ = \frac{\partial}{\partial h} \left( \frac{\beta(h, r^g, h^g)}{\tau} \right) - \log \left( \sum_{n \in \{ P, N \}} e^{\beta(n) / \tau} \right) \]
\[ + \sum_{(h, r^g, t_{k,j}) \in N^A} e^{\beta(h, r^g, t_{k,j}) / \tau} \]
\[ + \sum_{(h, r^g, \{t_{k,j}^g\}) \in N^A} e^{\left( \frac{\beta(h, r^g, t_{k,j}^g)}{\tau} \right)} \]
\[ = \frac{\beta'(h, h^g)}{\tau} - \frac{1}{\tau^g} \left( \frac{\beta(h, r^g, h^g)}{\tau} \right)^{\beta'(h, r^g, h^g)} \]
\[ + \sum_{(h, r^g, t_{k,j}) \in N^A} e^{\beta(h, r^g, t_{k,j}) / \tau} \]
\[ + \frac{1}{\tau^g} \left( \frac{\beta(h, r^g, h^g)}{\tau} \right)^{\beta'(h, r^g, h^g)} \]
\[ - \sum_{(h, r^g, t_{k,j}) \in N^A} e^{\beta(h, r^g, t_{k,j}) / \tau} \]
\[ - \sum_{(h, r^g, \{t_{k,j}^g\}) \in N^A} e^{\left( \frac{\beta(h, r^g, t_{k,j}^g)}{\tau} \right)} \]
\[ = \frac{1}{\tau^g} \left( \sum_{n \in N^A} e^{\left( \frac{\beta(n)}{\tau} \right)} \right) \beta'(h, r^g, h^g) \]
\[ + \sum_{n \in N^A} e^{\left( \frac{\beta(n)}{\tau} \right)} \beta'(n, h^g) \]
\[ - \sum_{n \in N^A} e^{\left( \frac{\beta(n)}{\tau} \right)} \beta'(h, n, h^g) \]
\[ \text{(24)} \]

### 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 \(U = 32\) for pretraining/finetuning stages. For our VMCL, the dimension \(D^L\) is set to 128, \(D^1 = \ldots = D^{l-1} = D\). The layers \(L\) of the encoder is selected from \{1, 3, 5\} and the best \(L\) is 3. The number \(U_n\) of negative meta-KGs is selected from \{0, 1, 4, 8\} and the best \(U_n\) is 4. The temperatures \(\tau, \tau_c\) is selected from \{0.01, 0.05, 1\} and the best \(\tau\) is 0.05 and the best \(\tau_c\) is 1. The loss weights \(\eta_1, \eta_2\) are selected from \{.002, .001, .0005\} and the best \(\eta_1\) is .001 and the best \(\eta_2\) 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, I

✗ 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?

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.