Alleviating Sparsity of Open Knowledge Graphs with Ternary Contrastive Learning

Anonymous ACL submission

Abstract

Sparsity of formal knowledge and roughness of non-ontological construction methods make sparsity problem particularly prominent in Open Knowledge Graphs (OpenKGs). Sparse links make few-shot entities unable to learn potential features. We hypothesize that negative samples could help sparse links highlight discriminative features. However, existing contrastive learning in Graphs model binary objects, none has studied contrastive learning to model ternary pattern in any KGs. In this paper, we propose a Ternary Contrastive Learning (TernaryCL) to alleviate the sparsity of OpenKGs. TernaryCL designs (1) *Contrastive Entity* and (2) *Contrastive Relation* to mine ternary discriminative features by both negative entities and relations. (3) *Contrastive Self* constructs a *self* positive sample to give zero-shot and few-shot entities chances to learn discriminative features. (4) *Contrastive Fusion* aggregates graph features by extending the pattern from 1-to-1 to 1-to-N. Extensive experiments on benchmarks show the superiority of TernaryCL over state-of-the-art models.

1 Introduction

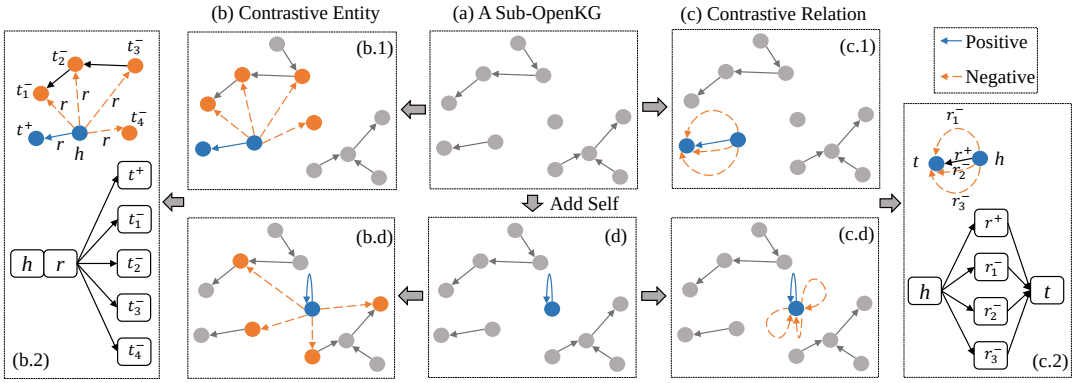
Knowledge Graphs (KGs) structure objective facts in the form of ("head entity", "relation", "tail entity") triples. KGs can be of diverse types, such as CuratedKGs (Li et al., 2021) whose construction relies on specification of ontology schema, CommonsenseKGs (Malaviya et al., 2020) whose entities are free-form text, and OpenKGs (Chandrahas and Talukdar, 2021) which take noun phrases as entities and relation phrases as relations. Representation learning (Bordes et al., 2013; Dettmers et al., 2018) of KGs aims to learn implicit embeddings of entities and relations, and has become an indispensable step in the application of KGs to downstream tasks (Gupta et al., 2019; Broscheit et al., 2020). Because of sparsity of formal grammatical knowledge, a common challenge in representation learning of KGs is the sparsity problem, where a large portion of entities have few- or zero-shot links.

Our work in this paper concerns OpenKGs, which are extracted from text corpora with OpenIE tools (Fader et al., 2011; Gashteovski et al., 2019), and generally they do not rely on specification of ontology schema. Roughness of non-ontological construction methods makes sparsity problem particularly prominent in OpenKGs. According to our statistics of standard OpenKGs, the degree of 55% entities in ReVerb20K and 89% entities in ReVerb45K is less than 3. Due to fewer training chances caused by sparse links, few-shot and zero-shot entities are not well trained, resulting in poor generalization performance. Although existing representation learning models have achieved promising performance (Dettmers et al., 2018; Gupta et al., 2019; Chandrahas and Talukdar, 2021), they do not effectively tackle the sparsity problem. This motivates us to develop a more effective method to alleviate the sparsity of OpenKGs.

Being popular in self-supervised representation learning, *contrastive learning* aims to learn discriminative features by introducing negative samples in contrast with positive samples (He et al., 2020; Gao et al., 2021; Zhu et al., 2021). These negative samples can enrich the understanding of positive samples in the form of a negative feedback. We hypothesize that negative samples could help existing sparse links to learn discriminative features.

In any KG, links imply ternary propagation patterns, where entities propagate to multi-neighbor-entities through multi-relations. However, traditional contrastive learning in Graphs (Velickovic et al., 2019) only models binary objects. To the best of our knowledge, none has studied contrastive learning to study ternary patterns in KGs.

In this work, we propose a Ternary Contrastive Learning (TernaryCL) framework to alleviate the sparsity of OpenKGs. We explore four key ideas: (1) *Contrastive Entity* to learn discriminative fea-



(d) Contrastive Self, where b.d shows its contrastive entities, c.d shows its contrastive relations.

Figure 1: The overall framework of our proposed TernaryCL model for alleviating sparsity of OpenKGs. (a) A given subgraph of an OpenKG. (b) Contrastive Entity generates negative entities (yellow) and contrasts them with a positive entity (blue). (c) Contrastive Relation generates negative relations (yellow) and contrasts them with a positive relation (blue). (d) Contrastive Self constructs a positive sample by adding a *self* relation (blue) to the entity (blue), generates negative entities (yellow) in (b.d) and negative relations (yellow) in (c.d), and contrasts them with the *self* positive sample.

tures of different entities under the same (head entity, relation)-pair, which alleviates sparsity by considering negative entities (Fig. 1b). (2) *Contrastive Relation* to learn discriminative features of different relations under the same (head entity, tail entity)-pair, which alleviates the sparsity by considering negative relations (Fig. 1c). (3) *Contrastive Self* to construct a positive sample by adding a *self* relation, then generate negative entities and relations to contrast with the self positive sample, which gives zero-shot and few-shot entities one or more chances to learn discriminative representations (Fig. 1d). (4) *Contrastive Fusion* to extend the ternary propagation pattern from the above 1-to-1 to 1-to-N: a head entity propagates to multiple tail entities through multiple relations. To gain insights into TernaryCL, we analyze the gradients it receives from the above components.

TernaryCL can learn effective embeddings from the KG itself without relying on pretrained language models or side information. This low dependency makes it easy to apply to downstream tasks. Extensive evaluation of overall performance, sparsity granularity, few- and zero-shot types and visualizations show that TernaryCL can significantly outperform state-of-the-art baselines.

In summary, our key contributions are:

- To the best of our knowledge, this is the first work to introduce contrastive learning for representation learning over knowledge graphs.
- We propose a ternary contrastive learning model to learn representations from complex propagation patterns of OpenKGs. It subtly generates negative samples from the perspective of entities and relations, and uses contrastive learning to incorporate structural information. It does not rely on external resources making it easy to scale and to apply to downstream tasks.
- We present *Contrastive Self* to generate *self* positive samples, which solves the problem of learning for zero-shot entities. This gives zero-shot or few-shot entities one or more chances to learn discriminative representations.
- We perform extensive experiments to show the superiority of our method over state-of-the-art baselines. We release our code at `ACLARRFeb`.

2 Related Work

2.1 Open Knowledge Graphs (OpenKGs)

OpenKGs represent factual knowledge in structured forms, especifically, as triples of *h*ead-*r*elation-*t*ail or (h, r, t). They are extracted with OpenIE tools (Fader et al., 2011; Gashteovski et al., 2019), and generally do not rely on specification of ontology. Although OpenKGs have the advantage that they can be easily bootstrapped to new domains, because of sparsity of formal grammatical knowledge and roughness of non-ontological construction, many relevant facts are often missing from such OpenKGs, This makes them difficult to be used effectively in downstream knowledge related tasks (Chandrahas and Talukdar, 2021).

Representation learning of KGs devotes to learning informative features of entities and relations, which can be used for other tasks such as predicting missing relations. General representation learning models over focus on inducing structural features with linear (Bordes et al., 2013), bilinear (Wang et al., 2014; Lin et al., 2015), complex (Yang

et al., 2015; Trouillon et al., 2016) or convolutional (Dettmers et al., 2018; Nguyen et al., 2018) operations, while OpenKG-specific models enhance the embeddings with side information (Gupta et al., 2019) and pretrained language models (Chandrahas and Talukdar, 2021). However, these methods are still limited in alleviating the sparsity issue. We propose a contrastive learning method that is more effective and does not rely on external resources.

2.2 Contrastive Learning

Contrastive learning aims to learn effective representation by pulling semantically close neighbors together and pushing apart non-neighbors (Hadsell et al., 2006; Gao et al., 2021), which has achieved great success in vision (He et al., 2020), text (Gao et al., 2021) and graph (Zhu et al., 2021). A number of graph representation learning methods attempt to leverage a contrastive learning loss at node (Velickovic et al., 2019), graph (Sun et al., 2020) and multi-view levels (Hassani and Ahmadi, 2020; Zhu et al., 2021). However, traditional contrastive learning in graphs (Velickovic et al., 2019) only model binary objects. In contrast, we propose contrastive learning to study ternary patterns in KGs.

3 Preliminaries

• Let $\mathcal{G} = (\mathcal{E}, \mathcal{R})$ be an **OpenKG** and (h, r, t) be a triple in $\mathcal{G}$, where $h, t \in \mathcal{E}$ represent head and tail entities, and $r \in \mathcal{R}$ represents the relation between them. Entities and relations are non-empty word sequences; $w_h = \{w_{h,i}\}_{i=1}^{|w_h|}$ and $w_r = \{w_{r,i}\}_{i=1}^{|w_r|}$ respectively represent word sequences of entity h and relation r. Representations of entities and relations are denoted as $\mathbf{E} \in \mathbb{R}^{|\mathcal{E}| \times D}$ and $\mathbf{R} \in \mathbb{R}^{|\mathcal{R}| \times D}$ respectively, with D being embedding dimension. We will use $\mathbf{h} \in \mathbf{E}$ and $\mathbf{r} \in \mathbf{R}$ to denote the embeddings of entity h and relation r, respectively.

• **Link prediction** of OpenKGs is the task of predicting answer entities for two questions: (1) predicting the tail $Q_t = (h, r, ?)$ and (2) predicting the head $Q_h = (?, r, t)$. For each such question, the possible true answer entities can be one or more, because there could be multiple entities with the same meaning but different surface textual forms in an OpenKG (Broscheit et al., 2020). For example, for question $Q_t =$ ("NBC-TV", "has office in", ?), we expect all answers from the set of entities {"New York", "NYC", "New York City"}.

• A **zero-shot** entity (relation) is an entity (relation) without links in the KG. A **few-shot** entity (relation) is an entity (relation) with few links in the KG, e.g. an one-shot entity has only one link. A zero-shot (few-shot) triple is a triple that contain at least one zero-shot (few-shot) entity or relation.

4 Proposed TernaryCL Model

The overall framework of TernaryCL for alleviating sparsity of OpenKGs is shown in Fig. 1. In the following subsections, we first introduce a simple *Ternary Similarity* function to compute a similarity score for each (h, r, t) triple by considering both textual and structural information (§4.1). We present *Contrastive Entity* in §4.2 to learn embeddings of different entities with the same (h, r)-pair, followed by *Contrastive Relation* in §4.3 to learn embeddings of different relations with the same (h, t)-pair. We describe *Contrastive Self* in §4.4 that constructs a positive sample $(h, \textit{self}, h^+)$ and contrasts it with negative samples to give zero- and few-shot entities chances to learn discriminative features. *Contrastive Fusion* in §4.5 extends propagation patterns from the above 1-1 to 1-N. Finally, the training procedure is described in §4.6.

4.1 Ternary Similarity

For a triple $(h, r, t) \in \mathcal{G}$, word sequence of head entity h is $w_h = \{w_{h,i}\}_{i=1}^{|w_h|}$, of relation r is $w_r = \{w_{r,i}\}_{i=1}^{|w_r|}$, and of tail entity t is $w_t = \{w_{t,i}\}_{i=1}^{|w_t|}$. We encode each of these sequences with a text encoder (Enc) such as BiGRU (Cho et al., 2014) and BERT (Devlin et al., 2019).

$$\mathbf{i}^{\mathbf{w}} = \text{Enc}(w_i) \text{ for } w_i \in \{w_h, w_r, w_t\} \quad (1)$$

This encoding yields the textual embeddings of h, r and t as $\mathbf{h}^{\mathbf{w}}$, $\mathbf{r}^{\mathbf{w}}$, and $\mathbf{t}^{\mathbf{w}}$, respectively.[1]

Then, we focus on exploiting potential connections between entities and relations. We use a two-dimensional convolutional network (Dettmers et al., 2018) to learn potential connections between a head entity h and a relation r as follows:

$$\varphi(h, r) = \rho\big(\text{Linear}(\rho(\text{Conv2d}_{\omega}([\widehat{\mathbf{h}}; \widehat{\mathbf{r}}])))\big) \quad (2)$$

where ρ represents a ReLU activation, and $\widehat{\mathbf{h}}$ and $\widehat{\mathbf{r}}$ denote a reshaping of $[\mathbf{h} + \mathbf{h}^{\mathbf{w}}]$ and $\mathbf{r}^{\mathbf{w}}$ respectively, with $\mathbf{h} \in \mathbf{E}$.[2] Specifically, the reshaping operation converts a vector $\mathbf{v} \in \mathbb{R}^D$ from one-dimension to two-dimensions $\mathbf{v} \in \mathbb{R}^{D_1 \times D_2}$, where

[1] For BiGRU, we take the concatenation of last states in the forward and backward directions as the sequence representation. For BERT, we take the [CLS] representation.

[2] For relations, adding $\mathbf{r} \in \mathbf{R}$ with the textual embedding $\mathbf{r}^{\mathbf{w}}$ worsen the performance.

$D = D_1 * D_2$, and $[\widehat{\mathbf{h}};\widehat{\mathbf{r}}] \in \mathbb{R}^{(D_1*2)\times D_2}$ represents the concatenation of the reshaped embeddings of $\widehat{\mathbf{h}}$ and $\widehat{\mathbf{r}}$. The Conv2d$_\omega$ symbol denotes a two-dimensional convolutional layer with filters ω. This layer returns a feature map tensor $\mathcal{C} \in \mathbb{R}^{C_1\times C_2\times C_3}$, where C_1 is the number of feature maps of dimensions $C_2 \times C_3$. $\mathcal{C}$ is then reshaped into $\mathbb{R}^{C_1*C_2*C_3}$, and projected to $\mathbb{R}^D$ with a Linear layer. Through the convolution module, potential embeddings of entity h and relation r are jointly encapsulated.

Finally, we compute a similarity score for each triple (h, r, t) with a dot product similarity function:

$$\beta(h, r, t) = \varphi(h, r) \cdot \mathbf{t} \tag{3}$$

where $\mathbf{t} \in \mathbf{E}$. When predicting the head h based on pair (r, t), we reverse the relation by adding a special symbol, and obtain a new triple (t, r_{rev}, h). The similarity score for (t, r_{rev}, h) is computed in a similar fashion as above following Eq. (2)-Eq. (3).

4.2 Contrastive Entity

Contrastive Entity (Fig. 1b) alleviates sparsity of OpenKGs from the perspective of *nagative* entities, and induces discriminative features of different entities with the same (h, r)-pair . The contrastive score for a triplet $p_e = (h, r, t^+)$ is:

$$S(h, r, t^+) = -\log \frac{e^{(\beta(h,r,t^+)/\tau)}}{\sum_{n\in\{p_e,\mathcal{N}_e\}} e^{(\beta(n)/\tau)}} \tag{4}$$

where τ is a temperature hyperparameter, $\beta(.)$ is the similarity score as in Eq. (3), and $p_e = (h, r, t^+)$ is a true triple in the OpenKG. $\mathcal{N}_e = \{(h, r, t_j^-)\}_{j=1}^{|\mathcal{N}_e|}$ is a set of negative samples, where a negative entity t_j^- is selected strategically from an entity list defined by: $\mathcal{E} - \mathcal{E}(h, r)$ with $\mathcal{E}(h, r)$ being the entity list of true answers (tail entities), that is, $t_i \in \mathcal{E}(h, r)$ if the triple $(h, r, t_i) \in \mathcal{G}$.

To analyse how this contrastive loss affects the learning, we perform gradient analysis. The gradient that the head entities get $(-\frac{\partial S(h,r,t^+)}{\partial \mathbf{h}})$ is as follows (see Appendix for a derivation):

$$\frac{\varphi'(h,r)}{\tau A}\Big([\sum_{(h,r,t_j^-)\in\mathcal{N}_e} e^{(\frac{\varphi(h,r)\cdot\mathbf{t_j^-}}{\tau})}]\mathbf{t}^+ - \sum_{(h,r,t_j^-)\in\mathcal{N}_e} [e^{(\frac{\varphi(h,r)\cdot\mathbf{t_j^-}}{\tau})}\mathbf{t_j^-}]\Big) \tag{5}$$

where A is a normalization constant. This is consistent with our assumption, where positive entity t^+ gives positive feedback while negative entities t_j^- give negative feedback. The Gradient for a relations r has a similar form as Eq. (5).

Similarly, we can derive the gradients for the positive t^+ and negative tail entities t_j^- as:

$$-\frac{\partial S(h,r,t^+)}{\partial \mathbf{t}^+} = \frac{\varphi(h,r)}{\tau A} \sum_{(h,r,t_j^-)\in\mathcal{N}_e} e^{(\frac{\varphi(h,r)\cdot\mathbf{t_j^-}}{\tau})} \tag{6}$$

$$-\frac{\partial S(h,r,t^+)}{\partial \mathbf{t_j^-}} = -\frac{\varphi(h,r)}{\tau A} e^{(\frac{\varphi(h,r)\cdot\mathbf{t_j^-}}{\tau})} \tag{7}$$

For a few-shot entity t, when it appears as a positive (Eq. (6)) or negative (Eq. (7)) sample headed by entity h of high degree, it gets sufficient gradients to learn informative representations. Similarly, through Eq. (7), the parameters of a zero-shot entity get updated when it appears as a negative sample with a (h, r)-pair. As discussed later in §4.4, with Contrastive Self, zero-shot entities get updated with their own contrastive losses. Overall, through the negative samples, training of few-shot, zero-shot and other entities (with many links) gets more balanced, while with existing approaches, zero-shot entities generally do not get trained as they are disconnected from the rest of the OpenKG.

Negative sampling strategy For sampling negative entities, we calculate the degree of each entity in the OpenKG, and normalize them into $[0, 1]$ to get a degree distribution, P_{deg}. We design three negative sampling strategies based on P_{deg}. (a) *ManyP* gives higher chances to entities with many links by sampling directly from P_{deg} with the hope to transfer sufficient knowledge to other entities. (b) *FewP* gives higher chances to few- or zero-shot entities by sampling from $(1 - P_{\text{deg}})$, so as to increase their exposure to get more training updates. (c) *Uniform* samples uniformly without considering the degree.

4.3 Contrastive Relation

Entities in a KG propagate information to neighboring entities through one or more relations, so the features of relations are as important as that of entities. Contrastive Relation (Fig. 1c) alleviates the sparsity from the perspective of *negative* relations, and captures potential features of different relations with the same (h, t)-pair. The contrastive score for a positive relation $p_r = (h, r^+, t)$ is:

$$S(h, r^+, t) = -\log \frac{e^{(\beta(h,r^+,t)/\tau)}}{\sum_{n\in\{p_r,\mathcal{N}_r\}} e^{(\beta(n)/\tau)}} \tag{8}$$

where p_r is a true triple in the OpenKG, and $\mathcal{N}_r = \{(h, r_j^-, t)\}_{j=1}^{|\mathcal{N}_r|}$ is a set of negative samples, where negative relation r_j^- is sampled strategically from a relation list $\mathcal{R} - \mathcal{R}(h, t)$ with $\mathcal{R}(h, t)$ being a relation list that satisfies the condition: $r_i \in \mathcal{R}(h, t)$ if the triple $(h, r_i, t) \in \mathcal{G}$. The gradients have the same form as above (see Appendix). In particular, the gradients that a tail entity gets is:

$$-\frac{\partial S(h,r^+,t)}{\partial \mathbf{t}} = \frac{1}{\tau B}\Big([\sum_{(h,r_j^-,t)\in\mathcal{N}_r} e^{(\frac{\varphi(h,r_j^-)\cdot\mathbf{t}}{\tau})}]\varphi(h,r^+) - \sum_{(h,r_j^-,t)\in\mathcal{N}_r}[e^{(\frac{\varphi(h,r_j^-)\cdot\mathbf{t}}{\tau})}\varphi(h,r_j^-)]\Big) \quad (9)$$

Different from Eq. (6), tail entities ($t = t^+$) get contrastive gradients, because the signals from the positive (r^+) and negative (r_j^-) samples are in opposite direction in Eq. (9). We follow the same *ManyP*, *FewP* and *Uniform* sampling strategies as in §4.2 for sampling a negative relation.

4.4 Contrastive Self

As described in §4.2 and §4.3, parameters of a few-shot entity get updated with contrastive gradients when it acts as a head (Eq. (5)) or tail entity (Eq. (9)). Parameters of a zero-shot entity are also updated when it appears as a negative entity (Eq. (7)), but not from the perspective of contrastive. This means zero-shot entities have no chance to learn discriminative representations, because they have no links with the rest of OpenKG.

In view of this, we propose to construct a positive sample $p_s = (h, \textit{self}, h^+)$ by adding a *Self* relation (Fig. 1d), where h and h^+ are the same entity but with different embeddings: the embedding of h is $[\mathbf{h} + \mathbf{h^w}]$ and the embedding of h^+ is $\mathbf{h} \in \mathbf{E}$. For such a positive sample, negative samples $\mathcal{N}_e = \{(h, \textit{self}, h_j^-)\}_{j=1}^{|\mathcal{N}_e|}$ are generated with negative entities by selecting h_j^- strategically from an entity list $\mathcal{E} - h$ (Fig. 1b.d). Similarly, negative relation samples $\mathcal{N}_r = \{(h, \textit{self}_j^-, h^+)\}_{j=1}^{|\mathcal{N}_r|}$ are generated with negative relations by selecting $\textit{self}_j^-$ strategically from a relation list $\mathcal{R} - \textit{self}$ (Fig. 1c.d). The contrastive scores for $p_s = (h, \textit{self}, h^+)$ can then be computed with Eq. (4) and Eq. (8).

Through the *Self* positive sample, parameters of zero-shot entities can have the chances to be updated from the contrastive perspective (Eqs. (5) and (9)), where h^+, r^+ give positive feedback while h_j^-, r_j^- give negative feedback.

4.5 Contrastive Fusion

The contrastive learning of the above modules has the form of 1-to-1, where a head entity propagates information to a tail entity through a relation, i.e., it involves only one positive sample. However, a head entity can connect to multiple tail entities through multiple relations. To model this multi-propagation pattern, we extend the 1-to-1 to 1-to-N, where multiple positive samples are considered.

There are two types of 1-to-N pattern: $p_e = \{(h, r, t_j^+)\}_{j=1}^{|p_e|}$ where a head entity h connects to multiple tail entities through a relation r, and $p_r = \{(h, r_j^+, t)\}_{j=1}^{|p_r|}$ where a head entity h connects to a tail entity t through multiple relations. We generate negative samples $\mathcal{N}_e = \{(h, r, t_j^-)\}_{j=1}^{|\mathcal{N}_e|}$ for p_e as in §4.2, and negative samples $\mathcal{N}_r = \{(h, r_j^-, t)\}_{j=1}^{|\mathcal{N}_r|}$ for p_r as in §4.3. Let $p_f = \{p_e, p_r\}$ and $\mathcal{N}_f = \{\mathcal{N}_e, \mathcal{N}_r\}$. With this, we design two types of contrastive scores to learn 1-to-N features.

$$S_a^f(p_f) = -\log \frac{\sum_{n^+\in p_f} e^{(\beta(n^+)/\tau)}}{\sum_{n^-\in\{p_f,\mathcal{N}_f\}} e^{(\beta(n^-)/\tau)}} \quad (10)$$

$$S_b^f(p_f) = -\sum_{n^+\in p_f} \log \frac{e^{(\beta(n^+)/\tau)}}{\sum_{n^-\in\{n^+,\mathcal{N}_f\}} e^{(\beta(n^-)/\tau)}} \quad (11)$$

In Eq. (10), all positive samples are trained together and normalized to a unified space, while in Eq. (11), different positive samples are trained separately and the independent similarity score of all positives are accumulated. From the perspective of negative samples, negative entity $\mathcal{N}_e$ and relation $\mathcal{N}_r$ samples are merged to train with the positive samples, which is different from Eq. (4) and Eq. (8) which only have one type of negatives.

4.6 Training Procedure

We train TernaryCL in Pretrain and Finetune stages, where Pretrain stage aims to learn discriminative representations with Eq. (10) or Eq. (11), and Finetune stage aims to optimize parameters for a target task. Note that TernaryCL is a general framework that can be applied to diverse KG types, such as OpenKGs, CuratedKGs and TemporalKGs, and to diverse applications, such as link prediction, relation prediction, node classification, relation extraction and other ternary tasks. In addition, the principle of Contrastive Self can be applied to dynamic tasks, e.g., when a new entity gets added to a KG, it can transfer embeddings of existing entities to the new entity. Since our focus in this work is the sparsity of OpenKGs, we design experiments with link prediction task in OpenKGs.

For a test triple (h_i, r_i, t_i), the jointly encapsulated representation $\varphi(h_i, r_i)$ of entity h_i and relation r_i is matched with the embeddings of all the entities in $\mathbf{E}$ to predict $\hat{Y}_i \in [0,1]^{|\mathcal{E}|}$ as:

$$\hat{Y}_i = \text{sigmoid}(\varphi(h_i, r_i) \cdot \mathbf{E}^\top) \quad (12)$$

We use a binary cross-entropy loss defined as:

$$-\frac{1}{|\mathcal{E}|}\sum_{j=1}^{|\mathcal{E}|}(Y_{i,j}\log\hat{Y}_{i,j} + (1-Y_{i,j})\log(1-\hat{Y}_{i,j})) \quad (13)$$

Dataset	Ent	Rel	Clust	Valid	Test	Train					Few-Shot		Zero-Shot	
						100%	80%	60%	40%	20%	Ent	Rel	Ent	Rel
ReVerb20K	11.1	11.1	10.8	1.6	2.4	15.5	12.4	9.3	6.2	3.1	2.8 / 1.0	2.5 / 0.8	8.1 / 1.1	8.5 / 1.0
ReVerb45K	27.0	21.6	18.6	3.6	5.4	36.0	28.8	21.6	14.4	7.2	7.7 / 1.8	5.1 / 1.5	18.8 / 2.5	16.4 / 1.8

Table 1: Dataset statistics. Ent, Rel, Clust show the number of entities, relations, entity clusters. Valid, Test, Train show the number of triples in valid, test, train sets, respectively. In Train, $x\%$ represents different sparsity levels. In Few- and Zero-Shot, left of / respresents the number of entities or relations, and right of / represents the number of related triples in its Test set.

where $Y_{i,j} = 1$ if $t_j \in \mathcal{E}(h_i, r_i)$ otherwise $Y_{i,j} = 0$ with $\mathcal{E}(h, r)$ being the set of all true tail entities for a pair (h, r). As mentioned earlier, when predicting the head h_i based on a pair (r_i, t_i), we reverse the relation to obtain a reversed triple (t, r_{rev}, h), then train it with Eqs. (12) and (13).

5 Experiments

5.1 Datasets and Experiment Setup

We use ReVerb20K and ReVerb45K OpenKG benchmarks (Vashishth et al., 2018), which are constructed through ReVerb (Fader et al., 2011). Table 1 presents the statistics about datasets. ReVerb45K with 27.0K entities and 21.6K relations is larger and sparser than ReVerb20K with 11.1K entities and 11.1K relations. Entity clusters, gold canonicalization clusters, are extracted through the Freebase entity linking tools (Gupta et al., 2019). Entities in an entity cluster have the same meaning. Usually, entity clusters are only used for evaluation.

To evaluate the sparsity problem, we design two sets of experiments: the first is about different sparsity levels and the second is about few-shot and zero-shot samples. For the first, we construct *train sets* at different sparsity granularity {100%, 80%, 60%, 40%, 20%} by respectively removing {0%, 20%, 40%, 60%, 80%} of the links from original train set. In this setup, we use the same original data for validation and testing.

For the second, we evaluate on few- or zero-shot samples separately, where few-shot refers to three-, two- and one-shot. Few- or zero-shot entities (relations) are extracted as per the definition in §3, e.g., a three-shot entity has three links in the test KG. For the test set of a few-shot (zero-shot) entity (relation), its related triples are extracted from original test set. For training here, we use the 20% train set, because it is sparse enough to contain more few-shot (zero-shot) samples.

5.2 Evaluation Metrics and Baselines

For a single test triple, we use Mention Ranking (MR) (Gupta et al., 2019) as a metric, which is the minimum ranking position of the answer entities. For all test triples, we use three most widely used ways to integrate the individual MR scores: (a) *H@N*: proportion of MR scores not higher than N, (b) *AR*: average all MR scores, and (c) *ARR*: compute the reciprocal of each MR score, and average all reciprocals. A model with better performance should have higher $H@N$, ARR and lower AR.[3]

To show the effectiveness of our approach, we compare TernaryCL against several strong baselines which fall in three groups:

- **General**: applies latest models for general KGs to OpenKGs. It includes TransE (Bordes et al., 2013), DistMult (Yang et al., 2015), ComplEx (Trouillon et al., 2016), ConvE (Dettmers et al., 2018), ConvTransE (Shang et al., 2019).
- **OpenKG**: comprises the baselines designed specifically for OpenKGs. It includes CaRe-TransE and CaReConvE (Gupta et al., 2019).
- **OpenKG+LM**: uses pretrained language models (LM) to improve the performance of the above OpenKG models. It includes OKGIT+Bert and OKGIT+Roberta (Chandrahas and Talukdar, 2021). To our knowledge, OKGIT is the state-of-the-art for OpenKGs.

For our model TernaryCL, default fusion strategy is $S_b^f(p_f)$, negative sampling strategy is *Uniform*, and text encoder is BiGRU. Due to space limitation, we provide details of the models, hyperparameters and settings in the Appendix.

5.3 Results

Full-data evaluation. We present the full-data results in Table 2, for which we use the original train set (100%). We notice that TernaryCL achieves substantial improvements in comparison to the baselines. For ReVerb20K, it outperforms all the baselines by a good margin across the metrics –

[3] In previous work, ARR and AR are called MRR, and MR respectively, where M stands for *Mean*. However, since *Mention Ranking* is abbreviated as MR, to prevent confusion, we use *Average* in the names in stead of *Mean*.

Type	Model	ReVerb20K						ReVerb45K					
		$AR\downarrow$	ARR	$H@1$	$H@10$	$H@50$	$H@100$	$AR\downarrow$	ARR	$H@1$	$H@10$	$H@50$	$H@100$
General	TransE	1497	13.3	2.2	29.6	43.0	49.2	2222	15.8	9.3	25.9	37.1	43.2
	DistMult	4569	1.9	1.3	2.7	5.2	7.1	5782	8.5	7.7	9.7	12.0	13.6
	ComlEx	4376	2.0	1.4	3.0	5.6	7.7	5173	8.9	7.5	11.3	16.0	18.9
	ConvE	1085	25.5	19.9	35.8	50.1	57.2	2483	22.1	16.6	32.4	43.3	47.9
	ConvTransE	1080	26.1	20.5	35.9	50.0	57.1	2490	23.4	17.9	33.8	44.4	48.8
OpenKG	CaReTransE	950	30.3	23.2	42.8	58.4	64.6	2414	19.5	7.8	37.5	47.5	51.4
	CaReConvE	801	31.6	25.6	42.9	56.7	63.4	1589	29.7	23.4	41.3	53.6	58.7
OpenKG +LM	OKGIT+Bert	524	35.1	27.5	49.5	65.9	72.7	**735**	**33.7**	**26.7**	47.1	59.8	65.2
	OKGIT+Rob	594	35.8	28.4	49.2	65.4	72.1	849	33.4	26.5	46.4	58.8	63.9
Our	**TernaryCL**	**421**	**38.1**	**29.9**	**53.3**	**68.5**	**75.2**	744	33.2	25.6	**48.1**	**61.9**	**67.7**

Table 2: Link prediction results on ReVerb20K and ReVerb45K in the standard full data (100%) setup. Best scores are made bold. Columns with ↓ denote lower is better, otherwise higher is better.

ARR increases by 2.3 point and H@1,10,50,100 increase by 1.5, 3.8, 2.6, 2.5 points respectively. For ReVerb45K, its performance is better than General and OpenKG baselines with sizeable improvements in all metrics, notably 3.5 point in ARR and 2.2, 6.8, 8.3, 9.0 points in H@1,10,50,100. It also outperforms OpenKG+LM baselines in all metrics for ReVerb20K and most metrics for ReVerb45K.

Overall, TernaryCL with a simple structure can achieve better performance through innovative training methods at lower costs than OpenKG+LM baselines that use a relatively complex structure and large pretrained LMs which can be memory/compute intensive. This can make TernaryCL a favorable choice against the baselines.

Targeted evaluation with sparsity. We now evaluate whether TernaryCL can alleviate the sparsity problem from the perspective of different sparsity levels. Fig. 2 reports the results on the (original) test set for different train sets with varying sparsity level. We observe that TernaryCL achieves the best scores at all sparsity levels on both datasets. Taking ReVerb20K as an example, it gives improvements of 2.9, 4.8, 2.1 and 3.9 points over OKGIT+Bert at 80%, 60%, 40%, 20%, respectively. Note that OKGIT+Bert enhances learning with large-scale pretrained LMs, which possess a lot of commonsense and factual knowledge within its huge parameter space by training on very large data. In contrast, TernaryCL does not use any side information or pretrained LMs, but gives significant performance gains over OKGIT+Bert, even when the sparsity level is high (e.g., 20%).

Targeted evaluation for few- and zero-shot. We now analyze whether TernaryCL can alleviate sparsity on test sets with few- or zero-shot entities (or relations). Table 3 shows the results on ReVerb20K.

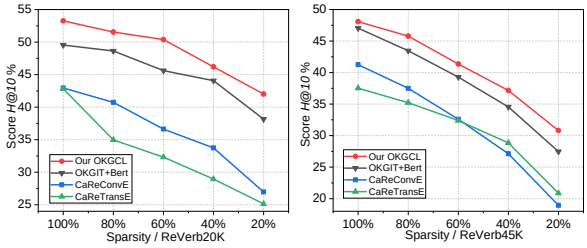


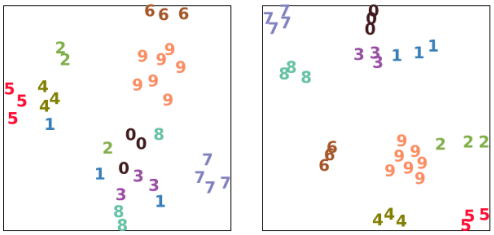
Figure 2: Link prediction results for different sparsity levels.

Type		Metric	CaRe TransE	CaRe ConvE	OKGIT +Bert	Ternary CL
Few Shot	Ent	AR	1215	2196	**285**	443
		ARR	10.6	24.4	29.4	**32.6**
		$H@1$	1.2	19.6	22.2	**25.3**
	Rel	AR	2022	2663	1696	**826**
		ARR	10.7	20.6	26.2	**28.6**
		$H@1$	1.7	16.2	19.3	**21.3**
Zero Shot	Ent	AR	3286	3304	3465	**1461**
		ARR	7.3	14.2	20.1	**20.9**
		$H@1$	0.0	11.1	15.1	**15.4**
	Rel	AR	2071	2557	1775	**887**
		ARR	10.4	22.3	27.4	**30.1**
		$H@1$	2.2	18.1	20.7	**23.4**

Table 3: Results of few- and zero-shot entities (Ent) / relations (Rel) on ReVerb20K. Models were trained on the 20% setup.

CaReTransE performs poorly in all metrics, especially, H@1 score is 0.0 for zero-shot entities. CaReConvE performs better than CaReTransE on ARR and H@1, but weaker on AR. OKGIT+Bert achieves better results than CaReTransE and CaReConvE, which shows that pretrained knowledge introduced by Bert could be helpful to few- and zero-shot entities and relations. TernaryCL, without using any pretrained LM or side information, outperforms all baselines by a good margin. We provide more results in the Appendix.

Visualization To qualitatively demonstrate that TernaryCL can make entities with the same meaning closer in the vector space, we show t-SNE visualization (Van der Maaten and Hinton, 2008) in Fig. 3. For this, we use the original train set



(a) OKGIT+Bert (b) TernaryCL

Figure 3: Visualization on ReVerb20K, where same numbers represent entities with the same meaning in an entity cluster. Visualization on ReVerb20K is shown in the Appendix.

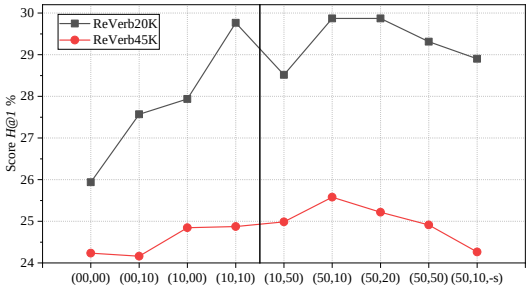


Figure 4: Results of the number collocations of negative entities and negative relations.

Model	$AR\downarrow$	ARR	$H@1$	$H@10$	$H@50$	$H@100$
$S_a(p_e)$	486	36.1	28.4	50.9	65.6	71.7
$S_a(p_r)$	481	36.5	28.4	52.3	66.4	72.8
$S_b(p_e)$	440	36.3	28.1	51.7	66.6	72.6
$S_b(p_r)$	482	36.3	28.0	52.3	66.9	73.5
$S_a^f(p_f)$	459	37.1	28.6	53.3	68.0	73.9
$S_b^f(p_f)$	**421**	**38.1**	**29.9**	53.3	**68.5**	**75.2**
ManyP	423	37.2	28.8	53.3	68.1	74.2
FewP	437	37.9	29.5	**53.7**	68.1	74.9

Table 4: Results for different fusion and negative sampling strategies on ReVerb20K. Default *Uniform*.

(100%) and compare with the state-of-the-art baseline OKGIT+Bert. For the visualization, we selected 10 entity clusters, where each cluster has more than three entities. We observe that embeddings of entities with the same meaning (same number) are closer in TernaryCL than in OKGIT+Bert. These qualitative results are consistent with the above quantitative results, which further verify the effectiveness of TernaryCL.

5.4 Ablation Study and Analysis

Fusion strategy. The ablation results for the fusion strategies (§4.5) are shown in Table 4, where $S_a(p_e)$, $S_a(p_r)$, $S_b(p_e)$, $S_b(p_r)$ use only one type of 1-to-N, as denoted by the subscript of p, and $S_a^f(p_f)$ and $S_b^f(p_f)$ are our two fusion variants. $S_a^f(p_f)$ which fuses both 1-to-N types, outperforms its counterparts $S_a(p_e)$ and $S_a(p_r)$. We see the same phenomenon with $S_b^f(p_f)$, which outperforms $S_b(p_e)$ and $S_b(p_r)$. This proves that both 1-to-N types carry different semantic features, and integration of them can achieve better results. When we compare our two variants, $S_b^f(p_f)$ show better performance than $S_a^f(p_f)$. $S_b^f(p_f)$ contrasts a positive sample with both negative entities and relations, which is able to highlight the discriminative features of this positive sample.

Negative sampling strategy. Table 4 shows the results for $S_b^f(p_f)$ with *ManyP*, *FewP* sampling methods (§4.2) in addition to the default (*Uniform*) sampling method. We observe that *Uniform* strategy, which gives each entity (relation) the same selection probability, is able to better balance the training times of entities and relations.

Number of negative entities and relations Fusing negative entities and relations in strategy $S_b^f(p_f)$ has been proved effective above. Our assumption is that the number of negative entities and relations could also affect the performance of TernaryCL. So, we design several collocations of negative entities (left) and negative relations (right): (00, 00), (00, 10), (10, 00), (10, 10), (10, 50), (50, 10), (50, 20), and (50, 50). We also include (50,10,-s) which removes the Contrastive Self module.

The results in Fig. 4 reveal several findings: First, TernaryCL achieves best results at (50,10) on both datasets. Second, negative entities are more important than negative relations in improving performance. For example, performance of (10, 00) is better than that of (00, 10), and performance of (50, 10) is also better than that of (10, 50). Third, the role of negative relations can be more stimulated with the help of negative entities. For example on ReVerb45K, performance of (00, 10) is worse than that of (00, 00), while performance of (10, 10) is a little better than that of (10, 00). Finally, Contrastive Self plays a great role in improving performance. Compared with (50,10), performance of (50,10,-s) decreases prominently when removing Contrastive Self, especially on sparser ReVerb45K.

6 Conclusion

In this work, we have provided empirical insights about the sparsity of OpenKGs, and proposed TernaryCL to alleviate the sparsity with contrastive entity, relation and self. Through extensive experiments and comprehensive analysis on benchmarks, we show that TernaryCL outperforms state-of-the-art baselines by a good margin.

References

Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, pages 2787–2795.

Samuel Broscheit, Kiril Gashteovski, Yanjie Wang, and Rainer Gemulla. 2020. Can we predict new facts with open knowledge graph embeddings? A benchmark for open link prediction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL, Online, July 5-10, 2020*, pages 2296–2308.

Chandrahas and Partha P. Talukdar. 2021. OKGIT: open knowledge graph link prediction with implicit types. In *Findings of the Association for Computational Linguistics: ACL/IJCNLP, Online Event, August 1-6*, pages 2546–2559. Association for Computational Linguistics.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1724–1734. ACL.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2d knowledge graph embeddings. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 1811–1818.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, EMNLP, 27-31 July 2011, John McIntyre Conference Centre, Edinburgh, UK, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1535–1545.

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

Kiril Gashteovski, Sebastian Wanner, Sven Hertling, Samuel Broscheit, and Rainer Gemulla. 2019. OPIEC: an open information extraction corpus. In *1st Conference on Automated Knowledge Base Construction, AKBC, Amherst, MA, USA, May 20-22, 2019*.

Swapnil Gupta, Sreyash Kenkre, and Partha P. Talukdar. 2019. Care: Open knowledge graph embeddings. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP, Hong Kong, China, November 3-7, 2019*, pages 378–388.

Raia Hadsell, Sumit Chopra, and Yann LeCun. 2006. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2006), 17-22 June 2006, New York, NY, USA*, pages 1735–1742. IEEE Computer Society.

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 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 4116–4126. PMLR.

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 2020, Seattle, WA, USA, June 13-19, 2020*, pages 9726–9735. Computer Vision Foundation / IEEE.

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*.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA*, pages 2181–2187.

Chaitanya Malaviya, Chandra Bhagavatula, Antoine Bosselut, and Yejin Choi. 2020. Commonsense knowledge base completion with structural and semantic context. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 2925–2933.

Dai Quoc Nguyen, Tu Dinh Nguyen, Dat Quoc Nguyen, and Dinh Q. Phung. 2018. A novel embedding model for knowledge base completion based on convolutional neural network. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT, New Orleans,*

Louisiana, USA, June 1-6, 2018, Volume 2 (Short Papers), pages 327–333.

Chao Shang, Yun Tang, Jing Huang, Jinbo Bi, Xiaodong He, and Bowen Zhou. 2019. End-to-end structure-aware convolutional networks for knowledge base completion. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 3060–3067.

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 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *Proceedings of the 33nd International Conference on Machine Learning, ICML, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 2071–2080.

Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(11).

Shikhar Vashishth, Prince Jain, and Partha P. Talukdar. 2018. CESI: canonicalizing open knowledge bases using embeddings and side information. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web, WWW, Lyon, France, April 23-27, 2018*, pages 1317–1327.

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 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27-31, 2014, Québec City, Québec, Canada*, pages 1112–1119.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding entities and relations for learning and inference in knowledge bases. In *3rd International Conference on Learning Representations, ICLR, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

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. ACM / IW3C2.

7 Appendix

7.1 Gradient Update

Gradients of Contrastive Entity Take the Eq. (4) as an example, we give the detailed reasoning steps of gradient about h, r, t^+, t^-.

$$
\begin{aligned}
&-\frac{\partial S(h,r,t^+)}{\partial h} \\
&= \frac{\partial}{\partial h}(\log \frac{e^{(\beta(h,r,t^+)/\tau)}}{\sum_{n\in\{p_e,\mathcal{N}_e\}} e^{(\beta(n)/\tau)}}) \\
&= \frac{\partial}{\partial h}(\frac{\beta(h,r,t^+)}{\tau} - \log \sum_{n\in\{p_e,\mathcal{N}_e\}} e^{(\frac{\beta(n)}{\tau})}) \\
&= \frac{\partial}{\partial h}(\frac{\beta(h,r,t^+)}{\tau} - \log(e^{(\frac{\beta(h,r,t^+)}{\tau})} \\
&\quad + \sum_{(h,r,t_j^-)\in\mathcal{N}_e} e^{(\frac{\beta(h,r,t_j^-)}{\tau})})) \\
&= \frac{\partial}{\partial h}(\frac{\varphi(h,r)\cdot\mathbf{t}^+}{\tau} - \log(e^{(\frac{\varphi(h,r)\cdot\mathbf{t}^+}{\tau})} \\
&\quad + \sum_{(h,r,t_j^-)\in\mathcal{N}_e} e^{(\frac{\varphi(h,r)\cdot\mathbf{t_j^-}}{\tau})})) \\
&= \frac{\varphi'(h,r)}{\tau}\mathbf{t}^+ - \\
&\quad \frac{e^{(\frac{\varphi(h,r)\cdot\mathbf{t}^+}{\tau})}\frac{\varphi'(h,r)}{\tau}\mathbf{t}^+ + \sum_{(h,r,t_j^-)\in\mathcal{N}_e} e^{(\frac{\varphi(h,r)\cdot\mathbf{t_j^-}}{\tau})}\frac{\varphi'(h,r)}{\tau}\mathbf{t_j^-}}{e^{(\frac{\varphi(h,r)\cdot\mathbf{t}^+}{\tau})} + \sum_{(h,r,t_j^-)\in\mathcal{N}_e} e^{(\frac{\varphi(h,r)\cdot\mathbf{t_j^-}}{\tau})}} \\
&= \frac{\varphi'(h,r)}{\tau}(\frac{\sum_{(h,r,t_j^-)\in\mathcal{N}_e} e^{(\frac{\varphi(h,r)\cdot\mathbf{t_j^-}}{\tau})}}{e^{(\frac{\varphi(h,r)\cdot\mathbf{t}^+}{\tau})} + \sum_{(h,r,t_j^-)\in\mathcal{N}_e} e^{(\frac{\varphi(h,r)\cdot\mathbf{t_j^-}}{\tau})}}\mathbf{t}^+ \\
&\quad - \frac{\sum_{(h,r,t_j^-)\in\mathcal{N}_e} e^{(\frac{\varphi(h,r)\cdot\mathbf{t_j^-}}{\tau})}\mathbf{t_j^-}}{e^{(\frac{\varphi(h,r)\cdot\mathbf{t}^+}{\tau})} + \sum_{(h,r,t_j^-)\in\mathcal{N}_e} e^{(\frac{\varphi(h,r)\cdot\mathbf{t_j^-}}{\tau})}}) \\
&= \frac{\varphi'(h,r)}{\tau A}(\sum_{(h,r,t_j^-)\in\mathcal{N}_e} e^{(\frac{\varphi(h,r)\cdot\mathbf{t_j^-}}{\tau})}\mathbf{t}^+ \\
&\quad - \sum_{(h,r,t_j^-)\in\mathcal{N}_e} e^{(\frac{\varphi(h,r)\cdot\mathbf{t_j^-}}{\tau})}\mathbf{t_j^-})
\end{aligned} \tag{14}
$$

$$
A = e^{(\frac{\varphi(h,r)\cdot\mathbf{t}^+}{\tau})} + \sum_{(h,r,t_j^-)\in\mathcal{N}_e} e^{(\frac{\varphi(h,r)\cdot\mathbf{t_j^-}}{\tau})} \tag{15}
$$

Parameters of entity h can be updated from contrastive with the gradient in Eq. (20):

$$
\begin{aligned}
\mathbf{h}^{l+1} &= \mathbf{h}^l - \eta(\frac{\partial S(h,r,t^+)}{\partial h}) \\
&= \mathbf{h}^l + \frac{\varphi'(h,r)\eta}{\tau A}\sum_{(h,r,t_j^-)\in\mathcal{N}_e} e^{(\frac{\varphi(h,r)\cdot\mathbf{t_j^-}}{\tau})}\mathbf{t}^+ \\
&\quad - \frac{\varphi'(h,r)\eta}{\tau A}\sum_{(h,r,t_j^-)\in\mathcal{N}_e} e^{(\frac{\varphi(h,r)\cdot\mathbf{t_j^-}}{\tau})}\mathbf{t_j^-}
\end{aligned} \tag{16}
$$

$$
\frac{\partial S(h,r,t^+)}{\partial r} = \frac{\partial S(h,r,t^+)}{\partial h}
$$

$$
\begin{aligned}
&-\frac{\partial S(h,r,t^+)}{\partial t^+} \\
&= \frac{\partial}{\partial t^+}(\frac{\varphi(h,r)\cdot\mathbf{t}^+}{\tau} - \log(e^{(\frac{\varphi(h,r)\cdot\mathbf{t}^+}{\tau})} \\
&\quad + \sum_{(h,r,t_j^-)\in\mathcal{N}_e} e^{(\frac{\varphi(h,r)\cdot\mathbf{t_j^-}}{\tau})})) \\
&= \frac{\varphi(h,r)}{\tau} - \\
&\quad \frac{e^{(\frac{\varphi(h,r)\cdot\mathbf{t}^+}{\tau})}\frac{\varphi(h,r)}{\tau}}{e^{(\frac{\varphi(h,r)\cdot\mathbf{t}^+}{\tau})} + \sum_{(h,r,t_j^-)\in\mathcal{N}_e} e^{(\frac{\varphi(h,r)\cdot\mathbf{t_j^-}}{\tau})}} \\
&= \frac{\varphi(h,r)}{\tau A}\sum_{(h,r,t_j^-)\in\mathcal{N}_e} e^{(\frac{\varphi(h,r)\cdot\mathbf{t_j^-}}{\tau})}
\end{aligned} \tag{17}
$$

$$
\begin{aligned}
&-\frac{\partial S(h,r,t^+)}{\partial t_j^-} \\
&= \frac{\partial}{\partial t_j^-}(\frac{\varphi(h,r)\cdot\mathbf{t}^+}{\tau} - \log(e^{(\frac{\varphi(h,r)\cdot\mathbf{t_j^-}}{\tau})} \\
&\quad + \sum_{(h,r,t_j^-)\in\{p_e\mathcal{N}_e-(h,r,t_j^-)\}} e^{(\frac{\varphi(h,r)\cdot\mathbf{t_j^-}}{\tau})})) \\
&= \frac{-e^{(\frac{\varphi(h,r)\cdot\mathbf{t_j^-}}{\tau})}\frac{\varphi(h,r)}{\tau}}{e^{(\frac{\varphi(h,r)\cdot\mathbf{t_j^-}}{\tau})} + \sum_{(h,r,t_j^-)\in\{p_e\mathcal{N}_e-(h,r,t_j^-)\}} e^{(\frac{\varphi(h,r)\cdot\mathbf{t_j^-}}{\tau})}} \\
&= -\frac{\varphi(h,r)}{\tau A}e^{(\frac{\varphi(h,r)\cdot\mathbf{t_j^-}}{\tau})}
\end{aligned} \tag{18}
$$

Gradients of Contrastive Relation Take the Eq. (8) as an example, we give the detailed reasoning steps of gradient about h, t, r^+ and r^-.

$$
\begin{aligned}
&-\frac{\partial S(h,r^+,t)}{\partial h} \\
&= \frac{\partial}{\partial h}(\frac{\varphi(h,r^+)\cdot\mathbf{t}}{\tau} - \log(e^{(\frac{\varphi(h,r^+)\cdot\mathbf{t}}{\tau})} \\
&\quad + \sum_{(h,r_j^-,t)\in\mathcal{N}_r} e^{(\frac{\varphi(h,r_j^-)\cdot\mathbf{t}}{\tau})})) \\
&= \frac{\varphi'(h,r^+)\mathbf{t}}{\tau} - \\
&\quad \frac{e^{(\frac{\varphi(h,r^+)\cdot\mathbf{t}}{\tau})}\frac{\varphi'(h,r^+)\mathbf{t}}{\tau} + \sum_{(h,r_j^-,t)\in\mathcal{N}_r} e^{(\frac{\varphi(h,r_j^-)\cdot\mathbf{t}}{\tau})}\frac{\varphi'(h,r_j^-)\mathbf{t}}{\tau}}{e^{(\frac{\varphi(h,r^+)\cdot\mathbf{t}}{\tau})} + \sum_{(h,r_j^-,t)\in\mathcal{N}_e} e^{(\frac{\varphi(h,r_j^-)\cdot\mathbf{t}}{\tau})}} \\
&= \frac{\varphi'(h,r^+)\mathbf{t}}{\tau B}\sum_{(h,r_j^-,t)\in\mathcal{N}_r} e^{(\frac{\varphi(h,r_j^-)\cdot\mathbf{t}}{\tau})} - \\
&\quad \frac{1}{B}\sum_{(h,r_j^-,t)\in\mathcal{N}_r} e^{(\frac{\varphi(h,r_j^-)\cdot\mathbf{t}}{\tau})}\frac{\varphi'(h,r_j^-)\mathbf{t}}{\tau}
\end{aligned} \tag{19}
$$

$$
\begin{aligned}
&-\frac{\partial S(h,r^+,t)}{\partial t} \\
&= \frac{\partial}{\partial t}(\frac{\varphi(h,r^+)\cdot\mathbf{t}}{\tau} - \log(e^{(\frac{\varphi(h,r^+)\cdot\mathbf{t}}{\tau})} \\
&\quad + \textstyle\sum_{(h,r_j^-,t)\in\mathcal{N}_r} e^{(\frac{\varphi(h,r_j^-)\cdot\mathbf{t}}{\tau})})) \\
&= \frac{\varphi(h,r^+)}{\tau} - \\
&\quad \frac{e^{(\frac{\varphi(h,r^+)\cdot\mathbf{t}}{\tau})}\frac{\varphi(h,r^+)}{\tau} + \sum\limits_{(h,r_j^-,t)\in\mathcal{N}_r} e^{(\frac{\varphi(h,r_j^-)\cdot\mathbf{t}}{\tau})}\frac{\varphi(h,r_j^-)}{\tau}}{e^{(\frac{\varphi(h,r^+)\cdot\mathbf{t}}{\tau})} + \sum\limits_{(h,r_j^-,t)\in\mathcal{N}_r} e^{(\frac{\varphi(h,r_j^-)\cdot\mathbf{t}}{\tau})}} \\
&= \frac{\varphi(h,r^+)}{\tau B} \sum_{(h,r_j^-,t)\in\mathcal{N}_r} e^{(\frac{\varphi(h,r_j^-)\cdot\mathbf{t}}{\tau})} - \\
&\quad \frac{1}{B} \sum_{(h,r_j^-,t)\in\mathcal{N}_r} e^{(\frac{\varphi(h,r_j^-)\cdot\mathbf{t}}{\tau})}\frac{\varphi(h,r_j^-)}{\tau}
\end{aligned}
\tag{20}
$$

$$
\begin{aligned}
&-\frac{\partial S(h,r^+,t)}{\partial r^+} \\
&= \frac{\partial}{\partial r^+}(\log\frac{e^{(\beta(h,r^+,t)/\tau)}}{\sum_{n\in\{p_r,\mathcal{N}_r\}} e^{(\beta(n)/\tau)}}) \\
&= \frac{\partial}{\partial r^+}(\frac{\beta(h,r^+,t)}{\tau} - \log\textstyle\sum_{n\in\{p_r,\mathcal{N}_r\}} e^{(\frac{\beta(n)}{\tau})}) \\
&= \frac{\partial}{\partial r^+}(\frac{\beta(h,r^+,t)}{\tau} - \log(e^{(\frac{\beta(h,r^+,t)}{\tau})} \\
&\quad + \textstyle\sum_{(h,r_j^-,t)\in\mathcal{N}_r} e^{(\frac{\beta(h,r_j^-,t)}{\tau})})) \\
&= \frac{\partial}{\partial r^+}(\frac{\varphi(h,r^+)\cdot\mathbf{t}}{\tau} - \log(e^{(\frac{\varphi(h,r^+)\cdot\mathbf{t}}{\tau})} \\
&\quad + \textstyle\sum_{(h,r_j^-,t)\in\mathcal{N}_r} e^{(\frac{\varphi(h,r_j^-)\cdot\mathbf{t}}{\tau})})) \\
&= \frac{\varphi'(h,r^+)\mathbf{t}}{\tau} - \\
&\quad \frac{e^{(\frac{\varphi(h,r^+)\cdot\mathbf{t}}{\tau})}\frac{\varphi'(h,r^+)\mathbf{t}}{\tau}}{e^{(\frac{\varphi(h,r^+)\cdot\mathbf{t}}{\tau})} + \sum\limits_{(h,r_j^-,t)\in\mathcal{N}_r} e^{(\frac{\varphi(h,r_j^-)\cdot\mathbf{t}}{\tau})}} \\
&= \frac{\varphi'(h,r^+)\mathbf{t}}{\tau}\Big(\\
&\quad \frac{\sum\limits_{(h,r_j^-,t)\in\mathcal{N}_r} e^{(\frac{\varphi(h,r_j^-)\cdot\mathbf{t}}{\tau})}}{e^{(\frac{\varphi(h,r^+)\cdot\mathbf{t}}{\tau})} + \sum\limits_{(h,r_j^-,t)\in\mathcal{N}_r} e^{(\frac{\varphi(h,r_j^-)\cdot\mathbf{t}}{\tau})}}\Big) \\
&= \frac{\varphi'(h,r^+)\mathbf{t}}{\tau B} \sum_{(h,r_j^-,t)\in\mathcal{N}_r} e^{(\frac{\varphi(h,r_j^-)\cdot\mathbf{t}}{\tau})}
\end{aligned}
\tag{21}
$$

$$
B = e^{(\frac{\varphi(h,r^+)\cdot\mathbf{t}}{\tau})} + \sum_{(h,r_j^-,t)\in\mathcal{N}_r} e^{(\frac{\varphi(h,r_j^-)\cdot\mathbf{t}}{\tau})} \tag{22}
$$

$$
\begin{aligned}
&-\frac{\partial S(h,r^+,t)}{\partial r_j^-} \\
&= \frac{\partial}{\partial r_j^-}(\frac{\varphi(h,r^+)\cdot\mathbf{t}}{\tau} - \log(e^{(\frac{\varphi(h,r_j^-)\cdot\mathbf{t}}{\tau})} \\
&\quad + \textstyle\sum_{(h,r_j^-,t)\in\{p_r\mathcal{N}_r-(h,r_j^-,t)\}} e^{(\frac{\varphi(h,r_j^-)\cdot\mathbf{t}}{\tau})})) \\
&= \frac{-e^{(\frac{\varphi(h,r_j^-)\cdot\mathbf{t}}{\tau})}\frac{\varphi'(h,r_j^-)\mathbf{t}}{\tau}}{e^{(\frac{\varphi(h,r_j^-)\cdot\mathbf{t}}{\tau})} + \sum_{(h,r_j^-,t)\in\{p_r\mathcal{N}_r-(h,r_j^-,t)\}} e^{(\frac{\varphi(h,r_j^-)\cdot\mathbf{t}}{\tau})}} \\
&= -\frac{\varphi'(h,r_j^-)\mathbf{t}}{\tau B} e^{(\frac{\varphi(h,r_j^-)\cdot\mathbf{t}}{\tau})}
\end{aligned}
\tag{23}
$$

7.2 Hyperparameter and Setting

For TernaryCL TernaryCL is implemented based on the PyTorch library with a single GeForce RTX 2080 GPU. The number of parameters is 18.06M for ReVerb20K and 33.63M for ReVerb45K. We run the model several times and report the maximum of results. For the training settings, the optimizer is set to Adam, the embedding size is set to 300. The entity and relation embeddings are initialized randomly, and the word vectors are initialized with the GloVe embeddings.

We tune our model with the grid search to select the optimal hyper-parameters based on the performance on the validation dataset. The list of hyperparameters are from two aspects: (1) Hyperparameters for Pretrain stage: learning rate $\in$ {1e-3, 1e-4, 5e-5, 1e-5}, temperature regulation value $\in$ {0.1, 0.05, 0.01}. (2) Hyperparameters for Finetune stage: learning rate $\in$ {1e-3, 1e-4, 8e-5, 5e-5, 1e-5}, batch size $\in$ {32, 64, 128, 256, 512}. The results of hyperparameters are shown in Fig. 5. For Pretrain stage, the optimal value of learning rate is 5e-5 on both datasets, where too large a learning rate, such as 1E-3, may skip the optimal results. And the optimal value of temperature regulation value is 0.05 on both datasets. For Finetune stage, the optimal value of learning rate is 5e-5 on both datasets, where too small a learning rate, such as 1E-5, brings worse performance than too large a learning rate, such as 1E-3. The optimal value of batch size is 128 on both datasets, where TernaryCL is not very sensitive to batch size, but too large a batch size could have a negative impact.

For Baselines The results of baselines are reproduced with open source implementations. Concretely, TransE, DistMult and ComplEx are repro-

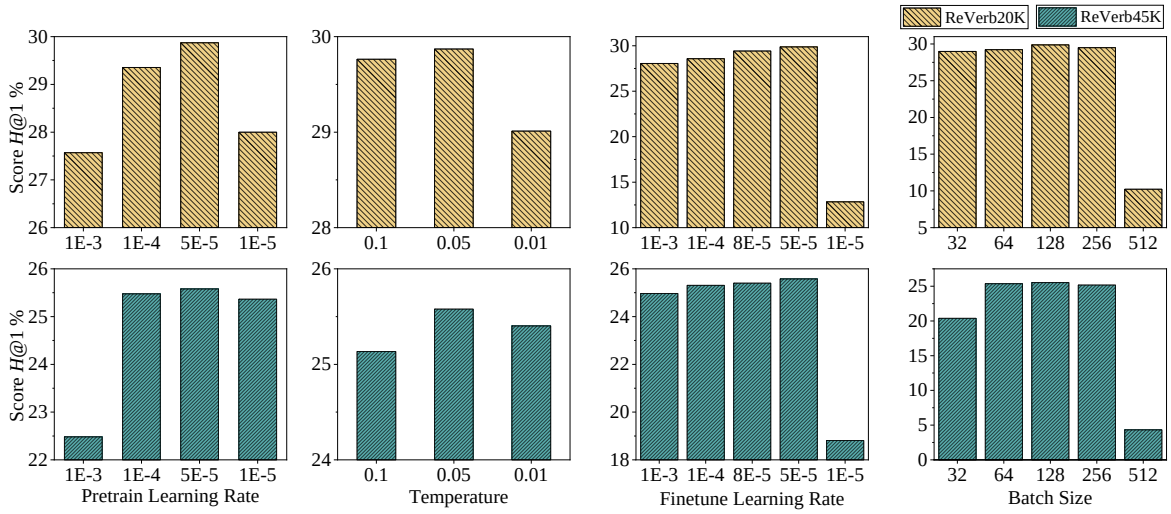
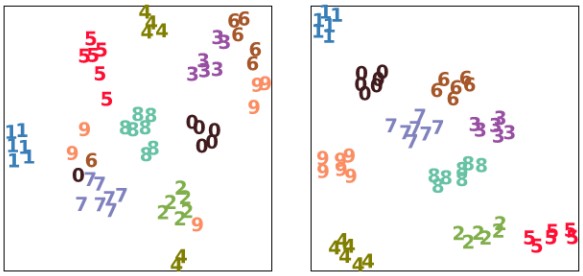


Figure 5: Results of hyperparameters on ReVerb20K (upper) and ReVerb45K (Below).

Model	$AR\downarrow$	ARR	$H@1$	$H@10$	$H@50$	$H@100$
Bert-F	754	28.1	22.1	39.1	53.8	61.2
Bert-U	1315	23.7	18.7	33.2	45.9	51.8
BiGRU	**421**	**38.1**	**29.9**	**53.3**	**68.5**	**75.2**
Bert-F	1165	28.8	21.6	42.8	55.3	60.4
Bert-U	2131	18.0	12.5	28.9	41.3	46.6
BiGRU	**744**	**33.2**	**25.6**	**48.1**	**61.9**	**67.7**

Table 5: Results of Textual Technology on ReVerb20K (upper) and ReVerb45K (below).



(a) OKGIT+Bert (b) TernaryCL

Figure 6: Visualization on ReVerb45K, where the same numbers represent entities with the same meaning.

duced with public code in [4]. ConvE is reproduced with public code in [5]. The code for ConvTransE is implemented by us and public in our code. We use the grid search technique to select the optimal values of hyperparameters for above baselines. For OpenKG and OpenKG+Pretrained baselines, CaReTransE, CaReConvE are reproduced with the public code in [5], OKGIT+Bert and OKGIT+Rob are reproduced with the public code in [6]. The optimal values of hyperparameters for this four baselines are consistent with that in their paper.

7.3 Textual Technology

Results of textual technologies to encoder sequences in §4.1: BiGRU and Bert, are shown in Table 4, where *Bert-F* fixes the parameters of Bert, *Bert-U* unfixes them. By observing the results, performance of model with Bert as encoder is weaker than that with BiGRU. Results of Bert with fixed parameters are better than results with unfixed parameters. This shows that BiGRU is more capable of capturing sequential information in KGs. State-of-the-art baseline OKGIT also points out that pretrained language models can not predict the correct entities on the top. It could be inadvisable to introduce the pretrained language model into KGs due to the difficulty of training and reproduction. The intention of constructing a KG is to convert text into structures for easy calculation, that is, KG itself is rich in world knowledge. So, our TernaryCL pays attention to mining own structure features in an effective way.

[4]https://github.com/uma-pi1/kge
[5]https://github.com/malllabiisc/CaRE
[6]https://github.com/Chandrahasd/OKGIT

Model	$AR\downarrow$	ARR	$H@1$	$H@10$	$H@50$	$H@100$
$S_a(p_e)$	**712**	32.8	24.8	**48.6**	**62.7**	**68.2**
$S_a(p_r)$	791	32.0	24.5	46.1	59.9	65.5
$S_a^f(p_f)$	**712**	33.1	25.3	48.5	62.3	68.0
$S_b(p_e)$	806	33.0	25.1	48.0	62.2	67.6
$S_b(p_r)$	752	31.5	23.5	46.9	60.5	66.0
ManyP	758	**33.3**	25.5	48.5	62.2	68.1
FewP	770	33.2	25.3	48.2	62.4	67.9
#	744	33.2	**25.6**	48.1	61.9	67.7

Table 6: Results of Fusion strategy and Negative Strategy on ReVerb45K, where # represents $S_b^f(p_f)$ and *Uniform*.

Type		Metric	ReVerb20K				ReVerb45K			
			CaRe TransE	CaRe ConvE	OKGIT +Bert	Ternary CL	CaRe TransE	CaRe ConvE	OKGIT +Bert	Ternary CL
Few Shot	Ent	AR	1215	2196	**285**	443	2610	2739	**644**	988
		ARR	10.6	24.4	29.4	**32.6**	12.6	19.2	23.5	**23.5**
		$H@1$	1.2	19.6	22.2	**25.3**	6.7	15.5	**18.2**	16.8
		$H@10$	25.6	33.1	43.1	**46.8**	21.4	26.2	33.7	**35.8**
		$H@50$	41.4	44.9	61.1	**65.0**	30.9	33.8	48.1	**50.1**
		$H@100$	47.5	50.9	68.4	**72.2**	35.6	38.2	55.0	**56.8**
	Rel	AR	2022	2663	1696	**826**	3323	3379	2211	**1503**
		ARR	10.7	20.6	26.2	**28.6**	13.6	14.8	20.0	**21.0**
		$H@1$	1.7	16.2	19.3	**21.3**	7.8	10.9	14.5	**14.6**
		$H@10$	26.4	29.3	38.8	**43.3**	23.0	22.8	31.0	**33.2**
		$H@50$	41.9	38.8	53.8	**56.8**	33.9	30.5	45.3	**47.4**
		$H@100$	48.6	43.4	60.0	**62.0**	38.9	35.7	51.4	**54.2**
Zero Shot	Ent	AR	3286	3304	3465	**1461**	5776	5662	4952	**2720**
		ARR	7.3	14.2	20.1	**20.9**	10.2	4.3	7.9	**12.7**
		$H@1$	0.0	11.1	15.1	**15.4**	8.8	3.0	5.4	**9.3**
		$H@10$	20.9	20.8	29.8	**31.2**	12.4	6.6	12.5	**19.1**
		$H@50$	34.7	27.3	40.6	**41.5**	17.1	11.2	20.5	**26.9**
		$H@100$	39.5	30.2	44.5	**45.1**	19.8	13.6	25.3	**31.7**
	Rel	AR	2071	2557	1775	**887**	3368	3429	2349	**1571**
		ARR	10.4	22.3	27.4	**30.1**	11.8	12.5	18.6	**18.6**
		$H@1$	2.2	18.1	20.7	**23.4**	6.4	8.8	**13.2**	12.5
		$H@10$	24.8	30.4	40.8	**42.6**	20.3	19.2	29.2	**30.6**
		$H@50$	40.7	39.2	54.0	**56.1**	31.4	28.4	42.4	**43.5**
		$H@100$	47.0	43.0	59.9	**61.4**	35.9	32.9	49.2	**50.4**

Table 7: Results of few-shot and zero-shot entities (Ent) / relations (Rel) on ReVerb20K and ReVerb45K.