Improving Knowledge Graph Completion with Generative Hard Negative Mining

Zile Qiao[1], **Wei Ye**[2] , **Dingyao Yu**[2], **Tong Mo**[1,†]
Weiping Li[1], **Shikun Zhang**[2]
[1] School of Software and Microelectronics, Peking University
[2] National Engineering Research Center for Software Engineering, Peking University
{zileq, wye, yudingyao, zhangsk}@pku.edu.cn
{motong, wpli}@ss.pku.edu.cn

Abstract

Contrastive learning has recently shown great potential to improve text-based knowledge graph completion (KGC). In this paper, we propose to learn a more semantically structured entity representation space in text-based KGC via hard negatives mining. Specifically, we novelly leverage a sequence-to-sequence architecture to generate high-quality hard negatives. These negatives are sampled from the same decoding distributions as the anchor (or correct entity), inherently being semantically close to the anchor and thus enjoying good hardness. A self-information-enhanced contrasting strategy is further incorporated into the Seq2Seq generator to systematically diversify the produced negatives. Extensive experiments on three KGC benchmarks demonstrate the sound hardness and diversity of our generated negatives and the resulting performance superiority on KGC.

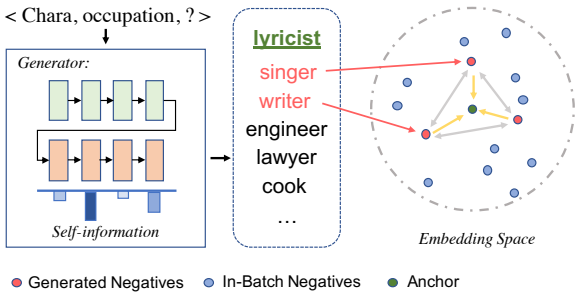

Figure 1: Conceptual illustration of our generative negative mining strategy. We feed the query (e.g., "<Chara, occupation, ?>") into a sequence-to-sequence model to generate multiple entities consisting of the correct answer and negatives. These entities are sampled from the same decoding distributions, making the negatives inherently semantically close to the target entity (as the yellow arrows show). A contrasting strategy regulated by token self-information is imposed on the Seq2Seq generator to make the negatives more diversified (or uniformly distributed, as the grey arrows depict).

1 Introduction

Knowledge Graph (KG) is an efficient method of representing global knowledge (Cui et al., 2019; Lv et al., 2022), playing a fundamental role in many Natural Language Processing (NLP) tasks like question answering (Sun et al., 2019a; Saxena et al., 2020), recommender systems (Huang et al., 2018), and web search (Ji et al., 2020), etc. The knowledge graph is composed of triples (h, r, t), where h, t, and r denote a head entity, a tail entity, and their relation, respectively. Modern public KGs such as Freebase, YAGO, and Wikidata, although covering massive knowledge with a large number of entities, are inevitably incomplete. Therefore, Knowledge graph completion (KGC) has become a popular area of research in recent years (Wang et al., 2017).

Generally, two types of methods are applied for this task. **Embedding-based methods** assign each entity/relation with a dense vector trained with structural information of graphs. Representative efforts include TransE (Bordes et al., 2013), TransH (Wang et al., 2014), Complex (Trouillon et al., 2016), and RotatE (Sun et al., 2019b), etc. **Text-based methods** (Wang et al., 2019; Yao et al., 2019; Lv et al., 2022; Saxena et al., 2022) exploit textual names or descriptions of entities and relations to facilitate representation learning. Among the two categories, the performance of text-based ones generally lags behind that of embedding-based ones, due to inadequate entity representation optimization of the former (Wang et al., 2022).

The recent robust text-based method, SimKGC (Wang et al., 2022), tackles the sub-optimal representation problem by contrasting carefully sampled in-batch, pre-batch and self negatives, inspiring us to leverage hard example mining (Kalantidis et al., 2020; Hu et al., 2020) to unlock the potential of contrastive learning for text-based KGC methods. Specifically, there

[†]Corresponding authors.

Findings of the Association for Computational Linguistics: ACL 2023, pages 5866–5878

are two essential properties for negative sample selection:

- **Vicinity** (Robinson et al., 2021; Tabassum et al., 2022a) is a metric to measure the distance between the negatives and the anchor. Lower vicinity can push the model to learn better representation boundaries.
- **Uniformity** (Wang and Isola, 2020; Tabassum et al., 2022a) is another one to measure the representativeness of the negatives. Higher uniformity means negative samples are uniformly distributed on the hypersphere and will make the produced representation more generalized.

To improve vicinity, we pioneeringly introduce a generative method to produce negatives in KGC. Unlike previous efforts that use in-batch or pre-batch entities, we apply a sequence-to-sequence model as a generator, which is trained to directly generate the tail entity t for a given $< h, r >$ pair. With a sampling strategy, we can get multiple tail entities from the generator, and the incorrect ones among them constitute our negatives for efficient contrastive learning. As shown in Figure 1, since the generated negative samples are decoded from the same hidden state as the correct answer, they inherently have better anchor similarity in the semantic space.

To improve uniformity, we further incorporate the generator itself with a novel self-information-enhanced contrastive learning strategy. In particular, we apply a prefix tree to obtain the conditional frequency of the tokens at each decoding step, which is converted to self-information (Shannon, 1948) and utilized as a token-wise weight for contrastive learning in the generator. Typically, the generator prefers to predict more frequent entity descriptions which consist of tokens of low self-information. Incorporating the weighting mechanism forces the generator to produce more distinctive representations for more informative tokens (e.g., in less frequent entity descriptions), yielding more diversified negatives for KGC.

In summary, our generative negative mining strategy innovatively balances the vicinity and uniformity of negative samples. Meanwhile, the hard negatives produced by the generator are natural candidates for inference. Compared to other text-based methods that have to enumerate all entities for inference, we can pick a small number of generated entities as a high-quality candidate set, significantly accelerating inference speed for large-scale KGs. We evaluate our method on three popular KGC datasets (WN18RR (Dettmers et al., 2017), FB15K237 (Toutanova and Chen, 2015a), and Wikidata (Wang et al., 2019)), and the empirical results verify the superiority of our method.

Our contributions are as follows:

- We pioneeringly incorporate generative methods to produce hard negatives in text-based KGC for better contrasting effects.
- We design a novel self-information-enhanced contrastive learning method for the negative generator, providing us with high-quality negative samples.
- The negatives we generated are proven to systematically balance the hardness and diversity (or vicinity and uniformity, more formally), leading to competitive performances on three KGC benchmarks.

2 Methodology

The link prediction task of KGC is to infer the missing triples given an incomplete $\mathcal{G}$. In this section, we mathematically describe the proposed **G**enerative **H**ard **N**egative Mining (GHN) method in detail. The core idea of GHN is to obtain better negatives in terms of vicinity and uniformity. We first introduce the overall process of GHN in 2.1 which includes how to generate and use these negative samples to enhance KGC model training. Then, to further improve the vicinity and uniformity of generated negatives, we propose a self-information-enhanced token-level contrastive learning method which is discussed in 2.2. Finally, we introduce how to facilitate inference on KGC tasks in 2.3.

2.1 Model Architecture

Our GHN model mainly consists of two parts, a generator that aims to provide hard negatives to facilitate efficient contrastive learning and a predictor which is supposed to predict the tail entity given a <head entity, relation> pair. Then, we will present how to leverage generated negatives for better training efficiency.

Generator Saxena et al. (2022) has proved that a simple encoder-decoder Transformer (Vaswani et al., 2017) can perform knowledge graph completion in the form of a sequence-to-sequence task.

The performance of directly using a Transformer to predict the tail entity may not appeal. However, using a sequence-to-sequence model to generate negatives has ideal properties, such as scalability (the inference speed is independent of the scale of KG) and the better vicinity of generated negatives, considering that the generation of negatives and golden entities share the same hidden state.

First, we construct a mapping between the textual descriptions and entities/relations following Saxena et al. (2022); Wang et al. (2022). The textual mentions we used are provided by KG-BERT (Yao et al., 2019). Then we formally describe how to convert link prediction queries to textual queries. Given a link prediction query $(h, r, ?)$, where h and r represent the head entity and the relation, respectively. The textual query is the concatenation of the text mentions of h and r, separated by $|$. For example, given a link prediction query $(\text{St.Louis}, \text{time zones location}, ?)$, the converted text query t_{hr} is $'\text{St.Louis} \mid \text{time zones location}'$. The text query is the input of our generator, which is supposed to output the correct answer $'\text{Central Time Zone}'$.

To obtain diverse and semantically similar negatives to the golden entity, we use the sampling strategy to get multiple predictions for the same input query. Specifically, we get a probability distribution over tokens at each step of decoding, then sample a token from the distribution and autoregressively decode until the 'stop' token. This procedure will be repeated multiple times to produce a negative set $\mathcal{N}^q$ for each query. Finally, we drop the correct entity to get a set of negatives.

Predictor Following SimKGC (Wang et al., 2022), our GHN adopts a bi-encoder architecture. The first encoder BERT_{hr} takes the text query t_{hr} as input and produces the relation-aware embedding e_{hr}, where t_{hr} is the same text query as the input of the generator. Similarly, the second encoder BERT_t takes the textual mentions of the tail entity and produces its embedding e_t. We use mean pooling followed by L2 normalization to obtain these embeddings. Then we compute cosine similarity to these embeddings and predict the one with the largest score:

$$\begin{aligned} &\phi(h, r, t_i) = \cos\left(e_{hr}, e_{t_i}\right), \\ &\underset{t_i}{\text{argmax}}\ \phi(h, r, t_i),\ t_i \in \mathcal{E}, \end{aligned} \tag{1}$$

Training with Generated Negatives We first briefly introduce the objective of the generator. The generator is optimized by minimizing a cross-entropy loss with the golden entity:

$$\mathcal{L}_{CE} = -\frac{1}{K}\sum_{k=1}^{K} \log p\left(y_k^* \mid \mathbf{y}_{<k}, \mathbf{x}\right), \tag{2}$$

where K is the length of the target entity. The probability $p\left(y_k^* \mid \mathbf{y}_{<k}, \mathbf{x}\right)$ is calculated by multiplying the last layer decoder hidden state of the generator s_k and the softmax embedding matrix $\mathbf{W_s}$ together:

$$p\left(y_k \mid \mathbf{y}_{<k}, \mathbf{x}\right) \propto \exp\left(\mathbf{W_s} \cdot s_k\right), \tag{3}$$

where $\mathbf{x}$ and $\mathbf{y}$ denote the input sequence and output sequence respectively.

Unlike other negative mining methods (Xiong et al., 2020; Kalantidis et al., 2020), our method does not need to calculate weights for negatives. Instead, we directly mix the generated negatives and in-batch negatives to produce a negative set $\mathcal{N} = \mathcal{N}_{ib} \cup \mathcal{N}_{gen}$, where $\mathcal{N}_{ib}$ and $\mathcal{N}_{gen}$ denote the set of in-batch negatives and generated negatives respectively. In training, the generated negative samples $\mathcal{N}_{gen}^q \subseteq \mathcal{N}^q$ are sampled from corresponding negative set $\mathcal{N}^q$:

$$p(n \mid q) = \sum_{t=1}^{T} \log\left(p\left(n_t \mid \mathbf{y}_{<t}, \mathbf{x}\right)\right), n \in \mathcal{N}^q. \tag{4}$$

After constructing the negative set, we use InfoNCE (Chen et al., 2020) loss to train the predictor:

$$\mathcal{L}_p = -\log \frac{e^{\phi(h,r,t)/\tau}}{e^{\phi(h,r,t)/\tau} + \sum_{i=1}^{|\mathcal{N}|} e^{\phi\left(h,r,t_i'\right)/\tau}}, \tag{5}$$

where τ is a learnable parameter.

2.2 Self-information enhanced Training

We have used a sequence-to-sequence model to provide hard negatives for the predictor. Through further investigation, we noticed that the target of generative link prediction is different from traditional generative tasks like machine translation. We call tasks like link prediction as target-constraint generation tasks. Specifically, one is able to obtain all available targets in link prediction, which is impossible for traditional generative tasks. Intuitively, more informative tokens are more critical than others. We hypothesize that informative tokens may impact the diversity of generated results since the generator prefers to predict more frequent

entity descriptions which consist of tokens of low self-information, typically. Our investigation (see section 4.3 for more details) verified this hypothesis.

Therefore, we propose a self-information-enhanced contrastive learning method to further improve the uniformity of generated negatives by producing more distinctive representations for more informative tokens. First, we will introduce how to obtain the self-information (Shannon, 1948) of tokens. Then, we will introduce how to facilitate training for the generator of GHN with self-information.

Self-Information is a measure of the information content associated with the outcome of a random variable (Shannon, 1948). Since all possible outputs are acquirable in KGC, we can easily calculate self-information for all tokens.

We first construct a prefix tree $\mathcal{T}$ where nodes are annotated with tokens from the vocabulary of entity mentions. The children of each node $t \in \mathcal{T}$ indicate all the available continuations from the prefix defined traversing the prefix tree from the root to t. Then, given the previous sequence generated by a sequence-to-sequence model $w_1, \cdots, w_{n-1}$ and the current label token w_n, the self-information of the current label token $\mathrm{I}(w_n|\mathbf{w}_{<n})$ can be calculated as:

$$\begin{aligned} \mathrm{I}(w_n|\mathbf{w}_{<n}) &= -\log\left(\mathrm{P}\left(w_n|\mathbf{w}_{<n}\right)\right) \\ \mathrm{P}\left(w_n|\mathbf{w}_{<n}\right) &:= \frac{\mathrm{Count}(w_n|\mathcal{A}_{w_1,\cdots w_{n-1}})}{\left|\mathcal{A}_{w_1,\cdots w_{n-1}}\right|}, \end{aligned} \tag{6}$$

where $\mathcal{A}$ is all the children nodes determined by the prefix $w_1, \cdots, w_{n-1}$ on the prefix tree. Then we average the self-information of tokens that are present in the different positions of the vocabulary to produce the self-information for each token:

$$\mathrm{I}(w_n) = \frac{1}{K}\sum_{k=1}^{K}\mathrm{I}(w_n^k), \tag{7}$$

where w_n^k denotes the kth token in vocabulary and K is the total times of occurrences of w_n.

Self-Information-aware Contrastive Learning Token-level contrastive learning methods widen the representations of tokens that have different labels. Since we want the representation of informative tokens to be more distinctive, we assign a set of weights for each contrasting pair (higher weight for contrasting pairs with higher self-information tokens in it) according to their self-information. As a result, the model can assign more distinctive representations for tokens with more self-information and facilitate the generation of these tokens.

Formally, given a target token t_i and a negative sample t_j, we assign a soft weight $w(i, j)$ for them. The weight is determined by the self-information of both t_i and t_j. We use the same strategy as token-level contrastive learning (Zhang et al., 2021) to build up the negative and positive samples. Given the sequence of former target tokens and current target label t_i, we formulate the information-aware soft weight $w(i, j)$ for t_i and corresponding negative sample t_j as:

$$w(i, j) = \lambda \mathrm{I}(t_i) \cdot \mathrm{I}(t_j), \tag{8}$$

where λ is a hyperparameter. In our implementation, the mean value of information-aware weights for all negatives of each anchor is normalized to be 1. The contrastive learning object is:

$$\mathcal{L}_{\mathrm{CL}} = -\log \frac{e^{cos(s_a, s_p)}}{\sum_{i=1}^{N} w(a, i) e^{cos(s_a, s_i)}}, \tag{9}$$

where s_a, s_p, and s_i denote the representation of anchor, positive token, and negative tokens, respectively. Finally, we weight the traditional generative object $\mathcal{L}_{\mathrm{CE}}$ and the self-information-aware contrastive objective $\mathcal{L}_{\mathrm{CL}}$ by a hyperparameter γ to compute the final loss $\mathcal{L}_{\mathrm{GEN}}$ for the generator:

$$\mathcal{L}_{\mathrm{GEN}} = (1 - \gamma)\mathcal{L}_{\mathrm{CE}} + \gamma \mathcal{L}_{\mathrm{CL}}. \tag{10}$$

2.3 Inference

Most text-based methods have to produce representation for each entity (Wang et al., 2022; Yao et al., 2019). Considering the large scale of modern KGs, the efficiency of link prediction is of concern. Therefore, we propose a simple generation-classification two-stage method to perform high-efficiency inference. Specifically, the generator we used to produce hard negatives is also able to provide candidates for inference. In the first stage, the generator produces a set of candidates $\mathcal{N}_c$ by the same sampling strategy as we discussed in 2.1. In the second stage, the predictor computes scores for candidates and then predicts the one with the largest score:

$$\underset{t_i}{\mathrm{argmax}}\ \phi(h, r, t_i), \quad t_i \in \mathcal{N}_c. \tag{11}$$

The number of candidates $|\mathcal{N}_c|$ is empirically set to be 50. Be aware that the two-stage inference is not

dataset	#entity	#relation	#train	#valid	#test
WN18RR	40,943	11	86,835	3,034	3,134
FB15k-237	14,541	237	272,115	17,535	20,466
Wikidata5M	4,594,485	822	20,614,279	5,163	5,163

Table 1: Statistics of the datasets used in this paper.

enforced for our method, it can speed up inference only when the scale of KG is relatively large.

3 Experiments Setup

3.1 Datasets

We evaluate our method on three widely-used link prediction datasets, **WN18RR** (Dettmers et al., 2017), **FB15K237** (Toutanova and Chen, 2015b), and **Wikidata** (Wang et al., 2019). The statistics are shown in Table 1. **WN18RR** consists of about 41k synsets and 11 relations from WordNet (Miller, 1995). This dataset is constructed by removing the inverse relations from FB15k (Bordes et al., 2013) which suffers from test set leakage. **FB15k-237** is a subset of Freebase (Bollacker et al., 2008), it consists of about 15k entities and 237 relations. Similar to the WN18RR dataset, Dettmers et al. (2017) removed the inverse relations to address the test set leakage problem. **Wikidata5M** has much larger scale which consists of $\sim$ 4.6M entities, 822 relations, and $\sim$ 20 million triples. Following most of KGC methods, we use the transductive version of Wikidata5M. All the textual descriptions for WN18RR and FB15k-237 are provided by Yao et al. (2019). The Wikidata5M dataset already provides descriptions for all entities and relations.

3.2 Baselines

TransE (Bordes et al., 2013) constructs a relation-specific translation from the head entity to the tail entity. Trouillon et al. (2016) introduces complex number embeddings.Tucker (Balažević et al., 2019) facilitate KGC based on Tucker decomposition of the binary representation of triples. DistMult (Yang et al., 2014) models the three-way interactions in triples. RotatE (Sun et al., 2019b) models a relation as rotation in complex space. DKRL (Xie et al., 2016) leverages a CNN network to obtain text representations. KEPLER (Wang et al., 2019) uses a Transformer-based encoder trained with the typical KGE objective and the masked language modeling objective. MTL-KGC (Kim et al., 2020) proposes a multi-task learning method that can learn more relational properties. KG-BERT (Yao et al., 2019) and StAR (Wang et al., 2021) both leverage pre-trained language models to produce the representation of entities. SimKGC (Wang et al., 2022) train the model with much more negatives by incorporating three types of negatives.

3.3 Implementation Details

The generator is implemented using the HuggingFace library (Wolf et al., 2019) with the pre-trained weight of BART-base (Lewis et al., 2019). The predictor is initialized with BERT (Devlin et al., 2018). And we first train the generator with $\mathcal{L}_{gen}$ as the objective until validation accuracy did not significantly increase for 5k steps. Most hyperparameters except training epochs are shared across all datasets to avoid dataset-specific tuning.

Follow SimKGC (Wang et al., 2022), the entity descriptions are truncated to a maximum of 50 tokens for a fair comparison. The size of $\mathcal{N}_{gen}^{q}$ is set to 10. We use AdamW optimizer with linear learning rate decay, the learning rate is initialized to 3×10^{-5}. Models are trained with batch size 1024. All experiments were performed using 4 NVIDIA A100 GPUs. For the WN18RR, FB15k-237, and Wikidata5M datasets, we train for 50, 10, and 1 epochs, respectively.

4 Experiment Results

4.1 Main Results

Table 2 shows the performance of the baselines and our method variants on WN18RR and FB15k-237 tasks (statistically significant with $p < 0.05$). While Table 3 further demonstrates the performance and inference speed of baselines and our methods on Wikidata5M dataset, which has a much larger scale. We have the following observations.

First, the superiority of GHN is significant in terms of performance. Compared to previous text-based methods, GHN shows consistent performance improvement in all three tasks. Compared to embedding-based methods, GHN has a significant performance advantage in WN18RR and Wikidata5M, while marginally trailing in FB15k-237. All of the text-based methods lag behind

Method	WN18RR				FB15k-237			
	MRR	**Hits@1**	**Hits@3**	**Hits@10**	**MRR**	**Hits@1**	**Hits@3**	**Hits@10**
embedding-based methods								
TransE	24.3	4.3	44.1	53.2	27.9	19.8	37.6	44.1
DistMult	44.4	41.2	47.0	50.4	28.1	19.9	30.1	44.6
RotatE	47.6	42.8	49.2	57.1	33.8	24.1	37.5	53.3
TuckER	47.0	44.3	48.2	52.6	**35.8**	**26.6**	**39.4**	**54.4**
text-based methods								
KG-BERT	21.6	4.1	30.2	52.4	-	-	-	42.0
MTL-KGC	33.1	20.3	38.3	59.7	26.7	17.2	29.8	45.8
StAR	40.1	24.3	49.1	70.9	29.6	20.5	32.2	48.2
SimKGC	66.6	58.7	71.7	80.0	33.6	24.9	36.2	51.1
GHN-SL	67.3	59.3	71.5	80.9	33.8	**25.1**	36.3	51.5
GHN	**67.8**	**59.6**	**71.9**	**82.1**	**33.9**	**25.1**	**36.4**	**51.8**

Table 2: Main results for **WN18RR** and **FB15k-237** datasets. GHN-SL denotes the proposed GHN without self-information-enhanced training. Following previous work, **MRR** and **Hits@k** are reported under the filtered setting (Bordes et al., 2013) for a fair comparison.

embedding-based methods on FB15k-237 dataset for now. The possible reason is that many links in FB15k-237 dataset are not predictable based on the available information (Cao et al., 2021) and may harm the training of text-based models.

Second, though GHN already surpasses most baselines without the self-information-enhanced training method, this training method further improves the performance on link prediction tasks, especially reflected in Hits@10 metric. This improvement shows that the self-information-enhanced training method may lead to a more generalized KGC model by introducing more diverse negatives. Please refer to section 4.3 for more details.

Third, GHN also significantly accelerates the inference speed on Wikidata5M dataset. To perform link prediction on a test set, SimKGC needs to do $2 \times |\mathcal{T}| + |\mathcal{E}|$ times BERT forward pass, where $|\mathcal{T}|$ and $|\mathcal{E}|$ denote the size of the test set and the number of entities in corresponding KG, respectively. While GHN needs up to $2 \times |\mathcal{T}| \times (|\mathcal{N}_c| + 1)$ times BERT forward pass and $\sim 2 \times |\mathcal{N}_c| \times |\mathcal{T}|$ times BART generation process. Since in large-scale datasets such as Wikidata5M, $|\mathcal{T}| << |\mathcal{E}|$ (5,163 test samples and $\sim$ 4.6M entities for Wikidata5M), the significant acceleration is as expected. Note that the two-stage inference is not enforced, in the case that there are too many queries or the KG's scale is relatively small, GHN has the same inference speed as SimKGC.

4.2 Overall Impacts of Generative Hard Negative mining

One of the key designs in GHN is leveraging a sequence-to-sequence model to generate hard negatives. To further investigate the impact of generated negatives, we construct an approximate nearest neighbor negative sampling strategy (ANN) inspired by Xiong et al. (2020); Kalantidis et al. (2020) and replace the generative negative sampling strategy in GHN with it. Specifically, we use the current predictor to find entities that are close to the golden tail entity among all entities as the hard negatives. Since the computation cost of producing representations for all entities is unaffordable, we randomly sample 500 entities for each query and pick the top 10 entities by their representations' cosine similarity with the representation of the golden tail entity. Table 4 shows the performance comparison between GHN, ANN, and four variants of SimKGC with different negative sampling strategies.

Intuitively, since ANN picks the nearest entities to the anchor as negatives which are more "harder" than generated ones, it should further improve link prediction performance. However, it does not show the expected performance improvement. The possible reason is that the balance between vicinity and uniformity is critical since these picked negatives apparently have better vicinity than generated neg-

Method	Wikidata5M				
	MRR	Hits@1	Hits@3	Hits@10	Inference Speed
embedding-based methods					
TransE	25.3	17.0	31.1	39.2	-
RotatE	29.0	23.4	32.2	39.0	-
text-based methods					
DKRL	16.0	12.0	18.1	22.9	$\sim 3\times$
KEPLER	21.0	17.3	22.4	27.7	$\sim 5\times$
SimKGC	35.8	31.3	37.6	44.1	$\sim 4\times$
GHN-SL	36.2	31.5	37.8	44.8	$\sim 1\times$
GHN	**36.4**	**31.7**	**38.0**	**45.3**	$\sim 1\times$

Table 3: Main results with inference speed comparison for **Wikidata5M** datasets.

Method	WN18RR			
	MRR	H@1	H@3	H@10
IB	67.1	58.5	**73.1**	81.7
IB+PB	66.6	57.8	72.3	81.7
IB+SN	66.7	58.8	72.1	80.5
IB+SN+PB	66.6	58.7	71.7	80.0
ANN	66.7	58.9	71.4	79.2
GHN	**67.8**	**59.6**	71.9	**82.1**

Table 4: Ablation study on WN18RR dataset. NN denotes that we replace the proposed negative sampling strategy with the approximate nearest neighbor sampling strategy to obtain negative samples. IB, IB+PB, IB+SN, and IB+SN+PB denote the four variants of SimKGC(IB, PB, and SN denote the variant of SimKGC with in-batch, pre-batch, and self-negative sampling strategy, respectively (Wang et al., 2022).).

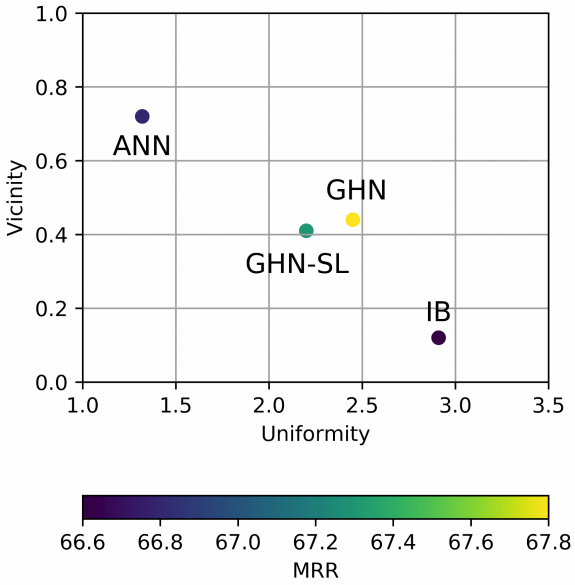

Figure 2: Vicinity, uniformity, and MRR metrics.The x/y axes correspond to $\mathcal{M}_{uni}$ and $\mathcal{M}_v$, respectively. GNH and GHN-SL denote the proposed negative sampling strategy with or without self-information, respectively. IB denotes the in-batch sampling strategy. ANN denotes the nearest neighbor negative sampling strategy.

atives but lead to worse performance(more details in section 4.3). Besides, GHN also shows performance superiority to SimKGC with four different negative sampling strategies, which further demonstrates the effectiveness of our generative negative sampling strategy.

4.3 Effects on Representation Learning

To further examine what makes GHN excel and how the self-information-enhanced training method impacts the generated negatives. We conduct a further investigation from the perspective of representation learning by computing two metrics $\mathcal{M}_v$ and $\mathcal{M}_{uni}$ to measure the vicinity and uniformity for negatives produced by different sampling strategies, inspired by Wang and Isola (2020) and Tabassum et al. (2022b). Specifically, we use the average cosine distance of negatives between the anchor to measure the vicinity $\mathcal{M}_v$ for each negative sampling strategy. To measure the uniformity $\mathcal{M}_{uni}$ of negatives' distribution on the hypersphere, we compute the logarithm of the average pairwise Gaussian potential between all negatives' embedding. Here we choose two variants of GHN, ANN, and IB negative sampling strategies to conduct the comparison. The reason we did not add PB and SN negative sampling strategy to the comparison is as follows: PB sampling strategy shares almost the

same $\mathcal{M}_v$ and $\mathcal{M}_{uni}$ with IB. SN sampling strategy only picks the head entity itself in the query as the only negative sample and it is impossible to calculate $\mathcal{M}_{uni}$ for this sampling strategy.

Figure 2 shows that our GHN is able to produce negatives with better uniformity but worse vicinity compared to the ANN sampling strategy, while the opposite is true when compared to IB sampling strategy. Meanwhile, our generative sampling strategy has a better MRR score than both NN and IB strategies (67.8 v.s. 66.7 and 67.1 MRR score).

Leveraging self-information-aware contrastive learning (SL) to enhance the training of the generator is another key design in GHN. Figure 2 also demonstrates that SL training method significantly improves the uniformity of generated negatives while the improvement of vicinity is also visible, which empirically verified that SL method could lead to a more diverse generation.

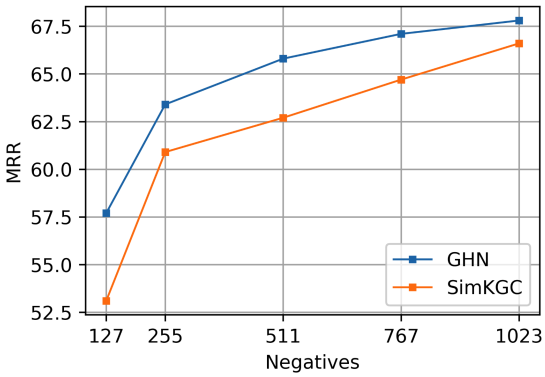

Figure 3: MRR score of GHN and SimKGC with different number of negatives.

4.4 Effects on Training Efficiency

An abundance of negative samples is critical for contrastive learning (Chen et al., 2020; Wang et al., 2022) while causing more demands on GPU memory. Figure 3 illustrates the MRR metrics on WN18RR dataset vary for GHN and SimKGC under different numbers of negatives for each query. By introducing generated negatives, our GHN leads to much efficiency training: GHN only needs 511 negatives to reach a similar performance as SimKGC does with 1023 negatives. Note that we do not count the pre-batch negatives used in SimKGC for a fair comparison. This observation re-emphasizes the effectiveness of the proposed generative negative mining strategy.

5 Related Work

Text-based KGC methods aims to leverage text information in KGs to assist KGC. DKRL (Xie et al., 2016) obtained text representations by CNN. KEPLER (Wang et al., 2019) leveraged the typical KGE training objective into the masked language model training. MTL-KGC (Kim et al., 2020) proposed to learn more relational properties in KGs with multi-task learning method. KG-BERT (Yao et al., 2019) and StAR (Wang et al., 2021) leveraged pre-trained language models to produce entity embeddings in the cross-encoder style and bi-encoder style, respectively. SimKGC (Wang et al., 2022) increase the number of negatives by incorporating three types of negatives and achieved notable performance improvements.

Negative Mining aims to find property negatives to assist contrastive representation learning (Mao et al., 2021). The most effective way is to use the samples within the same mini-batch for negative candidates (Wang et al., 2022; Chen et al., 2020). Another effective and widely used method is to store negative samples with an asynchronous update mechanism (Zhang et al., 2018; He et al., 2019), which allows more negative candidates to be involved during training. To make contrastive pairs difficult to discriminate, Zhang et al. (2013); Chen et al. (2017); Xiong et al. (2020) compute propensity score for each <query, sample candidate> pair. Ying et al. (2018) further utilizes PageRank score to calculate weights for negative candidates. Motivated by the generative adversarial networks (Goodfellow et al., 2014), Wang et al. (2020) proposed a sampling strategy by adaptively receiving knowledge-aware rewards. And Hu et al. (2020) proposed to adversarially generate the hard negative samples together with the representation network.

6 Conclusion

We have presented our method for KGC tasks, which incorporate generative methods with a novel self-information-enhanced training strategy to produce high-quality negatives. And we further reveal that the proposed method systematically balances uniformity and vicinity, two essential properties for negative sample selection. Empirical results on three widely-used datasets (WN18RR, FB15k-237, Wikidata5M) have verified the superiority of our method.

7 Limitations

For now, the superiority of the proposed two-stage inference speed-up method cannot adapt to inductive datasets since the generated sequences are difficult to map to unseen entities. Therefore, we will explore how to efficiently perform KGC under the inductive setting in the future.

Like the other text-based KGC methods, our GHN lag behind embedding-based methods on FB15k-237 dataset. Cao et al. (2021) claims that many links in the FB15k-237 dataset are not predictable based on the information in the KG and we hypothesize this may harm the training of text-based models. In the future, we intend to examine this more thoroughly.

Acknowledgements

We thank anonymous reviewers for their valuable comments. This work was supported in part by the National Key R&D Program of China under Grants No.2022YFF0902703.

References

Ivana Balažević, Carl Allen, and Timothy M Hospedales. 2019. Tucker: Tensor factorization for knowledge graph completion. *arXiv preprint arXiv:1901.09590*.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250.

Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *NIPS*.

Yixin Cao, Xiang Ji, Xin Lv, Juanzi Li, Yonggang Wen, and Hanwang Zhang. 2021. Are missing links predictable? an inferential benchmark for knowledge graph completion. *arXiv preprint arXiv:2108.01387*.

Long Chen, Fajie Yuan, Joemon M. Jose, and Weinan Zhang. 2017. Improving negative sampling for word representation using self-embedded features. *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*.

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

Wanyun Cui, Yanghua Xiao, Haixun Wang, Yangqiu Song, Seung won Hwang, and Wei Wang. 2019. Kbqa: Learning question answering over qa corpora and knowledge bases. *ArXiv*, abs/1903.02419.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2017. Convolutional 2d knowledge graph embeddings. In *AAAI Conference on Artificial Intelligence*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *NIPS*.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross B. Girshick. 2019. Momentum contrast for unsupervised visual representation learning. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9726–9735.

Qianjiang Hu, Xiao Wang, Wei Hu, and Guo-Jun Qi. 2020. Adco: Adversarial contrast for efficient learning of unsupervised representations from self-trained negative adversaries. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1074–1083.

Jin Huang, Wayne Xin Zhao, Hongjian Dou, Ji rong Wen, and Edward Y. Chang. 2018. Improving sequential recommendation with knowledge-enhanced memory networks. *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*.

Shaoxiong Ji, Shirui Pan, E. Cambria, Pekka Marttinen, and Philip S. Yu. 2020. A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE Transactions on Neural Networks and Learning Systems*, 33:494–514.

Yannis Kalantidis, Mert Bulent Sariyildiz, No'e Pion, Philippe Weinzaepfel, and Diane Larlus. 2020. Hard negative mixing for contrastive learning. *ArXiv*, abs/2010.01028.

Bosung Kim, Taesuk Hong, Youngjoong Ko, and Jungyun Seo. 2020. Multi-task learning for knowledge graph completion with pre-trained language models. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 1737–1743, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

Xin Lv, Yankai Lin, Yixin Cao, Lei Hou, Juanzi Li, Zhiyuan Liu, Peng Li, and Jie Zhou. 2022. Do pre-trained models benefit knowledge graph completion? a reliable evaluation and a reasonable approach. In *Findings*.

Kelong Mao, Jieming Zhu, Jinpeng Wang, Quanyu Dai, Zhenhua Dong, Xi Xiao, and Xiuqiang He. 2021. Simplex: A simple and strong baseline for collaborative filtering. *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*.

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

Joshua David Robinson, Ching-Yao Chuang, Suvrit Sra, and Stefanie Jegelka. 2021. Contrastive learning with hard negative samples. In *International Conference on Learning Representations*.

Apoorv Saxena, Adrian Kochsiek, and Rainer Gemulla. 2022. Sequence-to-sequence knowledge graph completion and question answering. In *Annual Meeting of the Association for Computational Linguistics*.

Apoorv Saxena, Aditay Tripathi, and Partha Pratim Talukdar. 2020. Improving multi-hop question answering over knowledge graphs using knowledge base embeddings. In *Annual Meeting of the Association for Computational Linguistics*.

Claude E. Shannon. 1948. A mathematical theory of communication. *Bell Syst. Tech. J.*, 27:623–656.

Haitian Sun, Tania Bedrax-Weiss, and William W. Cohen. 2019a. Pullnet: Open domain question answering with iterative retrieval on knowledge bases and text. *ArXiv*, abs/1904.09537.

Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019b. Rotate: Knowledge graph embedding by relational rotation in complex space. *arXiv preprint arXiv:1902.10197*.

Afrina Tabassum, Muntasir Wahed, Hoda Eldardiry, and Ismini Lourentzou. 2022a. Hard negative sampling strategies for contrastive representation learning. *ArXiv*, abs/2206.01197.

Afrina Tabassum, Muntasir Wahed, Hoda Eldardiry, and Ismini Lourentzou. 2022b. Hard negative sampling strategies for contrastive representation learning. *arXiv preprint arXiv:2206.01197*.

Kristina Toutanova and Danqi Chen. 2015a. Observed versus latent features for knowledge base and text inference. In *Workshop on Continuous Vector Space Models and their Compositionality*.

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

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *International conference on machine learning*, pages 2071–2080. PMLR.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Bo Wang, Tao Shen, Guodong Long, Tianyi Zhou, Ying Wang, and Yi Chang. 2021. Structure-augmented text representation learning for efficient knowledge graph completion. In *Proceedings of the Web Conference 2021*, pages 1737–1748.

Liang Wang, Wei Zhao, Zhuoyu Wei, and Jingming Liu. 2022. Simkgc: Simple contrastive knowledge graph completion with pre-trained language models. *ArXiv*, abs/2203.02167.

Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. 2017. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*, 29:2724–2743.

Tongzhou Wang and Phillip Isola. 2020. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *International Conference on Machine Learning*, pages 9929–9939. PMLR.

Xiang Wang, Yaokun Xu, Xiangnan He, Yixin Cao, Meng Wang, and Tat-Seng Chua. 2020. Reinforced negative sampling over knowledge graph for recommendation. *Proceedings of The Web Conference 2020*.

Xiaozhi Wang, Tianyu Gao, Zhaocheng Zhu, Zhiyuan Liu, Juan-Zi Li, and Jian Tang. 2019. Kepler: A unified model for knowledge embedding and pre-trained language representation. *Transactions of the Association for Computational Linguistics*, 9:176–194.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *AAAI Conference on Artificial Intelligence*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Ruobing Xie, Zhiyuan Liu, Jia Jia, Huanbo Luan, and Maosong Sun. 2016. Representation learning of knowledge graphs with entity descriptions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30.

Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul Bennett, Junaid Ahmed, and Arnold Overwijk. 2020. Approximate nearest neighbor negative contrastive learning for dense text retrieval. *ArXiv*, abs/2007.00808.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2014. Embedding entities and relations for learning and inference in knowledge bases. *arXiv preprint arXiv:1412.6575*.

Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. Kg-bert: Bert for knowledge graph completion. *ArXiv*, abs/1909.03193.

Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, and Jure Leskovec. 2018. Graph convolutional neural networks for web-scale recommender systems. *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*.

Tong Zhang, Wei Ye, Baosong Yang, Long Zhang, Xingzhang Ren, Dayiheng Liu, Jinan Sun, Shikun Zhang, Haibo Zhang, and Wen Zhao. 2021. Frequency-aware contrastive learning for neural machine translation. *ArXiv*, abs/2112.14484.

Weinan Zhang, Tianqi Chen, Jun Wang, and Yong Yu. 2013. Optimizing top-n collaborative filtering via dynamic negative item sampling. *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*.

Yongqi Zhang, Quanming Yao, Yingxia Shao, and Lei Chen. 2018. Nscaching: Simple and efficient negative sampling for knowledge graph embedding. *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, pages 614–625.

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?
Not applicable. Left blank.

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

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

B ☑ Did you use or create scientific artifacts?

1 3 4 6

☑ B1. Did you cite the creators of artifacts you used?
1 3 4 6

☐ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
Not applicable. Left blank.

☑ 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)?
3 4

☐ 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?
Not applicable. Left blank.

☐ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
Not applicable. Left blank.

☑ 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.
3

C ☑ Did you run computational experiments?

4

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
3

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?
3

☑ 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?
4

☑ 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.)?
3

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

Left blank.

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

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

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

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

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