Don't Mess with Mister-in-Between: Improved Negative Search for Knowledge Graph Completion

Fan Jiang and **Tom Drummond** and **Trevor Cohn**
School of Computing and Information Systems
The University of Melbourne, Victoria, Australia
fan.jiang1@student.unimelb.edu.au
{tom.drummond, t.cohn}@unimelb.edu.au

Abstract

The best methods for knowledge graph completion use a 'dual-encoding' framework, a form of neural model with a bottleneck that facilitates fast approximate search over a vast collection of candidates. These approaches are trained using contrastive learning to differentiate between known positive examples and sampled negative instances. The mechanism for sampling negatives to date has been very simple, driven by pragmatic engineering considerations (e.g., using mismatched instances from the same batch). We propose several novel means of finding more informative negatives, based on searching for candidates with high lexical overlaps, from the dual-encoder model and according to knowledge graph structures. Experimental results on four benchmarks show that our best single model improves consistently over previous methods and obtains new state-of-the-art performance, including the challenging large-scale Wikidata5M dataset. Combing different strategies through model ensembling results in a further performance boost.

1 Introduction

A Knowledge Graph (KG) is a structured form of human knowledge consisting of entities, facts, relationships between any pair of entities, and semantic descriptions of entities. As important structures that store millions of data records that represent a part of human knowledge, KGs have been proven to bring substantial benefits to a wide range of applications, including commonsense question answering (Yasunaga et al., 2021) and reasoning (Ren and Leskovec, 2020). Knowledge Graph Completion (KGC) supports the automatic construction or completion of a KG by finding the missing entity or link in incomplete triples.

Graph embedding and textual embedding methods are two mainstream techniques for KGC problems. The former typically map entities and relations into fixed dense vectors and maximises the probability of valid triples using specially-designed scoring functions (Bordes et al., 2013; Sun et al., 2019); while the latter additionally uses available textual descriptions associated with entities to gather more information (Wang et al., 2021b). Surprisingly, textual embedding methods lag behind graph embedding methods, perhaps due to their extra computational overheads in encoding textual inputs and, thus, inefficiency in incorporating a sufficient number of negatives (i.e., incorrect KG triples) to learn discriminative KG embeddings.

Recently, Wang et al. (2022) found that the key to making textual embedding methods outperform their graph embedding counterparts is to adopt a dual-encoder structure and train using contrastive learning to differentiate between positive training instances and randomly sampled negative instances (Karpukhin et al., 2020). Although their technique outperformed various graph embedding methods, achieving a new state-of-the-art, their means of negative sampling is not optimal: this in-batch negative method has been shown to be inefficient in training dual encoders (Lu et al., 2021). Instead, as they are highly similar to positives in terms of topics and lexicons, the use of so-called 'hard' negatives have been shown to result in better models in information retrieval (Xiong et al., 2021) through providing a more informative training signal and faster convergence. However, their effectiveness has not yet been established for KGC.

Therefore, to fill this gap, in this work, we aim to systematically investigate the effects of various hard negative sampling strategies for dual-encoder-based KGC. Specifically, we construct negative samples using three different ways. Our approach first evaluates the utility of negatives that share high lexical similarity with the head entity or the correct tail entity in terms of entity names and text descriptions. Based on the knowledge graph structures, we alternatively search negatives from the head or tail entity's local neighbourhood, hypothesising

Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics, pages 1818–1832

that the neighbourhoods of a certain entity that are not directly connected to it are highly related, but not so related to be false negatives (i.e., positives). Lastly, we investigate sampling so-called 'hard negatives' from top-k predictions generated by a baseline dual-encoder KGC model, as negatives that receive high scores are believed to be important and difficult to distinguish. In addition, in order to reduce possible false negatives, we also experimented with two variant neural negative sampling strategies according to heuristics. In summary, our contributions are:[1]

1. To the best of our knowledge, we are the first to systematically investigate the impacts of different types of negative sampling strategies for dual-encoder-based KGC.
2. We explore how best to combine the benefits of different negative sampling strategies to obtain further performance gains.
3. We compare our proposed negative searching methods on four benchmark datasets of different scales. Experimental results demonstrate that our best model significantly outperforms baselines, establishing a new state-of-the-art on all datasets, while ensembling leads to further performance gains.

2 Background

2.1 Task Formulation

In this paper, we deal with the task of predicting missing entities in knowledge graph completion. Formally, given a knowledge graph $\mathcal{G}$ which has a set of entities $\mathcal{E}$ and predefined relations $\mathcal{R}$, the tail entity retrieval task $(h, r, ?)$ requires retrieving a list of entities $\{t_1, t_2, \ldots, t_k\}$ from the entity set $\mathcal{E}$, ranked by their relevance to this head-relation pair (h, r).

2.2 Dual Encoder for KGC

Following the current state-of-the-art approach to KGC (Wang et al., 2022), we use a dual encoder framework.

Figure 1 shows the architecture. In this approach, a pair of encoders E_{hr} and E_t, which are usually initialised by pretrained language models (e.g., BERT (Devlin et al., 2019)), are used to map the head entity and relation (h, r) and tail entity t into dense vectors, respectively. More specifically, to encode the tuple (h, r), we first concatenate the

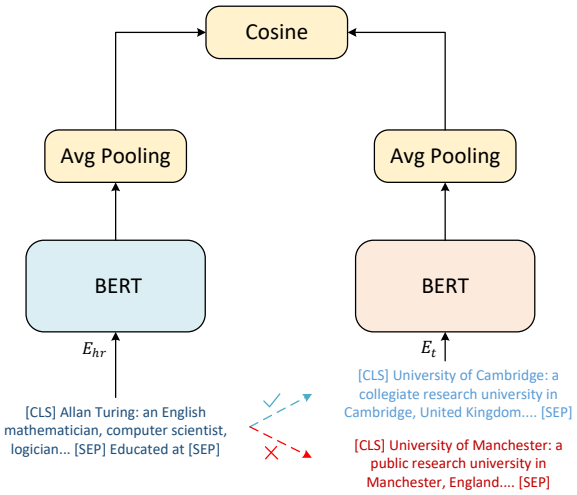

Figure 1: The Dual-Encoder Model for Knowledge Graph Completion, which learns to discriminate positive tail entities from negative ones.

text description of head entity h and relation r as: [CLS] *h: entity description* [SEP] *r* [SEP] and feed the sequence into a BERT encoder E_{hr}. Similarly, we use another encoder E_t to encode the tail entity t and its description, using an analogous input method. Mean pooling is used to obtain fix-sized embeddings, which are then l_2-normalised. The relevance score between (h, r) and t is calculated as the inner dot product of the two l_2-normalised vectors:

$$s(h, r, t) = \frac{E_{hr}(h; r) \cdot E_t(t)}{||E_{hr}(h; r)|| \cdot ||E_t(t)||} \quad (1)$$

The dual encoders are normally trained to maximise the similarity scores of all positive triples (h, r, t). Here, we use the InfoNCE loss, following Wang et al. (2022):

$$\mathcal{L}_{hr \to t} = -\log \frac{s(h, r, t)}{\sum_{t' \in \mathcal{E}} s(h, r, t')} \quad (2)$$

where the denominator sums over all $N = |\mathcal{E}|$ entities in the KG.

Since N is usually very large, the common practice is to use specific negative sampling strategies to select a subset of negative samples to replace the full normalisation term in Equation 2. In practice, the selection of negative samples is crucial to the performance of a trained model (Zhang and Stratos, 2021). So the critical question becomes how to sample informative negatives which could generate meaningful signals for training an effective model.

[1]Source code is available at https://github.com/Fantabulous-J/Improved-Negative-Search-for-KGC.

3 Negative Sampling Strategies

3.1 In-Batch Negatives

This strategy treats as negatives the tail entities of other samples in the same mini-batch, which is a cheap way to obtain a large number of negatives (Karpukhin et al., 2020). However, the downside of this approach is that the number of negatives is limited to batch size, and it usually requires a large batch size to work well, which is not always feasible with limited computing resources.

3.2 Hard Negatives

'Hard' negatives are informative negatives, which are difficult to distinguish as they share similar characteristics with true positives (e.g., high lexical overlap or semantic similarity). They normally receive larger similarity scores from the model, which in turn results in larger loss gradients and thus larger parameter updates in training. Selecting negatives from them can effectively mitigate the diminishing gradient norms when using uninformative negative instances (i.e., in-batch negatives), thus providing more optimal training signals and leading to faster convergence speed (Xiong et al., 2021). In this paper, we systematically investigate the effectiveness of different hard negative sampling strategies for knowledge graph completion. An illustrative example of the different types of hard negatives is shown in Table 1.

Sparse Negatives Hard negatives were first shown to be useful for improving the performance of dense passage retrievers in question answering (Karpukhin et al., 2020). Their approach, which we call **Sparse Negatives** henceforth, samples hard negatives from the top-k ranked list returned by a sparse retrieval system such as BM25 (Robertson and Zaragoza, 2009). We test the generality of this method to our task of knowledge graph completion. More specifically, given a KG triplet (h, r, t), we either use the concatenation of the head entity h and relation type r or the correct tail entity t[2] to query a sparse BM25 retriever to find top entities which share similar entity names or similar tokens in their entity descriptions.[3]

Structure-Aware Negative Entities in the local neighbourhood of an entity in the KG are typically semantically related, and can serve as good candidates (Ahrabian et al., 2020). For a triplet (h, r, t), we sample hard negatives from the n-hop neighbours of either the head entity h or the tail entity t, following random edges.[4]

Neural Negative samples hard negatives from the top-k predictions of a neural model (Xiong et al., 2021; Glass et al., 2021), which has been shown to boost the performance of neural retrievers. In this paper, we use a simple method of sampling static hard negatives from a fixed learned KGC model, which we denote as **Head-Relation Negative**:

Step 1: In-batch negatives are used to train a dual-encoder KGC model.

Step 2: FAISS (Johnson et al., 2021) is used to index all entities in the KG, using their dense vector encoding, E_t.

Step 3: Employ approximate search to find the top-k retrieved entities using head-relation pair (h, r) as the query, with encoding E_{hr}. Then any positive tail entity to this query are removed based on the training dataset and all other entities will be kept as hard negatives.

Although sampled negatives are both difficult and informative, the Neural Negative method may end up including many false negatives. This is because for a good KGC model, if the same head-relation pair (h, r) appears in both training and test graphs, the top-k predictions will likely include correct answers in the test set. [5] Therefore, we propose another two variant negative sampling strategies to limit false negatives while maintaining informativeness. In particular, **Entity Similar Negative** uses either the head entity h or the tail entity t of a given KG triplet (h, r, t) as the query to find the top-k nearest entities in the embedding space of a fixed learned KGC model. **Replaced Head-Relation Negative** replaces the head h in (h, r, t) with another entity h' that is nearest to h in the embedding space and uses (h', r) as query to sample negatives from the top-k predictions.

3.3 Other Negatives

Pre-batch negatives extend in-batch negatives by using stale entity embeddings from previous n batches, which can be considered a cheaper way

[2]The concatenation of an entity's name and text description will be used.

[3]Whether the head h or the tail t is used to retrieve negatives is based on the development set performance.

[4]We use $n = 2$, which we found to work well.

[5]This is more true to neural negatives than previous two methods, as neural models are superior in capturing relation semantics. See Table 1 for more intuitive examples.

Head Entity **Relation** **Tail Entity**	land_reform_NN_1: a redistribution of agricultural land (especially by government action) hypernym reform_NN_1: a change for the better as a result of correcting abuses; justice was for sale before the reform of the law courts
Sparse	1. landing_NN_3: the act of coming down to the earth (or other surface); "the plane made a smooth landing"; "his landing on his feet was catlike" 2. amphibious_landing_NN_1: a military action of coordinated land, sea, and air forces organized for an invasion; "MacArthur staged a massive amphibious landing behind enemy lines" 3. enderby_land_NN_1: a region of Antarctica between Queen Maud Land and Wilkes Land; claimed by Australia
Structure	1. event_planner_NN_1: someone who plans social events as a profession (usually for government or corporate officials) 2. price-fixing_NN_1: control (by agreement among producers or by government) of the price of a commodity in interstate commerce 3. lawlessness_NN_1: a state of lawlessness and disorder (usually resulting from a failure of government)
Neural	1. reform_NN_3: self-improvement in behavior or morals by abandoning some vice; "the family rejoiced in the drunkard's reform" 2. improvement_NN_1: a change for the better; progress in development 3. reform_NN_2: a campaign aimed to correct abuses or malpractices

Table 1: Examples of hard negatives using different types of sampling strategies, using the head entity and relation as query for sparse and neural negatives but only the head entity for structure negatives (see top block). Sparse negatives typically overlap with the head entity in surface form but ignore the relation, while structure negatives can find entities with similar types. Neural retrievers can successfully find entities that are informative as hard negatives, but the risk of including false negatives also increases. For instance, both *improvement_NN_1* and *reform_NN_2* are feasible answers to the above query.

to expand negatives compared to memory bank approaches (He et al., 2020). Self-negatives regard the head entity h in $(h, r, ?)$ as the hard negative to reduce the model's reliance on spurious text matches. We include these two kinds of negatives, following Wang et al. (2022).

3.4 Negatives for Training

During training, we use one of the hard negative sampling strategies proposed in §3.2 together with a combination of in-batch negatives, pre-batch negatives, and self-negatives. Since extra hard negatives are used, the total number of instances used for loss calculation in Eq. 2 will be increased. For fair comparison, we reduce the batch size to ensure the total number of negatives used in training remains identical to our baseline *SimKGC* method (Wang et al., 2022). More details can be found in Appendix B.

4 Model Ensembling

Since we propose multiple hard negative sampling strategies in §3.2, we train several models, each with different types of hard negatives. It is natural to combine these models to achieve further performance improvements through model ensembling. Inspired by Lu et al. (2021), we explore two methods to ensemble results from multiple models.

Rank Fusion Reciprocal Rank Fusion (RRF) (Cormack et al., 2009) is a simple yet effective algorithm for merging ranking results from multiple retrievers, and is a prevailing technique in information retrieval. Its utility in knowledge graph completion has not yet been evaluated. The technique works by merging the ranking positions from the various models. Suppose we have a query (h, r) and each model $p \in P$ returns a ranking list $\{\text{Rank}_e^p \,|e \in \mathcal{E}\}$, the final ranking score is:

$$RRF(e) = \sum_{p \in P} \frac{w_p}{\text{Rank}_e^p}$$

Rank_e^p is the ranking position of an entity e returned by a model p, ranging from $[1, |\mathcal{E}|]$. w_p is the weight of a model p, which is tuned based on the performance on the development set.

Embedding Fusion An alternative aggregation method is to compute the weighted sum of ranking scores from different models, i.e., $\sum_{p=1}^{P} w_p E_{hr}^p E_t^p$. This can be implemented cheaply using the models' embeddings, which has the benefit of allowing for efficient vector indexing and search. Specifically, for each query (h, r), we obtain vectors from each models' encoder, which are then scaled by a model specific weight (tuned manually) and concatenated. Each entity $e \in \mathcal{E}$, is also embedded by each model

and the vectors concatenated. This allows the aggregation method to be implemented as a simple dot product:[6]

$$s(h,r,t) = [w_P E^P_{hr}, \ldots, w_1 E^1_{hr}]^T [E^P_t, \ldots, E^1_t]$$

5 Experiments

5.1 Datasets

Our method is evaluated on WN18RR (Dettmers et al., 2018), FB15k-237 (Toutanova and Chen, 2015), DBPedia500k (Shi and Weninger, 2017) and Wikidata5M (*Transductive Setting*) (Wang et al., 2021b) datasets, where the number of entities ranges from 15K to 4.6M. More details about dataset statistics are shown in Table 7.

Following previous work (Wang et al., 2022), we focus on retrieving the missing entity of triples $(h, r, ?)$ and $(?, r, t)$, effectively doubling dataset sizes.[7] The adopted evaluation metrics are Mean Reciprocal Rank (MRR) and Hits@k($k \in \{1, 3, 10\}$). More specifically, MRR is calculated as $\text{MRR} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{\text{Rank}_i}$, where Rank_i is the rank of the correct tail entity in the predicted outputs and N is the total number of triples in the test set. Similarly, Hits@k is the proportion of correct tails that appear in the top-k ranked candidates.

We follow Bordes et al. (2013) and Wang et al. (2022) by conducting evaluations under *filter* settings, where for a given incomplete triple $(h, r, ?)$, the prediction scores of all its correct answers[8] except for the target to be predicted are removed. Both tail entity prediction $(h, r, ?)$ and head entity prediction $(?, r, t)$ are conducted, with their averages on all metrics reported.

5.2 Experimental Settings

We replicated the state-of-the-art SimKGC model (Wang et al., 2022) using PyTorch (Paszke et al., 2019) and Transformers (Wolf et al., 2020), and treat it as the baseline in our experiments. The uncased BERT base model is used for model initialisation.[9]

Most hyperparameters are adopted from Wang et al. (2022), and newly introduced hyperparameters are determined through grid search. We choose the best model according to Hits@1 on the dev set, which is then evaluated on the test set for all experiments. More details are in Appendix A.

5.3 Overall Results

Tables 2 and 3 show the results of our models compared with a range of high-performing graph embedding and text embedding models over four KGC datasets. We observe that the performance of our replicated baseline is competitive with that of Wang et al. (2022), achieving slightly better results on FB15k-237 and Wikidata5M but slightly worse on WN18RR. Moreover, our best single model either matches or outperforms the state-of-the-art results on all datasets for most metrics.

By looking into the models trained using different hard negative sampling strategies, the behaviours are quite different in each dataset. More specifically, simply taking the n-hop neighbours as structure-aware negatives results in significant improvement on WN18RR but marginal increases or even negative impacts on the other three datasets. Similar effects are also observed for sparse negatives. Our model achieves the best results on FB15k237, DBPedia500k, and Wikidata5M when using replaced head-relation negatives, which shows our heuristics are useful and can indeed lead to better performance. Gains over the baseline when using entity similar negatives are also significant, but it lags behind the other two types of neural negatives most of the time. Overall, we can conclude that there is no single negative sampling strategy that outperforms on all datasets. We believe this is due in part to the distinct characteristics of each dataset, which we explore in §6.1.

Furthermore, by ensembling models trained using different types of negatives at inference time, we can observe performance gains up to 1.3% MRR and 1.8% H@1 over the best single model. For model ensembling methods, we find that embedding fusion is more beneficial when improving the precision (MRR and H@1), while rank fusion is helpful to boost the recall (H@3 and H@10), especially on DBPedia500k. Besides, both methods do not help much on FB15k237 with only marginal gains. However, both ensembling methods come at the cost of increased inference latency with a factor of N, the number of ensembled models, which

[6] For both fusion methods, the weights were tuned through grid search. Empirically, we found that single model whose performance is better was generally given a higher weight than worse performing methods. See Table 8 and Appendix C.3 for more details.

[7] For reversed triples, a special r^{-1} relation type is used. For instance, we convert a triple (?, educated at, Cambridge University) to (Cambridge University, reverse educated at, ?). We did not conflate the reversed relations with existing relations, although this may be beneficial (e.g., hyponym ≡ reverse hypernym.)

[8] Triples that appear in training, validation and test sets.

[9] https://huggingface.co/bert-base-uncased

Method	WN18RR				FB15k-237				Wikidata5M			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
Graph Embedding Approach												
TransE (Bordes et al., 2013)*	24.3	4.3	44.1	53.2	27.9	19.8	37.6	44.1	25.3	17.0	31.1	39.2
DisMult (Yang et al., 2014)*	44.4	41.2	47.0	50.4	28.1	19.9	30.1	44.6	-	-	-	-
RotatE (Sun et al., 2019)*	47.6	42.8	49.2	57.1	33.8	24.1	37.5	53.3	29.0	23.4	32.2	39.0
TuckER (Balazevic et al., 2019)*	47.0	44.3	48.2	52.6	35.8	26.6	39.4	**54.4**	-	-	-	-
Text Embedding Approach												
KG-BERT (Yao et al., 2019)*	21.6	4.1	30.2	52.4	-	-	-	42.0	-	-	-	-
MTL-KGC (Kim et al., 2020)*	33.1	20.3	38.3	59.7	26.7	17.2	29.8	45.8	-	-	-	-
StAR (Wang et al., 2021a)*	40.1	24.3	49.1	70.9	29.6	20.5	32.2	48.2	-	-	-	-
DKRL (Xie et al., 2016)*	-	-	-	-	-	-	-	-	16.0	12.0	18.1	22.9
KEPLER (Wang et al., 2021b)*	-	-	-	-	-	-	-	-	21.0	17.3	22.4	27.7
SimKGC (Wang et al., 2022)*	66.7	58.8	72.1	80.5	33.6	24.9	36.2	51.1	35.8	31.3	37.6	44.1
SimKGC replicated	65.0	55.6	71.5	**81.4**	33.8	25.3	36.5	51.2	36.8	32.5	38.3	44.7
+ Head-Relation	64.3	57.1	68.6	77.6	36.2	27.4	39.2	53.7	40.2	36.4	41.8	46.8
+ Entity Similar	66.4	59.5	70.1	79.1	35.4	26.9	38.4	52.3	39.3	35.5	40.9	46.0
+ Replaced Head-Relation	65.3	58.8	68.8	77.5	**36.5**	**27.6**	**39.9**	54.2	**41.0**	**37.0**	**42.7**	**48.0**
+ Sparse	66.7	59.3	71.3	79.3	34.6	25.9	37.5	52.2	36.7	32.3	38.4	44.5
+ Structure-Aware	**67.8**	**60.9**	**72.5**	80.0	33.1	24.4	35.7	50.7	38.0	33.4	39.7	46.2
Ensemble												
Rank Fusion	68.9	61.9	**72.9**	**81.7**	36.6	27.6	**40.1**	**54.3**	41.7	37.6	**43.5**	48.7
Embedding Fusion	**69.2**	**62.7**	72.8	81.1	**36.7**	**27.8**	40.0	**54.3**	**42.0**	**38.1**	**43.5**	**49.0**

Table 2: Evaluation results on the test set of WN18RR, FB15k-237 and Wikidata5M (*Transductive Setting*) datasets. $*$ indicates results directly copied from Wang et al. (2022).

Method	MRR	H@1	H@3	H@10
TransE (Bordes et al., 2013)*	7.4	3.1	10.1	14.5
TransH (Wang et al., 2014)*	7.4	3.2	10.1	14.6
TransR (Lin et al., 2015)*	7.3	3.5	9.9	13.5
TransD (Ji et al., 2015)*	7.4	3.2	10.1	14.5
SimKGC replicated	25.7	18.9	27.6	**39.5**
+ Head-Relation	26.3	20.5	28.6	37.0
+ Entity Similar	26.3	20.7	28.4	36.5
+ Replaced Head-Relation	**27.1**	**21.1**	**29.3**	38.1
+ Sparse	24.9	18.4	26.7	37.6
+ Structure-Aware	25.6	18.8	27.5	39.4
Ensemble				
Rank Fusion	**28.3**	**21.6**	**30.6**	**41.7**
Embedding Fusion	27.9	21.5	30.0	40.0

Table 3: Evaluation results on the DBPedia500k dataset. $*$ indicates the results obtained by running the open-source OpenKE toolkit (Han et al., 2018).

hinders their utility for real-time deployment.

6 Analysis

6.1 Why the Effects Vary by Dataset

Next, we analyse why the benefits of different hard negative sampling strategies vary from dataset to dataset. We employ two measurements, namely *Difficulty* and *False Negative Rate*, to draw the connections to the performance on specific datasets. *Difficulty* is measured by the average model score between hard negatives and their corresponding head-relation pairs. More specifically, for a given KG triplet (h_i, r_i, t_i) and its associated hard negative pool $\mathcal{N}_i = \{t_j | 1 \leq j \leq k\}$, the difficulty is computed as follows:

$$D_i = \frac{1}{|\mathcal{N}_i|} \sum_{t_j \in \mathcal{N}_i} s(h_i, r_i, t_j)$$

where $s(h_i, r_i, t_j)$ is the score predicted by the *SimKGC replicated* model.[10] The overall difficulty is the average over training, $\frac{1}{|\mathcal{D}|} \sum_i^{|\mathcal{D}|} D_i$.

On the other hand, *False Negative Rate* is the proportion of hard negatives which are correct answers that appear in development and test graphs:[11]

$$FNR = \frac{1}{|D|} \sum_{i}^{|\mathcal{D}|} \frac{1}{|\mathcal{N}_i|} \sum_{t_j \in \mathcal{N}_i} \mathbb{I}(< h_i, r_i, t_j > \in \mathcal{T})$$

where $\mathcal{T}$ is the development and test sets. $\mathbb{I}(*) = 1$ if the corresponding triple appears in the development or test set; otherwise it is 0.

[10]The model is fixed to allow comparison of different negative sampling strategies.

[11]The real false negative rate will be higher than what we measure due to the incomplete nature of knowledge graphs.

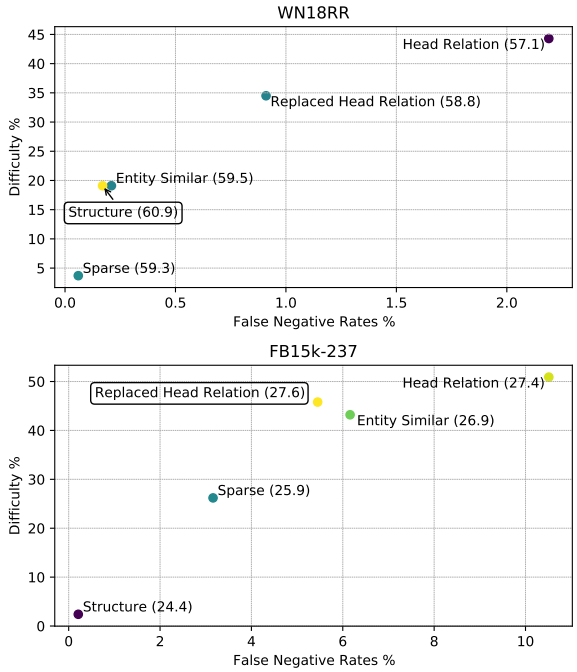

Figure 2: The difficulty and false negative rates of hard negatives extracted using different sampling strategies on WN18RR and FB15k237 datasets. Colours of points and numbers represent the Hit@1 score. The best results are circled with round boxes.

A model that is trained using hard negatives with high difficulty and low false negative rate is expected to achieve better performance. Figure 2 compares the difficulty and false negative rates for hard negative sampling strategies over two datasets. The most effective strategies are different on the two datasets: for Wordnet (WN18RR) the accuracy (Hit@1) is most sensitive to the false negative rate (horizontal axis), while for Freebase (FB15k237) it is most sensitive to difficulty (vertical) – despite the false negative rates being considerably higher on this dataset. We ascribe the difference to the underlying dataset. Wordnet is a sparsely-connected graph, and has relation types that connect entities with hierarchical structures. For instance, the *hypernym* and *derivationally related form* relations which satisfy the transitivity property account for 74% of triples in the graph. Including negatives with high difficulty for such relations (i.e., false negatives on the hierarchy that are not directly connected to a specific head entity) will inevitably encourage the model to learn embeddings that destroy such structures. However, the best-performing structure-aware negative sampling methods only sample negatives from local neighbourhoods, which we believe can effectively

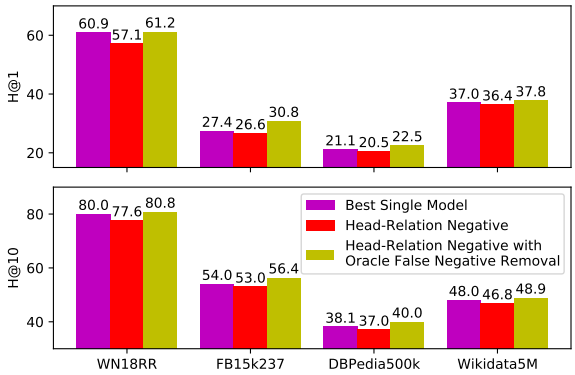

Figure 3: The comparison between the best single model and the model trained using **Head-Relation** negatives with and without oracle false negative removal.

reduce such side-effects. In contrast, Freebase is a much more denser graph compared to Wordnet (8× in terms of node degrees), with significantly fewer entities but more diverse relations. Moreover, over 70% triples include relations that have high arity (e.g., *has part*) and many-to-many mappings. Thus, the chance of having false negatives within globally sampled hard negatives will be much higher than that on Wordnet in nature. However, since FB15k237 is a much more difficult task compared to WN18RR, highly difficult negatives should be used to encourage the model to learn embeddings that can discriminate correct tails from very similar ones, and the importance of false negatives are comparatively less important. We leave the systematic analysis of different negative sampling methods with respect to KG structures as future work.

6.2 Oracle Upper Bound

One natural following question is how much improvement we may achieve if we remove all false negatives when using the most difficult head-relation hard negatives to train a model. As shown in Figure 3, the performance increases substantially in terms of H@1 and H@10 across four datasets when simply removing false negatives. This shows that there is much room for improvement and designing effective false negative elimination methods could potentially fill the performance gap and result in better-performing models. Besides, our best single model can obtain results that are close to the upper bound on WN18RR and Wikidata5M, but still lags behind on the other two datasets.

Rel Category		1-TO-1	1-TO-M.	M.-TO-1	M.-TO-M.
Size %		0.94	6.32	20.45	72.29
SimKGC	Forward	57.3	**4.6**	73.2	24.8
	Backward	64.6	37.1	**13.6**	15.3
	Avg.	60.9	20.8	**43.4**	20.1
Our model	Forward	**60.9**	4.3	**75.8**	**28.9**
	Backward	**67.7**	**40.8**	10.5	**17.4**
	Avg.	**64.3**	**22.6**	43.1	**23.2**

Table 4: Detailed results (H@1) on the FB15k-237 test dataset, broken down by category of relationships and prediction directions. Our model refers to the best single model trained with replaced head-relation negatives.

WN18RR	Model	MRR	H@1	H@3	H@10
Seen	SimKGC	65.5	56.1	72.0	**81.7**
	Our model	**68.2**	**61.3**	**72.6**	80.0
Unseen	SimKGC	59.1	48.6	65.2	77.6
	Our model	**63.7**	**54.1**	**70.2**	**81.0**
Wikidata5M	**Model**	**MRR**	**H@1**	**H@3**	**H@10**
Seen	SimKGC	36.6	32.3	38.1	44.5
	Our model	**40.7**	**36.7**	**42.4**	**47.7**
Unseen	SimKGC	43.4	38.4	45.3	51.6
	Our model	**49.7**	**45.6**	**51.9**	**57.2**
Inductive Unseen	SimKGC	42.5	37.8	43.9	51.6
	Our model	**47.4**	**43.3**	**49.3**	**55.1**

Table 5: Results on test examples when only containing seen and unseen entities, respectively. Our model refers to the best single model in each dataset.

6.3 Relation Category

To further understand the behaviour of our model, we follow Bordes et al. (2013) by classifying triples into different groups based on the category of relationships. Relationships are broken down into four categories according to the cardinalities of their head and tail arguments: one-to-one (1-TO-1), one-to-many (1-TO-M.), many-to-one (M.-TO-1) and many-to-many (M.-TO-M.). For a given relation r, if the average number of heads h appearing in the dataset for a tuple (r, t) is lower than 1.5, the head argument will be labeled as 1 and M. otherwise. The same is applied to the tail argument.

Table 4 shows the detailed results of four categories on the FB15k-237 dataset, together with the forward and backward prediction results. Firstly, we can find that both models perform the best on triples with 1 on the tail side, while predicting the M side is significantly more difficult. Secondly, our method can beat the baseline on most relation categories. Thirdly, the substantial improvement on 1-TO-1 relations (+3.4%) shows that our model is making more precise decisions. Besides, by looking into specific prediction directions, we find that the performance mainly comes from predicting triples with 1 on the target side. When the target side contains multiple answers, adding hard negatives even hurts the performance, especially in the backward direction of M.-TO-1. Further analysis finds that its false negative rate is almost three times the overall rate (15.9% vs 5.5%), and we believe it is the high false negative rate that misleads the model and results in negative impacts on this specific relation category.

6.4 Generalise to Unseen Entities

Textual embedding methods are known to generalise better to unseen entities than graph embedding ones (Wang et al., 2021a). We conduct experiments to testify whether their generalisation ability can be further improved by using harder negatives. We split test data based on whether the head or target entity is unseen in training. 210 out of 3134 and 159 out of 5163 test triples include unseen entities on WN18RR and Wikidata5M, respectively. Furthermore, we use another *Inductive* setting on Wikidata5M, which has different data splits compared to the *Transductive* setting. We extract 336 test triples from the whole graph of *Induction* setting which include entities unseen to the *Transductive* setting's training graph. The detailed results are shown in Table 5. We observe our best model achieves considerable improvements on three unseen entity settings across all metrics. Moreover, the absolute improvements are significantly higher than those on the seen entity setting, especially on recall metrics (e.g., H@3 and H@10). This demonstrates learning from hard negatives leads entity embeddings that generalise better.

7 Related Work

Knowledge Graph Completion KGC has been extensively studied for many years as a popular research topic. Conventional KGC methods adopt graph embedding methods to map entity and relation into low-dimensional dense vectors and design various scoring functions to measure the plausibility of KG triples, including TransE (Bordes et al., 2013), DistMult (Yang et al., 2014), ComplEx (Trouillon et al., 2016) and RotatE (Sun et al., 2019). Recent text embedding methods choose to include additional text descriptions related to en-

tities by making use of large pretrained language models, including KG-BERT (Yao et al., 2019), MTL-KGC (Kim et al., 2020) and StAR (Wang et al., 2021a). In this work, we follow Wang et al. (2022) by adopting the simple yet effective text embedding-based method using powerful pretrained language models. The above graph-based methods, which also rely on negatives to learn entity and relation embeddings, are orthogonal to the sampling strategies proposed in this work. We believe our techniques could lead to improvements when applied to graphs, but we leave this for future work.

Dual Encoder for Contrastive Learning A Dual Encoder, or Bi-Encoder, which adopts two encoders without weight sharing for feature encoding, has been widely used in many tasks, including image learning (He et al., 2020) and information retrieval (Karpukhin et al., 2020). Typically, an image with two different augmented views or query-document pairs is encoded into vectors separately by a dual encoder. The model learns to minimise the distance between positive pairs and push negative pairs further apart in the embedding space. Inspired by previous work, we decouple the encoding of (h, r) and t by dual encoder and use the contrastive learning framework to learn effective knowledge embeddings.

Hard Negatives for Contrastive Learning Hard negatives have been identified to be extremely helpful in learning better representations, including image learning (Robinson et al., 2021) and information retrieval (Karpukhin et al., 2020). For example, the DPR model (Karpukhin et al., 2020) mines hard negatives using a sparse retriever BM25. Xiong et al. (2021) proposed to sample hard negatives from the model itself by theoretically and empirically verifying such negatives can result in larger gradient norms and thus faster convergence speed. Zhang and Stratos (2021) also found doing negative sampling from the model being optimised leads to better performance, as they argued that contrastive learning is a biased estimator and sampling negatives from the model itself can reduce such bias. We follow this direction and propose various hard negative search methods in this paper and show that they can substantially improve KGC.

Negative Sampling Strategies for KGC Most KGC works employ a simple negative sampling strategy by corrupting the head entity h or tail entity t of a correct KG triplet (h, r, t) with uniformly sampled random entities from the whole knowledge graph (Yao et al., 2019). However, such easy negatives are identified to provide limited training signal, since most of them produce small scores and nearly zero gradients late in training (Sun et al., 2019). Therefore, various improved negative sampling strategies have been proposed. Self-adversarial negative sampling (Sun et al., 2019) uses the distribution generated by the model being optimised to weight sampled negatives. GAN-based methods (Wang et al., 2018) are also effective in extracting informative negatives but suffer from inefficiency and high training costs. NSCaching (Zhang et al., 2019) regards negative triplets with large scores as important and maintains a cache of such triples from which negatives are sampled. Ahrabian et al. (2020) takes the head or tail entity's n-hop neighbours as negatives and evaluates their utility on graph embedding methods. Our work also aims to explore the effects of improved negative sampling strategies for KGC but with different model architectures (dual-encoder vs graph embeddings). Many of the above negative sampling strategies are orthogonal to our work and we believe they have the potential to be employed within our method for further empirical gains.

8 Conclusion

In this paper, we successfully improve a powerful dual-encoder-based KGC model by introducing various improved negative search methods and investigating their combinatorial effects. Empirical results on four benchmarks with different scales confirm the superiority of our proposed methods, significantly beating a wide range of competitive methods and achieving state-of-the-art performance. For future work, we are interested in eliminating false negatives contained in sampled hard negatives. An exciting future direction is to use a model to filter out false negatives and potentially even identify pseudo positives, for more accurate models.

Limitations

Although our method is efficient and introduces no extra cost during inference time, it does incur the additional training cost of retrieving different types of hard negatives, which roughly doubles the time for completing the whole training pipeline. The model ensembling methods, especially the embedding fusion one, need to build indexes for entity embeddings and require extra cost for storage. De-

spite these drawbacks, the training of our methods has a fairly modest footprint by modern standards, taking about 36 hours of a server with 4×A100 GPUs and 200G CPU RAM for the largest datasets, Wikidata5M.

As an empirical study, we provide observations under different design choices for sampling hard negatives. We hope our findings can shed lights on future work. A theoretical analysis about each proposed negative sampling strategy with connection to KG properties (e.g., sparsity) would certainly strengthen our claims, but is out of scope of this paper. In addition, the state-of-the-art dual-encoder-based KGC model is used to verify the sampling methods proposed in this work. One would expect to test their generalisation to other underlying KGC methods, e.g., graph embedding methods, which we leave as future work.

Moreover, adapting the dual-encoder-based KGC model and our proposed negative sampling methods to multilingual KGs, e.g., for a KG with concepts in different languages, or multi-modal settings, for a KG with concepts in the form of images, videos, or audios would further test the generalisation ability, which can be a promising research direction for future work.

Acknowledgements

We thank the anonymous reviewers for their helpful feedback and suggestions. The first author is supported by the Graduate Research Scholarships funded by the University of Melbourne. This research was undertaken using the LIEF HPC-GPGPU Facility hosted at the University of Melbourne. This Facility was established with the assistance of LIEF Grant LE170100200.

References

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

Ivana Balazevic, Carl Allen, and Timothy Hospedales. 2019. TuckER: Tensor factorization for knowledge graph completion. 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)*, pages 5185–5194, Hong Kong, China. Association for Computational Linguistics.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc.

Gordon V. Cormack, Charles L A Clarke, and Stefan Buettcher. 2009. Reciprocal rank fusion outperforms condorcet and individual rank learning methods. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '09, page 758–759, New York, NY, USA. Association for Computing Machinery.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2d knowledge graph embeddings. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1).

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, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Michael Glass, Gaetano Rossiello, Md Faisal Mahbub Chowdhury, and Alfio Gliozzo. 2021. Robust retrieval augmented generation for zero-shot slot filling. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1939–1949, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Xu Han, Shulin Cao, Xin Lv, Yankai Lin, Zhiyuan Liu, Maosong Sun, and Juanzi Li. 2018. OpenKE: An open toolkit for knowledge embedding. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 139–144, Brussels, Belgium. Association for Computational Linguistics.

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

Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Knowledge graph embedding via dynamic mapping matrix. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 687–696, Beijing, China. Association for Computational Linguistics.

Jeff Johnson, Matthijs Douze, and Herve Jegou. 2021. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, 7(3):535–547.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online. Association for Computational Linguistics.

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.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015*.

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*, AAAI'15, page 2181–2187. AAAI Press.

Jing Lu, Gustavo Hernandez Abrego, Ji Ma, Jianmo Ni, and Yinfei Yang. 2021. Multi-stage training with improved negative contrast for neural passage retrieval. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6091–6103, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. Curran Associates Inc., Red Hook, NY, USA.

Hongyu Ren and Jure Leskovec. 2020. Beta embeddings for multi-hop logical reasoning in knowledge graphs. In *Neural Information Processing Systems*.

Stephen Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: Bm25 and beyond. *Found. Trends Inf. Retr.*, 3(4):333–389.

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

Baoxu Shi and Tim Weninger. 2017. Open-world knowledge graph completion.

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

Kristina Toutanova and Danqi Chen. 2015. Observed versus latent features for knowledge base and text inference. In *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*, pages 57–66, 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 *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, page 2071–2080. JMLR.org.

Bo Wang, Tao Shen, Guodong Long, Tianyi Zhou, Ying Wang, and Yi Chang. 2021a. Structure-augmented text representation learning for efficient knowledge graph completion. In *Proceedings of the Web Conference 2021*, WWW '21, page 1737–1748, New York, NY, USA. Association for Computing Machinery.

Liang Wang, Wei Zhao, Zhuoyu Wei, and Jingming Liu. 2022. SimKGC: Simple contrastive knowledge graph completion with pre-trained language models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4281–4294, Dublin, Ireland. Association for Computational Linguistics.

Peifeng Wang, Shuangyin Li, and Rong Pan. 2018. Incorporating gan for negative sampling in knowledge representation learning. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI'18/IAAI'18/EAAI'18. AAAI Press.

Xiaozhi Wang, Tianyu Gao, Zhaocheng Zhu, Zhengyan Zhang, Zhiyuan Liu, Juanzi Li, and Jian Tang. 2021b. 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. *Proceedings of the AAAI Conference on Artificial Intelligence*, 28(1).

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Ruobing Xie, Zhiyuan Liu, Jia Jia, Huanbo Luan, and Maosong Sun. 2016. Representation learning of

knowledge graphs with entity descriptions. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, page 2659–2665. AAAI Press.

Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul N. Bennett, Junaid Ahmed, and Arnold Overwijk. 2021. Approximate nearest neighbor negative contrastive learning for dense text retrieval. In *International Conference on Learning Representations*.

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

Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. Kg-bert: Bert for knowledge graph completion.

Michihiro Yasunaga, Hongyu Ren, Antoine Bosselut, Percy Liang, and Jure Leskovec. 2021. QA-GNN: Reasoning with language models and knowledge graphs for question answering. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 535–546, Online. Association for Computational Linguistics.

Wenzheng Zhang and Karl Stratos. 2021. Understanding hard negatives in noise contrastive estimation. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1090–1101, Online. Association for Computational Linguistics.

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

	WN18RR		FB15k-237	
	MRR	H@1	MRR	H@1
Best Single Model	**67.8**	**60.9**	**36.5**	**27.6**
Negative Combination	65.8	59.0	35.6	26.7

Table 6: Results when training a model by sampling from the union of all types of negatives.

A Hyperparameters

The learning rates are set to 5×10^{-5} on WN18RR, 3×10^{-5} on Wikidata5M and 1×10^{-5} on the remaining datasets. All models are trained using Adam optimizer (Kingma and Ba, 2015) with a warmup learning rate scheduler. The model is trained for 50, 10, 5, and 1 epochs on WN18RR, FB15k-237, DBPedia500k, and Wikidata5M. For each hard negative sampling strategy, we generate 30 negatives for each training example.[12] During each training step, we uniformly sample a subset of hard negatives from the pool for each training example, and the best number is chosen from $[1, 5]$. For both rank fusion and embedding fusion methods, their weights are shared and are tuned based on the performance on development sets. A summary of training details and hyperparameters is shown in Table 8.

B Number of Negatives for Training

For each training example in a mini-batch, we uniformly sample N negatives from its associated hard negative pool. We also treat the hard negatives and self-negatives of other examples in the same mini-batch as in-batch negatives. Suppose the batch size is B and pre-batches are M, the total number of instances used for loss calculation in Eq. 2 will be $(N + M + 2) \times B$ for each training example. By contrast, the number of negatives used by SimKGC is $(M+2) \times B$. If we keep the same batch size, the number of negatives used in our experiment will increase by $N \times B$, which would potentially weak our claims as more negatives are used for contrastive learning. To ensure fair comparison, we reduced the batch size so that the number of negatives used in our experiment is the same as Wang et al. (2022). For example, if we take $B = 768, N = 1, M = 1$ on WN18RR, the number of negatives equals to 3072; while we take $B = 1024$ for SimKGC, the number will also be 3072. Thus, our methods will not be affected by including more negatives. Similar settings apply to other datasets.

[12] Gold answers appearing in the training graph are removed for each (h, r). (i.e., $\mathcal{N} = \{t_j | (h, r, t_j) \notin \mathcal{D}\}_{j=1}^{k}$)

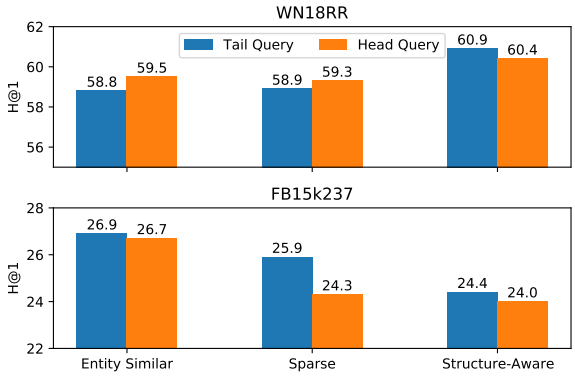

Figure 4: Results comparison when taking head or tail entity as query.

C Ablation Study

C.1 Negative Combination

Another possible way to benefit from all kinds of hard negatives is to train a model on their combinations. We experiment with training a model by uniformly sampling negatives from the union of all types of hard negatives generated from §3.2. As shown in Table 6, the model does not obtain improved results and this can even hurt the performance. We reason that although the combined negatives can provide more diverse supervisions, their false negative rates and difficulty also changes. More specifically, the false negative rate and difficulty on WN18RR change to 0.93% and 29.66%. Although more difficult negatives are included, the more important false negative rate is worse than the best structure-aware one. By contrast, the false negative rate in FB15k237 after combination reduces from Replaced Head Relation's 5.45% to 4.24%, but the more important difficulty also decreases from 45.8% to 33.7%, thus leading to inferior results.

C.2 Head Entity vs Tail Entity as Query

We analyse the performance difference between using head entity and tail entity as the query in the *sparse*, *structure-aware* and *entity similar* negative sampling strategies. Figure 4 shows that for similar and sparse negatives, using tail entity as query generally leads to worse performance on WN18RR; while in other cases, using the tail entity is more beneficial. One reason behind this is that taking the head entity as query would inevitably reduce

Dataset	#Ent	#Rel	Train	Dev	Test
WN18RR	40, 943	11	86, 835	3, 034	3, 134
FB15k-237	14, 541	237	272, 115	17, 535	20, 466
DBPedia500k	517, 475	654	3, 102, 677	10, 000	1, 155, 937
Wikidata5M	4, 594, 485	822	20, 614, 279	5, 163	5, 163

Table 7: Number of Entities, Relations and Triples in Train/Dev/Test splits of datasets used in our experiments.

Hyperparameters	WN18RR	FB15k-237	DBPedia500k	Wikidata5M
Learning rate	5e-5	1e-5	1e-5	3e-5
LR Scheduler	Linear Warmup	Linear Warmup	Linear Warmup	Linear Warmup
Warmup steps	400	400	400	400
Epochs	50	10	5	1
Batch Size for SimKGC	1024	1024	1024	1024
Batch size	768	512	512	768
Gradient clipping	10.0	10.0	10.0	10.0
#Hard negatives	1	3	3	1
Fusion weights	[0.1, 0.3, 0.3, 0.3, 1.2]	[0.6, 0.2, 0.9, 0.05, 0.01]	[1, 1, 1, 1, 1]	[0.9, 0.6, 1.2, 0.3, 0.45]

Table 8: Hyperparameter settings for different datasets. Entries in fusion weights correspond to Head-Relation, Entity Similar, Replaced Head-Relation, Sparse, and Structure-Aware sampling methods, respectively.

		MRR	H@1	H@3	H@10
Rank Fusion	Uniform	68.7	61.7	72.7	81.6
	Tuned	**68.9**	**61.9**	**72.9**	**81.7**
Embedding Fusion	Uniform	68.9	62.5	72.5	80.8
	Tuned	**69.2**	**62.7**	**72.8**	**81.1**

Table 9: Results comparison on WN18RR when using uniform and manually-tuned weights for model ensembling.

the difficulty but can avoid false negatives, a factor that is more significant for WN18RR. By contrast, the situation on FB15k237 is the other way around where the difficulty is more crucial. Again, we can confirm that there is no single design choice for each sampling strategy that works the best across datasets.

C.3 Uniform Weights for Ensembling

We show results when using uniform weights for model ensembling in Table 9. We find that simply using uniform weights has already boosted the performance. By carefully tuning the weights of each model through grid search, we can achieve further gains on all metrics, although the benefits are minimal. Making fusion weights learnable (Wang et al., 2021a) may lead to further improvements, which we leave as future work.

D Qualitative Analysis

Table 10 shows some examples of predictions from the SimKGC baseline and our best method. For the first example, both methods' predictions are wrong. However, our method ranks the correct tail entity much higher than the SimKGC baseline. Moreover, the top-1 prediction from our method *prosody_NN_1* is more reasonable as an answer compared to *articulation_NN_1*. For the second example, both methods fail according to the test set annotations. Our method returns correct tail entities based on our judgment. The same applies to SimKGC for its top-2 predictions. This also shows current automatic evaluation metrics cannot reflect the performance of KGC models precisely. For the third example, our method finds the correct answer out of other very semantically similar entities.

I. *Both SimKGC and our model fail, but our predictions are more related*	
Head Entity	accentuation_NN_1: the use or application of an accent; the relative prominence of syllables in a phrase
Relation	hypernym
Tail Entity	stress_NN_1: the relative prominence of a syllable or musical note (especially with regard to stress or pitch)
SimKGC (Rank 180)	1. articulation_NN_1: the aspect of pronunciation that involves bringing articulatory organs together so as to shape the sounds of speech 2. prosody_NN_1: the patterns of stress and intonation in a language 3. non-standard_speech_NN_1: speech that differs from the usual accepted, easily recognizable speech of native adult members of a speech community
Ours (Rank 108)	1. prosody_NN_1: *as above* 2. articulation_NN_1: *as above* 3. speech_pattern_NN_1: distinctive manner of oral expression; "he couldn't suppress his contemptuous accent".
II. *Both SimKGC and our model fail, but both models' predictions are correct based on human judgement*	
Head Entity	pea_family_NN_1: a large family of trees, shrubs, vines, and herbs bearing bean pods...
Relation	member meronym
Tail Entity	wild_pea_NN_1: any of various plants of the family Leguminosae that usually grow like vines.
SimKGC (Rank 7)	1. genus_sesbania_NN_1: small genus of tropical and subtropical leguminous herbs or shrubs or trees 2. genus_centrosema_NN_1: a genus of chiefly tropical American vines of the family Leguminosae... 3. torchwood_family_NN_1: resinous or aromatic chiefly tropical shrubs or trees
Ours (Rank 4)	1. genus_acacia_NN_1: large genus of shrubs and trees and some woody vines... 2. genus_sesbania_NN_1: *as above* 3. genus_dalbergia_NN_1: large genus of tropical trees having pinnate leaves and paniculate flowers...
III. *Our model succeeds but SimKGC fails*	
Head Entity	wive_VB_1: take (someone) as a wife
Relation	hypernym
Tail Entity	wed_VB_1: take in marriage
SimKGC (Rank 4)	1. wifely_JJ_1: befitting or characteristic of a wife 2. shack_up_VB_1: share living quarters; people who are not married and live together as a couple 3. wive_VB_2: marry a woman, take a wife
Ours (Rank 1)	1. **wed_VB_1:** ***as above*** 2. wifely_JJ_1: *as above* 3. shack_up_VB_1: *as above*

Table 10: Predictions of tail entities by the SimKGC baseline and our best model. The top-3 ranked outputs are reported, and the ranks of the correct results are also included. Correct predictions are in **bold**, and underline means predictions that do not exist in original KGs but are correct based on human judgment.