MixKG: Mixing for harder negative samples in knowledge graph

Feihu Che, Guohua Yang, Pengpeng Shao, Dawei Zhang, Jianhua Tao

Abstract

Knowledge graph embedding (KGE) aims to represent entities and relations into low-dimensional vectors for many real-world applications. The representations of entities and relations are learned via contrasting the positive and negative triplets. Thus, high-quality negative samples are extremely important in KGE. However, the present KGE models either rely on simple negative sampling methods, which makes it difficult to obtain informative negative triplets; or employ complex adversarial methods, which requires more training data and strategies. In addition, these methods can only construct negative triplets using the existing entities, which limits the potential to explore harder negative triplets. To address these issues, we adopt mixing operation in generating harder negative samples for knowledge graphs and introduce an inexpensive but effective method called MixKG. Technically, MixKG first proposes two kinds of criteria to filter hard negative triplets among the sampled negatives: based on scoring function and based on correct entity similarity. Then, MixKG synthesizes harder negative samples via the convex combinations of the paired selected hard negatives. Experiments on two public datasets and four classical KGE methods show MixKG is superior to previous negative sampling algorithms.

1 Introduction

Knowledge graphs (KGs) store information in the form of factual triplets \((h, r, t)\), where each triplet represents a particular relation \(r\) between head entity \(h\) and tail entity \(t\). Recently, with the help of KGs, many related applications have gained performance improvements, such as question answering [Yao and Van Durme, 2014], information retrieval [Xiong et al., 2017] and machine reading [Hao et al., 2019]. To benefit more applications, there exist many famous KGs, such as Freebase [Bollacker et al., 2008], Yago3 [Mahdisoltani et al., 2014] and WordNet [Miller, 1995].

However, directly using factual triplets in KGs is known to be difficult [Zhang et al., 2019]. The current popular methods are to embed entities and relations in the KGs into a vector space [Bordes et al., 2013; Nguyen, 2017; Sun et al., 2019]. Recently, some researchers try to employ graph embedding methods for KG embeddings [Shang et al., 2019; Schlichtkrull et al., 2018]. The great advantage of these models is that they map entities and relations in KGs into a low-dimensional vector space while preserving the information about graph structure. Therefore, they have shown promising performance in KG related tasks, such as link prediction [Shang et al., 2019] and triplet classification [Wang et al., 2017].

There is a tendency that more and more scoring functions are proposed to model the interactions between entities and relations in recent years [Hao et al., 2017; Bordes et al., 2013; Sun et al., 2019; Trouillon et al., 2016; Yang et al., 2014]. As a result, the performance of KG embeddings has greatly been improved. However, compared with scoring functions, fewer previous works focus on another aspect of KG embeddings: negative sampling. Negative sampling is essential because training KG embeddings is a process in which true triplets are given higher scores while false triplets are given lower scores [Ahrabian et al., 2020]. However, there are only ground truth triplets in KGs so negative triplets are used to generate corresponding false triplets.

Most current KG embedding models use uniform sampling to generate false triplets [Bordes et al., 2013; Trouillon et al., 2016; Yang et al., 2014]. Uniform sampling is to randomly select candidate entities with equal probability, so it is simple and efficient but only samples from a fixed distribution. Several pioneering works attempt to draw negative samples from a dynamical distribution. IGAN [Wang et al., 2018] and KBGAN [Cai and Wang, 2018] introduce generative adversarial network to select high-quality negative samples, while NSCaching [Zhang et al., 2019] utilizes cached-based mechanism to pay more attention to high-quality negative samples. Differently, SANS [Ahrabian et al., 2020] absorbs graph structure information into negative sampling process and achieves higher performances. Although these works optimize the negative sampling mechanism from different aspects, a common shortcoming of these models still exists: these models can only select pre-existing entities to construct the negative samples, which is a limitation in exploring harder negative triplets.

Recently, the mixing operation [Zhang et al., 2017] has shown its potential in many fields [Lee et al., 2020; Zhang et al., 2019]. However, directly using factual triplets in KGs is known to be difficult [Zhang et al., 2019]. The current popular methods are to embed entities and relations in the KGs into a vector space [Bordes et al., 2013; Nguyen, 2017; Sun et al., 2019]. Recently, some researchers try to employ graph embedding methods for KG embeddings [Shang et al., 2019; Schlichtkrull et al., 2018]. The great advantage of these models is that they map entities and relations in KGs into a low-dimensional vector space while preserving the information about graph structure. Therefore, they have shown promising performance in KG related tasks, such as link prediction [Shang et al., 2019] and triplet classification [Wang et al., 2017].
et al., 2020; Yoon et al., 2021]. Some researchers try to use the mixing operation to create better negative samples and the results show that the generated negative samples make the models more distinguishable [Huang et al., 2021; Kalantidis et al., 2020]. Considering that most negative triplets are easily distinguishable and cannot provide valuable information, we introduce MixKG, a new mechanism for negative sampling in KGs. The first step in MixKG is to filter out hard negative triplets and there are two criteria: a score function-based selector and a correct entity similarity selector. The first criterion is to take the negative triplets with high scores in the score function as hard samples, while the second criterion considers the negative entities that have higher similarity with the correct entity as hard samples. After selecting hard negative samples, MixKG then uses the mixing operation over these hard negatives to generate harder negative samples. Despite the simplicity and efficiency of our model, the performance shows the superiority of our model. Our contributions are as follows:

- To the best knowledge of us, we are the first to construct negative triplets using non-existent entities in KGs.
- We develop two criteria for selecting hard negative samples: score function-based selector and correct entity similarity selector.
- We generate harder negatives by mixing paired the selected hard negatives to make the model more distinguishable.
- We conduct extensive experiments on different datasets and KG embedding models to demonstrate the efficiency of the proposed model from different aspects.

2 Related Works

2.1 Negative Sampling

Negative sampling aims to generate negative samples, so the model can be trained by distinguishing observed positive data from negative samples. The advantage of negative sampling is abandoning to compute the normalization constant through sampling, which improves efficiency and effectiveness. From the perspective of sampling type, negative sampling can be divided into two categories: fixed negative sampling and dynamic negative sampling.

Fixed negative sampling

As a classic sampling strategy, the idea of fixed negative sampling is simple and intuitive. There are two representative methods for fixed negative sampling: uniform sampling [Bordes et al., 2013] and Bernoulli sampling [Wang et al., 2014]. However, due to the restriction to fixed sampling limitation, the fixed sampling strategy fails to form harder negative samples. and suffer from the vanishing gradient [Cai and Wang, 2018].

Dynamic negative sampling

To solve the problems in fixed negative sampling, several pioneering works have been proposed to extend negative sampling from fixed distribution to dynamic distribution. KBGAN [Cai and Wang, 2018] and IGAN [Wang et al., 2018] attempt to absorb generative adversarial network to generate high-quality negative triplets. However, both KBGAN and IGAN win effectiveness at the expense of instability and degeneracy [Zhang et al., 2019]. To reduce the complexity of the model while obtaining high-quality negative samples, NSCaching [Zhang et al., 2019] uses a cache to store negative triplets. In addition, SANS [Ahrabian et al., 2020] puts the graph structure information in KGs in the negative sampling and then dynamically selects negative samples only from the l-hop neighborhood of the head or tail entity. Although these methods try to generate high-quality negative samples from different aspects, they have a common shortcoming: the negative samples are generated only with pre-existing entities.

2.2 Mixing Method

Mixing [Zhang et al., 2017] is a data augmentation method, which generates new data by convex combinations of pairs of samples. Essentially, mixup encourages the model to behave linearly in-between training samples. Mixing methods have shown their superiority in many applications [Lee et al., 2020; Zhang et al., 2020; Yoon et al., 2021]. Overall, there are two popular types of domains for mixup: mixup for supervised learning and mixup for negative samples.

Mixing for supervised learning

Mixup [Zhang et al., 2017] generates fictitious training samples and their associated labels by linear interpolation, and the experimental results show mixup is universally applicable to image, speech and tabular datasets. SSMMix [Yoon et al., 2021] and MixText [Chen et al., 2020] extend the mixup operation to the domain of natural language processing through input and hidden space mixing respectively.

Mixing for hard negative samples

The core idea of mixing is to construct virtual samples by linear interpolation so some researchers use mixing for harder negative mining. MixGCF [Huang et al., 2021] uses positive mixing and hop mixing to obtain high-quality negative samples, which achieves a higher performance in the recommended scenario. MoCHi [Huang et al., 2021] shows that harder negative samples can be obtained by mixing between hard negative samples and labels.

3 Mix for Harder negative triplets

In this section, we introduce the proposed method in detail. To note that, the proposed method for generating harder negative samples via mixing operation is model-agnostic, so it is a general framework and can be easily plugged into the existing KGE models.

The core of the proposed model can be divided into two steps: first, selecting high-quality negative triplets from the sampled negative triplets; second, mixing the paired high-quality negative triplets and then generating novel harder negative triplets.

3.1 Select hard negative triplets

In KGs, there are only true triplets as positive samples, negative triplets can be obtained by replacing the head or tail entities in true triplets with other entities sampled from the
entire entity set. However, as is mentioned in previous works [Zhang et al., 2019; Ahrabian et al., 2020], many of the sampled negative triplets cannot provide discriminative information to help the model learn effective embeddings of entities and relations. Only a few negative samples help the model converge in a right direction, and these samples are hard negative samples, which are the raw material for mixing in the proposed method.

Hard negative samples are the core of the proposed model, but how to define hard negative samples remains a problem. Several previous works [Zhang et al., 2019; Kalantidis et al., 2020] use the score function $f$ and take the negative triplets with high scores as hard negative samples, and we call this method Score Function based Hard Negative Samples (HNS-SF). However, this definition only considers the negative samples in the negative set and ignores the help of true triplets. Therefore, we introduce a different definition for hard negative samples, which considers candidate entities that are more similar to the correct head or tail entity as hard negative samples. We name this kind of definition of hard negative samples Correct Entity Similarity based Hard Negative Samples (HNS-CES).

Given a KGE model scoring function $f$, a positive triplet $(h, r, t)$, the set of entities $E$. We first randomly sample entities from $E$ then get the sampled negative candidate entities $(t'_1, t'_2, ..., t'_M)$, and the negative triplets are $\{(h, r, t'_1), (h, r, t'_2), ..., (h, r, t'_M)\}$ by replacing the tail entity in $(h, r, t)$ with each element in $(t'_1, t'_2, ..., t'_M)$.

To note that, we take corrupting the tail entity as an example, and generating negative samples via corrupting the head entity is similar. The specific process of selecting negative hard samples under the two definitions above are as follows:

**Hard Negative Samples-Score Function**

For each triplet in $\{(h, r, t'_1), (h, r, t'_2), ..., (h, r, t'_M)\}$, we calculate the scores $s(t'_m) = f(h, r, t'_m)$ using scoring function $f$. Then we sort the scores $s(t'_1), s(t'_2), ..., s(t'_M)$ in a descending order, the negative triplets with $K$-largest scores are hard negative samples.

**Hard Negative Samples-Correct Entity Similarity**

We calculate the dot product between the tail entity and each candidate entity in $(t'_1, t'_2, ..., t'_M)$, then get the similarity $s(t'_m) = t \ast t'_m$. The candidate entities with top-$K$ largest $s(t'_m)$ and the head entity, relation in $(h, r, t)$ form hard negative triplets.

To note that, the size of the selected hard negative triplets $K$ is a hyperparameter, which will be discussed in more detail in Section 4.

### 3.2 Mixing among these hard triplets

After obtaining the hard negative triplets for mixing, we randomly select paired hard negative triplets then mix the tail entities to generate new harder negative samples. To be precise, suppose there are two negative triplets $\{(h, r, t'_1), (h, r, t'_2)\}$, then the tail entity of the newly generated harder negative triplet is

\[ \hat{t}_{i,j} = \alpha \ast t'_i + (1 - \alpha) \ast t'_j \quad (1) \]

### 3.3 Loss function

The last step of KGE is to construct a loss function to distinguish the positive and negative samples. Based on this, the embeddings of entities and relations are learned by forcing the positive triplets to have higher scores while lower scores for negative samples. The loss functions of the present KGE models can be divided into two categories: the first one is the translational distance model:

\[ \mathcal{L} = -\log \sigma(\gamma - f(h, r, t)) - \sum_{i=1}^{n} \log \sigma(f(h'_i, r, t'_i) - \gamma) \quad (2) \]

and the second one is the semantic matching model:

\[ \mathcal{L} = -\log \sigma(f(h, r, t)) - \sum_{i=1}^{n} \log \sigma(-f(h'_i, r, t'_i)) \quad (3) \]

where $(h'_i, r, t'_i)$ is the generated negative samples for $(h, r, t)$ by randomly replacing $h$ or $t$. $f$ is the scoring function to model the interactions between entities and relations. In the translational distance model, $f(h, r, t)$ is supposed to be smaller than $f(h'_i, r, t'_i)$, while $f(h, r, t)$ should be larger than $f(h'_i, r, t'_i)$ in semantic matching model, $\sigma$ is the sigmoid function, $\gamma$ is a fixed margin, and $n$ is the size of negative samples.

---

**Algorithm 1 MixKG: Mixing negative samples for KG embeddings**

**Input:** training set $S = \{(h, r, t)\}$, entity set $E$, relation set $R$, embedding dimension $d$, scoring function $f$, negative sample size $M$, generated negative sample size $N$, the size of selected harder negative number $K$, the size of epochs $E$.

1: initialize embeddings for each $e \in E$ and $r \in R$.
2: while epoch $< E$ do
3: sample a mini-batch $S_{\text{batch}} \in S$;
4: while $(h, r, t) \in S_{\text{batch}}$ do
5: uniformly sample $M$ entities from $E$ to form negative triplets $\{(h, r, t'_m), m = 1, 2,...M\}$
6: using scoring function or correct entity similarity to select negative triplets with $K$ highest scores $\{(h, r, t'_k), k = 1, 2,...K\}$
7: while $n < N$ do
8: uniformly pick up two triplets from negative triplets $\{(h, r, t'_k), k = 1, 2,...K\}$ and mix the tail entities using Equation 1 to get new negative triplet; 
9: end while
10: end while
11: end while

where $\alpha$ is randomly drawn from $(0, 1)$, then the generated harder negative triplet is $(h, r, \hat{t}_{i,j})$. 
4 Experiments

In this section, we conduct detailed experiments to demonstrate the effectiveness of the proposed model from various aspects. Our experiments seek to answer the following research questions (RQs):

- **RQ1**: Can the proposed method generate harder negative samples and have higher link prediction performances compared with previous negative sampling algorithms?
- **RQ2**: Are hard negative samples necessary for mixing?
- **RQ3**: How does the number of selected hard negative samples affect the performances?
- **RQ4**: Can more generated negative samples lead to better performance?

4.1 Datasets

We conduct experiments on two public KG datasets: FB15k-237 [Toutanova and Chen, 2015] and WN18RR [Dettmers et al., 2018]. FB15k-237 is a subset of FB15k, which comes from FreeBase [Bollacker et al., 2008] that contains lots of real-world triplets. Similarly, WN18RR is a subset of WN18 from WordNet KB, which is a large lexical English database. The details of the two datasets are given in Table 1.

| Dataset     | WN18RR | FB15k-237 |
|-------------|--------|-----------|
| #entity     | 14541  | 40943     |
| #relation   | 11     | 237       |
| #train      | 86835  | 272115    |
| #valid      | 3034   | 17535     |
| #test       | 3134   | 20466     |

4.2 Evaluation Protocols

Following previous works [Zhang et al., 2019; Ahrabian et al., 2020], we utilize two standard metrics to evaluate the performances for link prediction: mean reciprocal ranking (MRR) and Hits@10. Assuming \( N \) is the number of test triplets, MRR is the average of the reciprocal ranks \( \frac{1}{N} \sum_{i=1}^{N} \frac{1}{\text{rank}_i} \), where \( \text{rank}_i, i \in \{1, \ldots, N\} \) denotes the ranking results. In addition, Hits@10 is computed via \( \frac{1}{N} \sum_{i=1}^{N} \mathbf{1} (\text{rank}_i < 10) \), where \( \mathbf{1} (\cdot) \) is the indicator function. To be consistent with the previous and avoid underestimation, the results in our experiments are in a filtered setting, where all corrupted triplets in the dataset are removed.

4.3 Selected score functions

As is mentioned above, the KGE models can be divided into two groups: translational distance models and semantic matching models. We choose two representative models in each of the groups. For translational distance model, we utilize TransE [Bordes et al., 2013] and RotatE [Sun et al., 2019], then select ComplEx [Trouillon et al., 2016] and DistMult [Yang et al., 2014] for semantic matching model. The definitions of the four models are listed in Table 2.

| model         | scoring functions | definition                 |
|---------------|-------------------|----------------------------|
| translational distance | TransE, RotatE | \(|\mathbf{h} + \mathbf{r} - \mathbf{t}|\), \(|\mathbf{h} \circ \mathbf{r} - \mathbf{t}|\) |
| semantic matching | DistMult, ComplEx | \(\text{Re}(\mathbf{r}, \mathbf{h}, \mathbf{t})\) |

4.4 Hyperparameter settings

The optimizer in our experiments is Adam [Kingma and Ba, 2014] and we fine-tune the hyperparameters on the validation dataset. The embeddings of entities and relations are uniformly initialized. In addition, we use grid search to find the best hyperparameters, and the ranges of hyperparameters are as follows: batch size \( \in \{128, 256, 512\} \), negative sample set size \( M \in \{256, 512, 1024, 2048\} \), fixed margin \( \gamma \in \{3, 6, 9\} \), selected top hard negative sample size \( K \in \{5, 30, 50, 100\} \).

4.5 Baselines

We utilize the following negative sampling algorithms as baselines:

- **KBGAN** [Cai and Wang, 2018] first uniformly samples some entities to construct candidate negative triplets, then leverages one generator to pick up one high-quality negative triplet for training.
- **NSCaching** [Zhang et al., 2019] constructs a cache to store high-quality negative samples, then select negative samples from the cache and dynamically introduces new high-quality negative samples into the cache.
- **Uniform** [Sun et al., 2019] uniformly selects candidate entities from the entire entity set to form negative triplets.
- **SANS** [Ahrabian et al., 2020] incorporates graph structure information into negative sampling, and picks up negative samples from the \( l \)-hop neighborhood.
- **RW-SANS** [Ahrabian et al., 2020] is similar to SANS, and uses random walks of length \( l \) to approximate the \( l \)-hop neighborhood.

The results of TransE, RotatE and DistMult in Table 3 are taken from [Ahrabian et al., 2020], while the results of ComplEx for KBGAN, NSCaching are taken from [Zhang et al., 2019], and the results of ComplEx for Uniform, Uniform SANS, Uniform RW-SANS are our reproductions using codes in [Ahrabian et al., 2020].

4.6 Results analysis (RQ1)

The results of different negative sampling algorithms on four score functions are shown in Table 3. From Table 3, we can draw a conclusion that both HNM-SF and HNM-CES can improve the performances. We can see average improvement values of 0.0180 and 0.0115 in MRR for FB15k-237 dataset.
Table 3: Results of different negative sampling algorithms for link prediction on four scoring functions and two public datasets. In the table, HNM-SF indicates the hard negative samples for mixing are selected via score function, while HNM-CES denotes the selection criterion is correct entity similarity. Bold numbers represent the state-of-the-art performances, and the underlined numbers mean the best performances of the baselines.

| Score functions | Dataset       | FB15k-237 | WN18RR  |
|-----------------|---------------|-----------|---------|
|                 | Metrics       | MRR       | Hits@10(%) | MRR       | Hits@10(%) |
| TransE          | KBGAN         | 0.2926    | 46.59     | 0.1808    | 43.24      |
|                 | NSCaching     | 0.2993    | 47.64     | 0.2002    | 47.83      |
|                 | Uniform       | 0.2927    | 48.03     | 0.2022    | 49.63      |
|                 | Uniform SANS  | 0.2962    | 48.35     | 0.2254    | 51.15      |
|                 | Uniform RW-SANS | 0.2981   | 48.50     | 0.2317    | 53.41      |
|                 | HNM-SF        | 0.2983    | 50.29     | 0.2332    | 53.61      |
|                 | HNM-CES       | 0.3032    | 49.55     | **0.2386**| 52.97      |
|                 | Improvement   | +0.0039   | +1.94     | +0.0069   | +0.2       |
| RotatE          | Uniform       | 0.2946    | 47.85     | 0.4711    | 56.51      |
|                 | Uniform SANS  | 0.2985    | 48.22     | 0.4769    | 55.76      |
|                 | Uniform RW-SANS | 0.3003  | 48.47     | **0.4796**| **57.12**  |
|                 | HNM-SF        | 0.3292    | 51.75     | **0.4731**| 56.36      |
|                 | HNM-CES       | 0.3002    | 48.11     | 0.4753    | 56.37      |
|                 | Improvement   | +0.0289   | +3.28     | -0.0043   | -0.75      |
| DistMult        | KBGAN         | 0.2272    | 39.91     | 0.2039    | 29.52      |
|                 | NSCaching     | 0.2834    | 45.56     | 0.4128    | 45.45      |
|                 | Uniform       | 0.2537    | 40.26     | 0.3938    | 52.86      |
|                 | Uniform SANS  | 0.2595    | 41.00     | 0.4025    | 44.74      |
|                 | Uniform RW-SANS | 0.2621  | 41.46     | 0.4071    | 49.09      |
|                 | HNM-SF        | **0.3085**| **48.68** | **0.4420**| **53.22**  |
|                 | HNM-CES       | 0.2832    | 45.15     | 0.4299    | 50.91      |
|                 | Improvement   | +0.0251   | +3.12     | +0.0287   | +0.36      |
| ComplEx         | KBGAN         | 0.1910    | 32.07     | 0.3180    | 35.51      |
|                 | NSCaching     | 0.3021    | 48.05     | 0.4463    | 50.89      |
|                 | Uniform       | 0.2715    | 43.13     | 0.4506    | 56.05      |
|                 | Uniform SANS  | 0.2721    | 43.21     | 0.3832    | 41.16      |
|                 | Uniform RW-SANS | 0.2819  | 44.62     | 0.4247    | 46.38      |
|                 | HNM-SF        | **0.3160**| **50.43** | **0.4729**| **56.29**  |
|                 | HNM-CES       | 0.2712    | 43.03     | 0.4543    | 53.46      |
|                 | Improvement   | +0.0139   | +2.38     | +0.0223   | +0.24      |

and WN18RR dataset, respectively, while the improvement values are 1.07, 1.27, 1.74, 1.33 in Hits@10 for TransE, RotatE, DistMult and ComplEx respectively.

Overall, the two methods beat almost all previous negative sampling algorithms whether on the semantic translational distance models or on the semantic matching models, which shows the superiority of the proposed methods in general.

4.7 The importance of hard negative samples for mixing (RQ2)

To validate whether selecting hard negative samples is necessary for mixing, we also conduct experiments under the setting of Random Mix, which denotes the candidate entities for mixing are randomly drawn from the sampled negative samples without hard negative selection mechanism. The results are shown in Figure 1, and we have the following observations:

- Compared with Uniform on the four scoring functions, Random Mix has lower performances, which shows mixing among randomly drawn negative triplets cannot generate harder negatives.
- Both HNM-SF and HNM-CES improve Uniform by a large margin, therefore we can conclude that only selecting high-quality negative samples can lead to harder negatives and performance improvements.

4.8 How the size of hard negative samples affects performances (RQ3)

We use score function or correct entity similarity to calculate the hard extent of the negative samples and select the top-K hard negatives as hard negative samples for mixing. To see how K affects the performances, we conduct various experiments with different K in a range of {50, 150, 250, 350, 450, 550, 650}. The results are shown in Figure 2, we can have the observations:

- The performances gradually decrease with the increasing number of the selected hard negative samples on all the scoring functions, which means that only a small
number of negative samples are hard and helpful for guiding the KGE training process.

- With more unimportant negative samples joining the mixing process, it introduces noise and dilutes the proportion of the hard negative samples in final generated negative samples. Overall, the monotone decreasing line tells that the criteria for selecting hard negative samples are effective and essential.

![Figure 1: Comparisons of different mixing strategies on FB15k-237. Uniform denotes uniform sampling negative samples, Random Mix shows randomly picking up negative samples for mixing.](image)

4.9 How the size of generated harder negative samples affects performances (RQ4)

In this section, we analyze how the number of generated harder negatives affect performances. The final harder negative samples are generated via mixing two hard negative samples that are randomly drawn from the selected hard negative set. Theoretically, an ocean of harder negative samples can be generated by drawing two hard negative samples and mixing them endlessly. Based on this, we conduct detailed experiments on how the number of generated negative samples \( N \) affects the performances.

The results on the four score functions and FB15k-237 are in Figure 3, and the number of generated negatives is in range \( \{100, 200, 300, 400, 500, 600, 700\} \). The results on four different score functions all denote the performances remain consistency with a wide range number of generated hard negative samples, which shows that more generated negatives cannot improve performances. In other words, it is the generated high-quality harder negative samples which can benefit the performance not more generated negative samples, since more generated hard negative samples cannot provide extra effective information for training.

![Figure 3: The relationship between the number of generated harder negative samples via mixing operation and MRR on the TransE, RotatE, DistMult and ComplEx for FB15k-237.](image)

5 Conclusion

In this paper, we consider constructing harder negative samples via mixing hard negative samples, which is a simple but powerful method and can be absorbed into the present KGE models easily. Our work sheds light on the effectiveness and importance of generating non-existent entities as negative samples. We develop two kinds of criteria for picking up hard negative samples. One is taking the negative triplets with high scores as negative triplets, the other is selecting the candidate negative entities similar to a correct entity as negative samples, and the two types of hard negative samples can both lead to harder negative samples via mixing operation. We test the proposed method on four score functions and two public datasets, and the results show the method owns generalization under various experimental settings. There can be other sides of information to select hard negative samples for mixing, such as graph structure information [Ahrabian et al., 2020], and we leave this for future work.
References

[Ahrabian et al., 2020] Kian Ahrabian, Aarash Feizi, and et al. Salehi. Structure aware negative sampling in knowledge graphs. arXiv preprint arXiv:2009.11355, 2020.

[Bollacker et al., 2008] Kurt Bollacker, Colin Evans, and et al. Paritosh, Praveen. Freebase: a collaboratively created graph database for structuring human knowledge. In Proceedings of the 2008 ACM SIGMOD international conference on Management of data, 2008.

[Bordes et al., 2013] Antoine Bordes, Nicolas Usunier, and et al. Garcia-Duran, Alberto. Translating embeddings for modeling multi-relational data. Advances in neural information processing systems, 2013.

[Cai and Wang, 2018] Liwei Cai and William Yang Wang. Kbgan: Adversarial learning for knowledge graph embeddings. In Proc. of ACL, 2018.

[Chen et al., 2020] Jiaao Chen, Zichao Yang, and Diyi Yang. Mixtext: Linguistically-informed interpolation of hidden space for semi-supervised text classification. arXiv preprint arXiv:2004.12239, 2020.

[Dettmers et al., 2018] Tim Dettmers, Pasquale Minervini, and et al. Stenetorp, Pontus. Convolutional 2d knowledge graph embeddings. In Proc. of AAAI, 2018.

[Hao et al., 2017] Yanchao Hao, Yuanzhe Zhang, and et al. Liu, Kang. An end-to-end model for question answering over knowledge base with cross-attention combining global knowledge. In Proc. of ACL, 2017.

[Huang et al., 2021] Tinglin Huang, Yuxiao Dong, and et al. Ding, Ming. Mixgcf: An improved training method for graph neural network-based recommender systems. 2021.

[Kalantidis et al., 2020] Yannis Kalantidis, Mert Bulent Sariyildiz, and et al. Pion, Noe. Hard negative mixing for contrastive learning. arXiv preprint arXiv:2010.01028, 2020.

[Kingma and Ba, 2014] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014.

[Lee et al., 2020] Kibok Lee, Yian Zhu, and et al. Sohn, Ki-hyuk. I-mix: A domain-agnostic strategy for contrastive representation learning. arXiv preprint arXiv:2010.08887, 2020.

[Mahdisoltani et al., 2014] Farzaneh Mahdisoltani, Joanna Biega, and Fabian Suchanek. Yago3: A knowledge base from multilingual wikipedias. In 7th biennial conference on innovative data systems research, 2014.

[Miller, 1995] George A Miller. Wordnet: a lexical database for english. Communications of the ACM, 1995.

[Nguyen, 2017] Dat Quoc Nguyen. An overview of embedding models of entities and relationships for knowledge base completion. arXiv preprint arXiv:1703.08098, 2017.

[Schlichtkrull et al., 2018] Michael Schlichtkrull, Thomas N Kipf, and et al. Bloem, Peter. Modeling relational data with graph convolutional networks. In European semantic web conference, 2018.

[Shang et al., 2019] Chao Shang, Yun Tang, and et al. Huang, Jing. End-to-end structure-aware convolutional networks for knowledge base completion. In Proc. of AAAI, 2019.

[Sun et al., 2019] Zhiqing Sun, Zhi-Hong Deng, and et al. Nie, Jian-Yun. Rotate: Knowledge graph embedding by relational rotation in complex space. arXiv preprint arXiv:1902.10197, 2019.

[Toutanova and Chen, 2015] Kristina Toutanova and Danqi Chen. Observed versus latent features for knowledge base and text inference. In Proceedings of the 3rd workshop on continuous vector space models and their compositionality, 2015.

[Trouillon et al., 2016] Théo Trouillon, Johannes Welbl, and et al. Riedel, Sebastian. Complex embeddings for simple link prediction. In International conference on machine learning, 2016.

[Wang et al., 2014] Zhen Wang, Jianwen Zhang, and et al. Feng, Jianlin. Knowledge graph embedding by translating on hyperplanes. In Proc. of AAAI, 2014.

[Wang et al., 2017] Quan Wang, Zhendong Mao, and et al. Wang, Bin. Knowledge graph embedding: A survey of approaches and applications. IEEE Transactions on Knowledge and Data Engineering, 2017.

[Wang et al., 2018] Peifeng Wang, Shuangyi Li, and Rong Pan. Incorporating gan for negative sampling in knowledge representation learning. In Proc. of AAAI, 2018.

[Xiong et al., 2017] Chenyan Xiong, Russell Power, and Jamie Callan. Explicit semantic ranking for academic search via knowledge graph embedding. In Proceedings of the 26th international conference on world wide web, 2017.

[Yang et al., 2014] Bishan Yang, Wen-tau Yih, and et al. He, Xiaodong. Embedding entities and relations for learning and inference in knowledge bases. arXiv preprint arXiv:1412.6575, 2014.

[Yao and Van Durme, 2014] Xuchen Yao and Benjamin Van Durme. Information extraction over structured data: Question answering with freebase. In Proc. of ACL, 2014.

[Yoon et al., 2021] Soyoung Yoon, Gyuwan Kim, and Kyumin Park. Ssmix: Saliency-based span mixup for text classification. arXiv preprint arXiv:2106.08062, 2021.

[Zhang et al., 2017] Hongyi Zhang, Moustapha Cisse, and et al. Dauphin, Yann. Mixup: Beyond empirical risk minimization. arXiv preprint arXiv:1710.09412, 2017.

[Zhang et al., 2019] Yongqi Zhang, Quanming Yao, and et al. Shao, Yingxia. Nscaching: simple and efficient negative sampling for knowledge graph embedding. In 2019 IEEE 35th International Conference on Data Engineering (ICDE), 2019.

[Zhang et al., 2020] Rongzhi Zhang, Yue Yu, and Chao Zhang. Seqmix: Augmenting active sequence labeling via sequence mixup. arXiv preprint arXiv:2010.02322, 2020.