Investigating the Effects of Negative Sampling in Knowledge Graph Completion

Sooho Moon
Chung-Ang University
Seoul, Korea
moonwalk725@cau.ac.kr

Yunyong Ko*
Chung-Ang University
Seoul, Korea
yyko@cau.ac.kr

***Abstract*—Knowledge graphs (KGs) are powerful tools for representing structured knowledge, but their incompleteness limits the effectiveness in real-world applications. Knowledge graph completion (KGC) addresses this issue effectively, with negative sampling (NS) playing a crucial role in training KGC models. However, the impact of NS has not been systematically explored. In this paper, we investigate *how the number of negative triples per positive triple affects the performance of KGC models*. Experiments on four state-of-the-art models reveal the following findings: (1) NS consistently improves performance; (2) the performance gain diminishes as more negatives are used, with excessive negatives sometimes degrading model performance; and (3) sensitivity varies across KGC models, indicating that the optimal number of negatives is model-dependent. These insights provide practical guidance for designing more effective negative sampling strategies in KGC.**

***Index Terms*—Knowledge graph completion, Negative sampling, Knowledge graph embedding**

I. Introduction

A knowledge graph (KG) is a graph-structured representation of real-world knowledge, where an entity is represented as a node and a relation between two entities is represented as an edge in the form of a triple (h, r, t). KGs have been widely used across a range of applications [**?**], [1]–[3]. Unfortunately, real-world KGs are inherently incomplete [6], [7], i.e., a number of facts are missing. This fundamental limitation can hinder the potential of KGs. To address this, *knowledge graph completion* (KGC) has been widely studied [4]–[9], where the goal is to infer missing facts based on the observed KG structure, i.e., link prediction on KGs. Specifically, given a KG, a KGC model predicts the missing entity when either the head or the tail entity is unknown, i.e., in the form of $(h, r, ?)$ or $(?, r, t)$.

In real-world KGs, however, the number of existing triples is often limited [12]. Such sparsity is a fundamental cause of low accuracy in KGC. To address this problem, negative sampling (NS) is usually adopted [4]–[8], utilizing non-existing facts (i.e., negative triples) as contrastive information for training KGC models. Specifically, a KGC model is trained so that positive triples get higher scores (or lower ranks) while negative triples get lower scores (or higher ranks), thereby improving the distinguishing ability of the KGC model. Thus, it is critical to carefully sample negative triples to maximize the effect of negative sampling.

In spite of the critical role that negative sampling (NS) plays in training KGC models, it has not yet been thoroughly investigated. In particular, some fundamental questions remain largely unexplored: (**Q1**) *How many negative triples should be contrasted against each positive triple?* and (**Q2**) *How should negative triples be generated?* Most existing studies [4]–[8] adopt a simple heuristic sampling strategy, sampling a fixed number K (commonly 64 or 128) of negative triples uniformly at random for each positive triple, without deeper consideration of these design choices.

To bridge this gap, in this paper, we conduct a systematic investigation into the effect of negative sampling. Specifically, we focus on (Q1) to analyze how the number of negative triples per positive triple affects the performance of KGC models, aiming to provide new insights into the design of more effective negative sampling strategies.

Through extensive experiments with four state-of-the-art KGC models, we report the following key findings:

- **Consistent effectiveness**: Negative sampling consistently improves the performance of KGC models. Moreover, increasing the number of negative triples per positive triple generally leads to further performance gains.
- **Diminishing returns with more negatives**: The performance improvement becomes marginal as the number of negative triples grows. In some cases, using an excessively large number of negatives (e.g., greater than 256 or 1,024) even results in performance degradation.
- **Model-dependent sensitivity**: The degree of improvement and the point at which performance begins to drop vary across different models, suggesting that the optimal number of negatives should be tailored to each model.

II. Related Works

In this section, we introduce existing embeddings-based KGC models, which aim to represent entity and relation as embedding vectors in a latent space, while preserving the semantic meaning of triples through algebraic operations in the embedding space. TransE [4] represents each relation as a translation vector such that the head entity translated by the relation is closer to the tail entity. RotatE [7], the improved version of TransE, represents each relation as a

*Corresponding author

2-dimensional rotation in complex space to better capture diverse relation patterns such as symmetry and inversion. HousE [8] employs Householder transformations for both rotation and projection in the embedding space to capture more expressive relation patterns. DistMult [5] adopts a bi-linear scoring function with diagonal matrices to model symmetric relations. ComplEx [6] extends DistMult to complex number systems in order to model asymmetric relations. TuckEr [9] applies three-way Tucker tensor decomposition to KGC for capturing rich interactions between entities and relations.

III. Negative Sampling in KGC

In this section, we define the problem of knowledge graph completion and describe negative sampling in KGC.

Problem 1 (Knowledge Graph Completion). Given a knowledge graph (KG) $\mathcal{G} = (\mathcal{E}, \mathcal{R}, \mathcal{T})$, where $\mathcal{E}$ is the set of entities, $\mathcal{R}$ is the set of relations, and $\mathcal{T} = \{(h, r, t)|h, t \in \mathcal{E}, r \in \mathcal{R}\}$ is the set of triples (i.e., facts), the goal of knowledge graph completion (KGC) is to infer missing facts based on the observed KG.

KGC model training with negative triples. Given a KGC model $\theta(\cdot) : (|\mathcal{E}| \times |\mathcal{R}| \times |\mathcal{E}| \rightarrow \mathbb{R}^1)$, a positive triple $q = (h, r, t)$, and its K corresponding negative triples $\tilde{q}_{i(i \leq K)}$, the parameters of a KGC models are trained to minimize the following loss function:

$$\mathcal{L} = -\theta(q) + \frac{1}{K} \sum_{i=1}^{K} \theta(\tilde{q}_i). \tag{1}$$

Thus, the parameters of a KGC models are trained so that positive triples obtain higher scores than negative triples.

Negative sampling in KGC. Given a positive triple $q = (h, r, t)$, negative triples are typically generated by corrupting either the head entity h or the tail entity t with another entity e from the entity set $\mathcal{E}$. Thus, a negative triple $\tilde{q}$ takes the form of (h', r, t) where $h' \neq h$, or (h, r, t') where $t' \neq t$. There are some strategies for selecting which entity to corrupt:

- **Random sampling**: The most common strategy is to uniformly sample a replacement entity from the entire entity set $\mathcal{E}$. Thanks to its simplicity and generality, this method has been widely adopted in many existing works [4]–[8].
- **In-batch sampling**: Instead of sampling from the entire entity set $\mathcal{E}$, the replacement entity is restricted to those appearing in the current mini-batch B. This approach improves efficiency but often suffers from limited sample diversity, since candidate entities are extracted from a relatively small subset of entities.

In our experiments, we adopt the random sampling, in order to systematically investigate the effect of the number of negative triples per positive triple.

IV. Experimental Validation

A. Experimental Setup

1) KG dataset: In our experiments, we use a widely-used real-world knowledge graphs (KGs), FB15k237 [10], derived from a large-scale general-purpose knowledge base, Freebase. It contains facts about a wide range of real-world entities, including people, locations, films, organizations, and more. Table I shows the data statistics.

TABLE I
Statistics of FB15k237 dataset.

# of entities	*# of relations*	*# of triples*	*Avg. degree*	*Max. degree*
14,541	237	272,115	37.5	7,614

2) KGC models: We use four state-of-the-art KGC models: 4 embedding-based models (TransE [4], RotatE [7], ComplEx [6], DistMult [5]) For all KGC models, we use the official source codes provided by the authors.

3) Evaluation protocol: We evaluate all KGC models by using the protocol exactly same as that used in [11]. Each dataset is split into training, validation, and test sets, with different ratios depending on the dataset. Each KGC model is trained based on the training set, and its accuracy is evaluated on the validation set at every epoch. We save the model checkpoint that achieves the best accuracy on the validation set. As evaluation metrics, we use the following rank-based metrics:

- Mean rank (MR): The average rank position of the correct entity among all candidates (lower values indicate better performance).

$$MR = \frac{1}{Q} \sum_{i=1}^{Q} r_i \tag{2}$$

- Mean reciprocal rank (MRR): The average of the reciprocal ranks of the correct entity (higher values indicate better performance).

$$MRR = \frac{1}{Q} \sum_{i=1}^{Q} \frac{1}{r_i} \tag{3}$$

- Hits@K: The proportion of test triples for which the correct entity appears within the top-K ranked candidates (higher values indicate better performance).

$$Hit@K = \frac{1}{Q} \sum_{i=1}^{Q} \mathbb{I}[r_i \leq K] \tag{4}$$

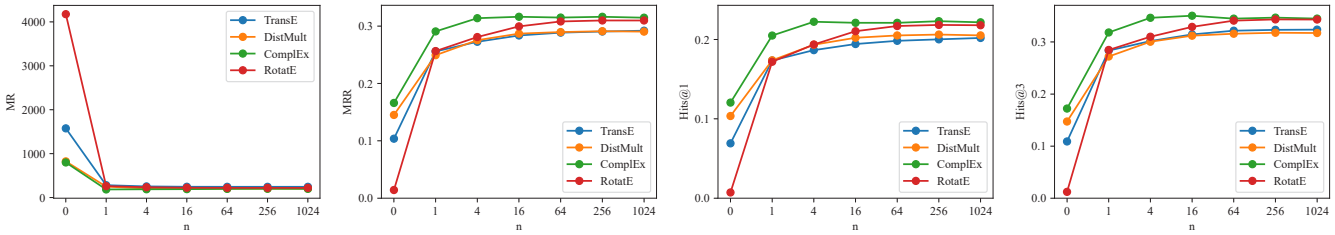
4) Implementation details: We implement our experimental codes by using python 3.11.2 on Ubuntu 22.04.4. We run all experiments on the machine equipped with an Intel i7-13700F CPU with 32GB main memory and two NVIDIA RTX 4080 GPUs, each of which has 16GB memory and is installed with CUDA 12.2 and cuDNN 8.9.7.

B. Experimental Results

Consistent effectiveness: We first compare the performance of KGC models with and without negative sampling (NS). For the baseline setting (w/o NS), no negative triples are used during training, whereas in the negative sampling setting (w/ NS), 256 negative triples are used per each positive triple. As shown in Table II, all KGC models consistently achieve significant

TABLE II
PERFORMANCE COMPARISON OF KGC MODELS DEPENDING ON WHETHER NEGATIVE SAMPLING (NS) IS APPLIED.

	TransE [4]				**DistMult [5]**				**ComplEx [6]**				**RotatE [7]**			
Metrics	MR	MRR	Hits@1	Hits@3	MR	MRR	Hits@1	Hits@3	MR	MRR	Hits@1	Hits@3	MR	MRR	Hits@1	Hits@3
w/o NS	1576.16	0.104	0.069	0.109	824.20	0.145	0.104	0.147	799.71	0.166	0.120	0.172	4176.94	0.014	0.007	0.012
w/ NS	244.71	0.291	0.200	0.323	203.40	0.291	0.206	0.318	201.00	0.316	0.223	0.347	221.07	0.310	0.218	0.343
Gain	+84%	+180%	+190%	+197%	+75%	+101%	+99%	+116%	+75%	+90%	+85%	+101%	+95%	+2,070%	+2,919%	+2,716%



(a) MR (b) MRR (c) Hits@1 (d) Hits@3

Fig. 1. Performance comparison of KGC models under varying numbers of negative triples per positive triple.

performance improvements when negative sampling is applied. This demonstrates that the use of negative sampling has a crucial impact on model performance.

Diminishing returns with more negatives: Next, we analyze how the performance of KGC models varies with different numbers of negatives per positive triple. Specifically, we evaluate each models with 0, 1, 4, 16, 64, 256, and 1,024 negatives per positive triple. As shown in Figure 1, the performance gain diminishes as the number of negative triples, and finally, most models reach performance saturation after a certain negative sampling size (64-256).

Model-dependent sensitivity: Finally, we observe that the effect of negative sampling varies across different models. Among the KGC models, RotatE exhibits the largest performance gain from negative sampling in our experiments, indicating that the effectiveness of negative sampling strategies is model-dependent.

V. CONCLUSION

In this paper, we conduct a systematic investigation into the effects of negative sampling in knowledge graph completion, especially focusing on the number of negative triples per positive triple. Our extensive experiments on four state-of-the-art KGC models reveal three key findings: (1) negative sampling consistently improves the performance of KGC models; (2) the performance gain diminishes as the number of negatives increases, and excessive negatives may even degrade performance; and (3) the sensitivity to the number of negatives varies across models, indicating that the optimal choice is model-dependent.

ACKNOWLEDGMENT

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (RS-2024-00459301).

REFERENCES

[1] L. Li, P. Wang, J. Yan, Y. Wang, S. Li, J. Jiang, Z. Sun, B. Tang, T.-H. Chang, and S. Wang, "Real-world data medical knowledge graph: Construction and applications," *Artif. Intell. Med.*, vol. 103, p. 101817, 2020.

[2] M. Rotmensch, Y. Halpern, A. Tlimat, S. Horng, and D. Sontag, "Learning a health knowledge graph from electronic medical records," *Sci. Rep.*, vol. 7, no. 1, p. 5994, 2017.

[3] Q. Guo, F. Zhuang, C. Qin, H. Zhu, X. Xie, H. Xiong, and Q. He, "A survey on knowledge graph-based recommender systems," *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 8, pp. 3549–3568, 2020.

[4] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, vol. 26, 2013.

[5] B. Yang, W.-t. Yih, X. He, J. Gao, and L. Deng, "Embedding entities and relations for learning and inference in knowledge bases," in *Proc. Int. Conf. Learn. Representations (ICLR)*, 2015.

[6] T. Trouillon, J. Welbl, S. Riedel, É. Gaussier, and G. Bouchard, "Complex embeddings for simple link prediction," in *Proc. Int. Conf. Mach. Learn. (ICML)*, pp. 2071–2080, 2016.

[7] Z. Sun, Z.-H. Deng, J.-Y. Nie, and J. Tang, "RotatE: Knowledge graph embedding by relational rotation in complex space," in *Proc. Int. Conf. Learn. Representations (ICLR)*, 2019.

[8] R. Li, J. Zhao, C. Li, D. He, Y. Wang, Y. Liu, H. Sun, S. Wang, W. Deng, Y. Shen, *et al.*, "House: Knowledge graph embedding with householder parameterization," in *Proc. Int. Conf. Mach. Learn. (ICML)*, pp. 13209–13224, 2022.

[9] I. Balazevic, C. Allen, and T. Hospedales, "TuckER: Tensor factorization for knowledge graph completion," in *Proc. Conf. Empirical Methods Nat. Lang. Process. (EMNLP-IJCNLP)*, pp. 5185–5194, 2019.

[10] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor, "Freebase: A collaboratively created graph database for structuring human knowledge," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, pp. 1247–1250, 2008.

[11] Z. Sun, S. Vashishth, S. Sanyal, P. Talukdar, and Y. Yang, "A re-evaluation of knowledge graph completion methods," in *Proc. Annu. Meeting Assoc. Comput. Linguistics (ACL)*, pp. 5516–5522, Jul. 2020.

[12] C. T. Hoyt, M. Berrendorf, M. Galkin, V. Tresp, and B. M. Gyori, "A unified framework for rank-based evaluation metrics for link prediction in knowledge graphs," in *Proc. Graph Learn. Benchmarks Workshop, TheWebConf*, 2022.