A Greedy Bit-flip Training Algorithm for Binarized Knowledge Graph Embeddings

Katsuhiko Hayashi[a,d,*] khayashi0201@gmail.com
Koki Kishimoto[b,*] ko.kishimoto8@gmail.com
Masashi Shimbo[c,d] shimbo@stair.center

[a]The University of Tokyo [b]Osaka University [c]Chiba Institute of Technology [d]Riken AIP

Abstract

This paper presents a simple and effective discrete optimization method for training binarized knowledge graph embedding model B-CP. Unlike the prior work using a SGD-based method and quantization of real-valued vectors, the proposed method directly optimizes binary embedding vectors by a series of bit flipping operations. On the standard knowledge graph completion tasks, the B-CP model trained with the proposed method achieved comparable performance with that trained with SGD as well as state-of-the-art real-valued models with similar embedding dimensions.

1 Introduction

Knowledge graph embedding (KGE) has a wide range of applications in AI and NLP, such as knowledge acquisition, question answering, and recommender systems. Most of the existing KGE models represent entities and relations as real or complex-valued vectors thus consuming a large amount of memory (Nickel et al., 2011; Bordes et al., 2013; Socher et al., 2013; Yang et al., 2014; Wang et al., 2014; Lin et al., 2015; Nickel et al., 2016; Trouillon et al., 2016; Hayashi and Shimbo, 2017; Liu et al., 2017; Manabe et al., 2018; Kazemi and Poole, 2018; Dettmers et al., 2018; Balažević et al., 2019a; Xu and Li, 2019; Balažević et al., 2019b). To deal with knowledge graphs with more than a million entities, more lightweight models are desirable for faster processing and to reduce memory consumption, as AI applications on mobile devices are becoming more common.

Kishimoto et al. (2019b) proposed a binarized KGE model B-CP, wherein all vector components are binarized, allowing them to be stored compactly with bitwise representation. Despite the reduced memory consumption by more than a magnitude, B-CP performed as well as the existing real-valued KGE models on benchmark tasks.

B-CP is based on the CP decomposition of a knowledge graph (Lacroix et al., 2018; Kazemi and Poole, 2018). It is *fully expressive* (Kishimoto et al., 2019a), meaning that any knowledge graph can be represented as a B-CP model.

During the training of B-CP, however, real-valued embeddings are maintained and are quantized at each training step (Kishimoto et al., 2019b). The loss function is computed with respect to the quantized vectors, but stochastic gradient descent is performed on the real vectors with the help of Hinton's "straight-through" estimator (HSTE) (Bengio et al., 2013). Thus, training does not benefit significantly from the compact bitwise representations, although score computation is faster by a bitwise technique. DKGE (Li et al., 2020) is another binary KGE model proposed recently, but it also maintains real-valued vectors during training, as it solves a relaxed optimization problem with continuous variables.

In this paper, we propose *greedy bit flipping*, a new training approach for B-CP in which binary vectors are directly optimized, i.e., without the intervention of real-valued vectors. A bit in binary vectors is sequentially flipped in a greedy manner so that the objective loss is improved. The advantages of greedy bit flipping are: (1) It does not need to maintain real-valued vectors even during training; (2) it is simple and is easy to implement; and (3) it has only a few hyperparameters.

2 Binarized CP Decomposition for Knowledge Graphs

A *knowledge graph* is a set of triples (e_i, e_j, r_k), where e_i, e_j are subject and object entities (represented as nodes in the graph), respectively, and r_k is

*The first and second authors equally contributed to this work.

Findings of the Association for Computational Linguistics: EMNLP 2020, pages 109–114

the label of the relation between them (corresponding to labeled arcs in the graph). When a triple is in a knowledge graph it is called a *fact*.

A knowledge graph can be equivalently represented by a third-order boolean tensor $\mathscr{X} = [x_{ijk}] \in \{0,1\}^{N_e \times N_e \times N_r}$, where N_e is the number of entities in the graph, and N_r is the number of relation labels; if a triple (e_i, e_j, r_k) is a fact, $x_{ijk} = 1$, and 0 otherwise.

CP decomposition (Hitchcock, 1927) is a general technique for decomposing a tensor into a sum of rank-1 tensors. For a third-order tensor $\mathscr{X}$ representing a knowledge graph, its approximate CP decomposition is given by $\mathscr{X} \approx \sum_{d=1}^{D} \mathbf{a}_d \otimes \mathbf{b}_d \otimes \mathbf{c}_d$ where $\otimes$ denotes outer product, and $\mathbf{a}_d, \mathbf{b}_d \in \mathbb{R}^{N_e}$ and $\mathbf{c}_d \in \mathbb{R}^{N_r}$ are real (column) vectors. In this case, matrices $\mathbf{A} = [\mathbf{a}_1\, \mathbf{a}_2 \cdots \mathbf{a}_D] \in \mathbb{R}^{N_e \times D}$, $\mathbf{B} = [\mathbf{b}_1\, \mathbf{b}_2 \cdots \mathbf{b}_D] \in \mathbb{R}^{N_e \times D}$, and $\mathbf{C} = [\mathbf{c}_1\, \mathbf{c}_2 \cdots \mathbf{c}_D] \in \mathbb{R}^{N_r \times D}$ are called *factor matrices*. For any matrix $\mathbf{M}$, let $\mathbf{m}_{i:}$ denote its ith row vectors. Then, the component x_{ijk} of $\mathscr{X}$ can be written as $x_{ijk} \approx \langle \mathbf{a}_{i:}, \mathbf{b}_{j:}, \mathbf{c}_{k:} \rangle = \sum_{d=1}^{D} a_{id} b_{jd} c_{kd}$. Thus, vectors $\mathbf{a}_{i:}, \mathbf{b}_{j:}, \mathbf{c}_{k:}$ can be regarded as the D-dimensional vectors representing the subject entity e_i, object entity e_j, and relation label r_k, respectively.

The B-CP decomposition of a knowledge graph (Kishimoto et al., 2019b) differs from the standard CP, in that $\mathscr{X}$ is decomposed in terms of binary vectors $\mathbf{a}_d, \mathbf{b}_d \in \{-1,+1\}^{N_e}, \mathbf{c}_d \in \{-1,+1\}^{N_r}$. As with CP, B-CP decomposition can be cast as a problem of binary classification, and solved by logistic regression. First, each x_{ijk} is assumed to be a random variable sampled independently from a probability distribution parameterized by $\mathbf{A}, \mathbf{B}, \mathbf{C}$:

$$p(\mathscr{X} | \mathbf{A}, \mathbf{B}, \mathbf{C}) = \prod_{i=1}^{N_e} \prod_{j=1}^{N_e} \prod_{k=1}^{N_r} p(x_{ijk} | \theta_{ijk}).$$

where $\theta_{ijk} = \langle \mathbf{a}_{i:}, \mathbf{b}_{j:}, \mathbf{c}_{k:} \rangle$ is called the *score* of triple (e_i, e_j, r_k), and

$$p(x_{ijk} | \theta_{ijk}) = \begin{cases} \sigma(\theta_{ijk}) & \text{if } x_{ijk} = 1, \\ 1 - \sigma(\theta_{ijk}) & \text{if } x_{ijk} = 0, \end{cases}$$

is a Bernoulli distribution. Function $\sigma(x) = 1/(1 + \exp(-x))$ is a sigmoid function.

To train factor matrices to match observed/unobserved facts encoded as $\mathscr{X}$, we minimize the negative log likelihood of the posterior probability:

$$\min_{\mathbf{A},\mathbf{B},\mathbf{C}} \sum_{i=1}^{N_e} \sum_{j=1}^{N_e} \sum_{k=1}^{N_r} E_{ijk} \tag{1}$$

$$\text{s.t.} \quad E_{ijk} = -x_{ijk} \log(\sigma(\theta_{ijk})) - (x_{ijk} - 1) \log(1 - \sigma(\theta_{ijk})). \tag{2}$$

Algorithm 1: Greedy Bit-flip Training

input: *Pos*: set of training triples (facts), including those for reciprocal relations (see Sec. 3.2)
input: N_e, N_r: numbers of entities and relations
input: I: maximum number of iterations
output: $\mathbf{A}, \mathbf{B} \in \{-1,+1\}^{N_e \times D}$: factor matrices of subject and object entity embeddings
output: $\mathbf{C} \in \{-1,+1\}^{N_r \times D}$: factor matrix of relation embeddings

1 Initialize binary factor matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}$ randomly
2 **foreach** *iter* $\in \{1, \ldots, I\}$ **do**
3 *Neg* ←negative samples (see Sec. 3.2)
4 Update($\mathbf{C}, N_r, D, Pos, Neg$) *// relation embeddings*
5 Update($\mathbf{A}, N_e, D, Pos, Neg$) *// subject embeddings*
6 Update($\mathbf{B}, N_e, D, Pos, Neg$) *// object embeddings*
7 Check convergence

Algorithm 2: Update($\mathbf{M}, N, D, Pos, Neg$)

input: $\mathbf{M} \in \{-1,+1\}^{N \times D}$: factor matrix to update
input: *Pos*: set of positive triples (facts)
input: *Neg*: set of negative triples (non-facts)
output: $\mathbf{M}$: updated factor matrix

1 C ←random permutation of indices $1, \ldots, D$
2 **foreach** $i \in \{1, \ldots, N\}$ **do** *// run in parallel*
3 **foreach** $j \in C$ **do** *// run sequentially, but in random order*
4 **if** $\Delta(m_{ij}) < 0$ **then** $m_{ij} \leftarrow -m_{ij}$

3 Greedy Bit-flip Training for B-CP

The proposed training method randomly samples an element (or a bit) of the factor matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}$ of B-CP, and negates its sign if this "bit flipping" reduces the objective loss. This process is repeated until the loss does not improve further or a specified number of iterations is reached. The pseudocode of the algorithm is depicted in Algorithms 1 and 2.

In Algorithm 1, when a factor matrix is updated, the other two factor matrices are fixed. As the number N_r of relations is considerably smaller than the number N_e of entities in general, the change in the relation matrix $\mathbf{C}$ influences the total loss much more significantly than entity matrices $\mathbf{A}$ and $\mathbf{B}$. For this reason, we update $\mathbf{C}$ prior to $\mathbf{A}$ and $\mathbf{B}$ in each iteration to promote faster convergence.

Actual update is carried out in Algorithm 2. As remarked on Line 2, a row of a factor matrix, which represents a single entity or a relation, can be processed in parallel, because the score of an individ-

ual triple depends only on a single subject, object, and relation it contains; for instance, even when all subject embeddings are updated simultaneously, only one of them can change the score of any given triple. This means that, when multiple rows of a factor matrix are updated, the change in the total loss in Eq. (1) is invariant to the order of the updates, as long as the other two factor matrices are fixed. Since Algorithm 2 updates only one matrix, we see that its rows can be processed in parallel.

By contrast, the loss is dependent on the order of updated columns (i.e., bits) within a row, i.e., components in an embedding vector. We thus change the order of updated columns every time Algorithm 2 is called, by shuffling the set $[D]$ of dimensions in Line 1.

In Algorithm 2, each bit in a factor matrix is examined to see if it is worth being flipped. For instance, consider a component (bit) a_{ij} of $\mathbf{A}$. Let $E(\mathbf{A},\mathbf{B},\mathbf{C}) = \sum_{i=1}^{N_e}\sum_{j=1}^{N_e}\sum_{k=1}^{N_r} E_{ijk}$ denote the loss in Eq. (1), and let $\mathbf{A}'$ denote the factor matrix $\mathbf{A}$ after a_{ij} is flipped to $-a_{ij}$. The change in the loss is then

$$\begin{aligned}\Delta(a_{ij}) &= E(\mathbf{A}',\mathbf{B},\mathbf{C}) - E(\mathbf{A},\mathbf{B},\mathbf{C}) \\ &= -\sum_{y=1}^{N_e}\sum_{z=1}^{N_r}\left(x_{iyz}\log\frac{\sigma(\theta_{iyz})}{\sigma(\theta_{iyz}-2a_{ij}b_{yj}c_{zj})}\right. \\ &\quad \left. -(1-x_{iyz})\log\frac{1-\sigma(\theta_{iyz})}{1-\sigma(\theta_{iyz}-2a_{ij}b_{yj}c_{zj})}\right),\end{aligned}$$

where θ_{ijk} is computed before the update (i.e., using $\mathbf{A}$, not $\mathbf{A}'$) by Eq. (2). Only if $\Delta(a_{ij})$ is found to be negative, i.e., the loss is decreased, a_{ij} is actually flipped. The same rule applies to the bits in the factor matrices $\mathbf{B}$ and $\mathbf{C}$. Repeated application of this update guarantees the loss to be non-increasing. However, the loss may be stuck in a local minimum, depending on the order of updates.[1] Training is terminated when the objective loss does not improve, or if a pre-determined number of epochs has elapsed.

3.1 Fast Score Computation by Bitwise Operations

In this section, we describe the implementation detail necessary to speed up training.

As Algorithm 2 involves repeated computation of scores θ_{ijk}, fast computation of scores is a key to speed up training. Although one easy approach is to cache all scores in memory, the number of facts may be huge in knowledge graphs. We thus consider bitwise computation to speed up score computation.

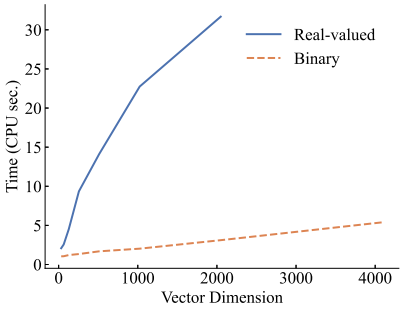

Figure 1: Real-valued vs. Bit vectors: Score computation time comparison.

We can compute the B-CP scores by bitwise operation, as follows.

$$\theta_{ijk} = D - 2h(\mathbf{a}_{i:}, \mathrm{XNOR}(\mathbf{b}_{j:}, \mathbf{c}_{k:})), \qquad (3)$$

where $h(\cdot,\cdot)$ is Hamming distance and $\mathrm{XNOR}(\cdot,\cdot)$ is the negation of exclusive-or. As shown in Figure 1, bitwise score computation is much faster than naive computation of scores by Eq. (3), making the cost of score computation negligible.

3.2 Negative Sampling and Reciprocal Relations

Before calling Algorithm 1, for each (e_i, e_j, r_k) in the training set *Pos*, we introduce its reciprocal triple (e_j, e_i, r_k^{-1}) in the set, with a new relation label r_k^{-1}. This technique was used by Lacroix et al. (2018) and Kazemi and Poole (2018), and is effective for models such as CP and B-CP, in which an entity has separate embeddings for subject and object.

Following prior work, we also approximate the objective loss by sampling negative examples (Algorithm 1, Line 3) to cope with the enormous size and sparsity of knowledge graphs. Specifically, for each (e_i, e_j, r_k) in the training set, a predetermined number of entities are first sampled randomly. Then, for each sampled entity e, we create a negative triple (e_i, e, r_k) and its reciprocal negative triple (e, e_i, r_k^{-1}).

4 Experiments

4.1 Experimental Setup

For evaluation, we performed entity prediction on two standard knowledge graph completion (KGC)

[1] In preliminary experiments, annealing strategies were tested with bit-flip training to mitigate overfitting. However, they had no impact on the KGC performance.

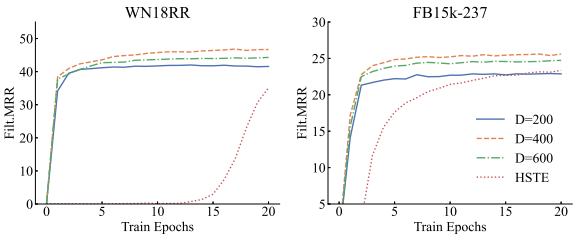

Figure 2: Training epochs vs. filtered MRR on the WN18RR and FB15k-237 validation datasets.

	WN18RR			FB15k-237	
Models	Memory	MRR	Hits@10	MRR	Hits@10
DistMult*	79.24	45.2	53.1	34.3	53.1
ComplEx*	39.62	47.5	54.7	34.8	**53.6**
ConvE*	79.24	44.2	50.4	33.9	52.1
HSTE B-CP**	3.87	45.0	52.0	29.2	46.1
DKGE***	2.62	35.0	50.6	**36.8**	50.7
HSTE B-CP	3.87	44.2	47.2	27.1	43.7
†HSTE B-CP	19.34	46.4	51.2	28.9	46.0
Bit-flip B-CP	3.87	47.7 (±0.2)	53.3 (±0.1)	27.6 (±0.0)	45.7 (±0.1)
†Bit-flip B-CP	19.34	**49.1**	**55.0**	29.5	47.8

Table 1: KGC results on WN18RR and FB15k-237: Memory consumption (MB), Filtered MRR and Hits@10 (%). *, ** and *** indicate the results taken from (Ruffinelli et al., 2020), (Kishimoto et al., 2019b) and (Li et al., 2020), respectively. The memory consumption figures for these models are estimated from the reported number of parameters.

datasets, WN18RR (Dettmers et al., 2018) and FB15k-237 (Toutanova and Chen, 2015) with their default training/validation/test splits.

In the entity prediction task, a KGE model is given a set of incomplete triples, each of which is generated by hiding one of the entities in a positive triple in the test set; i.e., from a positive triple (e_i, e_j, r_k), incomplete triples $(?, e_j, r_k)$ and $(e_i, ?, r_k)$ are generated. For each such incomplete triple, the KGE model must produce a list of all entities (including the correct entity, e_i or e_j) ranked by the score when each of these entities is plugged instead of the placeholder "?" in the triple. The quality of the output ranking list is then measured by two standard evaluation measures for the KGC task: Mean Reciprocal Rank (MRR) and Hits@10, in the "filtered" setting (Bordes et al., 2013).

We selected the hyperparameter D of the proposed method (henceforth denoted as "Bit-flip B-CP") via grid search over the range $D \in \{200, 400, 600\}$, such that the filtered MRR is maximized on the validation set. The maximum number of training epochs was set to 20. We generated 20 negative triples per positive training triple for FB15k-237 and 5 for WN18RR. Bit-flip B-CP was implemented in Java, and ran on a laptop PC with 2.7GHz Intel Core i7 CPU. Our implementation with $D = 400$ took about 5 minutes to finish 20 training epochs on the WN18RR training dataset.

4.2 Results

Training Convergence Figure 2 shows the MRR scores on the validation set at each training epoch. For comparison, we also trained B-CP using HSTE-based stochastic gradient descent for optimization and the best hyperparameters reported by Kishimoto et al. (2019a).

The figure shows greedy bit flipping (Bit-flip B-CP) requires a much smaller number of training epochs to converge than HSTE-based training (HSTE B-CP). For both datasets, the best MRR for Bit-flip B-CP was obtained when $D = 400$, and thus, we used this setting for the following test evaluations.

KGC Performance Table 1 summarizes the performance on the entity prediction task. The table lists the proposed Bit-flip B-CP, and several state-of-the-art models, including B-CP trained with HSTE (HSTE B-CP). We can see that Bit-flip B-CP achieved comparable results to other KGE models.

To examine the dependence on initial parameter values, we trained five bit-flip-trained B-CP models using different initial values generated with varied random seeds. The performance figures in the table for Bit-flip B-CP are the average over these five models, with the standard deviation shown in parentheses. The small standard deviations indicate that bit flipping training is stable over different random seeds.

Notice that B-CP consists of binary vectors, which makes the memory consumption approximately $1/20$ of that of real-valued models DistMult and ConvE. Taking advantage of the small memory consumption of B-CP, we created an ensemble of five B-CP models; i.e., the score θ_{ijk} is computed by the sum of the scores of all models in the ensemble. Its performance is shown in the rows titled "†Bit-flip B-CP" of Table 1. For comparison, we also show the result for the ensemble of five HSTE-trained B-CP models ("†HSTE B-CP"). As we can see from the table, ensemble improved the task performance. Note that even the ensemble models consume much less memory than existing models using 32-bit real embeddings.

5 Conclusion

In this paper, we have introduced greedy bit flipping, a simple yet effective discrete optimization method for training the binarized KGE model B-CP.

On the standard benchmark datasets of KGC, B-CP models trained by bit flipping were on per with HSTE-trained B-CP in terms of accuracy. Experimental results show that the KGC performance was stable over different initial values. Making ensemble of multiple B-CP models is made tractable by the small memory consumption of B-CP, which brought further performance improvement.

Bit flipping is unique in that it does not require the loss function to be differentiable, making it potentially applicable to a wide range of loss functions. We plan to investigate this direction in our future work. Application of bit flipping to other binarized KGE models is another interesting direction. A binary version of DistMult looks interesting as a starting point, as it is closely related to DKGE (Li et al., 2020), a recently proposed binarized model.

Acknowledgments

This work was partially supported by JSPS Kakenhi Grants 18K11457 and 19H04173.

References

Ivana Balažević, Carl Allen, and Timothy M Hospedales. 2019a. Hypernetwork knowledge graph embeddings. In *Proceedings of the 28th International Conference on Artificial Neural Networks (ICANN)*, pages 553–565.

Ivana Balažević, Carl Allen, and Timothy M. Hospedales. 2019b. TuckER: Tensor factorization for knowledge graph completion. In *Proceedings of the 2019 Conference on Empirical Methods in Natural*, pages 5184–5193.

Yoshua Bengio, Nicholas Léonard, and Aaron C. Courville. 2013. Estimating or propagating gradients through stochastic neurons for conditional computation. *CoRR*, abs/1308.3432.

Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems*, pages 2787–2795.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2d knowledge graph embeddings. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*.

Katsuhiko Hayashi and Masashi Shimbo. 2017. On the equivalence of holographic and complex embeddings for link prediction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 554–559.

F. L. Hitchcock. 1927. The expression of a tensor or a polyadic as a sum of products. *J. Math. Phys*, 6(1):164–189.

Seyed Mehran Kazemi and David Poole. 2018. Simple embedding for link prediction in knowledge graphs. In *Advances in Neural Information Processing Systems*, pages 4289–4300.

Koki Kishimoto, Katsuhiko Hayashi, Genki Akai, and Masashi Shimbo. 2019a. Binarized canonical polyadic decomposition for knowledge graph completion. *arXiv preprint arXiv:1912.02686*.

Koki Kishimoto, Katsuhiko Hayashi, Genki Akai, Masashi Shimbo, and Kazunori Komatani. 2019b. Binarized knowledge graph embeddings. In *Proceedings of the 41st European Conference on Information Retrieval*, pages 181–196.

Timothée Lacroix, Nicolas Usunier, and Guillaume Obozinski. 2018. Canonical tensor decomposition for knowledge base completion. In *Proceedings of the 35th International Conference on Machine Learning*, pages 2869–2878.

Yunqi Li, Shuyuan Xu, Bo Liu, Zuohui Fu, Shuchang Liu, Xu Chen, and Yongfeng Zhang. 2020. Discrete knowledge graph embedding based on discrete optimization. In *Proceedings of the AAAI-20 Workshop on Knowledge Discovery from Unstructured Data in Financial Services*.

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*, pages 2181–2187.

Hanxiao Liu, Yuexin Wu, and Yiming Yang. 2017. Analogical inference for multi-relational embeddings. In *Proceedings of the 34th International Conference on Machine Learning*, pages 2168–2178.

Hitoshi Manabe, Katsuhiko Hayashi, and Masashi Shimbo. 2018. Data-dependent learning of symmetric/antisymmetric relations for knowledge base completion. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*.

Maximilian Nickel, Lorenzo Rosasco, and Tomaso A. Poggio. 2016. Holographic embeddings of knowledge graphs. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 1955–1961.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. In *Proceedings of the 28th International Conference on Machine Learning*, pages 809–816.

Daniel Ruffinelli, Samuel Broscheit, and Rainer Gemulla. 2020. You CAN teach an old dog new tricks! On training knowledge graph embeddings. In *Proceedings of 8th International Conference on Learning Representations (ICLR)*.

Richard Socher, Danqi Chen, Christopher D. Manning, and Andrew Y. Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *Advances in Neural Information Processing Systems*, pages 926–934.

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.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *Proceedings of the 33nd International Conference on Machine Learning*, pages 2071–2080.

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

Canran Xu and Ruijiang Li. 2019. Relation embedding with dihedral group in knowledge graph. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 263–272.

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