Improving Knowledge Graph Embeddings with Inferred Entity Types

Esma Balkır
Amazon Research / Cambridge UK [*]
University of Edinburgh / Edinburgh, UK
esma.balkir@ed.ac.uk

Masha Naslidnyk
Amazon Research / Cambridge, UK
naslidny@amazon.co.uk

Dave Palfrey
Amazon Research / Cambridge, UK
dpalfrey@amazon.co.uk

Arpit Mittal
Amazon Research / Cambridge, UK
mitarpit@amazon.co.uk

Abstract

In this paper we study techniques to improve the performance of bilinear embedding methods for knowledge graph completion on large datasets, where at each epoch the model sees a very small percentage of the training data, and the number of generated negative examples for each positive example is limited to a small portion of the entire set of entities. We first present a heuristic method to infer the types and type constraints of entities and relations. We then use this method to construct both a joint learning model, and a straightforward method for increasing the quality of sampled negatives during training. We show that when these two techniques are combined, they give an improvement in performance of up to 5.6% for Hits@1. We find the improvement is especially significant when the batch size and the number of generated negative examples are low relative to the size of the dataset.

1 Introduction

A Knowledge Graph (KG) is a collection of facts which are stored as *entity-relation-entity* triples. Even though knowledge graphs are essential for various NLP tasks such as question answering and natural language understanding, open domain knowledge graphs have missing facts. To tackle this issue, there has recently been considerable interest in KG completion methods, where the goal is to assign scores to unseen triples that reflect how likely they are to be a missing correct fact.

A popular approach to this task is to learn vector representations for entities and relations, often referred to as *embeddings*. It has been shown that for learning these embeddings, simple methods often beat more complicated models when the hyperparameters are properly tuned [4, 7]. Among the parameters that have a big effect on performance are the size of minibatches during training, and the number of corrupted triples per positive triple. However, KGs in the wild are often several magnitudes bigger than the common benchmarking datasets. To reach peak performance, training on these datasets requires much larger batch sizes and more generated negatives than the hardware allows.

We break the batch size bottleneck by using the *implicit type information* contained in the dataset to train a joint model both on the correctness and the type-consistency of the triple. For improving training with a small number of negative examples, we focus on generating better quality corrupted triples by sampling type-consistent triples with a higher probability.

[*]Work done while the author was an intern

We show empirically that joint training is beneficial when the batch size is small, and that typed negative sampling helps the model learn higher quality embeddings with a fraction of the number of negative samples. Furthermore, the two techniques combined perform significantly better than either of them alone, with an uplift of up to 5.6% on hits@1.

2 Background

Formally, given a set of entities $\mathcal{E}$ and a set of relations $\mathcal{R}$, a knowledge graph (KG) is a set of triples in the form $\mathcal{G} = \{(h, r, t)\} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$, where if a triple $(h, r, t) \in \mathcal{G}$, then relation r holds between entities h and t. Given such a KG, the aim of knowledge base completion is to assign scores to each triple, so that correct triples are assigned higher scores than false ones.

A class of KG completion models such as RESCAL [8], DistMult [12], and ComplEx [11] define their scoring function to be a bilinear interaction of the embeddings of entities and relations in the triple. In this work, we will focus on ComplEx since we found it to perform consistently better than the others in our preliminary experiments.

ComplEx is a bilinear model which represents entities and relations as vectors of complex values, and defines the embedding of an entity in object position as the complex conjugate of the embedding in the subject position. The reason that ComplEx operates in the complex vector space is because dot product in the complex domain is non-symmetric unlike in the real domain, and this allows the model to be able to model anti-symmetric relations.

Let $\boldsymbol{h}, \boldsymbol{r}, \boldsymbol{t} \in \mathbb{C}^d$ be the embeddings for h, r, t. The score for ComplEx is defined as $s(h, r, t) = \text{Re}\left(\boldsymbol{h}^T \text{diag}(\boldsymbol{r})\overline{\boldsymbol{t}}\right)$, where $\overline{\boldsymbol{a}}$ denotes the complex conjugate of $\boldsymbol{a}$, and $\text{Re}(\boldsymbol{a})$ denotes the real part of the complex vector $\boldsymbol{a}$ and $\text{diag}(r)$ is a diagonal matrix with values of r on the diagonal.

The training set for KG completion tasks only includes the facts which are believed to be true. The standard way of generating negative examples is by corrupting the head or the tail entity randomly. The number of negative examples generated per positive example has a strong effect on the performance of the model [11].

There have been a number of loss functions proposed for optimizing KG embedding models. Among those are L2 loss [9], margin based ranking loss [1, 12], binary logistic loss [11], negative log likelihood (NLL) of sampled softmax [4], and NLL of softmax over the entire set of entities [3, 7].

In our experiments on bilinear models we found that using NLL of softmax over sampled negatives consistently outperformed other losses by significant margins, and results close to SOTA could be achieved by using the full-softmax. However, this technique can only be used when the KG is relatively small, and does not scale when the number of entities is very large. In these cases, it is necessary to revert to training with softmax over sampled negatives. In Section 4 we will present our method that improves the model performance for methods using sampled negatives. For comparison we also present the results using full-softmax.

3 Inferring and utilizing type-consistency

Consider the two triples 'Lyon is-the-capital-of Germany' and 'Lyon is-the-capital-of George Orwell'. In the first triple, the head and the tail entities obey the implicit semantic type restrictions the relation has for its arguments, but this is not the case for the second triple. We will henceforth refer to the former as a *type-consistent triple*, and the latter as a *type-inconsistent triple*.

Often type restrictions are explicitly provided in large datasets, but even if they are not available, the labels can be generated heuristically. In our experiments, we follow the method described in [6] and label a triple (h, r, t) type-consistent if there are triples (h, r, t') and (h', r, t) in the training set for any t' and h'.

Joint model. Our joint model contains two copies of ComplEx, with scoring functions s_{type} and s_{fact}. We tie the weights of the entity embeddings, but let the embeddings for the relations be optimized separately.

$$s_{\text{fact}}(h, r, t) = \text{Re}\left(\boldsymbol{h}^T \text{diag}(\boldsymbol{r}_{\text{fact}})\overline{\boldsymbol{t}}\right) \qquad s_{\text{type}}(h, r, t) = \text{Re}\left(\boldsymbol{h}^T \text{diag}(\boldsymbol{r}_{\text{type}})\overline{\boldsymbol{t}}\right) \tag{1}$$

This allows the joint model to have only a marginally larger number of parameters than the baselines, since the number of relations in KGs are often a fraction of the number of entities. It also forces the model to compress both the type constraints and the factual data in the latent features of entity embeddings, effectively acting as a regularizer against the sparsity in the data.

For each triple in the minibatch, we generate n_{neg} negative examples per positive triple by randomly corrupting the head or the tail entity. For s_{fact}, the task is then to differentiate the correct triple from all its corrupted versions. We accomplish this by applying softmax over all the candidates and then calculating the loss as the NLL of the true triple:

$$\mathcal{L}_{\text{fact}}(h,r,t) = -\log\left(\frac{exp\left(s_{\text{fact}}\left(h,r,t\right)\right)}{\sum_{t'} exp\left(s_{\text{fact}}\left(h,r,t'\right)\right) + \sum_{h'} exp\left(s_{\text{fact}}\left(h',r,t\right)\right)}\right) \tag{2}$$

For s_{type}, we generate the type-consistency labels as described above, and use binary cross entropy loss between the output and the type-consistency label:

$$\mathcal{L}_{\text{type}}(h,r,t) = -y_{\text{type}}\log s_{\text{type}}(h,r,t) - (1-y_{\text{type}})\log(1-s_{\text{type}}(h,r,t))) \tag{3}$$

We combine the two losses via a simple weighted addition with a tunable hyperparameter α:

$$\mathcal{L} = \mathcal{L}_{\text{fact}} + \alpha\mathcal{L}_{\text{type}} \tag{4}$$

Biased negative sampling. We also examine the effect of using the labelling method described in Section 3 for making the contrastive training more challenging for the model. For this, we keep the model as is, but with probability p, instead of corrupting the head or the tail of the triple with a random entity, we corrupt it with a type-consistent entity.

4 Experiments

We perform our experiments on YAGO3-10 [10, 2], which has 123,182 entities and 37 relations. We use the standard ranking task for evaluation, where for a triple (h,r,t), the model tries to score the original triple higher than the corrupted negative triples. Before ranking the triples, we filter out the corrupted triples that have occurred in the training, validation or test set as described in [1], and we report average hits@1, 3, 10 and mean reciprocal rank (MRR).

We optimize all models with stochastic gradient descent using Adam [5], and perform early stopping with hits@10 on the validation set, where evaluation is performed every five epochs. We fix initial learning rate to 0.001, and the embedding dimension to 200.

We perform a grid search over batch sizes: $\{200, 500, 1000\}$, negative ratios $n_{neg} : \{50, 100\}$, type-consistent sampling probability $p : \{0.1, 0.2, 0.3\}$ and type-loss weight $\alpha : \{0.25, 0.5\}$ where applicable. We choose the hyperparameters that give the highest hits@10 on the validation set, and use these hyperparameters to report the final results on the test. The results are presented in Table 1.

To isolate the effect of our techniques on varying batch sizes and negative ratios, we perform two further experiments. First, we keep everything but the batch size constant ($n_{neg} = 25$, $\alpha = 0.5$, $p = 0.3$) and plot the change in hits@10 as the batch size varies in $\{25, 50, 100, 200, 500, 1000\}$. Second, we keep everything but n_{neg} constant (batch size = 200, $\alpha = 0.5$, $p = 0.3$) and plot hits@10 as n_{neg} in $\{5, 10, 25, 50, 100, 200\}$. The results are presented in Figure 1.

4.1 Results and Discussion

In Table 1 it could be seen that our proposed methods improve the performance on the baseline model with sampled negatives (**Baseline**). Joint ComplEx (**Joint**) on its own provides a slight improvement over the baseline, and combining our two techniques (**Joint-TypedNeg**) gives 5.6% points improvement on hits@1. We note that none of the methods using sampled negatives beat the vanilla model which scores the triple against the entire set of candidate entities (full-softmax), the setting that is not feasible for a large KG.

ComplEx with only typed-negatives (**TypedNeg**) performs slightly under the baseline on all measures. We hypothesize that this is because the typed-negatives ratio is very sensitive to the batch size, and these two parameters needed to be optimized with a finer grid than we had. Note that this sensitivity disappears when **TypedNeg** is combined with **Joint**.

	Hits@1	Hits@3	Hits@10	MRR
ComplEx (full-softmax)	0.468	0.599	0.702	0.550
Baseline	0.277	0.44	0.589	0.383
TypedNeg	0.276	0.427	0.568	0.375
Joint	0.287	0.447	0.601	0.392
Joint-TypedNeg	**0.333**	**0.477**	**0.617**	**0.428**

Table 1: Results on the test set with the best parameters found in grid search for each model.

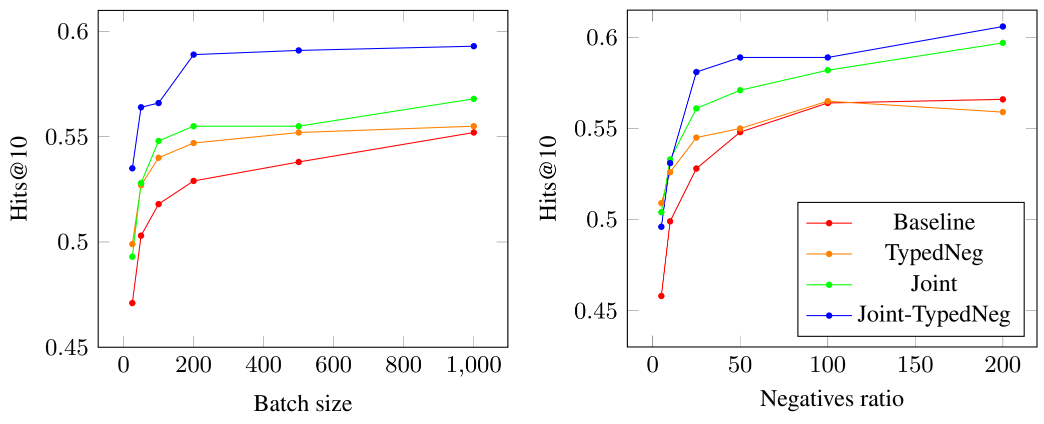

Figure 1: Performance on the test set with different batch sizes (on the left) and with different negative ratios (on the right).

The second set of experiments (Figure 1) shows that the **Joint-TypedNeg** not only consistently performs the best over the entire range of parameters, but also delivers a performance improvement that is especially large when the batch size or the negative ratio is small. This setting was designed to reflect the training conditions on very large datasets. On Figure 1 it can be seen that **TypedNeg** is more robust to low values of negative ratios, and both TypedNeg and **Joint** alone show less deterioration in performance as the batch size decreases. When these two methods are combined in **Joint-TypedNeg**, the training becomes more robust to different choices on both these parameters.

One reason why the two techniques complement each other especially well might be related to using binary cross entropy (BCE) loss for the type module in **Joint**. We speculate that as the negative ratio increases, the ratio of type-consistent to type-inconsistent triples becomes more skewed, and so the BCE loss becomes less effective in learning the implicit type constraints. Adding type-consistent negative triples in the training with higher probability alleviates this problem by making the classes more balanced, and allows the joint training to be more effective.

5 Conclusion and Further Work

We present two techniques that utilize inferred type-consistency labels for learning embeddings for KG completion, one by generating type-consistent negative examples with higher probability, and the second by doing joint training with the original task and the type-consistency labels. We have shown that the first technique makes the model more robust to different values of the number of generated negative samples, and the second one to smaller batch sizes. Furthermore, we have shown that these techniques complement each other well and perform above the baseline when combined.

We have demonstrated that our methods show large performance gains when the batch size and the negative ratio is small relative to the size of the dataset. We argue that this reflects the situation when training on very large Knowledge Graphs (KGs), and so our techniques show promise in improving performance when the datasets get very large. For future work, we plan to experiment on a wider range of datasets and with different bilinear models, integrate explicit type labels and constraints to our methods, and further investigate type-enforcing regularization techniques.

References

[1] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Durán, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *Proceedings of the 26th International Conference on Neural Information Processing Systems*, pages 2787–2795. Curran Associates Inc., 2013.

[2] Tim Dettmers, Minervini Pasquale, Stenetorp Pontus, and Sebastian Riedel. Convolutional 2D Knowledge Graph Embeddings. In *Proceedings of the 32th AAAI Conference on Artificial Intelligence*, pages 1811–1818, 2 2018.

[3] Armand Joulin, Edouard Grave, Piotr Bojanowski, Maximilian Nickel, and Tomas Mikolov. Fast Linear Model for Knowledge Graph Embeddings. *arXiv preprint arXiv:1710.10881*, 2017.

[4] Rudolf Kadlec, Ondrej Bajgar, and Jan Kleindienst. Knowledge Base Completion: Baselines Strike Back. *arXiv preprint arXiv:1705.10744*, 2017.

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

[6] Denis Krompaß, Stephan Baier, and Volker Tresp. Type-constrained representation learning in knowledge graphs. In *International Semantic Web Conference*, pages 640–655, 2015.

[7] Timothée Lacroix, Nicolas Usunier, and Guillaume Obozinski. Canonical Tensor Decomposition for Knowledge Base Completion. *arXiv preprint arXiv:1806.07297*, 2018.

[8] Maximilian Nickel. *Tensor Factorization for Relational Learning*. PhD thesis, Ludwig-Maximilians-Universität München, 2013.

[9] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. Factorizing YAGO: scalable machine learning for linked data. In *Proceedings of the 21st International Conference on World Wide Web*, pages 271–280, New York, New York, USA, 2012. ACM.

[10] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. YAGO. In *Proceedings of the 16th International Conference on World Wide Web - WWW '07*, page 697, New York, New York, USA, 2007. ACM Press.

[11] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Eric Gaussier, and Guillaume Bouchard. Complex Embeddings for Simple Link Prediction. In *Proceedings of The 33rd International Conference on Machine Learning*, pages 2071 – 2080, 2016.

[12] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding Entities and Relations for Learning and Inference in Knowledge Bases. *arXiv preprint arXiv:1412.6575*, 12 2014.