Vol.2019-NL-241 No.14
2019/8/29

A Simple Reranking Method for Knowledge Graph Completion

Lu Yuxun[1,a)] Yutaro Shigeto[2,b)] Katsuhiko Hayashi[3,c)] Masashi Shimbo[1,d)]

Abstract: It has recently been reported that the performance of knowledge graph completion (KGC) is improved by learning type embeddings of entities and relation arguments in addition to normal embeddings. We argue that the type information can be induced from the normal embedding of entities learned by existing KGC models. A simple reranking method is proposed that solely relies on normal entity and relation embeddings. This method requires computing two vectors from normal embeddings per relation, and has four hyperparameters per relation to be tuned on the validation data. Experimental results show that its performance is close to the approach based on type embeddings, although it does not require retraining of embeddings.

1. Introduction

1.1 Background

Knowledge graphs (KGx) have been recognized as an important resource to tackle many natural language processing tasks, such as information extraction [6], question answering [9], and named entity disambiguation [12]. A KG stores facts gathered from various sources in the form of triplets (h, r, t), which represents the proposition that relation r holds between "head" entity h and "tail" entity t. For example, a fact "Alice was born in London" can be represented as (*Alice*, *Born_in*, *London*) in a KG. A KG can be represented as a directed graph composed of entities as nodes and relations as arc labels. Every triplet thus corresponds to an arc in the graph.

In general, KGs are incomplete because of the difficulty of collecting an enormous number of facts in real world. Thus, one of the important tasks on KGs is *knowledge graph completion* (KGC), i.e., to supply knowledge graphs with new facts in order to make them more complete. Instead of directly inferring new facts, the task of KGC is often formulated as that of *link prediction*: Given a triplet with either head h or tail t missing in a query, predict most possible entities for the missing term.

Due to the discrete, symbolic nature of KGs, solving KGC directly using the topology of the graph is often difficult. For this reason, many popular KGC models are based on vector embeddings of graphs [1], [14], [19], [21]. These models map components of a KG (namely, entities and relations) into a continuous, high dimensional vector space, substituting operations in the vector space for manipulation of symbols. When applied to the link prediction task, they assign all entities a score by a *score function*, and rank them by their scores. The entities with higher scores are more likely to be an appropriate candidate for the missed term.

Despite KGC models serve as a feasible approach to solve link prediction, most models do not consider the type constraint of relations. For queries relevant to relation *Born_in*, such as (*Alice*, *Born_in*, ?) or (?, *Born_in*, *London*), entities of type "Food" (e.g., *Apples*, *Pears*, *Milk*) barely become a correct answer. As a result, these irrelevant entities in the KG lead to unsatisfying prediction.

There are many studies aiming to alleviate this issue [2], [5], [7], [16], [20]. Most of them [2], [5], [16], [20] require resources (attributes, type information, etc.) in addition to KG so that they can encode the type of entities as parameters, and use these parameters to verify if entities match the constraints for a relation. However, the required resources are not always available; they are usually handcrafted by humans and are expensive to construct.

Recently, Jain et al. [7] have extended two existing KGC models, DistMult [21] and ComplEx [19], to learn the embeddings of types of entities and relation arguments jointly with normal embeddings. The score functions are also modified to use the learned type embeddings in a way that soft constraints on types are imposed on relation arguments. Different from the work listed above, their method does not require additional resources on entity types, but induces type information from a KG.

In this technical report, we propose another method that learns and incorporates type information and constraints to KGC without using external resources. In contrast to Jains et al.'s, our method does not require training of additional embeddings. It works as a post-processing step following the normal training of DistMult and ComplEx, and can thus be regarded as a reranking method. Experimental results showed that our method improves the performance on KGC tasks over the base DistMult and ComplEx models, and is competitive with Jain et al.'s models.

1 Nara Institute of Science and Technology, Ikoma, Nara, Japan
2 Chiba Institute of Technology, Narashino, Chiba, Japan
3 Osaka University, Suita, Osaka, Japan
a) lu.yuxun.lp6@is.naist.jp
b) shigeto@stair.center
c) katsuhiko-h@sanken.osaka-u.ac.jp
d) shimbo@is.naist.jp; corresponding author

1.2 Notation

Vectors are denoted by lowercase boldface letters, and matrices by uppercase boldface letters. $\mathcal{G}$ denotes a KG, $\mathcal{E}$ denotes the set of entities in $\mathcal{G}$, and $\mathcal{R}$ denotes the set of relations in $\mathcal{G}$. KG $\mathcal{G}$ is identified with the facts (triplets) it contains; thus, $\mathcal{G} \subset \mathcal{E} \times \mathcal{R} \times \mathcal{E}$. For entity $i \in \mathcal{E}$, its vector embedding is written $\mathbf{e}_i$. For relation $j \in \mathcal{R}$, its embedding is $\mathbf{r}_j$. We often use letters h and t to denote the head and tail entities of a triplet, and letter r to denote a relation, e.g., $(h, r, t) \in \mathcal{E} \times \mathcal{R} \times \mathcal{E}$. For brevity, their vector embeddings $\mathbf{e}_h$, $\mathbf{e}_t$, and $\mathbf{r}_r$ are also written $\mathbf{h}$, $\mathbf{t}$, and $\mathbf{r}$. For a complex vector $\mathbf{v}$, $\overline{\mathbf{v}}$ denotes its complex conjugate, and $\mathrm{Re}(\mathbf{v})$ and $\mathrm{Im}(\mathbf{v})$ are operators that take the real part or imaginary part of $\mathbf{v}$, respectively. For two (complex) vectors $\mathbf{u}, \mathbf{v}$, $\mathbf{u} \odot \mathbf{v}$ denotes their elementwise product, and $\langle \mathbf{u}, \mathbf{v} \rangle_{\mathrm{H}} = \mathbf{u}^{\mathrm{T}} \overline{\mathbf{v}}$ denotes the Hermitian inner product. For real vectors $\mathbf{u}$, $\mathbf{v}$, $\mathbf{w} \in \mathbb{R}^d$, we write $\langle \mathbf{u}, \mathbf{v}, \mathbf{w} \rangle$ to denote $(\mathbf{u} \odot \mathbf{v})^{\mathrm{T}} \mathbf{w} = \mathbf{u}^{\mathrm{T}} (\mathbf{v} \odot \mathbf{w}) = \sum_{k=1}^{d} u_k v_k w_k$, where u_k, v_k, w_k are kth component of the respective vectors.

2. Related Work

2.1 Knowledge Graph and Knowledge Graph Completion Models

Due to the discrete nature of KGs, it is difficult to utilize them directly for KGC. Thus, many popular KGC methods do not directly operate on the structure of KGs, but are based on *knowledge graph embeddings*, i.e., embedding of entities and relations in vector space. More specifically, these methods define a *score function* $f(h, r, t)$ to assigns to triple (h, r, t) a score of its plausibility, in terms of the vector embeddings $\mathbf{h}$ and $\mathbf{t}$ of h and t, respectively, and embedding $\mathbf{r}$ of relation r. In typical benchmark tasks of KGC, this score function is used to evaluate the plausibility of entities in the KG as possible candidate of the missed term in a triplet $(h, r, ?)$ or $(?, r, t)$. The entities are ranked by their scores, and the prediction is a list of these ranked entities.

Embedding-based KGC models can be categorized into two types according to the style of their score function: *translation models* and *multiplicative models*.

Translation models define score functions based on distance. A large distance indicates low plausibility of an entity to be the right candidate for a query. TransE [1] is a representative translation model. In TransE, relations are represented as translations in the vector space. If a fact (h, t, r) is observed in the KG, then the embedding $\mathbf{h}$ of entity h should be close to the embedding $\mathbf{t}$ of t after translated by the embedding $\mathbf{r}$ of relation r, i.e., $\mathbf{h} + \mathbf{r} \approx \mathbf{t}$. The score function of TransE is thus given by

$$f_{\mathrm{TE}}(h, r, t) = -\|\mathbf{h} + \mathbf{r} - \mathbf{t}\|, \tag{1}$$

where $\|\cdot\|$ stands for 2-norm, but 1-norm could also be used. As the score function is not symmetric, i.e., $f_{\mathrm{TE}}(h, r, t) \neq f_{\mathrm{TE}}(t, r, h)$ unless $\mathbf{r} = \mathbf{0}$, TransE is not good at processing symmetric relations.

Multiplicative models build score functions using bilinear form. DistMult [21] is a classical model of this type. The score function of DistMult is

$$f_{\mathrm{DM}}(h, r, t) = \langle \mathbf{h}, \mathbf{r}, \mathbf{t} \rangle = \sum_{k=1}^{d} h_k r_k t_k, \qquad \mathbf{h}, \mathbf{r}, \mathbf{t} \in \mathbb{R}^d. \tag{2}$$

where h_k, r_k, and t_k represent the ith component of $\mathbf{h}$, $\mathbf{r}$, and $\mathbf{t}$. DistMult cannot provide good representations for asymmetric relations, as $f_{\mathrm{DM}}(h, r, t) = f_{\mathrm{DM}}(t, r, h)$. Another multiplicative model ComplEx [19] compensates this flaw by embedding entities and relations as vectors not in real space $\mathbb{R}^d$ but in complex space $\mathbb{C}^d$. The score function of ComplEx is defined as follows.

$$\begin{aligned}
f_{\mathrm{CP}}(h, r, t) &= \mathrm{Re}(\langle \mathbf{h}, \mathbf{r}, \overline{\mathbf{t}} \rangle) \\
&= \mathrm{Re}\left(\sum_{i=1}^{d} h_i r_i \overline{t_i} \right) \\
&= \langle \mathrm{Re}(\mathbf{h}), \mathrm{Re}(\mathbf{r}), \mathrm{Re}(\mathbf{t}) \rangle \\
&\quad + \langle \mathrm{Im}(\mathbf{h}), \mathrm{Re}(\mathbf{r}), \mathrm{Im}(\mathbf{t}) \rangle \\
&\quad + \langle \mathrm{Re}(\mathbf{h}), \mathrm{Im}(\mathbf{r}), \mathrm{Im}(\mathbf{t}) \rangle \\
&\quad - \langle \mathrm{Im}(\mathbf{h}), \mathrm{Im}(\mathbf{r}), \mathrm{Re}(\mathbf{t}) \rangle, \qquad \mathbf{h}, \mathbf{r}, \mathbf{t} \in \mathbb{C}^d.
\end{aligned} \tag{3}$$

To represent a symmetric relation in ComplEx, $\mathrm{Im}(\mathbf{r})$ must be set to 0 so that the last two terms on the right-hand side of Eq. (3) vanish and thereby $f_{\mathrm{CP}}(h, r, t) = f_{\mathrm{CP}}(t, r, h)$. Otherwise, it breaks the symmetry, allowing the model to accurately represent both symmetric and asymmetric relations.

2.2 Incorporating Type Constraints in Knowledge Graph Completion Models

The knowledge graph embedding models above do not consider type constraints of relations in their score function. However, types play an important role in prediction. For example, an entity of type "location" is apparently more appropriate as the tail for triplets with relation *Born_in*. Hence, many methods have been proposed for integrating type information into current KGC models. The central idea is to consider types of entities that a relation can take as its head and tail arguments, so that entities that do not match the requested types will not be ranked at the top of the prediction.

Krompaß et al. [10] integrate type information into RESCAL. They refer to `rdf:domain` and `rdf:range` tags to filter out facts that have entities mismatching the type constraints in the training set, and use filtered training examples to learn the entity and relation embeddings so that only entities that match the type constraints will be assigned high scores in prediction.

Type-embodied Knowledge Representation Learning (TKRL) proposed by Xie et al. [20] extend TransE with type encoding matrices. The score function of TKRL is defined as $f_{\mathrm{TKRL}}(h, r, t) = \|\mathbf{M}_r^{(\mathrm{head})}\mathbf{h} + \mathbf{r} - \mathbf{M}_r^{(\mathrm{tail})}\mathbf{t}\|$, where $\mathbf{h}$, $\mathbf{r}$, $\mathbf{t}$ are the vector embeddings of h, r, and t, respectively, and $\mathbf{M}_r^{(\mathrm{head})}$ and $\mathbf{M}_r^{(\mathrm{tail})}$ are the projection matrices for head and tail entities. These projection matrices are defined in terms of n type matrices $\{\mathbf{M}^{(k)}\}_{k=1}^{n}$, as in: $\mathbf{M}_r^{(\mathrm{head})} = \sum_{k=1}^{n} \alpha_{r,i}^{(\mathrm{head})} \mathbf{M}^{(k)} / \sum_{k=1}^{n} \alpha_{r,i}^{(\mathrm{head})}$. Here, if a type matches the heads of relation r, $\alpha_{r,k}^{(\mathrm{head})} = 1$, otherwise $\alpha_{r,k}^{(\mathrm{head})} = 0$. $\mathbf{M}_r^{(\mathrm{tail})}$ is defined similarly.

Semantically Smooth Embedding (SSE) proposed by Guo et al. [5] enforces the vector space to be semantically smooth. They assume that: (1) entities belonging to the same type shall stay close to each other in the vector space; (2) every entity shall be able to be constructed by a linear combination of its "neighbors." They want to enforce the neighbors of an entity e to be entities with the same category as e in the vector space. They use two regularizers R_1 and R_2 in the loss function to implement these assumptions. The first regularizer R_1 stands for assumption 1 and is defined as follows:

$$R_1 = \frac{1}{2}\sum_{i\in\mathcal{E}}\sum_{j\in\mathcal{E}} \left\|\mathbf{e}_i - \mathbf{e}_j\right\|^2 \omega_{ij}^{(1)} \quad (4)$$

where $\mathcal{E}$ is the set of all entities, $\mathbf{e}_i$ denotes the vector embedding of entity i, and $\omega_{ij}^{(1)} = 1$ if entities i and j belong to the same type; otherwise $\omega_{ij}^{(1)} = 0$. The second regularizer R_2 reflects assumption 2, and is given by:

$$R_2 = \sum_{i\in\mathcal{E}} \left\| \mathbf{e}_i - \sum_{j\in\mathcal{N}(i)} \omega_{ij}^{(2)} \mathbf{e}_j \right\|^2 . \quad (5)$$

In this equation, $\mathcal{N}(i) \subset \mathcal{E}$ denotes the set of neighbor entities[*1] of entity i, and if entities i and j are in the same category, $\omega_{ij}^{(2)} = 1$, otherwise $\omega_{ij}^{(2)} = 0$.

TKRL, SSE, and the approach from Nickel et al. [15] require external resources indicating entity types.

Recently, Jain et al. [7] proposed TypeDM and TypeComplEx that extend DistMult and ComplEx, respectively, to take type information into account. Differing from Krompaß et al.'s method, TKRL, and SSE, Jain et al.'s models learn soft constraints on the arguments of individual relations in a data-driven manner; i.e., from the knowledge graph given as training data alone, without additional information on entity types and type constraints on relation arguments. Along with normal entity embeddings of DistMult and ComplEx, an additional embedding $\mathbf{u}_i \in \mathbb{R}^{d'}$ is associated with each entity i as a representation of its type. Similarity, two vectors $\mathbf{v}_j, \mathbf{w}_j \in \mathbb{R}^{d'}$ are associated with each relation j, which indicate the type requirement of relation j in head and tail, respectively. With these additional vectors, the score functions for TypeDM and TypeComplEx are defined as:

$$f_{\text{TypeDM}}(h,r,t) = \sigma(f_{\text{DM}}(h,r,t))\, C^{(\text{head})}(h,r)\, C^{(\text{tail})}(t,r), \quad (6)$$

$$f_{\text{TypeComplEx}}(h,r,t) = \sigma(f_{\text{CP}}(h,r,t))\, C^{(\text{head})}(h,r)\, C^{(\text{tail})}(t,r), \quad (7)$$

with

$$C^{(\text{head})}(h,r) = \sigma(\langle \mathbf{u}_h, \mathbf{v}_r \rangle),$$
$$C^{(\text{tail})}(t,r) = \sigma(\langle \mathbf{u}_t, \mathbf{w}_r \rangle),$$

where $\sigma(\cdot)$ is the sigmoid function, and f_{DM} and f_{ComplEx} are the score functions of vanilla DistMult and ComplEx, given by Eqs. (2) and (3), respectively. The factor $C^{(\text{head})}(h,r)$ captures the compatibility of the given entity h as the head entity of relation r, and $C^{(\text{tail})}(t,r)$ captures the compatibility of entity t as its tail.

The score function for TypeComplEx is defined similarly, except that the score function f_{CP} (Eq. (3)) of ComplEx is used in place of f_{DM}. The new embeddings $\mathbf{u}_i, \mathbf{v}_j, \mathbf{w}_j \in \mathbb{R}^{d'}$ are trained jointly with the embeddings of the original DistMult or ComplEx model, with the modified score function $f_{\text{TypeDM}}(h,r,t)$ or $f_{\text{TypeComplEx}}(h,r,t)$ used in the training loss.

[*1] See [5] for the precise definition of the neighbor entities.

3. Proposed Method

As we have seen in the previous section, TypeDM and TypeComplEx learn extra vectors representing types from training data. However, the original embeddings of DistMult and ComplEx are also trained on the same training data, and similar entities are expected to have a similar vector embeddings. A question thus arises whether such information on types, which is basically a group of similar entities, are already present in the original embeddings. In this section, we propose a reranking method that extracts type information from the original DistMult and ComplEx embeddings.

We made two assumptions for our method: (1) entities with the same type have embeddings closed to each other in the vector space; (2) For a fact (h,r,t), the embedding $\mathbf{r} \odot \bar{\mathbf{t}}$ is close to embeddings of all heads that related with r, and $\mathbf{h} \odot \mathbf{r}$ is close to embeddings of all tails related with r, where $\odot$ is the elementwise multiplication.

We do not need any resource containing type information or extra parameters representing entities types in our method. Instead, we assign two factors to represent acceptable entity types in each relation. Afterwards, we modify the score functions in DistMult and ComplEx with these factors to incorporate type constraints.

The type constraints are represented in our method as vectors $\mathbf{c}_r^{(\text{head})}$ and $\mathbf{c}_r^{(\text{tail})}$, which are intended to provide the "prototypes" of typical head and tail entities of a relation r.

$$\mathbf{c}_r^{(\text{head})} = \frac{1}{|\mathcal{T}_r^{(\text{head})}|} \sum_{h\in\mathcal{T}_r^{(\text{head})}} \mathbf{e}_h, \quad (8)$$

$$\mathbf{c}_r^{(\text{tail})} = \frac{1}{|\mathcal{T}_r^{(\text{tail})}|} \sum_{t\in\mathcal{T}_r^{(\text{tail})}} \mathbf{e}_t. \quad (9)$$

Here, $\mathcal{T}_r^{(\text{head})}$ and $\mathcal{T}_r^{(\text{tail})}$ are multisets containing heads (or tails) in the training set related to r. $|T_r^{(\text{head})}|$ and $|T_r^{(\text{tail})}|$ are the cardinality of $\mathcal{T}_r^{(\text{head})}$ and $\mathcal{T}_r^{(\text{tail})}$. $\mathbf{e}_t$ and $\mathbf{e}_h$ are entity embeddings learned by DistMult or ComplEx.

Our score function is

$$\begin{aligned} f_{\text{rerank}}(h,r,t) = {} & f_{\text{base}}(h,r,t) \\ & + \alpha_r \ \mathrm{Re}\left(\langle \mathbf{h}, \mathbf{c}_r^{(\text{head})} \rangle_{\mathrm{H}}\right) \\ & + \beta_r \ \mathrm{Re}\left(\langle \mathbf{t}, \mathbf{c}_r^{(\text{tail})} \rangle_{\mathrm{H}}\right) \\ & + \gamma_r \ \mathrm{Re}\left(\langle \mathbf{r} \odot \bar{\mathbf{t}}, \mathbf{c}_r^{(\text{head})} \rangle_{\mathrm{H}}\right) \\ & + \epsilon_r \ \mathrm{Re}\left(\langle \mathbf{r} \odot \mathbf{h}, \mathbf{c}_r^{(\text{tail})} \rangle_{\mathrm{H}}\right), \quad (10) \end{aligned}$$

where $f_{\text{base}}(h,r,t)$ is the score function from the base model, specifically, DistMult (Eq. (2)) or ComplEx (Eq. (3)), $\mathbf{h}, \mathbf{r}, \mathbf{t}$ are the embeddings of h, r, and t in the base model, respectively, $\sigma(\cdot)$ is the sigmoid function, and $\alpha_r, \beta_r, \gamma_r, \epsilon_r \geq 0$ are four scalar hyperparameters introduced for relation r.

In Eq. (10), four extra terms are added to the score functions of the base model. If an entity is dissimilar to (i.e., having small inner product with) the prototype provided by the centroids (i.e., $\mathbf{c}_r^{(\text{head})}$ or $\mathbf{c}_r^{(\text{tail})}$) or its transformation by relation $\mathbf{r}$, its score is discounted by these terms since the inner products in the additional terms can be remain low, resulting in a low re-ranked position in the prediction given by f_{rerank}. On the other hand, entities matching type constraints will be re-ranked higher than they were as long as they do not have an extremely low score from base models.

A major difference from Jain et al.'s is that our model uses the embeddings of the base models to infer type constraints; i.e., in Eq. (10), $\mathbf{h}$, $\mathbf{t}$, $\mathbf{r}$ are vectors of the base model (DistMult or ComplEx), and $\mathbf{c}_r^{(\text{head})}$ and $\mathbf{c}_r^{(\text{tail})}$ are also computed from the entity vectors of the base models. By contrast, Jain et al.'s need to train additional type embeddings and constraint vectors.

4. Experiment

We conducted an empirical evaluation of the proposed method on the standard datasets for KBC.

4.1 Datasets

Three standard datasets were used in our experiment: FB15K237 [18], WN18RR [3], and YAGO3-10 [13][*2]. FB15K237 is a subset of Freebase. Most relations in FB15K237 are very specific, e.g., "influenced by", "jurisdiction of office", "honored for", etc. A large part of FB15K237 is about movies, sports, awards, and actors.

WN18RR is a subset of WordNet. It has 11 generic relations ("hypernym of", "hyponym of", "synonym of", etc.).

YAGO3-10 is a subset of YAGO. Most facts in YAGO3-10 are attributes of people, such as nationality, gender, and profession.

We followed the split of training, validation, and test data as distributed in their respective datasets. Table 4 shows the statistics of each dataset.

4.2 Compared Methods

We tested two proposed models given by Eq. (10); RerankDM, which is obtained when DistMult is used as the base model in Eq. (10), and RerankComplEx where ComplEx is the base model.

These models were compared with their base models, DistMult and ComplEx. They are also compared with TypeDM and TypeComplEx [7]; see Sec. 2.2.

4.3 Experiment Setup

Performance Metrics. We used Hits@N with $N = 1$ and Mean Reciprocal Rank (MRR) as performance metrics. Both metrics are calculated in a "filtered" manner: Given the prediction (i.e., list of all entities sorted by the score in descending order) for a query $(h, r, ?)$ with ground truth t^*, entities $t \neq t^*$ in prediction such that $(h, r, t) \in \mathcal{G}$ will be removed from the prediction, where $\mathcal{G}$ is the knowledge graph[*3]. For triplets (h, r, t) in test set $\mathcal{S}$, the model computes the rank of h with r and t for metrics related to heads, and similarly, rank of t with h and r for metrics related to tails. Let rank(e) be the rank of entity e in the filtered prediction. MRR and Hits@N are then computed for this model as follows.

[*2] We use the datasets FB15K237 and WN18RR provided by OpenKE at https://github.com/thunlp/OpenKE/tree/master/benchmarks. YAGO3-10 is downloaded from https://github.com/TimDettmers/ConvE and preprocessed with scripts from OpenKE.

[*3] For tuning hyperparameters, we only removed (h, r, t) with $t \neq t^*$ in training and validation set.

$$\text{Hits@}N\text{ (Head)} = \frac{1}{|\mathcal{S}|}\left|\{(h,r,t) \in \mathcal{S} \mid \text{rank}(h) \le N\}\right|,$$
$$\text{Hits@}N\text{ (Tail)} = \frac{1}{|\mathcal{S}|}\left|\{(h,r,t) \in \mathcal{S} \mid \text{rank}(t) \le N\}\right|,$$
$$\text{Hits@}N\text{ (Avg)} = \frac{1}{2}(\text{Hits@}N\text{ (Head)} + \text{Hits@}N\text{ (Tail)}),$$
$$\text{MRR (Head)} = \frac{1}{|\mathcal{S}|}\sum_{(h,r,t)\in\mathcal{S}} \frac{1}{\text{rank}(h)},$$
$$\text{MRR (Tail)} = \frac{1}{|\mathcal{S}|}\sum_{(h,r,t)\in\mathcal{S}} \frac{1}{\text{rank}(t)},$$
$$\text{MRR (Avg)} = \frac{1}{2}(\text{MRR (Head)} + \text{MRR (Tail)}).$$

We used Hits@1(Avg) as the main metric to pick hyperparameters and train the models, as it balances the Hits@1 (head) and Hits@1 (Tail).

Loss Function. We used logistic loss and negative sampling to learn embeddings of our base models (DistMult and ComplEx):

$$\mathcal{L}_{\text{logistic}} = \sum_{(h,r,t)\in\mathcal{T}} \log(1 + \exp(-f_{\text{base}}(h, r, t))) \tag{11}$$
$$+ \sum_{(h',r,t')\in\mathcal{T}'} \log(1 + \exp(f_{\text{base}}(h', r, t'))), \tag{12}$$

where $\mathcal{T}$ is the training set, and $\mathcal{T}'$ contains corrupted facts (h', r, t'), produced by corrupting a fact $(h, r, t) \in \mathcal{T}$ with either its head h or tail t replaced by a random entity (h' or t') uniformly sampled from $\mathcal{E}$ so that $(h', r, t') \notin \mathcal{T}$. f_{base} is the score function of DistMult (Eq. (2)) or ComplEx (Eq. (3)). We set the number of negative examples to be 20, as our benchmarks [7] performed best with this setting according to results obtained by grid search.

For our benchmarks TypeDM and TypeComplEx, we followed the way of training in [7]. The loss function was log-likelihood loss:

$$\mathcal{L}_{\text{log-likelihood}} = -\sum_{(h,r,t)\in\mathcal{T}} (\log \Pr(t \mid h, r) + \log \Pr(h \mid r, t)), \tag{13}$$

where

$$\Pr(t \mid h, r) = \frac{\exp(\theta f(h, r, t))}{\sum_{t'} \exp(\theta f(h, r, t'))}, \tag{14}$$
$$\Pr(h \mid r, t) = \frac{\exp(\theta f(h, r, t))}{\sum_{h'} \exp(\theta f(h', r, t))}. \tag{15}$$

In Eqs. (14) and (15), $f(h, r, t)$ is the score function of TypeDM or TypeComplEx defined in Eq. (6).

For all models, we used ℓ_2 regularizer of embeddings.

$$R = \sum_{i\in\mathcal{E}} \|\mathbf{e}_i\|^2 + \sum_{j\in\mathcal{R}} \|\mathbf{r}_j\|^2. \tag{16}$$

Parameter Setting and Optimization We set the dimension of embeddings in base models to be 200. As for benchmarks, the dimension of type embeddings ($\mathbf{u}_i, \mathbf{v}_j, \mathbf{w}_j$) in TypeDM and TypeComplEx was 19, and 180 for entity and relation embeddings, as Jain et al. did in [7]. The embeddings in DistMult and TypeDM were initialized by uniform Xavier initializer [4], and embeddings in ComplEx and TypeComplEx were initialized by Gaussian distribution with mean 0 and standard deviation 0.05.

Table 1 Statistics of FB15K237, WN18RR, and YAGO3-10.

	# entities	# relations	# triplets			
			training	validation	test	avg per relation (training)
FB15K237	14,541	237	272,115	17,535	20,466	1,148
WN18RR	40,943	11	86,835	3,034	3,134	7,894
YAGO3-10	123,182	37	1,079,040	5,000	5,000	29,163

We used Adam [8] as the optimizer. The number of training epochs was 1,000 with early stopping; the performance was evaluated every 50 epochs and the one with the best Hits@1 (avg) on the validation set was picked. We used grid search to set the batch size and learning rate. The scope of batch size was $\{4096, 8192, 16384, 32768, 65536\}$. The range for learning rate and the weight of ℓ^2 regularizer was $\{1\times10^{-5}, 5\times10^{-5}, 1\times10^{-4}, 5\times10^{-4}, 1\times10^{-3}, 5\times10^{-3}\}$.

For DistMult, the norm of entity embeddings was normalized to 1 at the end of every epoch. We applied gradient clipping for ComplEx and TypeComplEx; gradients with norm larger than 1 were normalized to norm 1.

After training, embeddings from DistMult and ComplEx were used for computing $\mathbf{c}_r^{(\text{head})}$ and $\mathbf{c}_r^{(\text{tail})}$ by Eq. (8) and Eq. (9). Hyperparameters $\alpha_r, \beta_r, \gamma_r, \epsilon_r$ in Eq. (10) were tuned on the validation set by grid search.

4.4 Results

The performances on FB15K237, WN18RR, and YAGO3-10 are shown in Tables 2, 3, and 4, respectively. The proposed RerankDM and RerankComplEx models improved performance upon their respective base models (DistMult and ComplEx). They also achieved competitive results compared wtih their counterparts on all three datasets. In particular, RerankComplEx outperfomed other approaches on WN18RR and YAGO30-10.

4.5 Discussion

The proposed methods showed the best performance on WN18RR and YAGO30-10, but not FB15K237. This is because in FB15K237, some entities e related to r are always close to centroids $\mathbf{c}_r^{(\text{head})}$ or $\mathbf{c}_r^{(\text{tail})}$ in the vector space, so that $\langle \mathbf{e}, \mathbf{c}_r^{(\text{head})} \rangle$ or $\langle \mathbf{e}, \mathbf{c}_r^{(\text{tail})} \rangle$ are high. Moreover, the score f_{base} in base model for those entities are not low enough to allow our method to distinguish the golden truth.

For instance, in the relation "influenced by" for entities with type "comedians", some famous comedians, like "Louis C.K.", "George Carlin", and "Sarah Silverman" have high score in the score function f_{base} of base models and with high scores regarding type constraint factors $\mathbf{c}_r^{(\text{head})}$, and $\mathbf{c}_r^{(\text{tail})}$. That is, not only the score from type factors $\langle \mathbf{e}_{\text{Louis C.K.}}, \mathbf{c}^{(\text{tail})}_{\text{influenced by}} \rangle$ is large, but also the score from our base models $f_{\text{base}}(h, \text{influenced by}, \text{Louis C.K.})$ is not very low no matter who the comedian h is, resulting in low rank for ground truth in the prediction, and lead to unsatisfying performance.

This phenomenon is caused by two aspects, one from statistical perspective, to be specific, hubs in high dimensional space [17], and the other one from the perspective of situations in real world. The analysis and solution for the statistical aspect are out of the scope in this report, so we will only concern the second aspect: in real world, these comedians are so popular that many facts are about them, and their frequency in the set of training examples related to "influenced by" $\mathcal{T}_{\text{influenced by}}$ are therefore higher than others, and for the relation "influenced by", an entity suitable for heads is often appropriate for tails as well.

For instance, there might be facts in $\mathcal{T}_{\text{influenced by}}$ whose head is always a popular entity A with various tails: $(A, \text{influenced\_by}, B_i)$ for $i = 1, 2, 3, ..., m$ and B_i can appear as heads in other training examples $(B_i, \text{influenced\_by}, C_j)$ for $i = 1, 2, 3, ..., m$. As a result, embeddings of these popular entities affect other unpopular entities in geometric distribution so that embeddings of unpopular entities approach towards their embeddings, leading to $\mathbf{c}_r^{(\text{head})}$ and $\mathbf{c}_r^{(\text{head})}$ too close to the embeddings of popular entities. To tackle this problem, we tried downsampling in calculating $\mathbf{c}_r^{(\text{tail})}$ and $\mathbf{c}_r^{(\text{head})}$: instead of using all triplets in $\mathcal{T}_r^{(\text{head})}$ and $\mathcal{T}_r^{(\text{tail})}$, we only use most representative entities. The representative entities are determined by their frequency. The results on FB15K237 with downsampling are in Table 5. In downsampling method, we used only entities with top 5 frequencies instead of all entities in $\mathcal{T}_r^{(\text{head})}$ and $\mathcal{T}_r^{(\text{tail})}$ to compute $\mathbf{c}_r^{(\text{head})}$ and $\mathbf{c}_r^{(\text{tail})}$.

From results in Table 5, downsampling alleviates the problem, but the performance is still not the best on FB15K237, compared with TypeDM and TypeComplEx. This indicates that the sampling method in the computation of $\mathbf{c}_r^{(\text{tail})}$ and $\mathbf{c}_r^{(\text{head})}$ matters. Sampling method considering the frequency, such as Determinantial Point Process (DPP) [11], may help improve the performance[*4].

On the other hand, TypeDM and TypeComplEx did not encounter such issue, because type embeddings in these two models are independent from embeddings of entities and relations. Therefore, the frequency of entity does not affect these two models severely. However, they also have more parameters (one type embeddings for entities, and two type embeddings for relations) than our method.

5. Conclusion and Future Work

We proposed a method for inferring type constraints of relations on the task of knowledge graph completion, or, the link prediction. Our method works in purely data-driven manner. It does not need any additional resources except facts in the knowledge graph, i.e., those already provided as training data. We represent type constraints by the centroid of embeddings of entities relevant to a specific relation, and modify the score functions of base models to incorporate these soft constraints. As a post-processing approach, our method does not need training for extra parameters. It has only four hyperparameters for each relation. These

[*4] Although the result is not listed in this report, naive DPP did not help with our methods.

Table 2 Hits@1 and MRR of our models, base models, and benchmarks on FB15K237.

	DistMult	TypeDM	RerankDM	ComplEx	TypeComplEx	RerankComplEx
Hits@1 (Head)	0.1177	**0.1382**	0.1228	0.1100	0.1222	0.1107
Hits@1 (Tail)	0.2736	**0.3077**	0.3057	0.2741	0.2724	0.2910
Hits@1 (Avg)	0.1956	**0.2230**	0.2142	0.1920	0.1973	0.2008
MRR (Head)	0.1967	**0.2283**	0.2007	0.1700	0.1952	0.1700
MRR (Tail)	0.3706	**0.4101**	0.3966	0.3549	0.3682	0.3692
MRR (Avg)	0.2837	**0.3192**	0.2987	0.2624	0.2817	0.2696

Table 3 Hits@1 and MRR of our models, base models, and benchmarks on WN18RR.

	DistMult	TypeDM	RerankDM	ComplEx	TypeComplEx	RerankComplEx
Hits@1 (Head)	0.3692	0.3666	0.3969	0.3950	0.3698	**0.4075**
Hits@1 (Tail)	0.4263	0.4218	0.4292	0.4228	0.4075	**0.4314**
Hits@1 (Avg)	0.3977	0.3942	0.4131	0.4089	0.3886	**0.4194**
MRR (Head)	0.4063	0.3945	0.4107	0.4176	0.3869	**0.4332**
MRR (Tail)	0.4613	0.4600	**0.4624**	0.4475	0.4343	0.4517
MRR (Avg)	0.4338	0.4273	0.4365	0.4325	0.4106	**0.4425**

Table 4 Hits@1 and MRR of our models, base models, and benchmarks on YAGO3-10.

	DistMult	TypeDM	RerankDM	ComplEx	TypeComplEx	RerankComplEx
Hits@1 (Head)	0.1414	0.1564	0.1390	0.2732	0.2364	**0.2794**
Hits@1 (Tail)	0.4298	0.4438	0.4329	0.4630	0.5168	**0.5372**
Hits@1 (Avg)	0.2842	0.3001	0.2859	0.3681	0.3766	**0.4083**
MRR (Head)	0.2441	0.2573	0.2451	0.3702	0.3361	**0.3766**
MRR (Tail)	0.5267	0.5224	0.5323	0.5325	0.5840	**0.6074**
MRR (Avg)	0.3854	0.3898	0.3887	0.4514	0.4600	**0.4920**

Table 5 Result on FB15K237 with/without downsampling. Results marked with "(D)" use downsampling, while results with "(A)" do not.

	RerankDM (A)	RerankDM (D)	RerankComplEx (A)	RerankComplEx (D)
Hits@1 (Head)	0.1228	**0.1256**	0.1107	0.1127
Hits@1 (Tail)	0.3057	**0.3096**	0.2910	0.2902
Hits@1 (Avg)	0.2142	**0.2176**	0.2008	0.2015
MRR (Head)	0.2007	**0.2032**	0.1700	0.1707
MRR (Tail)	0.3966	**0.3994**	0.3692	0.3689
MRR (Avg)	0.2987	**0.3013**	0.2696	0.2698

hyperparameters can be tuned on the validation set of a knowledge graph. Because the number of relations are in general more than the number of entities in knowledge graphs, our method is not as memory hungry as others.

This work can be extended in two ways. Firstly, as Table 5 shows, the method of sampling entities in training example to compute type constraint factors has a severe influence on our method. Method with only top 5 representative entities gave a better result than method with all entities sampled in training set. In the reported result, representative entities are merely sampled by its frequency. This process can be improved by Determinantal Point Process (DPP), which is a stochastic process that select representative elements from a set.

Secondly, the type constraint factors in our approach can be trained rather than computed. In future, we plan to design a proper regularizer helping factors in every relation converge to a appropriate location in training.

References

[1] Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J. and Yakhnenko, O.: Translating embeddings for modeling multi-relational data, *Advances in neural information processing systems*, pp. 2787–2795 (2013).

[2] Chang, K.-W., Yih, W.-t., Yang, B. and Meek, C.: Typed Tensor Decomposition of Knowledge Bases for Relation Extraction, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar, Association for Computational Linguistics, pp. 1568–1579 (online), DOI: 10.3115/v1/D14-1165 (2014).

[3] Dettmers, T., Minervini, P., Stenetorp, P. and Riedel, S.: Convolutional 2d knowledge graph embeddings, *arXiv preprint arXiv:1707.01476* (2017).

[4] Glorot, X. and Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks, *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256 (2010).

[5] Guo, S., Wang, Q., Wang, B., Wang, L. and Guo, L.: SSE: Semantically Smooth Embedding for Knowledge Graphs, *IEEE Transactions on Knowledge and Data Engineering*, Vol. 29, No. 4, pp. 884–897 (online), DOI: 10.1109/TKDE.2016.2638425 (2017).

[6] Hoffmann, R., Zhang, C., Ling, X., Zettlemoyer, L. and Weld, D. S.: Knowledge-based weak supervision for information extraction of overlapping relations, *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, Association for Computational Linguistics, pp. 541–550 (2011).

[7] Jain, P., Kumar, P., Mausam and Chakrabarti, S.: Type-Sensitive Knowledge Base Inference Without Explicit Type Supervision, *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, Melbourne, Australia, Association for Computational Linguistics, pp. 75–80 (online), DOI: 10.18653/v1/P18-2013 (2018).

[8] Kingma, D. P. and Ba, J.: Adam: A method for stochastic optimization, *arXiv preprint arXiv:1412.6980* (2014).

[9] Kiyota, Y., Kurohashi, S. and Kido, F.: Dialog navigator: A question answering system based on large text knowledge base, *Proceedings of the 19th international conference on Computational linguistics-Volume 1*, Association for Computational Linguistics, pp. 1–7 (2002).

[10] Krompaß, D., Baier, S. and Tresp, V.: Type-constrained representation learning in knowledge graphs, *International semantic web conference*, Springer, pp. 640–655 (2015).

[11] Kulesza, A., Taskar, B. et al.: Determinantal point processes for machine learning, *Foundations and Trends® in Machine Learning*, Vol. 5, No. 2–3, pp. 123–286 (2012).

[12] Li, Y., Tan, S., Sun, H., Han, J., Roth, D. and Yan, X.: Entity disambiguation with linkless knowledge bases, *Proceedings of the 25th International Conference on World Wide Web*, International World Wide

Web Conferences Steering Committee, pp. 1261–1270 (2016).

[13] Mahdisoltani, F., Biega, J. and Suchanek, F. M.: Yago3: A knowledge base from multilingual wikipedias (2013).

[14] Nickel, M., Rosasco, L., Poggio, T. A. et al.: Holographic Embeddings of Knowledge Graphs., *AAAI*, Vol. 2, pp. 3–2 (2016).

[15] Nickel, M., Tresp, V. and Kriegel, H.-P.: A Three-way Model for Collective Learning on Multi-relational Data, *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, USA, Omnipress, pp. 809–816 (online), available from ⟨http://dl.acm.org/citation.cfm?id=3104482.3104584⟩ (2011).

[16] Nickel, M., Tresp, V. and Kriegel, H.-P.: Factorizing YAGO: Scalable Machine Learning for Linked Data, *Proceedings of the 21st International Conference on World Wide Web*, WWW '12, New York, NY, USA, ACM, pp. 271–280 (online), DOI: 10.1145/2187836.2187874 (2012).

[17] Radovanović, M., Nanopoulos, A. and Ivanović, M.: Hubs in space: Popular nearest neighbors in high-dimensional data, *Journal of Machine Learning Research*, Vol. 11, No. Sep, pp. 2487–2531 (2010).

[18] Toutanova, K., Chen, D., Pantel, P., Poon, H., Choudhury, P. and Gamon, M.: Representing Text for Joint Embedding of Text and Knowledge Bases, *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, Lisbon, Portugal, Association for Computational Linguistics, pp. 1499–1509 (online), DOI: 10.18653/v1/D15-1174 (2015).

[19] Trouillon, T., Welbl, J., Riedel, S., Gaussier, É. and Bouchard, G.: Complex embeddings for simple link prediction, *International Conference on Machine Learning*, pp. 2071–2080 (2016).

[20] Xie, R., Liu, Z. and Sun, M.: Representation Learning of Knowledge Graphs with Hierarchical Types, *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, AAAI Press, pp. 2965–2971 (online), available from ⟨http://dl.acm.org/citation.cfm?id=3060832.3061036⟩ (2016).

[21] Yang, B., Yih, W.-t., He, X., Gao, J. and Deng, L.: Embedding entities and relations for learning and inference in knowledge bases, *arXiv preprint arXiv:1412.6575* (2014).