The Thirty-Second AAAI Conference
on Artificial Intelligence (AAAI-18)

Embedding of Hierarchically Typed Knowledge Bases

Richong Zhang,[1] **Fanshuang Kong,**[1] **Chenyue Wang,**[1] **Yongyi Mao**[2]
[1]BDBC and SKLSDE, School of Computer Science and Engineering, Beihang University
[2]School of Electrical Engineering and Computer Science, University of Ottawa
{zhangrc,kongfs,wangcy}@act.buaa.edu.cn; ymao@uottawa.ca

Abstract

Embedding has emerged as an important approach to prediction, inference, data mining and information retrieval based on knowledge bases and various embedding models have been presented. Most of these models are "typeless", namely, treating a knowledge base solely as a collection of instances without considering the types of the entities therein. In this paper, we investigate the use of entity type information for knowledge base embedding. We present a framework that augments a generic "typeless" embedding model to a typed one. The framework interprets an entity type as a constraint on the set of all entities and let these type constraints induce isomorphically a set of subsets in the embedding space. Additional cost functions are then introduced to model the fitness between these constraints and the embedding of entities and relations. A concrete example scheme of the framework is proposed. We demonstrate experimentally that this framework offers improved embedding performance over the typeless models and other typed models.

Introduction

Recently significant efforts from industry and academia have poured into constructing knowledge bases (Niu et al. 2012; Zhang et al. 2017) to exploit the enormous amount of information available on the Internet. The emerging of KBs such as YAGO, DBpedia, Freebase, etc is stimulating the development of software applications that may have great commercial and societal benefits. This has fostered active research in KBs (e.g., (Marin et al. 2014; Xiong and Callan 2015a)).

Among the diverse directions in KB research, KB embedding (Bordes et al. 2013; Wang et al. 2014; Wen et al. 2016) has received increasing attention. Briefly, KB embedding aims at representing knowledge, namely, the cross-linked entities and relations, as quantities in some Euclidean space. This approach appeals in many applications since it turns the inherent discrete structure of the linked knowledge into a continuous topology, avoiding potential combinatorial complexity. Among other applications, the embedding approach has demonstrated great power in reconstructing missing links in KBs (Xiong and Callan 2015b; Angeli and Manning 2013).

The challenges in KB embedding lies in finding low-complexity representations of knowledge which preserve the structure among the entities and relations and which support inference, induction, deduction and other reasoning mechanisms. As such, any breakthrough in knowledge representation has the potential to greatly advance machine intelligence.

There have been various models presented for KB embedding, with promising performances demonstrated (Wen et al. 2016; Lin et al. 2015; Bordes et al. 2014; 2011; Socher et al. 2013; Lin, Liu, and Sun 2015; Xiao, Huang, and Zhu 2016; He et al. 2015). All these models treat a KB simply as a collection of instances, and the type information of the entities is disregarded.

The type labels of entities arguably contain useful information, since entities of having the same type label are expected to be similar or related in certain ways. Indeed, in some other context, type information is proved useful (see, e.g., (Krompa, Baier, and Tresp 2015; Chang et al. 2014; Wang, Wang, and Guo 2015))

A recent work (Xie, Liu, and Sun 2016) suggests that when type information is built into the model, the embedding performance can be further improved. The approach of (Xie, Liu, and Sun 2016) to incorporating type information is to build it into a particular embedding model, known as TransR (Lin et al. 2015). This approach is specific to the TransR model, and can not be extended to other embedding models. Despite its good performance, the training of TransR and its typed version entails a time complexity that is orders of magnitude higher than other simpler models. It is then desirable to develop new low-complexity models that are capable of utilizing type information. In fact, given that various embedding models have already been proposed, it would be nice to develop a general framework that turns any "typeless model" (i.e. a model without considering type information) into a "typed model" (i.e. a model that exploits type information). This is precisely the objective and contribution of this work.

Types in a KB are usually organized in a hierarchical structure, to capture the ontology of the entities therein. Thus a principled approach is required to exploit such hierarchical structure rather than treating types as unrelated labels. Moreover, an entity usually has multiple types and some of the types may be missing. This requires the framework to be flexible enough to handle entities with varying numbers of types and robust enough to deal with missing types.

In this paper, we present a novel scheme that augments any typeless model to a typed one while considering these factors. Our key insight is that a type may be viewed as a constraint on the entities. The hierarchical type structure can then be understood as a partial order on a collection of such constraints. As embedding preserves the structure among entities, such a partial order induces an isomorphic partial order on a collection of subsets in the embedding space. Modelling the subsets as affine subspaces of the embedding space, we then define additional cost functions to model how entities and relations fit these spaces and augment the cost function of the original typeless model (the "base model") with these type-related cost functions. The KB embedding problem can then be solved by minimizing this new cost function. Via experiments we show that this scheme improves on the base typeless models and other typed embedding models of (Xie, Liu, and Sun 2016) .

Typeless Embedding

The problem of knowledge base (KB) embedding is to map the entities in a KB to vectors in some Euclidean space that preserve the structures among the entities. Most of these models can be unified under a common framework as was suggested in (Wen et al. 2016). In this section, we review this framework and various embedding models that can be understood as examples of this framework.

Typeless Embedding Framework

In (Wen et al. 2016), various existing models are unified under a common framework, of which we now give a concise review.

Let $\mathcal{N}$ denote the set of all entities in a KB. We will use $\mathcal{R}$ to index the set of all relations in the KB, namely, for each index $r \in \mathcal{R}$, there is a relation R_r. Here we will follow the general set up in (Wen et al. 2016) in which a relation R_r is allowed to have arbitrary fold, or arity. Let $\mathcal{M}$ be a set of roles in the KB and each relation R_r is associated with a set $\mathcal{M}(r)$ of roles, and R_r is understood as a set of functions mapping $\mathcal{M}(r)$ to $\mathcal{N}$. For any two sets $\mathcal{A}$ and $\mathcal{B}$, we will use $\mathcal{A}^{\mathcal{B}}$ to denote the set of all functions mapping $\mathcal{B}$ to $\mathcal{A}$. Then every relation R_r is understood as a subset of $\mathcal{N}^{\mathcal{M}(r)}$, and the cardinality $|\mathcal{M}(r)|$ is the *arity* or *fold* of R_r. Each element in relation R_r is called an *instance* of R_r. For any instance $t \in R_r$, will use $r(t)$ to denote the index r of the relation to which t belongs.

Usually a real-world KB is far from complete and many instances in each R_r are expectedly missing. Since for each relation R_r, the KB contains only a subset $\mathcal{T}_r$ of R_r, a typeless KB can be specified by the tuple $(\mathcal{N}, \mathcal{R}, \{\mathcal{T}_r : r \in \mathcal{R}\})$.

Let U be a Euclidean space and the notation $\circ$ denote function composition. The problem of embedding the KB $(\mathcal{N}, \mathcal{R}, \{\mathcal{T}_r : r \in \mathcal{R}\})$ can be formulated as finding 1) an embedding map $\phi : \mathcal{N} \to U$, which maps each entity in x to a vector $\phi(x)$ in U, and 2) for each relation R_r a subset $C_r \subset U^{\mathcal{M}(r)}$ such that $t \in R_r$ implies $\phi \circ t \in C_r$, and $t \in \mathcal{N}^{\mathcal{M}(r)} \setminus R_r$ implies $\phi \circ t \in U^{\mathcal{M}(r)} \setminus C_r$.

Since one does not have access to the entire R_r for each $r \in \mathcal{R}$, a commonly strategy to access the "boundary" of R_r (so as to model that of C_r) is to heuristically construct a set of negative examples $\mathcal{T}_r^-$ for each relation R_r. In below, for any negative example t' of relation R_r, we will use $r(t')$ to denote the index of the relation, with respect to which t' is regarded as a negative example.

The KB embedding problem can then be set up by introducing a non-negative cost function $f_r : U^{\mathcal{M}(r)} \to \mathbb{R}$ to each relation R_r and a positive margin T_{base}, where we force $f_r(\phi \circ t)$ to near zero for every instance $t \in \mathcal{T}_r$ and force $f_r(\phi \circ t)$ to be larger than a prescribed positive margin T_{base}, for every negative example $t \in \mathcal{T}_r^-$. For a given form of function f_r, let θ_r denote the parameter of function f_r, and Θ denote the collection $\{\theta_r : r \in \mathcal{R}\}$ of all θ_r's. Collectively, we denote $\mathcal{T} := \{\mathcal{T}_r : r \in \mathcal{R}\}$, and $\mathcal{T}^- := \{\mathcal{T}_r^- : r \in \mathcal{R}\}$. Then one can define a global cost function $F(\phi, \Theta)$ as

$$F(\phi, \Theta) := \sum_{t \in \mathcal{T}} f_{r(t)}(\phi \circ t) + \sum_{t' \in \mathcal{T}^-} \left[T_{\text{base}} - f_{r(t')}(\phi \circ t')\right]_+ \tag{1}$$

where function $[\cdot]_+$ is defined by $[a]_+ := \max(a, 0)$. An alternative "differential" formulation of the global cost function is also often used in practice: let $\mathcal{P} \subseteq \mathcal{T} \times \mathcal{T}^-$ be a collection of positive-negative example pairs; usually for each $(t, t') \in \mathcal{P}$, t' is made as a negative example for the relation to which t belongs, namely, $r(t) = r(t')$; the global cost function can then be formulated as

$$F(\phi, \Theta) := \sum_{(t,t') \in \mathcal{P}} \left[T_{\text{base}} + f_{r(t)}(\phi \circ t) - f_{r(t')}(\phi \circ t')\right]_+ \tag{2}$$

This cost function, when minimized, forces the the cost of each negative example t' to be at least T_{base} greater than the corresponding positive example t. Within this framework, the typeless KB embedding problem reduces to minimizing the cost function F (either in form (1) or (2)).

Existing Typeless Models

The typeless embedding models can be categorized into *binary models* and *multi-fold models*, where the former apply to binary relational data and the latter apply to multi-fold relational data. All known models presented so far are binary, with the exception of mTransH (Wen et al. 2016). We note that for binary relational data, an instance of a relation R_r is often written as triple (x, r, y), where x and y are two entities involved in the instance.

TransE (Bordes et al. 2013) is arguably the most influential embedding model. The cost function $f_r(\mathbf{x}, \mathbf{y})$ is parametrized by a vector $\mathbf{b}_r \in U$ and is defined by

$$f_r(\mathbf{x}, \mathbf{y}) := \|\mathbf{x} + \mathbf{b}_r - \mathbf{y}\|^2,$$

where $\|\cdot\|$ denotes the L-2 norm.

TransH (Wang et al. 2014) resolves the incapability of TransE in modelling symmetric, many-to-one and many-to-many relations. In TransH (Wang et al. 2014), $f_r(\mathbf{x}, \mathbf{y})$ is parametrized by a hyperplane in U with normal vector $\mathbf{n}_r$ and a vector $\mathbf{b}_r$ in the hyperplane. Let $\mathbb{P}_{\mathbf{n}_r}(\cdot)$ denote the projection operator onto the hyperplane, $f_r(\mathbf{x}, \mathbf{y})$ is defined as

$$f_r(\mathbf{x}, \mathbf{y}) = \|\mathbb{P}_{\mathbf{n}_r}(\mathbf{x}) + \mathbf{b}_r - \mathbb{P}_{\mathbf{n}_r}(\mathbf{y})\|^2.$$

TransR (Lin et al. 2015) parametrizes f_r by the pair $(\mathbf{b}_r, \mathbf{M}_r)$, where $\mathbf{r} \in \mathbb{R}^d$, $\mathbf{M}_r$ is a $d \times k$ matrix, and k is the dimension of U. The cost function f_r is defined as

$$f_r(\mathbf{x}, \mathbf{y}) = \|\mathbf{M}_r \mathbf{x} + \mathbf{b}_r - \mathbf{M}_r \mathbf{y}\|^2.$$

mTransH (Wen et al. 2016) extends the modelling philosophy of TransH to multi-fold relations. In mTransH, each cost function f_r is parametrized by two orthogonal vectors $\mathbf{n}_r$ and $\mathbf{b}_r$ in U and a function $\mathbf{a}_r \in \mathbb{R}^{\mathcal{M}(r)}$. More specifically, the function $f_r : U^{\mathcal{M}(r)} \to \mathbb{R}$ is defined by

$$f_r(\mathbf{t}) := \left\| \sum_{\rho \in \mathcal{M}(r)} \mathbf{a}_r(\rho) \mathbb{P}_{\mathbf{n}_r}(\mathbf{t}(\rho)) + \mathbf{b}_r \right\|^2, \ \mathbf{t} \in U^{\mathcal{M}(r)}.$$

Even for the models outside this framework, such as KG2E (He et al. 2015) and TransG (Xiao, Huang, and Zhu 2016), which are probabilistic models in nature, they still in effect minimize a global cost function F, and can also be adjusted so that the proposed augmentation framework applies.

Except for these models, which are the most relevant to this work, various other models have been proposed. They include, e.g. pTransE (Lin, Liu, and Sun 2015), Holographic embedding (Nickel, Rosasco, and Poggio 2016), Complex Embedding (Trouillon et al. 2016), etc. Most of these models fit in the above typeless embedding framework, and can be augmented by the proposed type-augmentation approach of this paper.

Type Augmentation Scheme

In this section, we first present a scheme which augment a typeless embedding model to a typed model, namely, one that takes into account the type information of the entities in modelling. Given a typeless embedding model (which we will call a *base model*) specified via a cost function F in the form of (1) or (2) (which we will call the *base cost*), our approach is to augment the base embedding model with two other costs: "entity-type cost" and "relation-type cost", where the entity-type cost measures the fitness of entities with types whereas the relation-type cost measures the compatibility between a relation and the types of the entities involved in the relation.

Entity-Type Cost

In a KB, an entity may be annotated with multiple types. For any entity x in a given KB, we will use $L(x)$ to denote the set of all types of x. Let $\mathcal{L} := \bigcup_{x \in \mathcal{N}} L(x)$ denote the set of all types given in the KB. To distinguish the elements $\mathcal{L}$ with a broader notion of type, we will refer to the elements in $\mathcal{L}$ as the *explicit types*.

It is worth noting that the type set $\mathcal{L}$ are usually organized in a hierarchical manner. For example, in a toy KB, the set $\mathcal{L}$ of types may contain sports.tournament, sports.team, sports.league, travel.destination, travel.hotel,

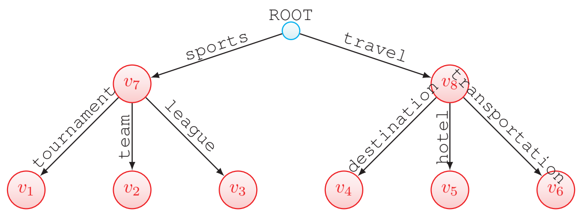


Figure 1: The tree Γ of types in the toy example.

and travel.transportation. Typically each type is expressed as a concatenated sequence of *type tokens*, such as `sports`, `travel`, `hotel`, etc. It is then possible to associate the types in $\mathcal{L}$ with an edge-labeled tree Γ as shown in Figure 1. Each type token serves as an edge label in Γ, and each type in $\mathcal{L}$ can be identified with a node v of Γ or a directed path from the root node (`ROOT`) to v. For example, the type `sports.tournament` can be identified with the node v_1 or with the path from `ROOT` to v_1 in Figure 1.

It is remarkable that we do not insist each type in $\mathcal{L}$ correspond to a *leaf* node in Γ or require entities to be typed at the finest level. For example, node v_7 could be included as a type in $\mathcal{L}$, if there is an entity that has `sports` as one of its types. Such "partial typing" may occur when, for example, the types v_1, v_2 and v_3 in Figure 1 are all inappropriate as a type for some entity related to "sports", and as a consequence, the entity is annotated in the KB to have type `sports`, or node v_7, rather than any of its child nodes. Such partial typing is also expected when the types of an entity are not annotated with sufficient care. This perspective also suggests that every node in $V(\Gamma)$ can be understood as a "type", whether or not it appears in $\mathcal{L}$. Consequently from here after, we will use the term "type" to broadly refer to any node in Γ.

For notation purpose, let $V(\Gamma)$ denote the set of all nodes in the tree Γ and $C(\Gamma)$ denote the set of all edge labels, or type tokens, in Γ. For any two nodes $v, v' \in V(\Gamma)$, if there is a directed path from v' to v, we may write $v \prec v'$. For example in Figure 1, $v_1 \prec v_7 \prec \texttt{ROOT}$.

A key insight of this work is that a node $v \in V(\Gamma)$ can be interpreted as a *constraint* on the entities in $\mathcal{N}$. This understanding is naturally justified since each type token essentially implies a *property* of certain entities, and if a type v of entity x contains a particular token c, then the entity x ought to satisfy the property or constraint associated with c. As every constraint on the entity set $\mathcal{N}$ can be equivalently expressed as a subset of $\mathcal{N}$, we denote the constraint associated with each node $v \in V(\Gamma)$ by a subset $\mathcal{N}_v$ of $\mathcal{N}$. The types (or nodes) of the type tree Γ then correspond to a partial order their associated subsets. More precisely, $\mathcal{N}_{\texttt{ROOT}} = \mathcal{N}$, and for every two nodes $v, v' \in V(\Gamma)$, if $v \prec v'$, then $\mathcal{N}_v \subset \mathcal{N}_{v'}$. For example, in Figure 1, the $\mathcal{N}_{v_1} \subset \mathcal{N}_{v_7}$. Interpreting types as constraints also suggest that if v is a type for entity x, then $x \in \mathcal{N}_v$. It then follows that for any entity $x \in \mathcal{N}$, $x \in \bigcap_{v \in L(x)} \mathcal{N}_v$. That is, each entity lives in the intersection of the constraints associated with its types.

For any node $v \in V(\Gamma)$, subset $\mathcal{N}_v \subset \mathcal{N}$ induces a subset

U_v of the embedding space U. Since we wish the embedding of the KB to preserve the structure of the KB, we require the embedding map ϕ to preserve the partial order "$\subset$" on the set $\{\mathcal{N}_v : v \in V(\Gamma)\}$ of subsets. This requirement immediately implies that $U_{\texttt{ROOT}} = U$ and that for every two nodes $v, v' \in V(\Gamma)$, if $v \prec v'$, then $U_v \subset U_{v'}$. In addition, this requirement implies:

Condition 1 *For any node $v \in V(\Gamma)$, if entity $x \in \mathcal{N}_v$, then $\phi(x) \in U_v$; and if entity $x \notin \mathcal{N}_v$, then $\phi(x) \in U \setminus U_v$.*

To implement this condition, for each entity x, we first construct a set $L^-(x)$ of negative types of x. That is, $L^-(x)$ contains a set of nodes $v \in V(\Gamma)$ for which $x \notin \mathcal{N}_v$. This may be carried out approximately by randomly sampling $V(\Gamma) \setminus L(x)$. We then associate each U_v a non-negative cost function $g_v : U \to \mathbb{R}$ satisfying the following property: For any entity x and any type v, if $v \in L(x)$, $g_v(\phi(x)) = 0$; if $v \in L^-(x)$, $g_v(\phi(x)) > T_{\rm ET}$ for some positive margin $T_{\rm ET}$. For any $v \in V(\Gamma)$ and for any given parametric form of function g_v, let ω_v denote the parameters of function g_v. Collectively, let $\Omega := \{\omega_v : v \in V(\Gamma)\}$ denote the collection of all parameters for functions g_v's. We can then define a global *entity-type cost* function [1] as follows.

$$G(\phi, \Omega) := \sum_{x \in \mathcal{N}} \left(\sum_{v \in L(x)} g_v(\phi(x)) + \sum_{v' \in L^-(x)} [T_{\rm ET} - g_{v'}(\phi(x))]_+ \right) \tag{3}$$

Function g_v: There can be many ways to model U_v and construct g_v accordingly. The simplest approach is to regard U_v as a (affine) subspace of U. More precisely, we propose the following construction.

For each type token $c \in C(\Gamma)$, we introduce a unit-length vector $\mathbf{a}_c \in U$ and a scalar $d_c \in \mathbb{R}$. Denote by U_c the solution space to linear equation

$$\langle \mathbf{a_c}, \mathbf{x} \rangle = d_c, \tag{4}$$

where $\mathbf{x} \in U$ is the unknown. Obviously U_c is an affine subspace in U. For each $v \in V(\Gamma)$, denote by $C(v)$ the set of all type tokens (i.e., all edge labels) on the path from $\texttt{ROOT}$ to v. We then define affine subspace U_v by

$$U_v := \{\mathbf{x} \in U : \langle \mathbf{a_c}, \mathbf{x} \rangle = d_c, \forall c \in C(v)\}. \tag{5}$$

It can then be verified that the affine spaces $\{U_v : v \in V(\Gamma)\}$ preserves the partial order on $\{N_v : v \in V(\Gamma)\}$.

Finally, for each $v \in V(\Gamma)$, define cost function $g_v : U \to \mathbb{R}$ by

$$g_v(\mathbf{x}) := \sum_{c \in C(v)} \|\langle \mathbf{a_c}, \mathbf{x} \rangle - d_c\|^2, \tag{6}$$

Note that for two different nodes v and v', g_v and $g_{v'}$ may share parameters, if they have common ancestors in Γ. It is easy to verify that $\mathbf{x} \in U_v$ if and only if $g_v(\mathbf{x}) = 0$. This completes the type-cost model, where the set Ω of parameters is $\{(\mathbf{a}_c, d_c) : c \in C(\Gamma)\}$.

[1]The entity-type cost function can be alternatively defined in a differential form similar to the way (1) is modified to (2).

Relation-Type Cost

The consideration so far has not made use of the fact that a given relation in fact only relates entities of certain specific types. We now consider this part of the modelling.

A function $\delta : \mathcal{M}(r) \to \mathcal{L}$ is said to be a r-compatible type configuration if there exists an instance $t \in R_r$ such that $\delta(\rho) \in L(t(\rho))$ for every $\rho \in \mathcal{M}(r)$. Let $\Delta(r)$ denote the set of all r-compatible type configurations. That is, $\Delta(r)$ is the set of all possible combination of types on the roles of R_r. Although in most KBs, it is observed that $\Delta(r)$ contains only one element, we do not make such an assumption and allow $\Delta(r)$ to contain an arbitrary number of elements. The definition of $\Delta(r)$ implies that a function $t : \mathcal{M}(r) \to \mathcal{N}$ can not be an instance of R_r if for every $\delta \in \Delta(r)$, there is some $\rho \in \mathcal{M}(r)$ with $\delta(\rho) \notin L(t(\rho))$. For any $\rho \in \mathcal{M}(r)$, let $\Delta(r; \rho) := \{\delta(\rho) : \delta \in \Delta(r)\}$. Then for any instance $t \in R_r$ and any role $\rho \in \mathcal{M}(r)$, $\phi(t(\rho)) \in \bigcup_{v \in \Delta(r;\rho)} U_v$. Thus the following condition is justified.

Condition 2 *For any function $\mathbf{t}' : \mathcal{M}(r) \to U$, if for some $\rho \in \mathcal{M}(r)$, $\mathbf{t}'(\rho) \notin \bigcup_{v \in \Delta(r;\rho)} U_v$, then $\mathbf{t}' \notin C_r$.*

That is, if we can take a positive instance t of R_r, obtain its embedding $\mathbf{t} := \phi(t)$, and replace the embedding vector for any entity, say $t(\rho)$, of t by a vector $\mathbf{x}'$ outside $\bigcup_{v \in \Delta(r;\rho)} U_v$, the resulting $\mathbf{t}'$ should be outside C_r and therefore have a large cost under f_r. We will call such $\mathbf{t}'$ a *mis-typed embedding* of instance t and call $\mathbf{x}'$ a *mistyped embedding* of $t(\rho)$.

We now build this condition into a "relation-type" cost function.

From implementation perspectives, for each relation R_r, we need to create a set $\mathcal{I}_r^\epsilon$ of mis-typed embeddings for a set of instances in R_r and insist that they have large costs under f_r. In particular, if $\mathbf{t}' \in \mathcal{I}_r^\epsilon$ is a mis-typed embedding for an instance $t \in R_r$ and $\mathbf{t}'(\rho)$ is the mis-typed embedding for entity $t(\rho)$, then the distance (say, L2) between $\mathbf{t}'(\rho)$ and U_v must be made at least ϵ for every $v \in \Delta(r; \rho)$. Here the distance between a point $\mathbf{x}$ and a set $S \subseteq U$ is the minimum distance between $\mathbf{x}$ and every point in S. If such $\mathcal{I}_r^\epsilon$ can be constructed, we will require that for each $\mathbf{t}' \in \mathcal{I}_r^\epsilon$, $f_r(\mathbf{t}') > T_{\rm ET}$. This gives rise to a *relation-type* cost function

$$h_r(\theta_r) := \sum_{\mathbf{t}' \in \mathcal{I}_r^\epsilon} [T_{\rm ET} - f_r(\mathbf{t}')]_+$$

and the overall relation-type cost function is then defined as

$$H(\Theta) = \sum_{r \in \mathcal{R}} h_r(\theta_r). \tag{7}$$

Set $\mathcal{I}_r^\epsilon$: We now propose a construction of the set $\mathcal{I}_r^\epsilon$ to complete the definition of h_r. For each $t \in \mathcal{T}_r$ and a role $\rho \in \mathcal{M}(r)$, we need to create a mis-typed embedding $\mathbf{t}'$ for $\phi(t)$ by modifying $\phi(t(\rho))$ to a random mis-typed embedding $\mathbf{x}'$ for $t(\rho)$ and use such a collection of $\mathbf{t}'$ to form the set $\mathcal{I}_r^\epsilon$. The random vector $\mathbf{x}'$ needs to be at least distance ϵ away to the affine space U_δ, for every $\delta \in \Delta(r; \rho)$. We now explain a procedure to create such an $\mathbf{x}'$. Pick a random point $\mathbf{y} \in U$; pick a random $\delta \in \Delta(r; \rho)$; project $\mathbf{y}$ onto the affine space U_δ and find its projection $\mathbf{z}$ on U_δ; move $\mathbf{y}$ along the line connecting $\mathbf{y}$ and $\mathbf{z}$ to a new point $\mathbf{y}'$ that is distance ϵ from $\mathbf{z}$; for all $\delta' \in \Delta(r;\rho) \setminus \{\delta\}$, check if the distance between $\mathbf{y}'$ and $U_{\delta'}$ is larger than ϵ; if this is the case, accept $\mathbf{y}'$ as $\mathbf{x}'$, otherwise repeat the process. Since the size of $\Delta(r;\rho)$ is usually small, this procedure can produce the desired $\mathbf{x}'$ efficiently.

Overall Cost Function and Type-Augmentation Scheme

At this point, we have completely specified G and H, we can then integrate them with the base cost F and complete the model. For binary relational data, when the base cost F is chosen as that for TransE (resp. TransH), the typed model is referred to as TransE-T (resp. TransH-T). For multi-fold relational data, when the base cost F is chosen as that for mTransH, the typed model is referred to as mTransH-T.

In this type augmentation scheme, the overall cost function is defined as a weighted sum of the base cost F, the entity-type cost G, and the relation-type cost H.

$$J(\phi, \Theta, \Omega) := F(\phi, \Theta) + \lambda_1 G(\phi, \Omega) + \lambda_2 H(\Theta), \qquad (8)$$

for some weighting factors $\lambda_1, \lambda_2 > 0$.

The KB embedding problem can then be formulated as minimizing J over its parameters (ϕ, Θ, Ω). The summation form in the definition of each component cost function in (8) allows such a problem to be solved efficiently using stochastic gradient descent (SGD).

Other Embedding Models Exploiting Type Information

Typed-Embodied Knowledge Representation Learning Model (Xie, Liu, and Sun 2016), or TKRL, proposes two typed embedding models. It extends TransR (Lin et al. 2015) to incorporate the type information. At the high level, it defines cost function in the style of TransR:

$$f_r(\mathbf{x}, \mathbf{y}) = \|\mathbf{M}_{v_x}\mathbf{x} + \mathbf{r} - \mathbf{M}_{v_y}\mathbf{y}\|^2,$$

parametrized by two matrices $\mathbf{M}_{v_x}$ and $\mathbf{M}_{v_y}$. The matrix $\mathbf{M}_{v_x}$ depends on the type v_x of entity x and likewise $\mathbf{M}_{v_y}$ depends on the type v_y of entity y. The two matrices are then parametrized based on two kinds of type information: 1) the set of types $L(x)$ of any entity x in a given KB; 2) additional relation-specific type constraints. Two different parameterizations are provided in TKRL: one assumes a product form (RHE) and the other assumes a summation form (WHE). Since TransR has large parameter space and is prone to overfitting, the TKRL models usually require pretraining by TransE.

While TKRL performs superior to the typeless models, it is a specific modification of TransR, which does not generalize to other typeless models. Its high training complexity also makes it unfavourable in some cases.

There are several other models, such as those in (Krompa, Baier, and Tresp 2015), (Chang et al. 2014) and (Wang, Wang, and Guo 2015), that have also exploited type information for embedding-based link prediction. These works however should not be regarded as "typed" embedding models, since the type information is not coded into the embedding vectors.

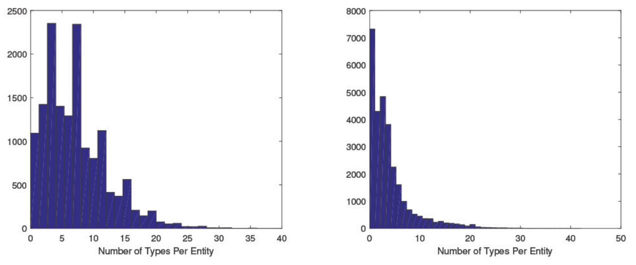


Figure 2: The histograms of the number of types per entity in FB15K (left) and JF17K (right).

Experiments

We conduct experiments to evaluate the proposed typed models (TransE-T and TransH-T on binary data and mTransH-T on multi-fold data) and their corresponding typeless models.

Datasets

Three datasets FB15K, FB15K*, and JF17K are used in the experiments. Both FB15K (Bordes et al. 2013) and JF17K (Wen et al. 2016) contain filtered data obtained from Freebase. FB15K represents data as triples or instances of binary relations, whereas JF17K preserves the original multi-fold relational representation in Freebase. The type annotation of FB15K adopts that used in (Xie, Liu, and Sun 2016). The type annotation of JF17K is obtained by retrieving its type information in Freebase. The types that are obviously ad hoc or non-indicative (such as those under domains `common`, `base`, and `user`) are removed.

In FB15K, for most triples (x, r, y), there is a triple (y, r', x) where relations r and r' are reciprocal of each other. For example, the following two triples are example of a reciprocal pair.

$$(\text{TerreHaute}, \text{location.location.containedby}, \text{Indiana})$$

$$(\text{Indiana}, \text{location.location.contains}, \text{TerreHaute})$$

For such a triple pair, we delete one of the two triples. This gives rise to the dataset FB15K*. The purpose of constructing the FB15K* dataset is to investigate to what extent the redundant reciprocal information may benefit embedding and to what extent the type information may help when such information is present or absent.

In all three datasets, the type tree Γ has depth 2. A summary of the datasets are shown in Table 1 and the distribution of the number of types per entity is shown in Figure 2.

Training and Testing

The models are trained by optimizing their respective overall cost functions. Entity-type cost function G takes the form of (3) and base cost F takes the differential form of (2). For the typed models, choosing λ_1 and λ_2 in (8) is equivalently

Table 1: Statistics of datasets

	#entities	#relations	#instances(total/train/test)	#types	#types/entity	#type tokens
FB15K	14951	1345	542213/483142/59071	886	7.35	971
FB15K*	14951	1260	318804/284292/ 34512	886	7.35	971
JF17K	29177	327	102648/77733/24915	989	4.28	1076

realized by choosing appropriate learning rate for the gradient of each cost term. More implementation details of the model can be found in the code on GITHUB[2].

In all the models, SGD initializes all entity embedding vectors and all $\mathbf{b}_r$ vectors to random unit-length vectors. For all typed models, each $\mathbf{a}_c$ vector is also initialized to random unit-length vectors, and each d_c is initialized to 1. For mTransH, each $\mathbf{a}_r(\rho)$ is randomly initialized to a value in the interval $(0, 1)$.

We will use the term "packet" to refer to a minimal unit in which the gradient of the global cost is computed with respect to all parameters. For all the base models, a packet contains one random instance and K random non-instances, where for TransE and TransH, K is set to 1, and for mTransH, K is set to be the arity of the instance. The gradient of the base cost is computed based on the cost difference between the instance and each of the K non-instances (noting that (2) is used to define the base cost). For each typed model, each packet is expanded (from its counterpart in its respective base model) to include all types of the entities in the positive instance and, for each such entity, 3 random "negative type tokens". We note that here instead of using negative types, we use "negative type tokens", where a negative type token for an entity x refers to a type token in $C(\Gamma)$ that is not an edge label along any path from $\texttt{ROOT}$ to a node $v \in L(x)$. Gradient of the type cost is computed using the entities in the positive instance, their types and the negative tokens. It is possible to show that using negative tokens has the same effect as using negative types, but has modestly reduced complexity.

In SGD, each mini-batch consists of 1000 packets. Each epoch loops over $\lceil M/1000 \rceil$ batches, where M is the number of instances in the training set. For each model, SGD runs for 1000 epochs. The learning rate in the updates based on the base-cost gradients is set to 0.001, and the learning rate in the updates based on the type-cost gradient is set to 0.0003.

Normalization procedures have been applied in the training of these models. In particular, for TransH and TransE, all entity embedding vectors and $\mathbf{b}_r$ vectors are normalized after every update. In mTransH, for each relation r, vector $\mathbf{b}_r$ and function $\mathbf{a}_r$ are also normalized after each update. From our observations, such heuristic tuning improves the embedding performance significantly, particularly for TransE. For each typed model, the same normalization procedure is applied according to its base model, to assure fair comparison.

The models are evaluated by a link prediction task using the testing set. For each testing instance/triple and for each entity x therein, x is held out. Every entity $x' \in \mathcal{N}$ is used to replace x in the instance, and the cost of this substituted instance is evaluated. For the typed models, the instance's cost is defined as the unweighted sum of the base cost $f_r(\phi \circ t)$ and the entity-type cost $\sum_{\rho \in \mathcal{M}(r)} \sum_{v \in \Delta(r;\rho)} g_v(t(\rho))$ of the substituted instance t. The relation-type cost is left out for evaluation since the h_r function are only concerned with the constructed mis-typed embeddings. The costs across all x' are ranked, and the rank for $x' = x$ is recorded. Hit@10 percentage value (HIT) and the mean rank (RANK) metrics proposed in (Bordes et al. 2013) are used as evaluation metrics.

Hit@K.

Hit@K is defined as the proportion of the observed triples or instances that are ranked in the top K position for each triple substitution or instance substitution. In this study, we choose K as 10, and refer to Hit@10 simply as HIT.

Mean Rank.

Mean Rank is the average rank position of the query entity, among all substituting entity, in a testing instance. From here after, we refer to this metric as RANK.

Results and Discussions

Link Prediction Performance on Binary Relational Data

Seen in Table 2, the two proposed typed models present a small performance advantage over the corresponding typeless models (1-2% in HIT and up to 15 in RANK, depending on the dimension of the embedding space). We estimate that this is due to the wide co-existence of reciprocal instances in FB15K. In particular, if a reciprocal instance of test triple exists in the training set, the typeless models are likely to predict the test instance correctly and the type information is less useful for further improvement.

Comparing TKRL models with our typed models, we see that at DIM=100 and 200, TKRL win slightly in HIT whereas our models win slightly in RANK. At DIM=300, the winners for both HIT and RANK are our models. We note that TKRL-RHE fails at high dimensions. We believe that this is due to the multiplicative form of the used matrix, which brings in high instability into the training process. In comparison, TKRL-WHE is less insensitive to increasing dimension. Overall, the best performances of our proposed models are comparable to those of TKRL models, where TKRL models win slightly in HIT and lose slightly in RANK. Furthermore, the number of parameters in TKRL is quadratic in DIM (as it is built upon TransR), whereas this number in the proposed scheme (applied on TransH/mTransH) is linear in DIM. This makes the training time of TKRL significantly longer than the proposed scheme.

On FB15K*(Table 3), where the reciprocal instances are

[2]https://github.com/kongfansh/Embedding_of_Hierarchically_Typed_KB

Table 2: HIT/RANK performance on FB15K

	DIM=100	DIM=200	DIM=300
TransE	46.25/190.46	48.43/193.76	48.82/193.09
TransH	46.68/189.30	48.52/193.02	49.20/192.34
TKRL-WHE	**51.97**/201.47	**51.58**/232.44	49.32/248.32
TKRL-RHE	51.84/196.09	11.40/4628.70	10.74/4926.71
TransE-T	47.00/**180.38**	49.51/**178.06**	50.17/**178.98**
TransH-T	48.73/186.55	50.34/191.49	**50.85**/198.20

Table 3: HIT/RANK performance on FB15K*.

	DIM=100	DIM=200	DIM=300
TransE	34.07/355.80	34.89/357.67	34.97/363.29
TransH	34.48/348.19	35.25/346.80	35.50/348.55
TKRL-WHE	36.78/360.94	35.93/399.37	35.28/418.31
TKRL-RHE	36.79/351.87	35.79/380.15	9.52/5824.83
TransE-T	37.35/313.73	38.35/**312.86**	38.73/**315.48**
TransH-T	**38.26/312.98**	**39.36**/315.44	**39.19**/318.46

no long present, the typeless models degrade their performances significantly. In this case, the proposed typed models offer larger performance advantage over the typeless models (3-4% in HIT and 30-50 in RANK). On this dataset, the two TKRL models appear only capable of improving HIT over the typeless models. With respect to the TKRL models, our models offer a significant RANK advantage (about 40-100). In addition, we notice that, for experiment implementation, TKRL is initialized with embeddings pre-trained by TransE model. However, in our models, we just randomly initialize the embeddings.

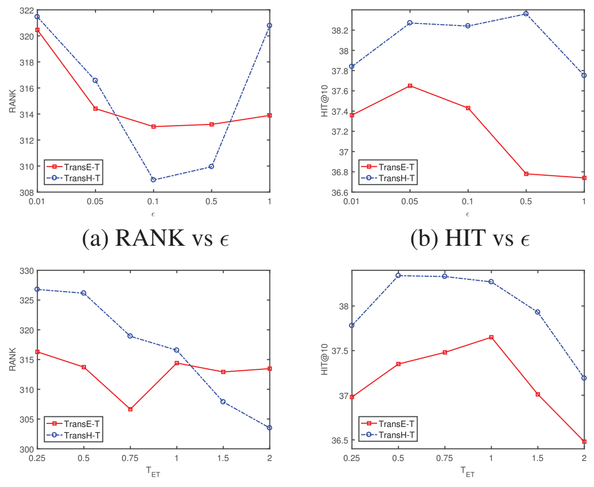
In the typed models, the hyperparameters T_{ET} and ϵ control the strength of the model constraints. Stronger constraints (larger T_{ET} or smaller ϵ) make the model reduce its capacity, which allows better generalization. However, its reduced solution space potentially excludes better parameter configurations and make the model harder to train. Weaker constraints gives the model higher capacity. But it may suffer from being trapped in local minima or overfitting. Thus the choice of these hyper-parameters may affect the performance of the model, as is observed in Figure 3. Built on different base models, in Figure 3, we see that TransH-T and TransE-T appear to have different "sweet spots" in the setting of $(T_{\mathrm{ET}}, \epsilon)$. This difference is presumably related to the mechanism by which the proposed augmentation scheme interact with the base model. Precise characterization of this mechanism seems a difficult open problem.

Link Prediction Performance on Multi-fold Relational Data

On JF17K (Table 4), we also see a similar performance gain in mTransH-T over the typeless mTransH model (HIT improvement by 4% and RANK improvement by 70-80). The performance trends of the TKRL models observed on FB15K* are similar to those observed in FB15K. Our typeless models behave steadily as DIM increases, trending slightly up in HIT and slightly down in RANK. On FB15K* and JF17K, our proposed typed models are winners for every examined value of DIM.

Table 4: HIT/RANK performance on JF17K

	DIM=100	DIM=200	DIM=300
mTransH	45.12/236.85	46.39/231.45	47.41/223.28
mTransH-T	**49.46/163.08**	**50.52/152.97**	**50.99/150.42**



(a) RANK vs ϵ (b) HIT vs ϵ

(c) RANK vs T_{ET} (d) HIT vs T_{ET}

Figure 3: Performance of typed models as functions of ϵ and T_{ET} on FB15K*.

Conclusion

In this paper, we recognize that a type can be naturally modelled as a constraint in the embedding space and that hierarchical types correspond to a partial order of their corresponding constraints. Based on this understanding, we propose a model-augmentation scheme that turns a typeless model into a typed one so as to explore information contained in the type labels of the entities. Via an experimental study, we investigated the performance of this scheme. Our experiments suggest that on FB15K a TKRL model wins at low dimensions and our proposed typed modes win at high dimensions. When the reciprocal instances are removed from FB15K, the proposed typed models win in all cases. Both TKRL models and our proposed typed models outperform the typeless models, confirming the usefulness of type information for embedding as was observed in (Lin et al. 2015). It is however remarkable that the TKRL models tend to degrade their performance or even fail as the embedding dimension increases. This is however not the case for the proposed models. Moreover, the complexity of the proposed models scales linearly with the embedding dimension, contrasting the quadratic scaling in TKRL. Finally the generality of the proposed type-augmentation framework allows one to plug in any base embedding cost F and any sensible entity-type cost G and relation-type cost H. This is verified by our experiments testing mTransH and its augmented version mTransH-T on the JF17K dataset.

Acknowledgments

This work is supported partly by China 973 program (No. 2014CB340305,2015CB358700), by the National Natural Science Foundation of China (No. 61772059, 61421003), by the Project of the State Key Laboratory of Software Development Environment of China (SKLSDE-2016ZX-09) and by the Beijing Advanced Innovation Center for Big Data and Brain Computing.

References

Angeli, G., and Manning, C. D. 2013. Philosophers are mortal: Inferring the truth of unseen facts. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, CoNLL'13, 133–142.

Bordes, A.; Weston, J.; Collobert, R.; and Bengio, Y. 2011. Learning structured embeddings of knowledge bases. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*, AAAI'11, 301–306.

Bordes, A.; Usunier, N.; Garcia-Duran, A.; Weston, J.; and Yakhnenko, O. 2013. Translating embeddings for modeling multi-relational data. In *Proceedings of the International Conference on Neural Information Processing Systems*, NIPS'13, 2787–2795.

Bordes, A.; Glorot, X.; Weston, J.; and Bengio, Y. 2014. A semantic matching energy function for learning with multi-relational data. *Machine Learning* 94(2):233–259.

Chang, K.; Yih, W.; Yang, B.; and Meek, C. 2014. Typed tensor decomposition of knowledge bases for relation extraction. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, EMNLP'14, 1568–1579.

He, S.; Liu, K.; Ji, G.; and Zhao, J. 2015. Learning to represent knowledge graphs with gaussian embedding. In *Proceedings of the Twenty-Fourth ACM International on Conference on Information and Knowledge Management*, CIKM'15, 623–632.

Krompa, D.; Baier, S.; and Tresp, V. 2015. Type-constrained representation learning in knowledge graphs. In *Proceedings of the 14th International Conference on The Semantic Web*, number 8 in ISWC'15, 640–655.

Lin, Y.; Liu, Z.; Sun, M.; Liu, Y.; and Zhu, X. 2015. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, 2181–2187.

Lin, Y.; Liu, Z.; and Sun, M. 2015. Modeling relation paths for representation learning of knowledge bases. 705–714.

Marin, A.; Holenstein, R.; Sarikaya, R.; and Ostendorf, M. 2014. Learning phrase patterns for text classification using a knowledge graph and unlabeled data. In *Proceedings of the Fifteenth Annual Conference of the International Speech Communication Association*, INTERSPEECH'14, 253–257.

Nickel, M.; Rosasco, L.; and Poggio, T. A. 2016. Holographic embeddings of knowledge graphs. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, 1955–1961.

Niu, F.; Zhang, C.; Ré, C.; and Shavlik, J. 2012. Elementary: Large-scale knowledge-base construction via machine learning and statistical inference. *International Journal on Semantic Web and Information Systems (IJSWIS)* 8(3):42–73.

Socher, R.; Chen, D.; Manning, C. D.; and Ng, A. 2013. Reasoning with neural tensor networks for knowledge base completion. In *Proceedings of the International Conference on Neural Information Processing Systems*, NIPS'13, 926–934.

Trouillon, T.; Welbl, J.; Riedel, S.; Gaussier, É.; and Bouchard, G. 2016. Complex embeddings for simple link prediction. In *Proceedings of the Thirty-Third International Conference on Machine Learning*, ICML'16, 2071–2080.

Wang, Z.; Zhang, J.; Feng, J.; and Chen, Z. 2014. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, AAAI'14, 1112–1119.

Wang, Q.; Wang, B.; and Guo, L. 2015. Knowledge base completion using embeddings and rules. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence*, IJCAI'15, 1859–1866.

Wen, J.; Li, J.; Mao, Y.; Chen, S.; and Zhang, R. 2016. On the representation and embedding of knowledge bases beyond binary relations. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, 1300–1307.

Xiao, H.; Huang, M.; and Zhu, X. 2016. TransG : A generative model for knowledge graph embedding. In *Proceedings of the Fifty-Fourth Annual Meeting of the Association for Computational Linguistics*, ACL'16, 2316–2325.

Xie, R.; Liu, Z.; and Sun, M. 2016. Representation learning of knowledge graphs with hierarchical types. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, 2965–2971.

Xiong, C., and Callan, J. 2015a. Esdrank: Connecting query and documents through external semi-structured data. In *Proceedings of the Twenty-Fourth ACM International on Conference on Information and Knowledge Management*, CIKM'15, 951–960.

Xiong, C., and Callan, J. 2015b. Query expansion with Freebase. In *Proceedings of the Fifth ACM International Conference on the Theory of Information Retrieval*, ICTIR'15, 111–120.

Zhang, C.; Ré, C.; Cafarella, M.; De Sa, C.; Ratner, A.; Shin, J.; Wang, F.; and Wu, S. 2017. Deepdive: Declarative knowledge base construction. *Commun. ACM* 60(5):93–102.