Connecting Embeddings for Knowledge Graph Entity Typing

Yu Zhao[1,*], **Anxiang Zhang**[2,*], **Ruobing Xie**[3], **Kang Liu**[4,5], **Xiaojie Wang**[6]
[1]Fintech Innovation Center, School of Economic Information Engineering, Southwestern University of Finance and Economics, Chengdu, China
[2]School of Computer Science, Carnegie Mellon University, Pittsburgh, USA
[3]WeChat Search Application Department, Tencent, Beijing, China
[4]National Laboratory of Pattern Recognition (NLPR), Institute of Automation, Chinese Academy of Sciences, Beijing, 100190, China
[5]University of Chinese Academy of Sciences, Beijing, 100049, China
[6] School of Computer Science, Beijing University of Posts and Telecommunications, Beijing, China

Abstract

Knowledge graph (KG) entity typing aims at inferring possible missing entity type instances in KG, which is a very significant but still under-explored subtask of knowledge graph completion. In this paper, we propose a novel approach for KG entity typing which is trained by jointly utilizing *local typing knowledge* from existing entity type assertions and *global triple knowledge* from KGs. Specifically, we present two distinct knowledge-driven effective mechanisms of entity type inference. Accordingly, we build two novel embedding models to realize the mechanisms. Afterward, a joint model with them is used to infer missing entity type instances, which favors inferences that agree with both entity type instances and triple knowledge in KGs. Experimental results on two real-world datasets (Freebase and YAGO) demonstrate the effectiveness of our proposed mechanisms and models for improving KG entity typing. The source code and data of this paper can be obtained from: `https://github.com/Adam1679/ConnectE`

1 Introduction

The past decade has witnessed great thrive in building web-scale knowledge graphs (KGs), such as Freebase (Bollacker et al., 2008), YAGO (Suchanek et al., 2007), Google Knowledge Graph (Dong et al., 2014), which usually consists of a huge amount of triples in the form of (*head entity*, *relation*, *tail entity*) (denoted $(e, r, \tilde{e})$). KGs usually suffer from *incompleteness* and miss important facts, jeopardizing their usefulness in downstream tasks such as question answering (Elsahar et al., 2018), semantic parsing (Berant et al., 2013), relation classification (Zeng et al., 2014). Hence, the task of

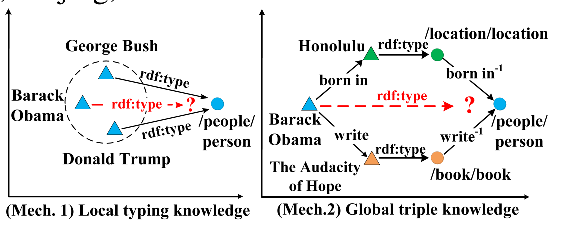

Figure 1: Effective mechanisms of entity type inference with local typing knowledge and global triple knowledge.

knowledge graph completion (KGC, i.e. completing knowledge graph entries) is extremely significant and attracts wide attention.

This paper concentrates on KG entity typing, i.e. inferring missing entity type instances in KGs, which is an important sub-problem of KGC. Entity type instances, each of which is in the formed of (*entity, entity type*) (denoted (e, t)), are essential entries of KGs and widely used in many NLP tasks such as relation extraction (Zhang et al., 2018; Jain et al., 2018), coreference resolution (Hajishirzi et al., 2013), entity linking (Gupta et al., 2017). Most previous works of KGC focus on inferring missing entities and relationships (Bordes et al., 2013; Wang et al., 2014; Lin et al., 2015; Dettmers et al., 2017; Ding et al., 2018; Nathani et al., 2019), paying less attention to entity type prediction. However, KGs also usually suffer from entity types incompleteness. For instance, 10% of entities in *FB15k* (Bordes et al., 2013), which have the */music/artist* type, miss the */people/person* type (Moon et al., 2017). KG entity type incompleteness leads to some type-involved algorithms in KG-driven tasks grossly inefficient or even unavailable.

To solve KG entity type incompleteness issue, in this paper we propose a novel embedding methodology to infer missing entity type instances that employs not only *local typing knowledge* from entity type assertions, as most conventional mod-

[*] Equal Contribution. Corresponding author: Y. Zhao (zhaoyu@swufe.edu.cn).

els do, but also leverages *global triple knowledge* from KGs. Accordingly, we build two distinct knowledge-driven type inference mechanisms with these two kinds of structural knowledge.

Mechanism 1. ***Missing entity types of an entity can be found from other entities that are close to the entity in the embedding space, using local typing knowledge as in Fig.1(Mech.1).***

Mechanism 2. ***Missing entity types of an (head or tail) entity can be inferred from the types of other (tail or head) entities through their relationships, using global triple knowledge as in Fig.1(Mech.2).***

The main idea behind Mech.1 is based on the observation that the learned entities' embeddings by conventional KG embedding methods (Ji et al., 2016; Xie et al., 2016) cluster well according to their types in vector space. For instance, in Fig.1(Mech.1), given an entity *Barack Obama*, it's missing hierarchical type */people/person* can be induced by the given hierarchical type of similar entity *Donald Trump*. In addition, the key motivation behind Mech.2 is that the relationship shall remain unchanged if the entities in a triple fact are replaced with their corresponding hierarchical types. For instance, given a global triple fact (*Barack Obama, born_in, Honolulu*), under this assumption, we can induce a new type triple (*/people/person, born_in, /location/location*)[1]. Formally, $\overrightarrow{\textbf{\textit{Honolulu}}} - \overrightarrow{\textbf{\textit{Barack Obama}}} = \overrightarrow{\textbf{\textit{/location/location}}} - \overrightarrow{\textbf{\textit{/people/person}}}$ (= $\overrightarrow{\textbf{\textit{born\_in}}}$), which can be used to infer missing entity types, e.g. (*Barack Obama, type=?*) via $\overrightarrow{\textit{Barack Obama}} - \overrightarrow{\textit{Honolulu}} + \overrightarrow{\textit{/location/location}} = \overrightarrow{\textit{/people/person}}$, as Mech.2 does. Fig.1 demonstrates a simple illustration of effective mechanisms of entity type inference. Both mechanisms are utilized to build our final composite model.

Specifically, we build two embedding models to realize the two mechanisms respectively. First, considering entities and entity types are completely distinct objects, we build two distinct embedding spaces for them, i.e., **entity space** and **entity type space**. Accordingly, we encode (e, t) entity type instance by projecting the entity from entity space to entity type space with mapping matrix $\mathbf{M}$, hence we have (1): $\boxed{\mathbf{M} \cdot \mathbf{e} \simeq \mathbf{t}}$, called **E2T**. Moreover, we learn the plausibility of $(t_e, r, t_{\tilde{e}})$ global type triple by newly generalizing from $(e, r, \tilde{e})$ global triple fact, even though this type triple is not present originally. Following translating assumption (Bordes et al., 2013), we have (2): $\boxed{\mathbf{t}_{\tilde{e}} - \mathbf{r}^{\circ} \simeq \mathbf{t}_e}$, called **TRT**. E2T and TRT are the implementation models of the two mechanisms. Fig.2 demonstrates a brief illustration of our models. A ranking-based embedding framework is used to train our models. Thereby, entities, entity hierarchical types, and relationships are all embedded into low-dimensional vector spaces, where the composite energy score of both E2T and TRT are computed and utilized to determine the optimal types for (*entity, entity type*=?) incomplete assertions. The experimental results on real-world datasets show that our composite model achieves significant and consistent improvement compared to all baselines in entity type prediction and achieves comparable performance in entity type classification.

Our contributions are as follows:

- We propose a novel framework for inferring missing entity type instances in KGs by connecting entity type instances and global triple information and correspondingly present two effective mechanisms.
- Under these mechanisms, we propose two novel embedding-based models: one for predicting entity types given entities and another one to encode the interactions among entity types and relationships from KGs. A combination of both models are utilized to conduct entity type inference.
- We conduct empirical experiments on two real-world datasets for entity type inference, which demonstrate our model can successfully take into account global triple information to improve KG entity typing.

2 Related Works

Entity typing is valuable for many NLP tasks (Yaghoobzadeh et al., 2018), such as knowledge base population (Zhou et al., 2018), question answering (Elsahar et al., 2018), etc. In recent years, researchers attempt to mine fine-grained entity types (Yogatama et al., 2015; Choi et al., 2018; Xu and Barbosa, 2018; Yuan and Downey, 2018) with external text information, such as web search query logs (Pantel et al., 2012), the textual surface patterns (Yao et al., 2013), context representation (Abhishek et al., 2017), Wikipedia (Zhou et al.,

[1]For more clarity, we represent it as (*/location/location, born_in*$^{-1}$*, /people/person*) in Fig.1(Mech.2).

Table 1: Entity type embedding models.

Models	**Energy function**		**Parameters**	**Sources**	**Training strategy**
	$\mathbf{S}_{e2t}(e,t)$	$\mathbf{S}_{triple}(\cdot)$			
LM (Neelakantan et al., 2015)	$\mathbf{e}^\top \mathbf{t}$	N/A	$\mathbf{e}, \mathbf{t} \in \mathbb{R}^\kappa$	entity type instances	N/A
PEM (Neelakantan et al., 2015)	$\mathbf{e}^\top \mathbf{U} \mathbf{V}^\top \mathbf{t}$	N/A	$\mathbf{e} \in \mathbb{R}^\kappa, \mathbf{t} \in \mathbb{R}^\ell$, $\mathbf{U} \in \mathbb{R}^{\kappa \times d}, \mathbf{V} \in \mathbb{R}^{\ell \times d}$	entity type instance	N/A
RESCAL (Nickel et al., 2011)	N/A	$\mathbf{e}^\top \mathbf{M}_r \tilde{\mathbf{e}}$	$\mathbf{e}, \tilde{\mathbf{e}} \in \mathbb{R}^\kappa, \mathbf{M}_r \in \mathbb{R}^{\kappa \times \kappa}$	mixed triple knowledge	syn.
RESCAL-ET (Moon et al., 2017)	$\Vert\mathbf{e} - \mathbf{t}\Vert_1$	$\mathbf{e}^\top \mathbf{M}_r \tilde{\mathbf{e}}$	$\mathbf{e}, \tilde{\mathbf{e}}, \mathbf{t} \in \mathbb{R}^\kappa, \mathbf{M}_r \in \mathbb{R}^{\kappa \times \kappa}$	entity type inst./ triple know.	asyn.
HOLE (Nickel et al., 2016)	N/A	$\mathbf{r}^\top (\mathbf{e} \star \tilde{\mathbf{e}})$	$\mathbf{e}, \mathbf{r}, \tilde{\mathbf{e}} \in \mathbb{R}^\kappa$	mixed triple knowledge	syn.
HOLE-ET (Moon et al., 2017)	$\Vert\mathbf{e} - \mathbf{t}\Vert_1$	$\mathbf{r}^\top (\mathbf{e} \star \tilde{\mathbf{e}})$	$\mathbf{e}, \mathbf{r}, \tilde{\mathbf{e}}, \mathbf{t} \in \mathbb{R}^\kappa$	entity type inst./ triple know.	asyn.
TransE (Bordes et al., 2013)	N/A	$\Vert\mathbf{e} + \mathbf{r} - \tilde{\mathbf{e}}\Vert$	$\mathbf{e}, \mathbf{r}, \tilde{\mathbf{e}} \in \mathbb{R}^\kappa$	mixed triple knowledge	syn.
TransE-ET (Moon et al., 2017)	$\Vert\mathbf{e} - \mathbf{t}\Vert_1$	$\Vert\mathbf{e} + \mathbf{r} - \tilde{\mathbf{e}}\Vert$	$\mathbf{e}, \mathbf{r}, \tilde{\mathbf{e}}, \mathbf{t} \in \mathbb{R}^\kappa$	entity type inst./ triple know.	asyn.
ETE (Moon et al., 2017)	$\Vert\mathbf{e} - \mathbf{t}\Vert_1$	$\Vert\mathbf{e} + \tilde{\mathbf{e}} + \mathbf{c} - \mathbf{r}\Vert$	$\mathbf{e}, \mathbf{r}, \tilde{\mathbf{e}}, \mathbf{c}, \mathbf{t} \in \mathbb{R}^\kappa$	entity type inst./ triple know.	asyn.
ConnectE (our proposed)	$\Vert\mathbf{M} \cdot \mathbf{e} - \mathbf{t}\Vert_2^2$	$\Vert\mathbf{e} + \mathbf{r}^\star - \tilde{\mathbf{e}}\Vert_2^2$, $\Vert\mathbf{t}_e + \mathbf{r}^\circ - \mathbf{t}_{\tilde{e}}\Vert_2^2$	$\mathbf{e}, \mathbf{r}^\star \in \mathbb{R}^\kappa, \mathbf{t}, \mathbf{r}^\circ \in \mathbb{R}^\ell$, $\mathbf{M} \in \mathbb{R}^{\ell \times \kappa}$	entity type inst./ triple know.	syn.

2018). Despite their success, existing methods rely on additional external sources, which might not be feasible for some KGs.

To be more universal, Neelakantan et al. (2015) propose two embedding models, i.e. linear model (LM) and projection embedding model (PEM), which can infer missing entity types only with KG itself. Although PEM has more expressive power than LM, however, both of them ignore global triple knowledge, which could also be helpful for encoding entity type assertions via shared entities' embeddings. To address this issue, Moon et al. (2017) propose a state-of-the-art model (ETE) to combine triple knowledge and entity type instances for entity type prediction, and build two entity type embedding methodologies: (1) Synchronous training: treat (*entity, entity type*) assertions as special triple facts that have a unique relationship "*rdf:type*", e.g. (*Barack Obama, "rdf:type", person*), and encode all mixed triple facts (original triple data fused with all generated special ones) by conventional entity relation embedding models, such as RESCAL (Nickel et al., 2011), HOLE (Nickel et al., 2016) and TransE (Bordes et al., 2013). (2) Asynchronous training: first learn the entities' embeddings $\mathbf{e}$ by conventional entity relation embedding models mentioned above, and then only update entity types' embeddings $\mathbf{t}$ for $\min \Vert\mathbf{e} - \mathbf{t}\Vert_{\ell 1}$ while keeping $\mathbf{e}$ fixed, called RESCAL-ET, HOLE-ET, TransE-ET and ETE. Although these approaches expect to explore global triple knowledge for entity type prediction, they still lack of expressive ability due to its simplicity of embeddings. In addition, they irrationally assume both the embeddings of entities and entity types being in the same latent space ($\in \mathbb{R}^\kappa$). Since entities and entity types are completely distinct objects, it may not be reasonable to represent them in a common semantic space.

In this paper, we introduce an enhanced KG entity type embedding model with better expressing and reasoning capability considering both local entity typing information and global triple knowledge in KGs. Note that incorporating more external information (Jin et al., 2018; Neelakantan et al., 2015) is not the main focus in this paper, as we only consider the internal structural information in KGs instead, which correspondingly makes our work much more challenging but also more universal and flexible due to the limited information. Recently, (Lv et al., 2018; Hao et al., 2019) also attempt to embedding structural information in KG. However, the goals and models are very different from ours. They encodes the concepts, not hierarchical types. On the contrary, we focus on the latter not the former. Table 1 summarizes the energy functions and other different settings of entity type embedding models.

3 Embedding-based Framework

We consider a KG containing entity type instances of the form $(e, t) \in \mathcal{H}$ ($\mathcal{H}$ is the training set consists of lots of (*entity, entity type*) assertions), where $e \in \mathcal{E}$ ($\mathcal{E}$ is the set of all entities) is an entity in the KG with the type $t \in \mathcal{T}$ ($\mathcal{T}$ is the set of all types). For example, e could be *Barack Obama* and t could be */people/person*. As a single entity can have multiple types, entities in KG often miss some of their types. The aim of this work is to infer missing entity type instances in KGs.

Our work concerns energy-based methods,

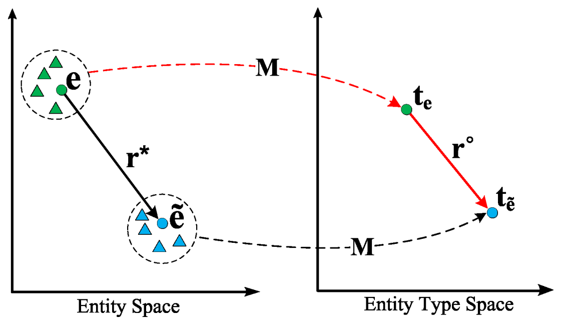

Figure 2: Simple illustration of E2T and TRT.

which learn low-dimensional vector representations (*embeddings*) of atomic symbols (i.e. *entities, entity hierarchical types, relationships*). In this framework, we learn two submodels: (1) one for predicting entity types given entities, and (2) another one to encode the interactions among entity types and relationships from KGs. The joint action of both models in prediction allows us to use the connection between triple knowledge and entity type instances to perform KG entity typing.

3.1 E2T: Mapping Entities to Types

The first model (E2T) of the framework concerns the learning of a function $\mathbf{S}_{e2t}(e,t)$ with local typing knowledge from entity type instances, which is designed to score the similarity of an entity e and a type t. The main ideas behind this model are as follows: (1) Since the learned entity embeddings cluster well when they have the same or similar types, therefore, it is rather intuitive that the entity type embedding represents the projective common concept representation of a cluster of entities, i.e., $f_{proj}(\mathbf{e}) \simeq \mathbf{t}_e, \forall e \in \mathcal{E}$. $\mathbf{e}$ $(\in \mathbb{R}^{\kappa})$ is the embedding of the entity e, $\mathbf{t}_e$ $(\in \mathbb{R}^{\ell})$ is the embedding of the type t_e. The entity type embedding represents common information of their entities, it thus should have fewer variates, i.e., $\ell < \kappa$. (2) Since the entities and entity types are totally distinct objects, we respectively build two embedding space for them, i.e., **entity space** and **entity type space**. (3) Inspired by the previous work TranSparse (Ji et al., 2016) projecting entities from entity space to relation space with operation matrix $\mathbf{M}$, which we adapted, replacing relation space with entity type space, we thus define $f_{proj}(\mathbf{e}) = \mathbf{M} \cdot \mathbf{e}$ $(\simeq \mathbf{t}_e)$. Therefore, this model consists of first projecting entity embedding into entity type space, and then computing a similarity measure between this projection and an entity type embedding. The scoring function of E2T given (e,t) is:

$$\mathbf{S}_{e2t}(e,t) = \|\mathbf{M}\cdot\mathbf{e} - \mathbf{t}\|^2_{\ell 2}\ , \qquad (1)$$

where $\mathbf{M} \in \mathbb{R}^{\ell\times\kappa}$ is a transfer matrix mapping entity embeddings into entity type space. The score is expected to be lower for a golden entity type instance and higher for an incorrect one.

3.2 TRT: Encoding Triples in KGs

Using only entity type instances for training ignores much of relational knowledge that can leverage from triple facts in KGs. In order to connect this relational data with our model, we propose to learn entity type and relationship embeddings from global triple knowledge from KGs. The key motivations behind this model are: (1) As mentioned above, the entities cluster well according to their types. Therefore, we believe that an essential premise of a triple (*head entity, relationship, tail entity*) holds is that its corresponding entity types should first conform to this relationship. Hence, we can build a new **entity type triple** (*head type, relationship, tail type*) by replacing both head entity and tail entity with their corresponding types: i.e. $(e, r, \tilde{e}) \overset{replace}{\longrightarrow} (t_e, r, t_{\tilde{e}})$. $(e, r, \tilde{e}) \in \mathcal{D}$, $\mathcal{D}$ is the training set consists of a lot of triples. $r \in \mathcal{R}$ ($\mathcal{R}$ is the set of relationships). t_e and $t_{\tilde{e}}$ stand for the hierarchical types of left entity e and right entity $\tilde{e}$ respectively. (2) Since the relationship r remains unchanged in replacement, we build two differentiated embeddings for the i-th relationship r_i in two embedding spaces: $\mathbf{r}^{\star}_i (\in \mathbb{R}^{\kappa})$ in entity space and $\mathbf{r}^{\circ}_i$ $(\in \mathbb{R}^{\ell})$ in entity type space. (3) Given entity type triple $(t_e, r, t_{\tilde{e}})$, under translation assumption [2] as in (Bordes et al., 2013), we have: $\mathbf{t}_{\tilde{e}} - \mathbf{r}^{\circ} \simeq \mathbf{t}_e$. Hence, the scoring function is defined as:

$$\mathbf{S}_{trt}(t_e, r, t_{\tilde{e}}) = \|\mathbf{t}_e + \mathbf{r}^{\circ} - \mathbf{t}_{\tilde{e}}\|^2_{\ell 2}\ , \qquad (2)$$

where $\mathbf{t}_e, \mathbf{r}^{\circ}, \mathbf{t}_{\tilde{e}} \in \mathbb{R}^{\ell}$. The model returns a lower score if the two entity types is close under this relationship and a higher one otherwise.

Fig.2 shows an illustration of E2T and TRT.

3.3 Implementation for Entity Type Prediction

Our framework can be used for entity type prediction in the following way. First, for each entity e

[2] We chose TransE in this paper, and it is not difficult for other enhanced translation-based methods to model triple knowledge, such as Trans(H, R, D and G) (Wang et al., 2017).

that appears in the testing set, a prediction by E2T is performed with:

$$\hat{t}_e = \underset{t \in \mathcal{T}}{\arg\min} \ \mathbf{S}_{e2t}(e, t). \tag{3}$$

In addition, a composite score (E2T+TRT) by connecting entity type instances and entity type triples with embedding model, which we call ConnectE [3], is defined as follows:

$$\begin{aligned}\mathbf{S}_{e2t+trt}(e, t_e) = \lambda \cdot \mathbf{S}_{e2t}(e, t_e) + \\ (1-\lambda) \cdot \Big\{ \frac{1}{|P|} \sum_{t_{\tilde{e}} \in P} \mathbf{S}_{trt}(t_e, r, t_{\tilde{e}}) \\ + \frac{1}{|Q|} \sum_{t_{\bar{e}} \in Q} \mathbf{S}_{trt}(t_{\bar{e}}, r, t_e) \Big\},\end{aligned}$$

where λ is a hyperparameter for the trade-off. $P = \{t_{\tilde{e}} | t_{\tilde{e}} \in \mathcal{T}, (e, r, \tilde{e}) \in \mathcal{D}\}$ (i.e. given e is head entity, P is the set of all corresponding tail entities' types.), and $Q = \{t_{\bar{e}} | t_{\bar{e}} \in \mathcal{T}, (\bar{e}, r, e) \in \mathcal{D}\}$ (i.e. given e is tail entity, Q is the set of all corresponding head entities' types.). $|P|$ and $|Q|$ represent the total number of entity types in P and Q respectively. A prediction is performed with:

$$\hat{t}_e = \underset{t_e \in \mathcal{T}}{\arg\min} \ \mathbf{S}_{e2t+trt}(e, t_e). \tag{4}$$

Hence, our final composite model ConnectE-(E2T+TRT) favors predictions that agree with both entity type instances and global triple information in KGs.

3.4 Optimization

We use ranking loss algorithm for training ConnectE-(E2T+TRT), in which the parameter set $\Theta = \{\mathbf{E}, \mathbf{T}, \mathbf{R}^{\star}, \mathbf{R}^{\circ}, \mathbf{M}\}$. $\mathbf{E}, \mathbf{T}$ stand for the collection of all entities' and types' embeddings respectively. $(\mathbf{R}^{\star}, \mathbf{R}^{\circ})$ denotes the collections of relationships' differentiated embeddings. The ranking objectives are designed to assign lower scores to true facts (including $(e, r, \tilde{e})$ triple facts, (e, t) entity type instances and $(t_e, r, t_{\tilde{e}})$ type triples) versus any corrupt ones. We build three sub-objective functions, i.e., $\mathbf{J}_1, \mathbf{J}_2, \mathbf{J}_3$, and implement dynamic optimization strategy, i.e., fix a partial of parameters and only update the rest when minimizing each function. (1) $\mathbf{J}_1$: We choose TransE (see Bordes et al. (2013)) to model triple facts as $\mathbf{S}(e, r, \tilde{e})$, in which we update the embeddings of entities ($\forall \mathbf{e} \in \mathbf{E}$) and the embeddings of relationships ($\forall \mathbf{r}^{\star} \in \mathbf{R}^{\star}$). (2) $\mathbf{J}_2$: We only update the embeddings of entity types ($\forall \mathbf{t} \in \mathbf{T}$) and projecting matrix $\mathbf{M}$, not the entities' embeddings that have been trained in $\mathbf{J}_1$. (3) $\mathbf{J}_3$: We only update the embeddings of relationships ($\forall \mathbf{r}^{\circ} \in \mathbf{R}^{\circ}$) while keeping the entity types' embeddings fixed. The training is performed using Adagrad (Kingma and Ba, 2014). All embeddings in Θ are initialized with uniform distribution. The procedure, from $\mathbf{J}_1$, $\mathbf{J}_2$ to $\mathbf{J}_3$, is iterated for a given number of iterations. We have:

$$\begin{aligned}\mathbf{J}_1 &= \sum_{\mathcal{D}} \sum_{\mathcal{D}'} [\gamma_1 + \mathbf{S}(e, r, \tilde{e}) - \mathbf{S}(e', r, \tilde{e}')]_+ , \\ \mathbf{J}_2 &= \sum_{\mathcal{H}} \sum_{\mathcal{H}'} [\gamma_2 + \mathbf{S}_{e2t}(e, t_e) - \mathbf{S}_{e2t}(e', t'_e)]_+ , \\ \mathbf{J}_3 &= \sum_{\mathcal{Z}} \sum_{\mathcal{Z}'} [\gamma_3 + \mathbf{S}_{trt}(t_e, r, t_{\tilde{e}}) - \mathbf{S}_{trt}(t'_e, r, t'_{\tilde{e}})]_+\end{aligned}$$

$\gamma_1, \gamma_2, \gamma_3 > 0$ are margin hyperparameters, and the corrupted datasets are built as follows:

$$\begin{aligned}\mathcal{D}' :=& \{(e', r, \tilde{e}) | (e, r, \tilde{e}) \in \mathcal{D}, e' \in \mathcal{E}, e' \neq e\} \\ & \cup \{(e, r, \tilde{e}') | (e, r, \tilde{e}) \in \mathcal{D}, \tilde{e}' \in \mathcal{E}, \tilde{e}' \neq \tilde{e}\} , \\ \mathcal{H}' :=& \{(e', t_e) | (e, t_e) \in \mathcal{H}, e' \in \mathcal{E}, e' \neq e\} \\ & \cup \{(e, t'_e) | (e, t_e) \in \mathcal{H}, t'_e \in \mathcal{T}, t'_e \neq t_e\} , \\ \mathcal{Z}' :=& \{(t'_e, r, t_{\tilde{e}}) | (t_e, r, t_{\tilde{e}}) \in \mathcal{Z}, t'_e \in \mathcal{T}, t'_e \neq t_e\} \\ & \cup \{(t_e, r, t'_{\tilde{e}}) | (t_e, r, t_{\tilde{e}}) \in \mathcal{Z}, t'_{\tilde{e}} \in \mathcal{T}, t'_{\tilde{e}} \neq t_{\tilde{e}}\}\end{aligned}$$

$\mathcal{D}, \mathcal{H}$ are training datasets of triple facts and entity type instances in KG. $\mathcal{Z}$ is the training data of type triples, built by replacing entities in $\mathcal{D}$ with their corresponding entity types.

4 Experiments

4.1 Datasets

We conduct the experiments on two real-world datasets ($\mathcal{D}$) widely used in KG embedding literature, i.e. FB15k (Bordes et al., 2013) and YAGO43k (Moon et al., 2017), which are subsets of Freebase (Bollacker et al., 2008) and YAGO (Suchanek et al., 2007) respectively. They consist of triples, each of which is formed as (*left entity, relationship, right entity*). We utilize two entity type data ($\mathcal{H}$, each of it is formed as (*entity, entity type*)) built in (Moon et al., 2017), called FB15kET and YAGO43kET, in which the entity types are mapped to entities from FB15k and YAGO43k respectively.

Moreover, we build new type triple datasets ($\mathcal{Z}$, each one in it is formed as (*head type, relationship, tail type*)), to train our model. They are built based on $\mathcal{D}$ and $\mathcal{H}$. First, for each triple $(e, r, \tilde{e}) \in \mathcal{D}$, we replace the head and the tail with their types

[3]We also call it ConnectE-(E2T+TRT), and use ConnectE-(E2T+0) to denote E2T for uniformity in the experiments.

according to $\mathcal{H}$. The generated datasets are called FB15kTRT(full) and YAGO43kTRT(full). Second, considering about the scalability of the proposed approach for full KGs, we further modify the generation method of type triples, which is the major training bottleneck. We discard newly generated ones with low-frequency (i.e. #frequency = 1). After that the size of both FB15kTRT(full) and YAGO43kTRT(full) decreased by about 90%, called FB15kTRT(disc.) and YAGO43kTRT(disc.) respectively. The statistics of the datasets are showed in Table 2. For saving space, we put more data processing details (include cleaning $\mathcal{H}$, building $\mathcal{Z}$, etc.) on our `github` website.

Table 2: Statistics of $\mathcal{D}, \mathcal{H}, \mathcal{Z}$.

Dataset	**#Ent**	**#Rel**	**#Train**	**#Valid**	**#Test**
FB15k	14,951	1,345	483,142	50,000	59,071
YAGO43k	42,335	37	331,687	29,599	29,593

Dataset	**#Ent**	**#Type**	**#Train**	**#Valid**	**#Test**
FB15kET	14,951	3,851	136,618	15,749	15,780
YAGO43kET	41,723	45,182	375,853	42,739	42,750

Dataset	**#Type**	**#Rel**	**#Train**	**Valid**	**Test**
FB15kTRT(full)	3,851	1,345	2,015,338	–	–
FB15kTRT(disc.)	2,060	614	231,315	–	–
YAGO43kTRT(full)	45,128	37	1,727,708	–	–
YAGO43kTRT(disc.)	17,910	32	189,781	–	–

4.2 Entity Type Prediction

This task concentrates to complete a pair (*entity, entity type*) when its type is missing, which aims to verify the capability of our model for inferring missing entity type instances.

Evaluation Protocol. We focus on entity type prediction determined by Formula (3) and (4). We use ranking criteria for evaluation. Firstly for each test pair, we remove the type and replace it by each of the types in $\mathcal{T}$ in turn. The function value of the negative pairs would be computed by the related models and then sorted by ascending order. We can obtain the exact rank of the correct type in the candidates. Finally, we use two metrics for comparison: (1) the mean reciprocal rank (MRR), and (2) the proportion of correct entities ranked in the top 1/3/10 (HITS@1/3/10)(%). Since the evaluation setting of "Raw" is not as accurate as "Filter" (Bordes et al., 2013), we only report the experimental results with latter setting in this paper.

$$MRR = \frac{1}{|C|} \sum_{i=1}^{|C|} \frac{1}{rank_i} \, ,$$

where C is a set of test pairs, and $rank_i$ is the rank position of the true entity type for the i-th pair.

Implementation. The results of entity type prediction are shown in Table 3, where the results for the baselines are directly taken from original literature (Moon et al., 2017). We do not choose LM and PEM (Neelakantan et al., 2015) as baselines since they do not utilize triple knowledge, thus it is not fair to compare with them. For training our model, we select the learning rate $\alpha \in \{0.1, 0.05, 0.001\}$, the margins $\gamma_1, \gamma_2, \gamma_3 \in \{0.5, 1, 2, 5, 10\}$, the embedding dimension pairs $(\kappa, \ell) \in \{(100, 50), (150, 75), (200, 100), (250, 125)\}$, and the weight $\lambda \in \{0.5, 0.65, 0.85, 0.95\}$. We use negative sampling, and gradient descent with AdaGrad as our optimization approach to improve convergence performance. During the initialization process, each embedding vector of the entities, entity types and relationships is initialized with a random number following a uniform distribution $-\sqrt{6}/(m+n)$, where $n \in \{\#\text{Ent}, \#\text{Type}, \#\text{Rel}\}$ and $m \in \{\kappa, \ell\}$. During the whole training process, we normalize the entity embeddings after each epoch.

We select the parameters based on MRR in valid dataset. The optimal configurations are: $\{\alpha = 0.1, \gamma_1 = \gamma_2 = \gamma_3 = 2, \kappa = 200, \ell = 100, \lambda = 0.85\}$ on FB15k/ET/TRT; $\{\alpha = 0.1, \gamma_1 = \gamma_2 = \gamma_3 = 1, \kappa = 250, \ell = 125, \lambda = 0.85\}$ on YAGO43k/ET/TRT. We run 800 epochs on both datasets, and the batch size is 4096.

Experimental Results. We can see from Table 3 that our ConnectEs outperform all baselines for entity type prediction in terms of all metrics on FB15kET and YAGO43kET. It confirms the capability of ConnectEs in modeling with local typing and global triple knowledge and inferring missing entity type instances in KGs. The model ConnectE-(E2T+TRT)(full) achieves the highest scores.

Analysis. (1) In E2T, we utilize a mapping matrix M which compresses entity embeddings into type embedding space, considering that entity type embedding represents common information of all the entities which belong to this type. The type embedding should be in a sharing subspace of entity embeddings. The experimental results of E2T compared with the baselines demonstrate that this assumption would be quite reasonable. (2) In E2T+TRT, we build new type-relation-type data, and then connect them with entity type instances. This approach provides more direct useful information to (weakly) supervise entity type prediction. For example, given a fact that head entity *Barack Obama* belongs to type */people/person*

Table 3: **Entity type prediction results.** Evaluation of different models on FB15kET and YAGO43kET.

DATASET	FB15kET				YAGO43kET			
METRICS	**MRR**	**HITS@1**	**HITS@3**	**HITS@10**	**MRR**	**HITS@1**	**HITS@3**	**HITS@10**
RESCAL (Nickel et al., 2011)	0.19	9.71	19.58	37.58	0.08	4.24	8.31	15.31
RES.-ET (Moon et al., 2017)	0.24	12.17	27.92	50.72	0.09	4.32	9.62	19.40
HOLE (Nickel et al., 2016)	0.22	13.29	23.35	38.16	0.16	9.02	17.28	29.25
HOLE-ET (Moon et al., 2017)	0.42	29.40	48.04	66.73	0.18	10.28	20.13	34.90
TransE (Bordes et al., 2013)	0.45	31.51	51.45	73.93	0.21	12.63	23.24	38.93
TransE-ET (Moon et al., 2017)	0.46	33.56	52.96	71.16	0.18	9.19	19.41	35.58
ETE (Moon et al., 2017)	0.50	38.51	55.33	71.93	0.23	13.73	26.28	42.18
ConnectE-(E2T+0)	0.57 +- .00	45.54 +- .28	62.31 +- .29	78.12 +- .12	0.24 +- .01	13.54 +- .12	26.20 +- .18	44.51 +- .09
ConnectE-(E2T+TRT)(disc.)	0.59 +- .01	48.54 +- .71	63.66 +- .39	78.27 +- .16	0.27 +- .01	15.1 +- .15	29.14 +- .13	47.08 +- .09
ConnectE-(E2T+TRT)(full)	**0.59 +- .00**	**49.55 +- .62**	**64.32 +- .37**	**79.92 +- .14**	**0.28 +- .01**	**16.01 +- .12**	**30.85 +- .13**	**47.92 +- .07**

and the relationship *born_in*, we could make the best guess of the type of tail entity *Honolulu* as */location/location*. Hence, the addition of type triples in ConnectE-(E2T+TRT) provides superior performance than ConnectE-(E2T+0). (3) Concerning about the scalability of our approach for big KGs, we utilize FB15kTRT(disc.) and YAGO43kTRT(disc.) for prediction, the training time of which reduced by 90% as the training data size decreased by 90%. Moreover, the results of ConnectE-(E2T+TRT)(disc.) show that it's comparable with the best ConnectE-(E2T+TRT)(full).

4.3 Entity Type Classification

This task aims to judge whether each entity type instance in testing data holds or not, which could be viewed as a binary classification problem.

Evaluation Protocol. Since there are no explicit negative entity type instances in existing KGs, in order to create datasets for classification, we build negative facts by randomly switching type from entity type pairs in validation and testing set with equal number of positive and negative examples. Inspired by the evaluation metric of triple classification in (Socher et al., 2013), we calculate the scores of all entity type instances based on model energy function, and rank all instances in testing set with these scores. Those instances with lower scores are considered to be true. We use **precision/recall curves** to show the performances of all models. Moreover, we also compare the **accuracy** among different models. We first use validate set to find best threshold η. For instance, if the model score $\mathbf{S}_{e2t+trt}(e, t_e) \leq \eta$ in classification, the entity type instance will be classified to be positive, otherwise to be negative. The final accuracy is based on how many facts are classified correctly.

Implementation. We utilize the source codes and parameter settings of several baselines provided by (Moon et al., 2017) for this task. The optimal parameter settings for our proposed models are: $\{\alpha = 0.1, \gamma_1 = \gamma_2 = \gamma_3 = 2, \kappa = 200, \ell = 100, \lambda = 0.85\}$ on FB15kET; $\{\alpha = 0.1, \gamma_1 = \gamma_2 = \gamma_3 = 1, \kappa = 250, \ell = 125, \lambda = 0.85\}$ on YAGO43kET. In both datasets, we learn all the training data for 800 epochs and the batch size is 4096. After training, we firstly draw PR-curves with dynamic thresholds. We select the best threshold based on the accuracy in valid dataset, which is used to calculate the accuracy in test dataset.

Experimental Results. We draw the PR-curves for type classification task on both datasets in Fig.3. Note that we only report the results of ConnectE-(E2T+TRT)(disc.) not ConnectE-(E2T+TRT)(full), since the learning speed of the former is much more faster than the latter and its results are close to the best results of the latter. We can see from Fig.3 that when the recall rate is between 0.88 $\sim$ 0.97, ConnectE-(E2T+TRT)(disc.) model could achieve the highest precision rate on FB15kET. In other ranges, our ConnectE-(E2T+TRT)(disc.) model also shows comparable performance. The result is consistent on YAGO43kET. Specifically, ConnectE-(E2T+TRT)(disc.) achieves the best F1 score of 94.66% when recall = 94.27% and precision = 95.05% on FB15kET. Also, ConnectE-(E2T+TRT)(disc.) surpasses other models and gets F1 score of 92.13% when precision = 93.18% and recall = 91.11% on YAGO43kET. It confirms the capability of our model, for they could not only infer missing types in KGs, but also perform well in KG entity type classification.

Table 4 demonstrates the evaluation accuracy results of entity type classification, from which we can observe that: (1) On FB15kET, ConnectE-(E2T+TRT)(disc.) achieves the best accuracy score (94.49%). Compared to the mostly related model ETE, our model shows 0.48% absolute performance improvement. On YAGO43kET, ConnectE-(E2T+TRT)(disc.) model outperforms other models as well. The improvement of our model com-

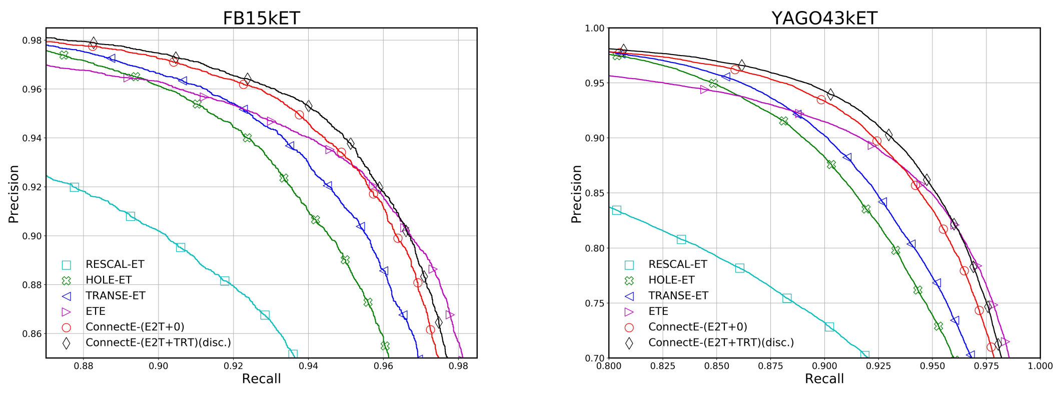

Figure 3: **Entity type classification results (Precision/Recall Curve).** Evaluate on FB15kET, YAGO43kET.

pared to ETE is almost 1.51%. (2) Comparing to the improvement on YAGO43kET, the advantage ConnectE-(E2T+TRT)(disc.) has over ConnectE-(E2T+0) in this task on FB15kET seems to be insignificant, which indicates that the type triples in FB15kTRT have fewer contribution on entity type classification than ones in YAGO43kTRT. It may be partially caused by the fact that the number of relations in YAGO43k (#Rel=37) is far less than that in FB15k (#Rel=1,345), which could considerably influence the effectiveness of the type-relation-type training set. Due to the rareness of relationships in YAGO43k, each entity usually connects with a large number of other entities through one single relationships, which means that the magnitude of $|P|$ and $|Q|$ in the composite model scoring function are large. After averaging in ConnectE-(E2T+TRT)(disc.), it could achieve more stable and significant results on YAGO43kET.

Table 4: **Entity type classification results (accuracy).**

Dataset	**FB15kET**	**YAGO43kET**
RESCAL-ET	90.02%	82.28%
HOLE-ET	93.23%	90.14%
TransE-ET	93.88%	90.76%
ETE	94.01%	90.82%
ConnectE (E2T+0)	94.45%	91.78%
ConnectE (E2T+TRT)(disc.)	**94.49%**	**92.33%**

4.4 Case Study

Table 5 shows the examples of entity type prediction by our model from *FB15k/ET/TRT*, which demonstrate our motivation of Mech. 2 that head type and tail type really maintain the relationship between head entity and tail entity. Given entity *Peter Berg*, TRT can find HITS@1 type prediction */people/person* for it via the existing entity type assertion (*New Youk, /location/location*) and the relationship (*/loc./loc./people_born_here*) between them, i.e. $\overrightarrow{Peter\ Berg} - \overrightarrow{New\ York} + \overrightarrow{/location/location} = \overrightarrow{/people/person}$.

Table 5: **Entity type prediction examples.** Extraction from *FB15k/ET/TRT*.

	Type prediction: HIT@1		**Rel**	**Tail type**	
1	**type=? */people/person***		*/location/location/ people_born_here*	***/location/location***	
	head entity	*Peter Berg*		*New York*	tail entity
		Gus Van Sant		*Louisville*	
2	**type=? /americancomedy/movie**		*/film/film/ directed_by*	***/film/director***	
	head entity	*Very Bad Things*		*Peter Berg*	tail entity
		Rush Hour		*Brett Ratner*	
3	**type=? */medicine/disease***		*people/cause_of _death/people*	***/people/person***	
	head entity	*Myocardial infarction*		*Dick Clark*	tail entity
		Pancreatic cancer		*John Hurt*	

5 Conclusion and Future Work

In this paper, we described a framework for leveraging global triple knowledge to improve KG entity typing by training not only on (*entity, entity type*) assertions but also using newly generated (*head type, relationship, tail type*) type triples. Specifically, we propose two novel embedding-based models to encode entity type instances and entity type triples respectively. The connection of both models is utilized to infer missing entity type instances. The empirical experiments demonstrate the effectiveness of our proposed model. Our modeling method is general and should apply to other type-oriented tasks. Next, we are considering to use this framework to conduct KG entity type noise detection.

Acknowledgments

The authors would like to thank all anonymous reviewers for their insightful comments. We also want to thank Zhiyuan Liu (Tsinghua University) and Linmei Hu (BUPT) for their useful suggestions and comments on early drafts. This work was supported by the National Natural Science Foundation of China under Grant No.61922085, 61906159, the Sichuan Science and Technology Program under Grant No.2018JY0607, the Fundamental Research Funds for the Central Universities under Grant No.JBK2003008, Fintech Innovation Center, and Financial Intelligence and Financial Engineering Key Laboratory of Sichuan Province.

References

Abhishek Abhishek, Ashish Anand, and Amit Awekar. 2017. Fine-grained entity type classification by jointly learning representations and label embeddings. In *Proceedings of EACL*, page 797807.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proceedings of EMNLP*, pages 1533–1544.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of SIGMOD*, pages 1247–1250.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Proceedings of NeurIPS*, pages 2787–2795.

Eunsol Choi, Omer Levy, Yejin Choi, and Luke S. Zettlemoyer. 2018. Ultra-fine entity typing. In *Proceddings of ACL*.

Tim Dettmers, Minervini Pasquale, Stenetorp Pontus, and Sebastian Riedel. 2017. Convolutional 2d knowledge graph embeddings. In *Proceedings of AAAI*, pages 1811–1818.

Boyang Ding, Quan Wang, Bin Wang, and Li Guo. 2018. Improving knowledge graph embedding using simple constraints. In *Proceedings of ACL*, pages 110–121, Melbourne, Australia.

Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. 2014. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *Proceedings of SIGKDD*, pages 601–610.

Hady Elsahar, Christophe Gravier, and Frederique Laforest. 2018. Zero-shot question generation from knowledge graphs for unseen predicates and entity types. In *Proceedings of NAACL*, pages 218–228.

Nitish Gupta, Sameer Singh, and Dan Roth. 2017. Entity linking via joint encoding of types, descriptions, and context. In *Proceedings of EMNLP*, pages 2681–2690.

Hannaneh Hajishirzi, Leila Zilles, Daniel S. Weld, and Luke Zettlemoyer. 2013. Joint coreference resolution and named-entity linking with multi-pass sieves. In *Proceddings of EMNLP*, pages 289–299.

Junheng Hao, Muhao Chen, Wenchao Yu, Yizhou Sun, , and Wei Wang. 2019. Universal representation learning of knowledge bases by jointly embedding instances and ontological concepts. In *Proceedings of KDD 2019*.

Prachi Jain, Pankaj Kumar, and Soumen Chakrabarti. 2018. Type-sensitive knowledge base inference without explicit type supervision. In *Proceedings of ACL*.

Guoliang Ji, Kang Liu, Shizhu He, and Jun Zhao. 2016. Knowledge graph completion with adaptive sparse transfer matrix. In *Proceddings of AAAI*.

Hailong Jin, Lei Hou, Juanzi Li, and Tiansi Dong. 2018. Attributed and predictive entity embedding for fine-grained entity typing in knowledge bases. In *Proceedings of COLING 2018*, pages 282–292.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. In *arXiv preprint arXiv:1412.6980*.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of AAAI*, pages 2181–2187.

Xin Lv, Lei Hou, Juanzi Li, and Zhiyuan Liu. 2018. Differentiating concepts and instances for knowledge graph embedding. In *Proceedings of EMNLP 2018*, page 19711979.

Chang Moon, Paul Jones, and Nagiza F. Samatova. 2017. Learning entity type embeddings for knowledge graph completion. In *Proceedings of CIKM*, pages 2215–2218.

Deepak Nathani, Jatin Chauhan, Charu Sharma, and Manohar Kaul. 2019. Learning attention-based embeddings for relation prediction in knowledge graphs. In *Proceedings of ACL*.

Arvind Neelakantan, Ming-Wei Chang, and . 2015. Inferring missing entity type instances for knowledge base completion: New dataset and methods. In *Proceedings of NAACL 2012*, page 515525.

Maximilian Nickel, Lorenzo Rosasco, and Tomaso Poggio. 2016. Holographic embeddings of knowledge graphs. In *Proceedings of AAAI*, pages 1955–1961.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. In *Proceedings of ICML*, pages 809–816.

Patrick Pantel, Thomas Lin, and Michael Gamon. 2012. Mining entity types from query logs via user intent modeling. In *Proceedings of ACL*, pages 563–571.

Richard Socher, Danqi Chen, Christopher D. Manning, and Andrew Y. Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *Proceedings of NeurIPS*, pages 926–934.

Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *Proceedings of WWW*, pages 697–706.

Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. 2017. Knowledge graph embedding: A survey of approaches and applications. *TKDE*, 29(12):2724–2743.

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

Ruobing Xie, Zhiyuan Liu, and Maosong Sun. 2016. Representation learning of knowledge graphs with hierarchical types. In *Proceedings of IJCAI*, pages 2965–2971.

Peng Xu and Denilson Barbosa. 2018. Neural fine grained entity type classification with hierarchy aware loss. In *Proceedings of NAACL*.

Yadollah Yaghoobzadeh, Heike Adel, and Hinrich Schtze. 2018. Corpus-level fine-grained entity typing. *Journal of Artificial Intelligence Research*, 61:835–862.

Limin Yao, Sebastian Riedel, , and Andrew McCallum. 2013. Universal schema for entity type prediction. In *Proceedings of the 2013 Workshop on Automated Knowledge Base Construction*, pages 79–84.

Dani Yogatama, Daniel Gillick, and Nevena Lazic. 2015. Embedding methods for fine grained entity type classification. In *Proceedings of ACL*, page 291296.

Zheng Yuan and Doug Downey. 2018. Otyper: A neural architecture for open named entity typing. In *Proceddings of AAAI*.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. In *Proceedings of COLING*, pages 23–29.

Richong Zhang, Fanshuang Kong, Chenyue Wang, and Yongyi Mao. 2018. Embedding of hierarchically typed knowledge bases. In *Proceddings of AAAI*.

Ben Zhou, Daniel Khashabi, Chen-Tse Tsai, and Dan Roth. 2018. Zero-shot open entity typing as type-compatible grounding. In *Procedddings of EMNLP*, pages 2065–2076.