Attributed and Predictive Entity Embedding for Fine-Grained Entity Typing in Knowledge Bases

Hailong Jin and **Lei Hou** and **Juanzi Li**
DCST, Tsinghua University
Beijing 100084, China

Tiansi Dong
B-IT, Bonn University
53113 Bonn, Germany

{jinhl15@mails.,houlei@mail.,lijuanzi@}tsinghua.edu.cn
dongt@bit.uni-bonn.de

Abstract

Fine-grained entity typing aims at identifying the semantic type of an entity in $\mathcal{KB}$. Type information is very important in knowledge bases, but is unfortunately incomplete even in some large knowledge bases. Limitations of existing methods are either ignoring the structure and type information in $\mathcal{KB}$ or requiring large scale annotated corpus. To address these issues, we propose an attributed and predictive entity embedding method, which can fully utilize various kinds of information comprehensively. Extensive experiments on real DBpedia dataset show that our proposed method significantly outperforms 8 state-of-the-art methods, with 3.4% and 2.9% improvement in Mi-F1 and Ma-F1, respectively.

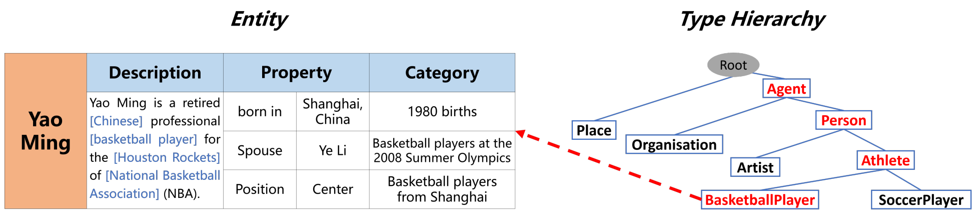
1 Introduction

Knowledge about entities in Knowledge Base ($\mathcal{KB}$) is essential for language understanding. This knowledge can be attributional (e.g., *canFly*, *hasAge*), relational (e.g., *marriedTo*, *bornIn*) or type-based (e.g., *isFood*, *isLocation*). Type information, which clusters a group of entities with same properties, is valuable in $\mathcal{KB}$, because it is the glue that holds our mental world together (Murphy, 2004). Types in $\mathcal{KB}$ are usually organized as a hierarchical structure, namely *type hierarchy*. **Entity typing** assigns types (e.g., *Person*, *Athlete*, *BasketballPlayer*) to an entity (e.g., *Yao Ming*) in $\mathcal{KB}$, and is a fundamental task in knowledge base construction.

Traditional entity typing focuses on a small set of types, such as *Person*, *Location* and *Organization* (Ratinov and Roth, 2009; Nadeau and Sekine, 2007), while **fine-grained entity typing** assigns more specific types to an entity, which normally forms a *type-path* in the type hierarchy in $\mathcal{KB}$ (Ren et al., 2016a). As shown in Fig. 1, *Yao Ming* is associated with a type-path */Thing/Agent/Person/Athlete/BasketballPlayer*. Fine-grained types (e.g., *Athlete* and *BasketballPlayer*) are more informative than coarse-grained types (e.g., *Person*), because they provide more specific semantic information about the entity (Xu et al., 2016). Characterizing an entity with fine-grained types (type-paths) benefits many real applications, such as *knowledge base completion* (Dong et al., 2014), *entity linking* (Ling et al., 2015; Durrett and Klein, 2014), *relation extraction* (Liu et al., 2014), and *question answering* (Yahya et al., 2014).

Many researches have been carried out on *fine-grained entity typing* ***in text***, which detects local types for an entity mention in a particular plain text from predefined fine-grained entity types (Ling and Weld, 2012; Yogatama et al., 2015; Ren et al., 2016a; Shimaoka et al., 2017; Xin et al., 2018). However, only a few works aim at *fine-grained entity typing* ***in knowledge base***, that is, inferring missing types for an entity in $\mathcal{KB}$, which is the focus of this paper. Most $\mathcal{KB}$s are incomplete and lack of type information, in part because the world is always changing. For example, 36.53% entities in DBpedia do not have type information. Therefore, $\mathcal{KB}$s are aways in need of updating and completing. To this end, there are two main methods in the literature as follow: One method exploits various kinds of information to construct the feature representation of an entity, such as entity textual description, property and category (Neelakantan and Chang, 2015; Xu et al., 2016). After that, a predict function $P(t|e)$ is learned to infer

Corresponding author: Lei Hou (houlei@mail.tsinghua.edu.cn)



	Description	Property		Category
Yao Ming	Yao Ming is a retired [Chinese] professional [basketball player] for the [Houston Rockets] of [National Basketball Association] (NBA).	born in	Shanghai, China	1980 births
		Spouse	Ye Li	Basketball players at the 2008 Summer Olympics
		Position	Center	Basketball players from Shanghai

Figure 1: An example of fine-grained entity typing in $\mathcal{KB}$. The left part is an entity with various information. The red ones are assigned types, which forms a type-path.

whether entity *e* is an instance of type *t*. This method ignores rich structural information in $\mathcal{KB}$ (i.e., link relation between entities). It is a promising and challenging problem to utilize structural information for type inference. The second method utilizes **annotated corpus** to learn entity low-dimensional representations then makes type inference based on the learned embeddings (Yaghoobzadeh and Schütze, 2015; Yaghoobzadeh and Schütze, 2017). Apparently, it requires large scale corpus in which entity mentions are annotated. But large scale highly-qualified annotated corpus is often difficult to obtain.

To address the above issues, we propose a novel *attributed and predictive entity embedding* method, which can be described as follow: We first construct a partially-labeled attributed entity network, which contains structural, attribute and type information for entities. Here "partially-labeled" means some entities have been labeled with type information, while others not. Then, we employ a deep neural network to integrate the structural and attribute information for entity representation learning. For an unlabeled entity, we jointly utilize its structural and attribute information for neighbor prediction. For a labeled entity, besides neighbors, we also predict its types and type hierarchy via a margin-based loss function. Finally, a multi-label classification model is used for type inference based on the learned entity embeddings.

The main contributions of this paper can be summarized as follows:

- We encode various kinds of information into a partially-labeled entity network and propose a deep neural network model to effectively integrate the structural and attribute information for entity representation learning.
- We utilize type information of labeled entities to equip the learned embeddings with strong predictive power in entity typing task, and further incorporate the structure of type hierarchy via a margin-based loss function to meet the fine-grained demand.
- We construct a dataset from DBpedia. Extensive experiments show that our proposed method significantly outperforms 8 state-of-the-art methods, with 3.4% and 2.9% improvement in Mi-F1 and Ma-F1, respectively.

The rest of this paper is organized as follows. In Section 2, we formally define the problem of fine-grained entity typing in knowledge bases. Section 3 demonstrates the proposed approach in detail. The dataset description and evaluation are presented in Section 4. Section 5 outlines some related works. Section 6 concludes our work.

2 Problem Formulation

In this section, we formally define the problem of *Fine-Grained Entity Typing* in $\mathcal{KB}$.

Definition 1 ***Knowledge Base*** *is defined a tuple of entities* $\mathcal{E}$ *and a type hierarchy* $\mathcal{H}$*, formally,* $\mathcal{KB} = \langle \mathcal{E}, \mathcal{H} \rangle$*, where* $\mathcal{H}$ *is used to organize entities in* $\mathcal{E}$*. Each entity* $e \in \mathcal{E}$ *denotes a specific thing in the real world.*

Definition 2 ***Type Hierarchy*** $\mathcal{H}$ *is defined as a tree or a directed acyclic graph (DAG). It provides a natural way to categorize and organize entities in large* $\mathcal{KB}$*. It is formalized* $(\mathcal{T}, \mathcal{R})$*, where* $\mathcal{T}$ *is the type set and* $\mathcal{R} = \{(t_i, t_j) | t_i, t_j \in \mathcal{T} \wedge i \neq j\}$ *is the relation set, in which* (t_i, t_j) *means that* t_i *is the subtype of* t_j*.*

We distinguish three kinds of entity knowledge which are beneficial to *fine-grained entity typing* task.

- **Entity Link Relation:** $\mathcal{KB}$ contains a large number of link relations between entities. It is natural to convert entities in $\mathcal{E}$ into a network structure via these link relations. If an entity e_i links to another entity e_j in $\mathcal{KB}$, there will be a directed edge from entity e_i to entity e_j. These link relations reflect the overall structure of $\mathcal{KB}$ and provide important semantic relatedness between entities.

- **Entity Attribute:** Besides the above link structure, each entity $e \in \mathcal{E}$ also has textual description, properties and categories (shown in Fig. 1). We term all such auxiliary information of an entity as attributes. The attributes of an entity provide valuable clues to predict its types.

- **Entity Type:** Part of entities in $\mathcal{E}$ have been assigned with types, we refer them as $\mathcal{E}^l$. The assigned types for an entity form a type-path in type hierarchy $\mathcal{H}$, which starts from the root and ends at one specific type (not necessary to end at the leaf type). Those entities in $\mathcal{E}$ without type information are referred as $\mathcal{E}^u$. The type information observed in $\mathcal{E}^l$ can be a useful signal to infer missing type information in $\mathcal{E}^u$.

From the above definitions, we achieve a **partially-labeled attributed entity network**, including link structure, entity attributes, and type information. We formally define this entity network as follow:

Definition 3 ***Entity Network:*** *Let $\mathcal{G} = (\mathcal{E}, \mathcal{L})$ be an entity network. $\mathcal{E} = \mathcal{E}^l \cup \mathcal{E}^u$ is the entity set, where $|\mathcal{E}| = N$ and $|\mathcal{E}^l| = N^l$. $\mathcal{L} = \{(e_i, e_j) \,|e_i, e_j \in \mathcal{E}$ and $i \neq j\}$ is the link relation set.*

For the convenience of formula deduction, we redefine this entity network in matrix form, $\mathbf{G} = (\mathbf{A}, \mathbf{X}, \mathbf{Y})$. Adjacency matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$ is equivalent to link set $\mathcal{L}$, $\mathbf{A}_{ij} = 1$ if there is a link from e_i to e_j, otherwise 0. Each row in $\mathbf{A}$ refers to an entity's link vector $\mathbf{A}_i$. Attribute matrix $\mathbf{X} \in \mathbb{R}^{N \times M}$ collects attributes for all entities, where *M* is the number of attributes. Each row in $\mathbf{X}$ refers to an entity's attribute vector $\mathbf{X}_i$. $\mathbf{X}$ can either be binary or take any real value. Type matrix $\mathbf{Y} \in \mathbb{R}^{N^l \times K}$ collects type information for entities in $\mathcal{E}^l$, where *K* is the number of types. Each row in $\mathbf{Y}$ refers to the type vector $\mathbf{Y}_i$ of a labeled entity in $\mathcal{E}^l$. $\mathbf{Y}_{ij} = 1$ if e_i has been assigned with type t_j, otherwise 0.

Based on the terminologies described above, we formally define the problem of *fine-grained entity typing in $\mathcal{KB}$* as follows:

Definition 4 ***Fine-grained entity typing in KB:*** *Given a partially-labeled attributed entity network $\mathcal{G} = (\mathcal{E}, \mathcal{L})$, we aim at learning a type predictor from $\mathcal{E}^l$, which comprehensively takes link structure, entity attributes and type information into account. Then we utilize the learned type predictor to predict the type of $e \in \mathcal{E}^u$, i.e., infer whether entity $e \in \mathcal{E}^u$ is an instance of type $t \in \mathcal{T}$.*

3 The Proposed Approach

3.1 General Architecture

A general state-of-the-art architecture for fine-grained entity typing consists of two components.

- **Entity Representation:** It learns distributed representation $\vec{v}(e)$ for each entity $e \in \mathcal{E}$.

- **Type Predictor:** It learns a predict function $P(t|e)$ using $\mathcal{E}^l$ as training examples. $P(t|e)$ denotes the probability that entity *e* belongs to type *t*, and it is often used to infer types of those unlabeled entities in $\mathcal{E}^u$.

Type predictor is regarded as a multi-label classification model. $P(t|e)$ is calculated through a two-layer Multi-Layer Perceptron (MLP).

$$[P(t_1|e) \cdots P(t_K|e)] = \sigma\left(\mathbf{W}_{out} f\left(\mathbf{W}_{in} \vec{v}(e)\right)\right) \tag{1}$$

Each entry of output layer indicates the predicted probability of type t_i for a given entity *e*. $\mathbf{W}_{in}$ and $\mathbf{W}_{out}$ are MLP parameter matrices. σ is the sigmoid function. *f* is the nonlinear activation function, such as *ReLU* and *tanh*.

The loss L for a prediction $\mathbf{y} = [P(t_1|e) \cdots P(t_K|e)]$ when the true labels are encoded in a binary vector $\mathbf{y}^* \in \{0,1\}^{1\times K}$ is the following cross entropy loss function:

$$L(\mathbf{y}, \mathbf{y}^*) = \sum_{i=1}^{K} -\mathbf{y}^{*(i)} \log \mathbf{y}^{(i)} - \left(1 - \mathbf{y}^{*(i)}\right) \log \left(1 - \mathbf{y}^{(i)}\right) \tag{2}$$

The key difficulty in computing $P(t|e)$ is to learn a good representation $\vec{v}(e)$ for entity e. Several models have achieved the state-of-the-art performance. However, these models still face the following non-trivial challenges:

- Entity attributes and link structure are typically regraded as separated features without considering their correlations. Recent research shows the importance of integrating link structure and entity attributes for learning more informative representations (Liao et al., 2017; Li et al., 2017). Could we seamlessly integrate entity attributes and link structure into a unified entity network? Since the link structure and entity attributes offer different sources of information, how shall we capture the complex non-linear relationship between them?

- The type information of the labeled entities in $\mathcal{E}^l$ is only used in the learning phase of the type predictor, but totally ignored in the representation learning phase. Recent research shows incorporating labeled information in representation learning could enhance the predictive power of the learning embeddings for specific tasks (Tang et al., 2015a; Yang et al., 2016). In our task, the entity types and their formed type hierarchy $\mathcal{H}$ are valuable labeled information, could we design a predictive model to make full use of the type information in $\mathcal{E}^l$ and the structure of type hierarchy?

To address the above challenges, we propose a novel attributed and predictive entity embedding method to learn entity representations. It combines the advantages of attributed network embedding method and semi-supervised learning. We integrate entity attributes and link structure via a deep neural network model (Section 3.2). For an entity e_i in $\mathcal{E}^l$, we jointly predict its type and its neighbor entities in the entity network (Section 3.3). In this way, we can fully utilize the type information in $\mathcal{E}^l$.

3.2 Attributed Entity Embedding

We aim at learning low-dimensional vector representations for each entity in the entity network, such that original link structure and entity attribute proximity can be preserved in vectors. The most intuitive idea is to directly concatenate the embeddings learned from different parts. However, this way can not capture the complex non-linear relationship between them, because each part is trained separately without interaction. To address this issue, we present a neural network model to integrate link structure (structural information) and entity attributes (attribute information). For an entity $e_i \in \mathcal{E}$, we jointly utilize its link vector $\mathbf{A}_i$ and attribute vector $\mathbf{X}_i$ to predict the link probability between e_i and other entities (shown in Fig. 2). Our model integrates structure and attribute information on the input layer, so that these two parts closely interact with each other and all parameters can be optimized simultaneously.

In our design, each entity *e* has two vector representations, *self representation* and *context representation*. $\mathbf{e}$ is self representation of *e* when it is treated as an entity in entity network $\mathcal{G}$. While $\tilde{\mathbf{e}}$ is context representation of *e* when it is treated as a specific context. To comprehensively represent an entity *e*, we add $\mathbf{e}$ and $\tilde{\mathbf{e}}$ as the final representation $\vec{v}(e)$.

Input and Embeddings. For an entity $e_i \in \mathcal{E}$, $\mathbf{A}_i$ and $\mathbf{X}_i$ denote its link vector and attribute vector as defined in Section 2. Our model takes both structural information ($\mathbf{A}_i$) and attribute information ($\mathbf{X}_i$) as input, and utilizes two embedding weight matrices to achieve the final input embeddings. $\mathbf{W}_{stru}$ projects the link vector $\mathbf{A}_i$ to a dense vector $\mathbf{u}_{stru}$ which captures structural information. $\mathbf{W}_{attr}$ encodes the attribute vector $\mathbf{X}_i$ and generates a compact vector $\mathbf{u}_{attr}$ which aggregates attribute information. We concatenate $\mathbf{u}_{stru}$ and $\mathbf{u}_{attr}$, then feed this concatenated vector into a Multi-Layer Perceptron.

Information Fusion. We employ a Multi-Layer Perceptron to integrate structural and attribute information. Recent research shows stacked multiple non-linear layers can help learn better representations

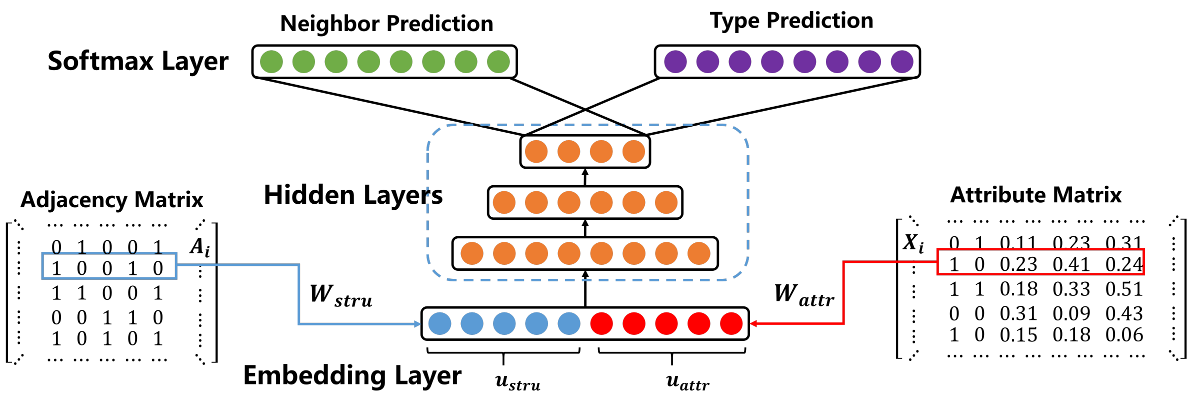


Figure 2: Attributed entity embedding framework

of data (Liao et al., 2017). Each layer is calculated as follows:

$$\mathbf{h}^{(1)} = f\left(\mathbf{W}^{(1)}\left[\mathbf{u}_{stru}, \lambda\mathbf{u}_{attr}\right] + \mathbf{b}^{(1)}\right) \tag{3}$$

$$\mathbf{h}^{(k)} = f\left(\mathbf{W}^{(k)}\mathbf{h}^{(k-1)} + \mathbf{b}^{(k)}\right), k = 1, 2, \cdots, n \tag{4}$$

λ is the trade-off factor for structure and attribute information. f denotes the activation function. n is the number of hidden layers. From the last hidden layer, we obtain an informative representation $\mathbf{h}^{(n)}$, which is regarded as the self representation $\mathbf{e}_i$ of the input entity e_i.

Neighbor Prediction. Finally, we predict neighbors for an input entity e_i (if there is a link from e_i to e_j, e_j is e_i's neighbor). We call it neighbor prediction, which is widely used to model network structure (Tang et al., 2015a; Liao et al., 2017). Specifically, we utilize the self representation $\mathbf{e}_i$ to predict the link probability between entity e_i and other entities. The predictive link probability is defined through a softmax layer as follows:

$$p\left(e_j|e_i\right) = \frac{\exp\left(\tilde{\mathbf{e}}_j \cdot \mathbf{e}_i\right)}{\sum_{j'=1}^{N}\exp\left(\tilde{\mathbf{e}}_{j'} \cdot \mathbf{e}_i\right)} \tag{5}$$

$\tilde{\mathbf{e}}_j$ denotes the context representation of entity e_j. Let $\mathbf{y}_i$ denotes the predicted link probability vector for entity e_i, $\mathbf{y}_i^{(j)}$ is the j-th entry of the vector $\mathbf{y}_i$. $\mathbf{A}_i$ is e_i's link vector (in Section 2), $\mathbf{A}_{ij}$ indicates whether there is a link from e_i to e_j. The loss function L_{attr} is defined as cross entropy over all entities in $\mathcal{E}$

$$L_{attr} = \sum_{i=1}^{N}\sum_{j=1}^{N} -\mathbf{A}_{ij}\log\mathbf{y}_i^{(j)} - \left(1 - \mathbf{A}_{ij}\right)\log\left(1 - \mathbf{y}_i^{(j)}\right) \tag{6}$$

3.3 Predictive Entity Embedding

In this section, we propose predictive entity embedding learning that makes full use of the type information in $\mathcal{E}^l$, which is often ignored by existing methods in representation learning phase. Inspired by (Tang et al., 2015a; Yang et al., 2016), we first introduce a type prediction loss to equip learned embeddings with strong predictive power for entity typing task (Section 3.3.1). Then, to meet the fine-grained demand, we model the tree-structured type hierarchy in a margin-based framework (Section 3.3.2).

3.3.1 Type Prediction

We fully utilize type information in $\mathcal{E}^l$ to make the learned embeddings more predictive. For each entity $e_i \in \mathcal{E}^l$, we not only predict its neighbors in $\mathcal{G}$ (Section 3.2), but also predict its types. We add another softmax layer, the probability of entity e_i being an instance of type t_j is defined as follows:

$$p\left(t_j|e_i\right) = \frac{\exp\left(\mathbf{t}_j \cdot \mathbf{e}_i\right)}{\sum_{j'=1}^{K}\exp\left(\mathbf{t}_{j'} \cdot \mathbf{e}_i\right)} \tag{7}$$

$\mathbf{t}_j$ denotes the representation of type t_j. Let $\mathbf{y}_i$ denotes the predicted type probability vector for entity e_i, $\mathbf{y}_i^{(j)}$ is the *j*-th entry of the vector $\mathbf{y}_i$. $\mathbf{Y}_i$ is e_i's type vector (in Section 2), $\mathbf{Y}_{ij}$ indicates whether e_i is an instance of t_j. The loss function L_{type} is defined as cross entropy over all entities in $\mathcal{E}^l$:

$$L_{type} = \sum_{i=1}^{N^l} \sum_{j=1}^{K} -\mathbf{Y}_{ij} \log \mathbf{y}_i^{(j)} - (1 - \mathbf{Y}_{ij}) \log \left(1 - \mathbf{y}_i^{(j)}\right) \tag{8}$$

3.3.2 Type Hierarchy Modeling

Fine-grained entity typing requires us to pay more attention to specific types (fine-grained types) than to general types (coarse-grained types), because specific types are more informative. For example, *Yao Ming* belongs to *BasketballPlayer* and *Athlete*, and it should get higher score on *BasketballPlayer* than *Athlete*, because the former characterizes *Yao Ming* more accurately. Besides, *Yao Ming* should get higher score on the correct type *BasketballPlayer* than its wrong sibling type *SoccerPlayer* although both types belong to *Athlete*. If different types are treated equally, the structural information between types will be lost. Therefore, we use *type order* to model different relatedness between an entity and its related types, which reflects the structural information of type hierarchy to a certain degree. In particular, we define two kinds of *type order* as follow:

Definition 5 ***Ancestor order:*** *In the type-path of an entity, specific types should get higher score than its ancestor. For example, Yao Ming should get higher score on BasketballPlayer than Athlete.*

Ancestor order reflects the relationship between a type and its ancestor types. For each triple (e, t, t^a), where *t* is one of the type for entity *e*, and t^a is *t*'s ancestor in type hierarchy. We define adaptive margin $\gamma_{tt^a} = \xi \times \frac{l(t^a,t)}{l(root,t)}$, where $l(t^a, t)$ denotes the number of steps going down from type t^a to type t. The hinge loss is defined as:

$$\max\left(0, s\left(e, t^a\right) - s\left(e, t\right) + \gamma_{tt^a}\right) \tag{9}$$

$$L_{ancestor} = \mathbb{E}_{(e,t,t^a)} \left[\max\left(0, s\left(e, t^a\right) - s\left(e, t\right) + \gamma_{tt^a}\right)\right] \tag{10}$$

where $s\left(e, t\right)$ is the score function, defined as the inner product of $\mathbf{e}$ and $\mathbf{t}$.

Definition 6 ***Sibling order:*** *For an entity, the correct type should get higher score than its wrong sibling type in type hierarchy. For example, Yao Ming should get higher score on BasketballPlayer than SoccerPlayer.*

Sibling order reflects the relationship between a type and its sibling types. For each triple (e, t, t^s), where *t* is one of the true type for entity *e*, and t^s is its wrong sibling type of entity *e*. We use the fixed margin ξ. The hinge loss is defined as:

$$\max\left(0, s\left(e, t^s\right) - s\left(e, t\right) + \xi\right) \tag{11}$$

$$L_{sibling} = \mathbb{E}_{(e,t,t^s)} \left[\max\left(0, s\left(e, t^s\right) - s\left(e, t\right) + \xi\right)\right] \tag{12}$$

We achieve the loss function L_{hier} for type hierarchy modeling: $L_{hier} = L_{ancestor} + L_{slibing}$.
Objective and Training. Finally, we sum these three loss function to get the final objective function of entity representation component.

$$Loss = L_{attr} + \lambda_1 \cdot L_{type} + \lambda_2 \cdot L_{hier} \tag{13}$$

λ_1 and λ_2 are weight parameters. We apply negative sampling (Mikolov et al., 2013; Tang et al., 2015b), where only a very small subset of entities are sampled from $\mathcal{E}$. Parameters are optimized through Adaptive Moment Estimation (Adam). In the embedding layer and each hidden layer, we also add dropout component to alleviate over fitting.

4 Experiments and Analysis

In this section, we evaluate the proposed method using real world dataset collected from DBpedia. We will introduce the dataset and experiment settings in Section 4.1, present the comparison results in Section 4.2 and investigate some method details in Section 4.3. Our source code is available[1] for reference.

4.1 Datasets and Experimental Setup

Datasets: To the best of our knowledge, there is no public large-scale dataset available for fine-grained entity typing in $\mathcal{KB}$. We construct a dataset from DBpedia. The type hierarchy in DBpedia is of tree structure. We extract link relation and attribute information from DBpedia[2]. For each entity, we use the single type-path assigned in DBpedia as the ground truth.

For this dataset, we remove those entities which are only labeled as *Thing*. Because they do not provide any semantic information. In representation learning phase, we utilize the whole dataset with only 50% entities are labeled with types. In type predictor phase, we follow a 50/30/20 ratio to split the dataset into training, dev and test sets. Table 1 shows the detailed statistics. Each entity has 3.27 types on average.

Table 1: Statistics of the dataset

Type	**Entity**	**Link**	**Attribute**		
			word	**property**	**category**
214	300,000	5,243,230	5,350	200	350

Metrics: To evaluate the performance of our proposed method, we use Accuracy (**Strict-F1**), Micro-averaged F1 (**Mi-F1**) and Macro-averaged F1 (**Ma-F1**), which have been used in many fine-grained typing systems (Yaghoobzadeh and Schütze, 2015; Yaghoobzadeh and Schütze, 2017; Ren et al., 2016a).
Parameter Settings: Our implementation of **APE** is based on TensorFlow[3]. Regarding the choice of activation function of hidden layers, we have tried *ReLU*, *softsign* and *tanh*, finding *softsign* leads to the best performance in general. We randomly initialize model parameters with a Gaussian distribution (with a mean of 0.0 and standard deviation of 0.01), optimizing the model with mini-batch Adam (Kingma and Ba, 2014). We set fixed margin $\xi = 1$ in our experiments. Both λ_1 and λ_2 are chosen among $\{0.2, 0.4, 0.6, 0.8\}$, we found $\lambda_1 = 0.8$ and $\lambda_2 = 0.4$ achieve best performance.
Baseline: We denote our model as **APE**, and compare it with three sets of methods:

- **Entity typing methods**: We compare **APE** with five state-of-the-art entity typing methods. **FIGMENT** uses entity representation learned from annotated corpus for type inference (Yaghoobzadeh and Schütze, 2015). **CUTE** employs a multi-label hierarchical classification method to address entity typing task (Xu et al., 2016). **MuLR** uses multi-level representations of entities (character, word and entity) to make type inference in $\mathcal{KB}$ (Yaghoobzadeh and Schütze, 2017). **Global** utilizes entity text description to infer missing entity type instances (Neelakantan and Chang, 2015). **Corpus** designs global and context model to make type inference jointly (Yaghoobzadeh et al., 2017).

- **Network Embedding methods**: Attributed and predictive network embedding methods can learn informative entity representations, so we compare **APE** with them. **PTE** learns predictive text embeddings from heterogeneous network (Tang et al., 2015a). **Planetoid** is a semi-supervised network embedding method which makes neighbor and label prediction jointly (Yang et al., 2016). **ASNE** integrates structural and attribute information via a deep neural architecture (Liao et al., 2017).

- **APE's variants**: If we ignore the type information, that is, utilize the embeddings learned from section 3.2 to make type inference, the method is denoted as **APE**$_{no\_type}$. Similarly, if we omit the type hierarchy modeling in Section 3.3.2, the method is denoted as **APE**$_{no\_hierarchy}$.

4.2 Overall Comparison Results

Table 2 shows the overall performance and we have the following conclusions:

[1]https://github.com/Tsinghua-PhD/APE
[2]http://wiki.dbpedia.org/downloads-2016-10
[3]https://www.tensorflow.org/

Table 2: Typing performance on DBpedia dataset

Type	Algorithm	Acc	Ma-F1	Mi-F1
Entity Typing	**FIGMENT**	0.463	0.619	0.628
	CUTE	0.515	0.673	0.677
	MuLR	0.501	0.654	0.662
	Corpus	0.488	0.662	0.659
	Global	0.457	0.608	0.615
Network Embedding	**Planetoid**	0.434	0.585	0.590
	ASNE	0.417	0.568	0.573
	PTE	0.352	0.512	0.518
Our Variants	**APE**$_{no\_type}$	0.492	0.649	0.657
	APE$_{no\_hierarchy}$	0.531	0.694	0.689
	APE	**0.545**	**0.702**	**0.711**

Comparison with Entity Typing methods. Our model outperforms the state-of-the-art entity typing methods, and achieves 3.4% and 2.9% improvement in Mi-F1 and Ma-F1 respectively, because it employs multiple kinds of information for type inference. Each part can provide complementary information about entities for type inference. Our model leverages label information (type information) in representation learning phase, equip the learned embeddings with strong predictive power in entity typing task.

Comparison with Network Embedding methods. Our model outperforms the state-of-the-art network embedding methods (i.e., **Planetoid**, **ASNE** and **PTE**), and achieves 12.1% and 11.7% improvement in Mi-F1 and Ma-F1 respectively. We model the structure of type hierarchy in a margin-based loss function. Specifically, ancestor order makes entities closer to specific types than general types, and sibling order makes the learned entity embeddings have strong discriminative power between sibling types.

Comparison with Variants. **APE** consistently outperforms $\mathbf{APE}_{no\_type}$ and $\mathbf{APE}_{no\_hierarchy}$, and the simplest version $\mathbf{APE}_{no\_type}$ could achieve comparable results with state-of-the-art methods. The results verify that each of these information sources contributes complementary information for *fine-grained typing of entities*. Structural information (link relation) provides important semantic relatedness between entities. Attribute information (text, property and category) provides additional clues for type inference. Type information (type hierarchy) makes our model more powerful in classification task.

4.3 Result Analysis

4.3.1 Impact of Features

We investigate the effect of information sources used to learn entity embeddings. In Table 3, **Stru**, **T**, **P** and **C** is short for structural information, text description, property and category, respectively. From Table 3, we find different information sources play different roles in this task. *Structural information* is most common in $\mathcal{KB}$, it makes related entities have similar representations. It models the relation between entities, while neglecting entities's self-information. *Attribute information* (i.e., text description, property and category) can be viewed as entity features, which is often informative for type inference. We compare our model **APE**, $\mathbf{APE}_{no\_type}$ and $\mathbf{APE}_{no\_hierarchy}$ with different kinds of information combination. For attribute information, category information is more import than property information and text descriptions.

4.3.2 Effects of Labeled Data

We investigate the effect of labeled data proportion. Intuitively, our model can learn better embeddings and achieve better performance if we have more labeled entities. The results in Fig. 3 show that three metrics strikingly increase with more labeled data added, and gradually become stable when the proportion is over 0.5. It proves that our model can achieve a satisfactory result with not too much labeled data, and this advantage benefits from the entity link relations and network embedding.

Table 3: Performance on different information source combination.

Information Type	APE			APE_{no_type}			$\text{APE}_{no\_hierarchy}$		
	Acc	Ma-F1	Mi-F1	Acc	Ma-F1	Mi-F1	Acc	Ma-F1	Mi-F1
Stru+T	0.467	0.623	0.631	0.463	0.625	0.634	0.472	0.632	0.633
Stru+P	0.475	0.633	0.637	0.480	0.638	0.642	0.489	0.651	0.653
Stru+C	0.481	0.641	0.646	0.487	0.649	0.647	0.486	0.650	0.647
Stru+P+C	0.479	0.642	0.649	0.520	0.685	0.679	0.523	0.677	0.669
Stru+T+P	0.484	0.647	0.655	0.507	0.658	0.659	0.510	0.657	0.660
Stru+T+C	0.481	0.643	0.653	0.518	0.683	0.681	0.525	0.678	0.673
Stru+T+P+C	**0.492**	**0.649**	**0.657**	**0.531**	**0.694**	**0.689**	**0.545**	**0.702**	**0.711**

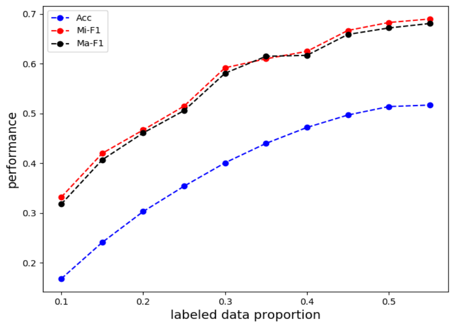


Figure 3: Performance on different labeled data proportion.

Table 4: Type macro average F1 on test for all, frequent and infrequent types, *#training/test* means the average entity number for each type.

Type	infrequent	frequent	all
# training	1013	13,513	8,673
# test	417	5,217	3,462
FIGMENT	0.327	0.620	0.557
MuLR	0.374	0.718	0.636
Corpus	0.447	0.783	0.675
CUTE	0.418	0.730	0.652
APE	**0.461**	**0.792**	**0.681**

4.3.3 Results of Frequent/Infrequent Types

We use type macro average F1 proposed by (Yosef et al., 2012) to evaluate the performance on *frequent types* and *infrequent types*. Note that it is different from Ma-F1 reported in Table 2. The results for infrequent types (occur less than 2,000 times) and frequent types (occur more than 10,000 times) are shown in Table 4. Generally, the performance on infrequent types is worse than frequent ones. Our model consistently outperforms the other methods on infrequent types, which demonstrates its ability on dealing with rare types.

5 Related Work

Fine-grained entity typing in $\mathcal{KB}$ is an important sub-task of knowledge base completion. One way to address this problem is making type inference based on entity's attribute information in $\mathcal{KB}$, such as textual description, property, category and so on. (Neelakantan and Chang, 2015) infer missing types for entities in $\mathcal{KB}$ based on text descriptions. (Xu et al., 2016) employ a multi-label hierarchical classification method to assign Chinese entities with DBpedia types based on property and category information. The other way relies on large-scale annotated corpus, then extracts type information from the annotated corpus for entities in $\mathcal{KB}$. (Yaghoobzadeh and Schütze, 2015) first propose FIGMENT to address this problem. They only used contextual information to assign types for entities in $\mathcal{KB}$. After that, they present FIGMENT-Multi to learn multi-level representations of entities on three complementary levels (character, word and entity). FIGMENT-Multi predicts whether an entity is a member of a type based on the learned embeddings (Yaghoobzadeh and Schütze, 2017). Finally, they propose an embedding based method which combines a global model with a context model. A global model that scores based on aggregated context information and a context model that aggregates the scores of individual contexts (Yaghoobzadeh et al., 2017).

Fine-grained entity typing in text is related to our task. Different from the works mentioned above, it seeks to detect local types of an entity mention inside an individual sentence, and the same entity could have different types in different sentences (Ling and Weld, 2012; Yosef et al., 2012; Gillick et al., 2014; Dong et al., 2015; Del Corro et al., 2015; Yogatama et al., 2015; Ren et al., 2016b; Ren et al., 2016a; Shimaoka et al., 2017; Rabinovich and Dan, 2017).

6 Conclusion

In this paper, we propose an attributed and predictive entity embedding method to address the task of fine-grained entity typing in $\mathcal{KB}$. It combines the advantages of attributed network embedding method and semi-supervised learning. Specifically, it employs a deep neural network architecture to integrate structure and attribute information of an entity. We jointly predict a labeled entity's neighbors and types, and model the structure of type hierarchy in a margin-based way. Experiments on real world datasets demonstrate the effectiveness of the proposed model.

Acknowledgements

The work is supported by National Key Research and Development Program of China (2017YFB1002101), NSFC key project (U1736204), Ministry of Education and China Mobile Research Fund (No. 20181770250), and THUNUS NExT Co-Lab.

References

Luciano Del Corro, Abdalghani Abujabal, Rainer Gemulla, and Gerhard Weikum. 2015. Finet: Context-aware fine-grained named entity typing. In *Conference on Empirical Methods in Natural Language Processing*, pages 868–878.

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 the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 601–610. ACM.

Li Dong, Furu Wei, Hong Sun, Ming Zhou, and Ke Xu. 2015. A hybrid neural model for type classification of entity mentions. In *International Conference on Artificial Intelligence*, pages 1243–1249.

Greg Durrett and Dan Klein. 2014. A joint model for entity analysis : Coreference , typing , and linking. *TACL*, pages 477–490.

Dan Gillick, Nevena Lazic, Kuzman Ganchev, Jesse Kirchner, and David Huynh. 2014. Context-dependent fine-grained entity type tagging. *arXiv:1412.1820.*

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

Hang Li, Haozheng Wang, Zhenglu Yang, and Masato Odagaki. 2017. Variation autoencoder based network representation learning for classification. In *ACL 2017, Student Research Workshop*, pages 56–61.

Lizi Liao, Xiangnan He, Hanwang Zhang, and Tat Seng Chua. 2017. Attributed social network embedding. *arXiv:1705.04969.*

Xiao Ling and Daniel S Weld. 2012. Fine-grained entity recognition. In *AAAI*, pages 94–100.

Xiao Ling, Sameer Singh, and Daniel S Weld. 2015. Design challenges for entity linking. *TACL*, pages 315–328.

Yang Liu, Kang Liu, Liheng Xu, and Jun Zhao. 2014. Exploring fine-grained entity type constraints for distantly supervised relation extraction. In *COLING*, pages 2107–2116.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *International Conference on Neural Information Processing Systems*, pages 3111–3119.

Gregory Murphy. 2004. *The big book of concepts.* MIT press.

David Nadeau and Satoshi Sekine. 2007. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26.

Arvind Neelakantan and Ming Wei Chang. 2015. Inferring missing entity type instances for knowledge base completion: New dataset and methods. *Computer Science*, pages 35–40.

Maxim Rabinovich and Klein Dan. 2017. Fine-grained entity typing with high-multiplicity assignments. In *Meeting of the Association for Computational Linguistics*, pages 330–334.

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, pages 147–155. Association for Computational Linguistics.

Xiang Ren, Wenqi He, Meng Qu, Lifu Huang, Heng Ji, and Jiawei Han. 2016a. Afet: Automatic fine-grained entity typing by hierarchical partial-label embedding. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1369–1378.

Xiang Ren, Wenqi He, Meng Qu, Clare R Voss, Heng Ji, and Jiawei Han. 2016b. Label noise reduction in entity typing by heterogeneous partial-label embedding. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1825–1834.

Sonse Shimaoka, Pontus Stenetorp, Kentaro Inui, and Sebastian Riedel. 2017. Neural architectures for fine-grained entity type classification. In *EACL*, pages 1271–1280.

Jian Tang, Meng Qu, and Qiaozhu Mei. 2015a. Pte: Predictive text embedding through large-scale heterogeneous text networks. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1165–1174. ACM.

Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015b. Line: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1067–1077. ACM.

Ji Xin, Yankai Lin, Zhiyuan Liu, and Maosong Sun. 2018. Improving neural fine-grained entity typing with knowledge attention. In *AAAI*.

Bo Xu, Yi Zhang, Jiaqing Liang, Yanghua Xiao, Seung-won Hwang, and Wei Wang. 2016. Cross-lingual type inference. In *International Conference on Database Systems for Advanced Applications*, pages 447–462.

Yadollah Yaghoobzadeh and Hinrich Schütze. 2015. Corpus-level fine-grained entity typing using contextual information. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 715–725.

Yadollah Yaghoobzadeh and Hinrich Schütze. 2017. Multi-level representations for fine-grained typing of knowledge base entities. *arXiv:1701.02025*.

Yadollah Yaghoobzadeh, Heike Adel, and Hinrich Schütze. 2017. Corpus-level fine-grained entity typing. *arXiv:1708.02275*.

Mohamed Yahya, Klaus Berberich, Shady Elbassuoni, and Gerhard Weikum. 2014. Robust question answering over the web of linked data. In *CIKM*, pages 1107–1116.

Zhilin Yang, William W Cohen, and Ruslan Salakhutdinov. 2016. Revisiting semi-supervised learning with graph embeddings. In *ICML*, pages 40–48.

Dani Yogatama, Dan Gillick, and Nevena Lazic. 2015. Embedding methods for fine grained entity type classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL*, pages 26–31.

M Amir Yosef, Sandro Bauer, Johannes Hoffart, Marc Spaniol, and Gerhard Weikum. 2012. Hyena: Hierarchical type classification for entity names. In *Proceedings of the 24th International Conference on Computational Linguistics*, pages 1361–1370.