Fine-Grained Entity Typing via Hierarchical Multi Graph Convolutional Networks

Hailong Jin[1,2,3], **Lei Hou**[1,2,3]*, **Juanzi Li**[1,2,3], **Tiansi Dong**[4]
[1]Department of Computer Science and Technology, Tsinghua University
[2]KIRC, Institute for Artificial Intelligence, Tsinghua University
[3]Beijing National Research Center for Information Science and Technology
[4]B-IT, University of Bonn, Bonn, Germany
{jinhl15@mails., houlei@, lijuanzi@}tsinghua.edu.cn
dongt@bit.uni-bonn.de

Abstract

This paper addresses the problem of inferring the fine-grained type of an entity from a knowledge base. We convert this problem into the task of graph-based semi-supervised classification, and propose Hierarchical Multi Graph Convolutional Network (**HMGCN**), a novel Deep Learning architecture to tackle this problem. We construct three kinds of connectivity matrices to capture different kinds of semantic correlations between entities. A recursive regularization is proposed to model the *subClassOf* relations between types in given type hierarchy. Extensive experiments with two large-scale public datasets show that our proposed method significantly outperforms four state-of-the-art methods.

1 Introduction

Nowadays, Knowledge Base ($\mathcal{KB}$ for short) attracts increasing research interests in various areas. One of the fundamental components in $\mathcal{KB}$ is the information of entity type, which clusters a group of entities with same properties, and is the glue that holds our mental world together (Murphy, 2004). 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., 2016). For example, *Messi* is classified as having the path of following types: $FootballPlayer \subset Athlete \subset Person$. Types in $\mathcal{KB}$ are usually organized as a hierarchical structure, namely *type hierarchy*. Unfortunately, most $\mathcal{KB}$s are incomplete and lack of type information. For example, in DBpedia, the average number of types is only 2.9 (5,044,223 entities with 14,760,728 types), while 36.53% entities do not have type information. As Figure 1 shows, for each unlabeled entity, we will fully utilize its textual description, category and property to predict missing types. In this paper, we aim at assigning Fine-grained Entity Typing for entities in $\mathcal{KB}$ (such as Wikipedia and DBpedia).

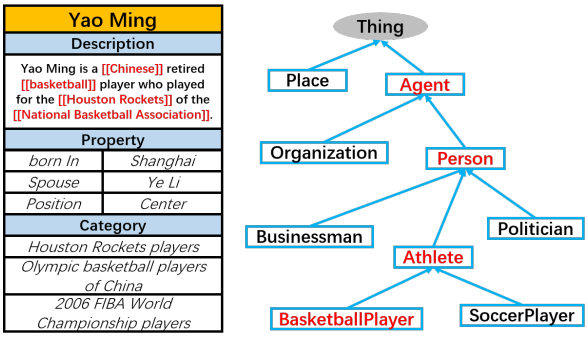


Figure 1: Entity (i.e., Yao Ming) with textual description (with anchor text in red), property (e.g., bornIn Shanghai) and category (e.g., Houston Rockets players). Yao Ming is associated with a type-path: $BasketballPlayer \subset Athlete \subset Person \subset Agent \subset Thing$

Many researches have been carried out in this field. State-of-the-art methods normally learn distributed representation for each entity, and apply a multi-label classification model to make type inference. For example, (Neelakantan and Chang, 2015) and (Xu et al., 2016) exploit various kinds of information to construct the feature representation of an entity, such as entity textual description, property and category. After that, a predict function is learned to infer whether entity e is an instance of type t. (Yaghoobzadeh et al., 2017) focus on the names of entities and the context of entity mentions (in text), and design two scoring models for pairs of entities and types. All these works ignore the internal relations among entities, and assign types for each entity in isolation. (Jin et al., 2018) viewed the internal relations

* Corresponding author

Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, pages 4969–4978, Hong Kong, China, November 3–7, 2019.

among entities as the structural information, and constructed an Entity Graph, further proposed a network embedding framework to learn the correlation among entities. It will be profitable to enrich entity features to entity graph structures, as recent studies have suggested to bring both node feature and graph structure information in a convolutional manner (Defferrard et al., 2016; Atwood and Towsley, 2016). Among these works, graph convolutional network (GCN) is the most widely-used model which can directly operate on graphs with arbitrary sizes and shapes (Kipf and Welling, 2016). The inputs of a GCN are feature vectors of nodes and graph structure. The most significant aspect of GCN is information diffusion, with which node's feature vector can be enriched by all the feature vectors of its neighbors.

Here, we convert entities in $\mathcal{KB}$ into three semantic graphs, each encoding a specific kind of correlation among entities, as follows: $\mathbf{A}_{co}$ a graph of the topical co-occurrence relations among entities, $\mathbf{A}_{cat}$ a category-based graph encoding the category-proximity between entities, and $\mathbf{A}_{prop}$ a property-based graph encoding the property-proximity between entities. We propose Hierarchical Multi Graph Convolutional Network (**HMGCN**), a novel deep learning architecture consisting of three Graph Convolutional Networks (GCNs): $\mathbf{GCN}_{co}$ and $\mathbf{GCN}_{cat}$ and $\mathbf{GCN}_{prop}$. Each connectivity matrix is fed to its corresponding GCN model, and the three GCN models are learned by shared parameters and consistency regularization. We adopt a simple and effective recursive regularization to deal with *subClassOf* relations between types. The main contributions of this paper are as follows:

- Graph convolutional network is applied for fine-grained entity typing task, which effectively integrates entity feature and structural information.
- Multi connectivity matrices are constructed to encode different kinds of semantic relatedness between entities. A recursive regularization is proposed to model *subClassOf* relations between types.
- Extensive experiments show that our proposed method significantly outperforms four state-of-the-art methods, with 1.7% and 1.3% improvement in Mi-F1 and Ma-F1 (on FIGER dataset), respectively.

The rest of this paper is organized as follows. Section 2 formally defines the problem of Fine-Grained Entity Typing in knowledge bases. Section 3 describes the proposed approach in detail. Section 4 reports a number of experiments with evaluations. 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*** *$\mathcal{KB} = (\mathcal{E}, \mathcal{T}, \mathcal{R})$, where $\mathcal{E}$, $\mathcal{T}$ and $\mathcal{R}$ are sets of entities, types and isA relations, respectively. $|\mathcal{E}| = N$ and $|\mathcal{T}| = K$. $\mathcal{R} = \mathcal{R}_i \cup \mathcal{R}_s$ consists of instanceOf and subClassOf triples. $\mathcal{R}_i = \{(e_i, isA, t_j) | e_i \in \mathcal{E}, t_j \in \mathcal{T}\}$ collects all instanceOf triples and $\mathcal{R}_s = \{(t_i, isA, t_j) | t_i, t_j \in \mathcal{T}\}$ collects all subClassOf triples.*

Note that not each entity in a knowledge base $\mathcal{KB}$ is assigned with types. We split the entity set $\mathcal{E}$ into two subsets: $\mathcal{E}^l$ and $\mathcal{E}^u$, $\mathcal{E} = \mathcal{E}^l \cup \mathcal{E}^u$, $\mathcal{E}^l \cap \mathcal{E}^u = \emptyset$. $\mathcal{E}^l$ denotes the entity set whose member is assigned with type information. $\mathcal{E}^u$ denotes the entity set whose member has no type information. $|\mathcal{E}^l| = N^l$, $|\mathcal{E}^u| = N^u$. Types in $\mathcal{T}$ form a tree or a directed acyclic graph (DAG) via *subClassOf* relations, we refer it as type hierarchy $\mathcal{H} = (\mathcal{T}, \mathcal{R}_s)$.

For each entity, we distinguish three kinds of features, namely, *Text Description*, *Category*, and *Property*, which are beneficial to *Fine-Grained Entity Typing* task.

- *Text Description* is a succinct summary of an entity, providing valuable clues to predict entity types. As Figure 1 shows, there are a large number of anchor texts in textual description.
- *Category* refers, in Wikipedia, actually to tags/topics that group entities on similar subjects. For instance, *Yao Ming* in Wikipedia has category *Olympic basketball players of China*, which is the topic of the entity instead of its type. *Category* can be a useful information to infer missing type information.
- *Property* of entities is important cue in type inference. For instance, from *Yao Ming*'s property *playing position*, one may infer that *Yao Ming* is an *Athlete*.

Besides entity features above, there are many kinds of semantic correlations between entities, which organize entities into graph structures. We refer it as **entity graph**. Each kind of semantic correlation corresponds to a kind of connectivity matrix. There are many ways to construct entity graph via different connectivity manner, which will be discussed in Section 3.

Definition 2 ***Entity Graph*** $\mathcal{G} = (\mathbf{A}, \mathbf{X}, \mathbf{Y})$. $\mathbf{A} \in \mathbb{R}^{N \times N}$ *is the connectivity matrix representing a kind of link relation between entities.* $\mathbf{X} \in \mathbb{R}^{N \times M}$ *is the feature matrix representing inherent features for all entities, where M is the number of features. Each row in* $\mathbf{X}$ *refers to an entity's feature vector* $\mathbf{X}_i$. $\mathbf{X}$ *can either be binary or take any real value.* $\mathbf{Y} \in \mathbb{R}^{N^l \times K}$ *is the type matrix collecting type information for entities in* $\mathcal{E}^l$. *Each row in* $\mathbf{Y}$ *refers to the instanceOf relations between an entity and all types.* $\mathbf{Y}_{ij} = 1$ *if* e_i *has been assigned with type* t_j, *otherwise 0.*

The task of Fine-Grained Entity Typing can be formally described as follows: Given a partially-labeled Entity Graph $\mathcal{G} = (\mathbf{A}, \mathbf{X}, \mathbf{Y})$, we aim at learning a type predictor from $\mathbf{Y}$, which comprehensively takes graph structure and entity features into account. Then we utilize the learned type predictor to predict the missing *instanceOf* relations in $\mathcal{E}^u$, i.e., whether $(e_i, instanceOf, t_j)$ is true. In this way, we convert the task of Fine-Grained Entity Typing into a task of graph-based semi-supervised classification.

3 Hierarchical Multi Graph Convolutional Network (HMGCN)

State-of-the-art methods convert entities in $\mathcal{KB}$ into entity graph, then apply graph-based algorithm. The difficulties lie in two facts:

- Firstly, it is hard to effectively integrate entity features and structural information. Although network embedding methods, in terms of attributed network embedding, can take both graph structure and features into consideration (Huang et al., 2017), the node features are usually only served as auxiliary information in structure learning (Yang et al., 2016; Liao et al., 2017).

- Secondly, it is not easy to effectively integrate different connectivity matrix which capture different kinds of structural information (Wang et al., 2018; Zhuang and Ma, 2018).

We propose a Hierarchical Multi GCN model (**HMGCN**) to encode heterogeneous structural information in an ensemble manner, as illustrated in Figure 2. We construct three undirected entity graphs to capture different kinds of semantic correlations between entities, i.e., co-occurrence graph $\mathbf{A}_{co}$, category-based graph $\mathbf{A}_{cat}$, and property-based graph $\mathbf{A}_{prop}$. $\mathbf{A}_{co}$ encodes the topical relevance between entities, and is derived from textual anchor texts. $\mathbf{A}_{cat}$ is constructed through similarity computation based on category information, based on the assumption that entities with similar categories tend to have the same type. In the same way, we construct property-based graph $\mathbf{A}_{prop}$. Each entity graph is fed to corresponding GCN model, namely, $\mathbf{GCN}_{co}$, $\mathbf{GCN}_{cat}$ and $\mathbf{GCN}_{prop}$. These models use shared parameters and unsupervised consistency regularization, so that they can jointly consider opinions from three semantic perspectives. To achieve this, a simple and effective recursive regularization is adopted to deal with the *subClassOf* relations between types.

3.1 GCN-Based Entity Type Classification

GCN model consists of multiple stacked GCN layers. Given the input feature matrix $\mathbf{X} \in \mathbb{R}^{N \times M}$ and adjacency matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$, the output of the i-th hidden layer of the network $\mathbf{Z}^{(i)}$ is defined as:

$$\mathbf{Z}^{(i)} = \sigma(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{Z}^{(i-1)}\mathbf{W}^{(i)}) \quad (1)$$

$\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I_n}$ is the adjacency matrix with self-loops, where $\mathbf{I_n} \in \mathbb{R}^{N \times N}$ is the identity matrix. $\tilde{\mathbf{D}}_{ii} = \sum_j \tilde{\mathbf{A}}_{ij}$. Accordingly, $\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}$ is the normalized adjacency matrix. $\mathbf{Z}_A^{(i-1)}$ is the output of the $(i-1)$-th layer, and $\mathbf{Z}^{(0)} = \mathbf{X}$. $\mathbf{W}^{(i)}$ are the trainable parameters of the network, and $\sigma(\cdot)$ denotes an activation function (e.g., *ReLU*, *Sigmoid*). A detailed explanation of GCN model is presented in the original work (Kipf and Welling, 2016).

Since Fine-grained Typing is a multi-label classification problem. The final layer further connects to a set of K Sigmoid functions, which correspond to the K types in the type hierarchy. Given the set of labeled entities $\mathcal{E}^l$, our model optimizes

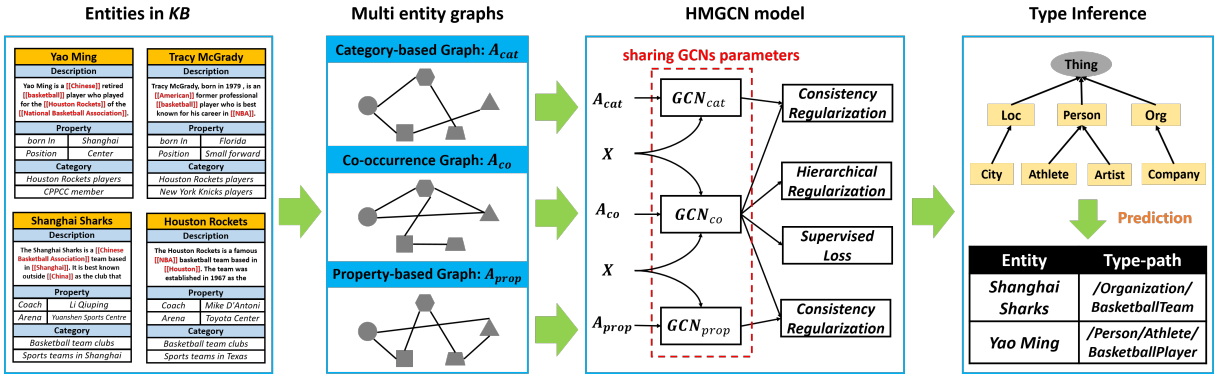


Figure 2: Framework of **HMGCN**. We convert entities in $\mathcal{KB}$ into multi kinds of semantic graphs. The co-occurrence graph $\mathbf{A}_{co}$ is constructed from anchor text in textual description. The category-based graph $\mathbf{A}_{cat}$ is derived from category proximity. The property-based graph $\mathbf{A}_{prop}$ is derived from property proximity. Each graph is fed to corresponding GCN model with same feature matrix $\mathbf{X}$. The three GCN models are learned by sharing parameters and consistency regularization. A hierarchical regularization is used to deal with the subClassOf relations between types.

the cross-entropy between the true type distribution and the predicted distribution.

$$L_s = -\sum_{i=1}^{N_l}\sum_{j=1}^{K} \mathbf{Y}_{i,j} \ln \hat{\mathbf{Z}}_{i,j} + (1-\mathbf{Y}_{i,j}) \ln (1-\hat{\mathbf{Z}}_{i,j}) \tag{2}$$

Where $\mathbf{Y}_{i,j}$ indicates whether entity e_i belongs to type t_j. $\hat{\mathbf{Z}}_{i,j}$ refers to the probability of GCN prediction.

The role of $\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{Z}^{(i-1)}\mathbf{W}^{(i)}$ is to exactly conduct a 1-hop diffusion process in each layer. Namely, a node's feature vector is enriched in the way of linearly adding all the feature vectors of its neighbors. With more hidden layers, an entity's feature can diffuse to further entities (2-hop, 3-hop and so on). Research shows that there's no significant increase when the number of layers is higher than 3 (Kipf and Welling, 2016). Here, we assume adjacency matrix $\mathbf{A}$ and feature matrix $\mathbf{X}$ are given. In the next part of this section, we show how to compute $\mathbf{A}$ and $\mathbf{X}$.

3.2 Connectivity Matrix Designing

In Knowledge Graphs, there are large numbers of descriptive anchor texts that are helpful for type inference. For an entity e, $CXT(e)$ is an entity set which contains all entities occur in its textual description. For example in Figure 1, *Houston Rockets* is in $CXT(YaoMing)$. If entity $e_i \in CXT(e_j)$, we can say that e_i occurs in e_j'context or e_j cites e_i in its textual description. These are co-occurrence information among entities, and can be explicitly extracted and represented by the connectivity matrix $\mathbf{A}_{co}$.

$$A_{co}[i,j] = \begin{cases} 1 & e_i \in CXT(e_j) \text{ or } e_j \in CXT(e_i) \\ 0 & otherwise \end{cases} \tag{3}$$

$\mathbf{A}_{co}$ can capture co-occurrence relations, though not precise enough. For example, *Houston Rockets* and *NBA* occur in *Yao Ming*'s context, but they do not belong to same type.

Besides $\mathbf{A}_{co}$, we calculate the correlation between entities via category information. If entity e_i and e_j share common categories, e_i and e_j will be likely belong to a same type. For example in Figure 2, both *Houston Rockets* and *Shanghai Sharks* have category *Basketball team clubs*, they have same type *BasketballTeam*. We construct $\mathbf{A}_{cat}$, and use Jaccard similarity coefficient to calculate each element $\mathbf{A}_{cat}[i,j]$:

$$\mathbf{A}_{cat}[i,j] = \frac{|Cat(e_i) \cap Cat(e_j)|}{|Cat(e_i) \cup Cat(e_j)|} \tag{4}$$

in which, $Cat(e)$ is the category set of entity e. In the same way, we can calculate the correlation between entities via property information, and construct $\mathbf{A}_{prop}$, whose element is calculated as follows.

$$\mathbf{A}_{prop}[i,j] = \frac{|Prop(e_i) \cap Prop(e_j)|}{|Prop(e_i) \cup Prop(e_j)|} \tag{5}$$

in which $Prop(e)$ is the property set of entity e.

Each connectivity matrix is fed to its corresponding GCN model, i.e., $\mathbf{GCN}_{co}$, $\mathbf{GCN}_{cat}$ and

$\mathbf{GCN}_{prop}$, respectively. As mentioned in Section 2, each entity is associated with text description, category and property. We used category and property information to construct connectivity matrix, therefore, words appeared in the text description of an entity can be approximately viewed as features of this entity. We apply fastText method, which treats the average of word/n-grams embeddings as entity feature (Joulin et al., 2016), and construct the feature matrix $\mathbf{X}$. As shown in Figure 2, both $\mathbf{GCN}_{co}$, $\mathbf{GCN}_{cat}$ and $\mathbf{GCN}_{prop}$ take $\mathbf{X}$ as input feature matrix.

3.3 Parameters Sharing and Consistency Regularization

The success of our model largely relies on the strategy that three GCN models share common parameters (i.e., neural network weights $\mathbf{W}^{(i)}$ in Eqs. 1), as shown in Figure 2. By doing so, our model (which is characterized by the parameters $\mathbf{W}^{(i)}$) can simultaneously consider the knowledge encoded in $\mathbf{A}_{co}$, $\mathbf{A}_{cat}$ and $\mathbf{A}_{prop}$. The prediction of $\mathbf{GCN}_{co}$ and $\mathbf{GCN}_{cat}$ are denoted as $\hat{\mathbf{Z}}^{co}$ and $\hat{\mathbf{Z}}^{cat} \in \mathbb{R}^{N\times K}$, respectively. To jointly consider the opinions from $\mathbf{GCN}_{co}$ and $\mathbf{GCN}_{cat}$, we apply an unsupervised regularizer for the ensemble. We minimize the mean squared difference between $\hat{\mathbf{Z}}^{co}$ and $\hat{\mathbf{Z}}^{cat}$ for all N entities.

$$L_{c1} = \sum_{i=1}^{N} \left\| \hat{\mathbf{Z}}^{cat}_{i,:} - \hat{\mathbf{Z}}^{co}_{i,:} \right\|^2 \tag{6}$$

The prediction of $\mathbf{GCN}_{prop}$ is denoted as $\hat{\mathbf{Z}}^{prop}$. In the same way, we minimize the mean squared differences between $\hat{\mathbf{Z}}^{co}$ and $\hat{\mathbf{Z}}^{prop}$ over all N entities.

$$L_{c2} = \sum_{i=1}^{N} \left\| \hat{\mathbf{Z}}^{prop}_{i,:} - \hat{\mathbf{Z}}^{co}_{i,:} \right\|^2 \tag{7}$$

Our model can jointly consider the opinions of $\mathbf{GCN}_{co}$, $\mathbf{GCN}_{cat}$ and $\mathbf{GCN}_{prop}$. Although sharing the same parameters $\mathbf{W}^{(i)}$, three GCN models utilize different connectivity matrices as input, which capture different semantic correlations between entities. This difference may cause the predictions of three GCN models to differ. Our model is expected to give a consistent prediction via the proposed unsupervised consistency regularizations, i.e., minimizing Eqs. 6 and 7. As a result, the learned parameter matrix $\mathbf{W}^{(i)}$ considers the judgments from both $\mathbf{GCN}_{co}$, $\mathbf{GCN}_{cat}$ and $\mathbf{GCN}_{prop}$.

3.4 Recursive Hierarchical Regularization

Leaf nodes (in type hierarchy) may have insufficient training examples. In that case, decisions can be regularized by its parent, if a type hierarchy is available. We introduce dependency relations among types to improve the classification performance. Similar to (Peng et al., 2018; Gopal and Yang, 2013), we use a recursive regularization over the final layer. Two types shall have similar embeddings, if they are close in a graph. In particular, types that are related by *subClassOf* relation shall have similar embeddings. For example, in Figure 1, there is an edge between *Athlete* and *BasketballPlayer*, so the parameters of the two types could be similar to each other. Assuming there are n hidden layers, the last layer parameter matrix $\mathbf{W}^{(n)}$ can be regarded as the type embedding matrix, each row in $\mathbf{W}^{(n)}$ refers to a type representation. Let $C(t)$ refer to the sub-type set of t (t's children in the hierarchy). We use the following recursive regularization term to regularize the parameter of each type:

$$L_{hie} = \sum_{i=1}^{K} \sum_{j\in C(t_i)} \frac{1}{2} \left\| \mathbf{W}^{(n)}_{i,:} - \mathbf{W}^{(n)}_{j,:} \right\|^2 \tag{8}$$

3.5 Model Training

We calculate the supervised loss over all labeled entities for $\mathbf{GCN}_{co}$ (Eqs. 2). Although $\mathbf{Z}^{co}_i$ is regarded as the final type prediction result, our model can consider the opinions from three GCN models via parameters sharing and consistency regularization. The total loss is the sum of supervised loss, two consistency regularizations and recursive regularization.

$$Loss = L_s + \lambda(t)(L_{c1} + L_{c2}) + \lambda' L_{hie} \tag{9}$$

in which $\lambda(t)$ is a dynamic weight function, and λ' is chosen among a fixed set based on performance on the dev set. At the beginning of the training process, $\lambda(t)$ is small, the loss function is mainly dominated by L_s. **HMGCN** is inclined to agree with $\mathbf{GCN}_{co}$ when making decisions, which encodes co-occurrence relation between entities. $\lambda(t)$ increases as time goes on, our model will obtain posterior distribution over the types using $\mathbf{GCN}_{co}$, and $\lambda(t)$ will force our model to consider simultaneously the knowledge encoded in $\mathbf{GCN}_{cat}$ and $\mathbf{GCN}_{prop}$. $\mathbf{GCN}_{cat}$ and $\mathbf{GCN}_{prop}$ play more important role when making decision. We adopt Adam (Kingma and Ba, 2014) to minimize the

above loss functions. For each $e_i \in \mathcal{E}^u$, we use $\mathbf{Z}_i^{co}$ as the type prediction result.

4 Experiments

4.1 Datasets and Metrics

Datasets: There are two public large-scale datasets available for Fine-Grained Entity Typing in $\mathcal{KB}$, *FIGER*[1] and *DBpedia* (Zhang et al., 2015). *FIGER* is a widely used dataset which is proposed by Yaghoobzadeh et al.. The *DBpedia* dataset is constructed by picking 14 non-overlapping types for text classification originally, we expand these 14 types and form a hierarchy. For each dataset, we extract entity feature from *DBpedia*[2]. For *FIGER* and *DBpedia*, we split entities into train (50%), dev (20%) and test (30%) sets. Since *DBpedia* dataset contains 630,000 entities, which is too big for computation, we divide it equally into three parts (i.e., DB-1, -2 and -3). Table 1 shows some statistics about the two datasets. Our source code is available[3] for reference. More detailed information can be found there.

Table 1: Statistics of the datasets.

Dataset	**FIGER**	**DB-1**	**DB-2**	**DB-3**
#Entities	200,000	210,000	210,000	210,000
#Types	102	48	48	48

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 (Ling and Weld, 2012; Ren et al., 2016; Yaghoobzadeh et al., 2017).

4.2 Methods for Comparison

Baseline: we compare **HMGCN** with four state-of-the-art of methods and three variants of **HMGCN**:

- **CUTE:** (Xu et al., 2016) utilizes three kinds of entity features: category, property and property-value pair, and employs a hierarchical multi-label classification method.
- **MuLR:** (Yaghoobzadeh and Schütze, 2017) applies embeddings of words, entities and types in entity typing task, and uses multi-level representations of entities via character, word, and entity embedding technologies.
- **FIGMENT:** (Yaghoobzadeh et al., 2017) introduces global model and context model to provide complementary information for entity typing.
- **APE:** (Jin et al., 2018) applies an attributed and predictive entity embedding method to learn entity representations. In a way, it takes graph structure and entity features into account.
- **HMGCN's variants:** $\textbf{HMGCN}_{no\_cat}$ only use $\textbf{GCN}_{co}$ and $\textbf{GCN}_{prop}$ to make type inference. $\textbf{HMGCN}_{no\_prop}$ only use $\textbf{GCN}_{co}$ and $\textbf{GCN}_{cat}$ to make type inference. $\textbf{HMGCN}_{no\_co}$ only use $\textbf{GCN}_{cat}$ and $\textbf{GCN}_{prop}$ to make type inference. $\textbf{HMGCN}_{flat}$ ignores the correlation between types, i.e., removes hierarchical recursive regularization.

Parameter Settings: **HMGCN** is implemented based on the original GCN model (Kipf and Welling, 2016). In our implementation, both $\textbf{GCN}_{co}$, $\textbf{GCN}_{cat}$ and $\textbf{GCN}_{prop}$ have two hidden layers. Namely, there are two separate parameter vectors, $\mathbf{W}^{(1)}$ and $\mathbf{W}^{(2)}$, that need training. Table 2 presents detailed information about the implementation of our method for each dataset, including (1) number of hidden units; (2) dropout rate; and (3) learning rate η. λ' is chosen among $\{0.00625, 0.0125, 0.025, 0.05, 0.1, 0.2, 0.4, 0.8\}$, we found $\lambda' = 0.2$ achieves best performance. We set $\lambda(t)$ in several ways, which is discussed in Section 4.4. We train all models for a maximum of 200 epochs (training iterations) using Adam (Kingma and Ba, 2014) and early stopping with a window size of 10 to minimize loss functions.

Table 2: Values of the Hyper-Parameters in **HMGCN** model.

Dataset	$\mathbf{W}^{(1)}$	$\mathbf{W}^{(2)}$	Dropout rate	Learning rate
DBpedia	32	48	30%	0.001
FIGER	64	102	10%	0.002

[1] https://github.com/yyaghoobzadeh/figment
[2] http://wiki.dbpedia.org/downloads-2016-10
[3] https://github.com/SIGKDD/HMGCN

4.3 Overall Comparison Results

Table 3 and 4 shows the overall performance on two datasets and we have the following conclusions:

Comparison with Entity Typing methods. Our model significantly outperforms the state-of-the-art entity typing methods over all datasets. **HMGCN** outperforms the best baseline with 1.7% and 1.3% improvement in Mi-F1 and Ma-F1 on *FIGER* dataset, 3.2% and 3.6% improvement in Mi-F1 and Ma-F1 on *DB-1* dataset, 3.9% and 3.5% improvement in Mi-F1 and Ma-F1 on *DB-2* dataset, 3.7% and 3.8% improvement in Mi-F1 and Ma-F1 on *DB-3* dataset, respectively. Compared with other methods, the performance of **CUTE** is relatively poor, as it ignores the rich structure information both in entity graph and type hierarchy. **FIGMENT** and **MuLR** achieve better performance, in part because powerful embedding technologies are applied and the decision is made in global-view and context-view. **APE** considers graph structure and entity features in attributed network embedding manner, while **HMGCN** applies this in a convolutional manner, which can embed the graph knowledge more sufficiently (Kipf and Welling, 2016; Zhuang and Ma, 2018).

Comparison with Variants. **HMGCN** consistently outperforms all four variants, because in **HMGCN** three GCN models provide complementary information. The key of **HMGCN** is parameter-sharing ($\mathbf{W}^{(i)}$ in $\mathbf{GCN}_{co}$, $\mathbf{GCN}_{cat}$ and $\mathbf{GCN}_{prop}$) and unsupervised regularizer for ensemble. The three GCNs encode different kinds of semantic correlations between entities, so that **HMGCN** could have a trade-off in decision-making. As a result, the trained parameters $\mathbf{W}^{(i)}$ have considered opinions from all three GCNs. $\mathbf{HMGCN}_{no\_co}$ and $\mathbf{HMGCN}_{no\_prop}$ demonstrate better performance than $\mathbf{HMGCN}_{no\_cat}$, which suggests that category information should be more import than co-occurrence and property information. $\mathbf{HMGCN}_{flat}$ ignores the correlations between types (i.e., *subClassOf* relations), while **HMGCN** take the structure of type hierarchy as prior knowledge. Infrequent types benefit from the recursive regularization in **HMGCN**. Because when an infrequent type has less training example, the decision can be regularized by its parent (super-type).

4.4 Result Analysis

Effects of Labeled Data. Intuitively, our model can learn better embeddings and achieve better performance, if we have more labeled entities. We investigate the effect of labeled data proportion. The results show that three metrics strikingly increase with more labeled data added, and gradually become stable when the proportion is over 0.5, as illustrated in Figure 3(a). This shows that our model can achieve a satisfactory result, even with not enough labeled data, and this advantage is benefited from the information diffusion of the GCN model, i.e., similar entities should share information between each other.

Results of Frequent/Infrequent Types. We evaluate the performance on *frequent types* ($frequency > 3,000$; 15 types) and *infrequent types* ($frequency < 200$; 36 types). The classification measure is the type macro average F1 (F1 of entities assigned to a type, then averaged over types) (Yosef et al., 2012). Note that it is different from Ma-F1 reported in Table 3. 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. The results for infrequent and frequent types in *FIGER* dataset are illustrated in Figure 3(b).

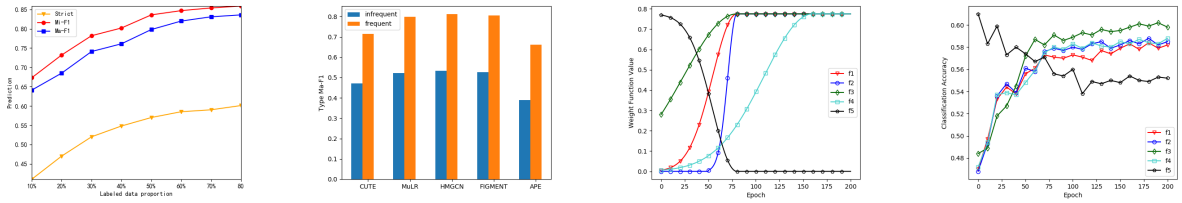
Effect of Regularization Weight $\lambda(\mathbf{t})$. Our model uses a weight function $\lambda(t)$ to balance the trade-off between the supervised loss and unsupervised consistency regularizer. We devised several different weight functions (Zhuang and Ma, 2018), as shown in Figure 3(c). Here, t is defined as the number of epochs. f_i $(1 \leq i \leq 4)$ increases with different increment speed, as the number of epochs increases; while f_5 decreases as the number of epochs increases. f_i $(1 \leq i \leq 4)$ strikingly outperforms f_5, as shown in Figure 3(d). We can see that the knowledge embedded in $\mathbf{GCN}_{cat}$ and $\mathbf{GCN}_{prop}$ is beneficial to entity typing task. At the beginning of training process, supervised loss function plays the leading role. After several epochs, $\lambda(t)$ forces our model to simultaneously consider the knowledge encoded in $\mathbf{GCN}_{cat}$ and $\mathbf{GCN}_{prop}$.

5 Related Work

Fine-grained entity typing in $\mathcal{KB}$ is an important sub-task of knowledge base completion. Most ex-

Table 3: Typing performance on DBpedia.

Methods	DB-1			DB-2			DB-3		
	Strict	Ma-F1	Mi-F1	Strict	Ma-F1	Mi-F1	Strict	Ma-F1	Mi-F1
CUTE	0.518	0.679	0.702	0.520	0.681	0.713	0.524	0.685	0.717
MuLR	0.608	0.748	0.771	0.612	0.757	0.784	0.610	0.752	0.775
FIGMENT	0.601	0.740	0.766	0.597	0.738	0.765	0.605	0.745	0.769
APE	0.612	0.758	0.784	0.614	0.761	0.785	0.611	0.760	0.782
HMGCN$_{no\_cat}$	0.619	0.769	0.792	0.622	0.772	0.796	0.620	0.770	0.793
HMGCN$_{no\_prop}$	0.621	0.771	0.793	0.625	0.782	0.801	0.623	0.775	0.799
HMGCN$_{no\_co}$	0.627	0.789	0.808	0.629	0.790	0.814	0.626	0.786	0.812
HMGCN$_{flat}$	0.626	0.785	0.812	0.633	0.794	0.820	0.628	0.791	0.817
HMGCN	**0.635**	**0.794**	**0.816**	**0.637**	**0.796**	**0.824**	**0.631**	**0.798**	**0.819**



(a) Label proportion (b) Frequent/infrequent types (c) $\lambda(t)$ implementations (d) $\lambda(t)$ performances

Figure 3: (a). Performance on different labeled data proportion. (b). Performance on frequent and infrequent types in FIGER dataset (c). Five different implementations of the weight function. X-axis represents the function variable t and the Y-axis represents the function value. (d). Classification accuracy using different $\lambda(t)$ functions for training on FIGER dataset.

Table 4: Typing performance on FIGER.

Methods	Strict	Ma-F1	Mi-F1
CUTE	0.531	0.743	0.782
MuLR	0.548	0.776	0.812
FIGMENT	0.563	0.785	0.819
APE	0.515	0.722	0.756
HMGCN$_{no\_cat}$	0.546	0.773	0.805
HMGCN$_{no\_prop}$	0.552	0.781	0.814
HMGCN$_{no\_co}$	0.557	0.784	0.821
HMGCN$_{flat}$	0.564	0.789	0.827
HMGCN	**0.570**	**0.798**	**0.836**

isting methods do type inference by utilizing entity's inherent feature (textual description, property and category) or mention in text (anchor text). (Neelakantan and Chang, 2015) use text description as entity feature representation, then design a global objective function to predict missing entity types in a $\mathcal{KB}$. (Xu et al., 2016) use property and category information, along with a multi-label hierarchical classifier, to assign DBpedia types to Chinese entities. (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) (Yaghoobzadeh and Schütze, 2017). FIGMENT-Multi predicts whether an entity is a member of a type based on the learned embeddings. 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). Recent research convert entities in $\mathcal{KB}$ to an entity graph and apply graph-based algorithm on it. (Jin et al., 2018) use links between entities to construct an entity graph, jointly utilize entity feature and graph structure to make type inference. They apply an attributed and predictive network embedding method to encode entity feature and graph structure. After that, a multi-label classifier is employed to carry out type classification.

6 Conclusion

We convert the task of Fine-Grained Entity Typing in $\mathcal{KB}$ to graph-based semi-supervised classification task, and propose a hierarchical multi graph

convolutional network (**HMGCN**) that fully utilizes entity features, entity graph structure, and type hierarchy structure to address this task. Three GCN models are devised to embed multi kinds of semantic correlations between entities. A recursive regularization is proposed to make the model understand, to a certain extent, the structure of type hierarchy. Experiments on two real datasets demonstrate the effectiveness of the proposed model.

7 Acknowledgements

This work is supported by National Key Research and Development Program of China (2017YFB1002101), NSFC key projects (U1736204, 6153301), a research fund by Alibaba, and THUNUS NExT Co-Lab.

References

James Atwood and Don Towsley. 2016. Diffusion-convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1993–2001.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in neural information processing systems*, pages 3844–3852.

Siddharth Gopal and Yiming Yang. 2013. Recursive regularization for large-scale classification with hierarchical and graphical dependencies. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 257–265. ACM.

Xiao Huang, Jundong Li, and Xia Hu. 2017. Label informed attributed network embedding. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, pages 731–739. ACM.

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 the 27th International Conference on Computational Linguistics*, pages 282–292.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

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

Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

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.

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.

Hao Peng, Jianxin Li, Yu He, Yaopeng Liu, Mengjiao Bao, Lihong Wang, Yangqiu Song, and Qiang Yang. 2018. Large-scale hierarchical text classification with recursively regularized deep graph-cnn. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*, pages 1063–1072. International World Wide Web Conferences Steering Committee.

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. 2016. 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.

Zhichun Wang, Qingsong Lv, Xiaohan Lan, and Yu Zhang. 2018. Cross-lingual knowledge graph alignment via graph convolutional networks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 349–357.

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

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*.

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

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.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657.

Chenyi Zhuang and Qiang Ma. 2018. Dual graph convolutional networks for graph-based semi-supervised classification. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*, pages 499–508. International World Wide Web Conferences Steering Committee.