Learning from Sibling Mentions with Scalable Graph Inference in Fine-Grained Entity Typing

Yi Chen[1,3]*, **Jiayang Cheng**[2]*, **Haiyun Jiang**†, **Lemao Liu**
Haisong Zhang, Shuming Shi, Ruifeng Xu[1,4]†
[1]School of Computer Science and Technology,
Harbin Institute of Technology, Shenzhen, China
[2]Department of Computer Science and Engineering, HKUST
[3]Joint Lab of HITSZ and China Merchants Securities, Shenzhen, China
[4]Peng Cheng Laboratory, Shenzhen, China
yichennlp@gmail.com, jchengaj@cse.ust.hk
xuruifeng@hit.edu.cn

Abstract

In this paper, we firstly empirically find that existing models struggle to handle hard mentions due to their insufficient contexts, which consequently limits their overall typing performance. To this end, we propose to exploit sibling mentions for enhancing the mention representations. Specifically, we present two different metrics for sibling selection and employ an attentive graph neural network to aggregate information from sibling mentions. The proposed graph model is scalable in that unseen test mentions are allowed to be added as new nodes for inference. Exhaustive experiments demonstrate the effectiveness of our sibling learning strategy, where our model outperforms ten strong baselines. Moreover, our experiments indeed prove the superiority of sibling mentions in helping clarify the types for hard mentions.

1 Introduction

Fine-Grained Entity Typing (FGET) aims to assign one or more fine-grained types to an entity mention given its context. For instance, the mention *Steve Jobs* should be classified as *Person* and *Entrepreneur* under the context "Steve Jobs cofounded Apple ...". Many tasks have witnessed the importance of FGET, such as relation extraction (Jiang et al., 2020b; Chu et al., 2020; Jiang et al., 2020a; Cheng et al., 2021), entity linking (Onoe and Durrett, 2020), and other tasks (Jiang et al., 2020c; Zhang et al., 2020b; Liu et al., 2021b).

It is challenging to learn effective representations for contextualized mentions[1] in many information extraction tasks (Gao et al., 2022), especially in FGET, since the representations are required to well distinguish fine-grained types with similar but different semantics. Noticeable efforts have been made to learn type-aware representations for mentions (Ren et al., 2016; Xin et al., 2018; Choi et al., 2018; Zhang et al., 2018; Lin and Ji, 2019; Abhishek et al., 2017; Xu and Barbosa, 2018; Ali et al., 2020; Chen et al., 2021) and significant progress has been achieved. However, as supported by our empirical experiments, existing SOTA models perform poorly on a certain number of "*hard*" mentions, leading to limited overall performance. The main reasons are the following challenges. First, the structure of some contexts surrounding the hard mentions are inherently too complex to extract informative features for identifying entity types. Second, the contexts of some hard mentions are ambiguous and thus it is insufficient to handle these mentions by learning from their contexts only.

In this paper, we show that representation learning of such hard mentions can be well handled by learning informative knowledge from their *sibling mentions*. Sibling mentions refer to the mentions that *potentially share the same or semantically similar types* (e.g., *country* and *nation*) with the target mention. We illustrate how sibling mentions assist classifying hard mentions in Figure 1. Intuitively, the context of the target mention *Sharp* is ambiguous and insufficient for inferring the ground-truth types (i.e., *organization*, *company*, and *tech company*), since both a *person* and a *company* can "sign a deal with Qualcomm". Fortunately, the sibling mentions provide rich information that works as an important supplement for the target mention *Sharp*. By aggregating the supplementary information from siblings, it is promising to learn effective representations with less ambiguity for hard target mentions.

*Equal Contribution
†Corresponding Authors
[1]To simplify the statement, in the rest of this paper, the term "mention" is referred to as the *contextualized* mention, i.e., a mention accompanied with its context.

Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics
Volume 1: Long Papers, pages 2076 - 2087

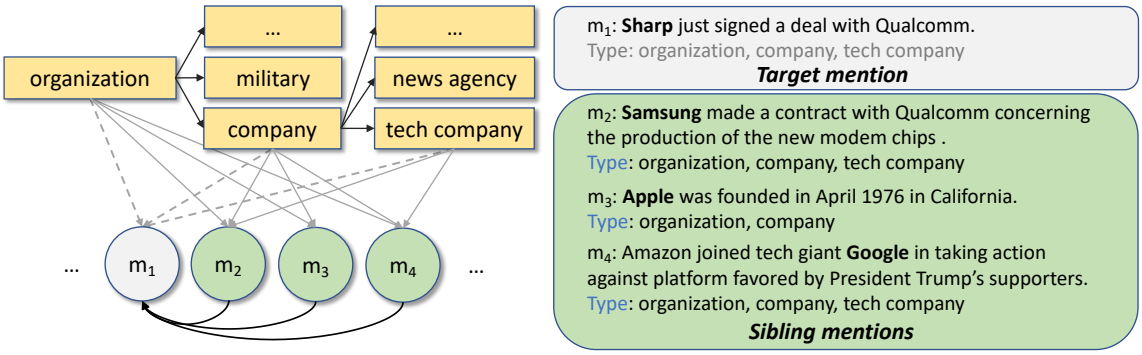

Figure 1: Illustration of the proposed sibling enhanced heterogeneous graph model.

To utilize sibling mentions, we model FGET as a heterogeneous graph learning problem. The graph is composed of two kinds of nodes, namely the mentions and the types. Besides, there are three kinds of edges connecting the nodes as shown in the left part of Figure 1, which represent the *sibling* relationship between mentions, the *hierarchical* relationship between types, and the *isLabel* relationship between mentions and types, respectively. The sibling relationship is considered as the most important part in our graph. For detecting it, we propose two similarity metrics, based on which we design an effective sibling selection algorithm. Upon the constructed nodes and edges, we employ an attentive graph neural module to learn their representations. Particularly, the representations of mention nodes are enriched by aggregating the information from their sibling and type neighbors. It is also noteworthy that, during inference stage, our graph model is scalable to include the unseen test mentions as new nodes and connect them with their existing sibling mention nodes in the graph to derive reliable representations for predictions.

Extensive experiments are conducted to verify the effectiveness of our model. Our experimental results demonstrate that our model outperforms several strong baselines on the standard test sets with a large margin. Moreover, our model is indeed able to well handle hard mentions with the help from sibling mentions.

We summarize our contributions as follows:

- We are the first to point out a bottleneck issue suffered by existing SOTA models, i.e., they perform poorly on a certain number of hard mentions, and we quantitatively analyze its influence on typing accuracy via measuring hard mentions by entropy.
- We are the first to exploit sibling information for mention representation learning in FGET. We design two effective metrics for sibling detection and propose a scalable graph model to take advantages of sibling mentions.

2 Methodology Overview

Given a mention m and the type set $\mathcal{Y}$, an FGET model needs to predict the correct types Y_m ($Y_m \subset \mathcal{Y}$) for m based on its context.

In this paper, mention representations are learned and refined with the help of sibling mentions and ground-truth types. To achieve it, we propose a heterogeneous graph model enhanced by sibling mentions for FGET, as illustrated in Figure 1. First, a mention-type graph $\mathcal{G}$ is constructed from training samples (Sec 3). Then, the features for mentions and types are learned by an attentive graph neural module upon $\mathcal{G}$ (Sec 4).

During inference stage (Sec 5), we add test mentions into graph $\mathcal{G}$ by connecting them to their sibling mentions in the training set. By aggregating sibling information, the representations of test mentions are generated and used for type prediction.

3 Graph Construction

3.1 Graph Definition

Consider graph $\mathcal{G} = (\mathcal{V}_m, \mathcal{V}_y, \mathcal{E}_m, \mathcal{E}_y, \mathcal{E}_{m,y})$, where $\mathcal{V}_m$ and $\mathcal{V}_y$ are the set of mention nodes and type nodes, respectively. $\mathcal{E}_m$ is the set of edges between the target mentions and their sibling mentions, while $\mathcal{E}_y$ is the set of edges between types. $\mathcal{E}_{m,y}$ denotes the edges connecting the target mentions and the ground-truth types. $\mathcal{E}_m$, $\mathcal{E}_y$ and $\mathcal{E}_{m,y}$ are obtained as follows:

$$\mathcal{E}_m = \{(m_i, m_j) | m_i, m_j \in \mathcal{V}_m, \\ isSib(m_i, m_j) = 1\} \quad (1)$$

$$\mathcal{E}_y = \{(y_i, y_j) | y_i, y_j \in \mathcal{V}_y, isA(y_i, y_j) = 1\} \quad (2)$$

$$\mathcal{E}_{m,y} = \{(m_i, y_j) | m_i \in \mathcal{V}_m,\ y_j \in \mathcal{V}_y, \\ isLabel(m_i, y_j) = 1\} \quad (3)$$

where $isA(y_i, y_j) = 1$ indicates y_j is the parent or child type of y_i in the type hierarchy [2], and $isLabel(m_i, y_j) = 1$ means mention m_i is labeled with the type y_j in the training set. Since type hierarchy and the ground-truth types of mentions are available in the training set, $\mathcal{V}_m$, $\mathcal{V}_y$, $\mathcal{E}_y$ and $\mathcal{E}_{m,y}$ can be easily derived. $isSib(m_i, m_j) = 1$ means m_j is the sibling mention of m_i, which will be discussed in Sec 3.2.

3.2 Sibling Selection

The key to construct $\mathcal{E}_m$ is to define $isSib(m_i, m_j)$, i.e., the criterion for judging whether m_j is the sibling of m_i. We design two metrics to detect the sibling relationships between mentions, named (unsupervised) *word distribution-based* and (supervised) *typing distribution-based* metrics.

Word distribution-based metric The basic assumption for this metric is that mentions sharing more contextual words tend to have more similar ground-truth types. We use TF-IDF to encode mentions as sparse feature vectors. Then the sibling similarity between any two mentions is measured by the cosine similarity of their vectors.

Typing distribution-based metric In this metric, we first derive the prior score distributions over the type set $\mathcal{Y}$ for all the mentions in the dataset from an extra base model (Lin and Ji, 2019) trained on the same dataset. Then the sibling mentions are selected by their cosine similarities to the target mention based on the score distributions.

Sibling mention selection Given one of the metrics above, we obtain the sibling mentions according to Algorithm 1. Note that for each target mention $m_i \in \mathcal{V}_m$, we first select a subset $\mathcal{V}'_m$ from $\mathcal{V}_m$ and only calculate the similarities between m_i and the mentions $m_j \in \mathcal{V}'_m$. The contexts of mentions from $\mathcal{V}'_m$ share at least one word with that of the target mention and $|\mathcal{V}'_m| \ll |\mathcal{V}_m|$, which greatly reduces time complexity. Then, based on the similarity scores, we choose the top-K most similar mentions $\mathcal{V}'_{m,K}$ as the siblings for m_i and let $isSib(m_i, m_j) = 1$ for each $m_j \in \mathcal{V}'_{m,K}$. Be aware that, by definition, the sibling relationship is directed, i.e., $isSib(m_j, m_i) = 1$ does not ensure $isSib(m_i, m_j) = 1$ holds.

[2] The edges between y_i and its parent or child type y_j are directed, as detailed in Eq.(5)

Algorithm 1: Sibling mention selection

Input : the set of mention nodes $\mathcal{V}_m$

```
for m_i, m_j ∈ V_m do
    isSib(m_i, m_j) ← 0
end
for m_i ∈ V_m do
    ▷ select a candidate set V'_m from V_m
    for m_j ∈ V'_m do
        ▷ compute similarity sim(m_i, m_j)
    end
    ▷ select the top-K similar mentions
      V'_{m,K} from V'_m
    for m_j ∈ V'_{m,K} do
        isSib(m_i, m_j) ← 1
    end
end
```

4 Graph-based Typing Model

4.1 Attentive Graph Neural Module

We employs graph neural networks (GNNs) with L layers (Velickovic et al., 2018; Xu et al., 2019) to aggregate the information of sibling mentions and types for learning mention representations. At the first layer of $\mathcal{G}$, the embedding of each type $y_i \in \mathcal{Y}$ (denoted by $\boldsymbol{y}_i^{(1)} \in \mathbb{R}^{d_r}$) is randomly initialized. In contrast, to capture the rich features from contexts, the initial embeddings for mentions are derived by a parameterized mention encoder $g(\cdot)$, i.e., $\boldsymbol{m}_i^{(1)} = g(m_i; \theta_{\mathrm{M}}) \in \mathbb{R}^{d_r}$ (details in Sec 4.2). Given the initial mention and type embeddings (i.e., $\boldsymbol{m}_i^{(1)}$ and $\boldsymbol{y}_i^{(1)}$), the graph module iteratively updates them to obtain $\boldsymbol{m}_i^{(l+1)}$ and $\boldsymbol{y}_i^{(l+1)}$.

Update of $\boldsymbol{y}_i^{(l+1)}$ In the l-th ($l = 1, ..., L-1$) layer, the updating formula for type embedding $\boldsymbol{y}_i^{(l+1)} \in \mathbb{R}^{d_r}$ is:

$$\boldsymbol{y}_i^{(l+1)} = f_0\left(\sum_{y_k \in \mathcal{Y}_{y_i}} \alpha_{i,k}^{(l)} f_1(\boldsymbol{y}_k^{(l)}) + f_1(\boldsymbol{y}_i^{(l)})\right), \quad (4)$$

where f_0 and f_1 are linear layers with *ReLU* activation. $\mathcal{Y}_{y_i}$ denotes the type neighbors for y_i in graph $\mathcal{G}$, which are the parent or child types of y_i in the type hierarchy. $\alpha_{i,k}^{(l)}$ is the attention weight from type y_i to y_k defined as

$$\alpha_{i,k}^{(l)} = \begin{cases} \sigma\left(\boldsymbol{y}_i^{(l)\top} \boldsymbol{W}_1^{(l)} \boldsymbol{y}_k^{(l)}\right), & y_k \text{ is a child type;} \\ \sigma\left(\boldsymbol{y}_k^{(l)\top} \boldsymbol{W}_1^{(l)} \boldsymbol{y}_i^{(l)}\right), & y_k \text{ is a parent type,} \end{cases} \quad (5)$$

$\boldsymbol{W}_1^{(l)} \in \mathbb{R}^{d_r \times d_r}$ is the weight matrix to model the parent-child relationship. Note that Eq.(4) does not involve mention embeddings and only focuses on learning the hierarchical structure of types. The interaction between types and mentions will be modeled by Eq.(6) during the update process of mention embeddings.

Update of $\boldsymbol{m}_i^{(l+1)}$ The updating formula for the mention embedding $\boldsymbol{m}_i^{(l+1)} \in \mathbb{R}^{d_r}$ is:

$$\boldsymbol{m}_i^{(l+1)} = f_2\Bigg(\sum_{m_j \in \mathcal{M}_{m_i}} \mu_{i,j}^{(l)} f_3(\boldsymbol{m}_j^{(l)}) + \sum_{y_k \in \mathcal{Y}_{m_i}} \nu_{i,k}^{(l)} f_4(\boldsymbol{y}_k^{(l)})\Bigg) \tag{6}$$

where $\mathcal{M}_{m_i}$ and $\mathcal{Y}_{m_i}$ are the sibling and type neighbors[3] of m_i in graph $\mathcal{G}$. $\mu_{i,j}^{(l)}$ and $\nu_{i,k}^{(l)}$ are the attention weights from m_i to mention m_j and type y_k in the l-th layer, respectively. Specifically,

$$\begin{aligned} \mu_{i,j}^{(l)} &= \sigma\big(\boldsymbol{m}_i^{(l)\top} \boldsymbol{W}_2^{(l)} \boldsymbol{m}_j^{(l)}\big), \\ \nu_{i,k}^{(l)} &= \sigma\big(\boldsymbol{m}_i^{(l)\top} \boldsymbol{W}_3^{(l)} \boldsymbol{y}_k^{(l)}\big), \end{aligned} \tag{7}$$

$\boldsymbol{W}_2^{(l)}, \boldsymbol{W}_3^{(l)} \in \mathbb{R}^{d_r \times d_r}$ are learnable parameters. f_2, f_3, f_4 are linear layers with *ReLU* activation.

Here, we use the attention mechanism to distinguish informative neighbors. Besides, the update process of target mentions involves both the sibling and type neighbors, whose representations are also updated at the same. In this way, the learned representations for both mentions and types are more consistent and thus more reliable for prediction.

4.2 Mention Encoder $g(m_i; \theta_{\mathrm{M}})$

The mention encoder uses the backbone from Lin and Ji (2019). Given a mention, we first encode the mention span and the surrounding context as the weighted sum of their ELMo (Peters et al., 2018) word representations respectively. Then, the unified feature vector for the mention is derived by concatenating both representations.

4.3 Type Prediction

Given a mention m_i, the predicted score distribution $\boldsymbol{p}_i \in \mathbb{R}^{|\mathcal{Y}|}$ over the type set $\mathcal{Y}$ is computed as:

$$\boldsymbol{p}_i = \sigma\big(\boldsymbol{Y}^{(L)} \boldsymbol{W}_4 \boldsymbol{m}_i^{(L)} + \boldsymbol{W}_5 \boldsymbol{m}_i^{(L)}\big), \tag{8}$$

where $\boldsymbol{Y}^{(L)} = \Big[\boldsymbol{y}_1^{(L)}, \boldsymbol{y}_2^{(L)}, ..., \boldsymbol{y}_{|\mathcal{Y}|}^{(L)}\Big] \in \mathbb{R}^{|\mathcal{Y}| \times d_r}$, $\boldsymbol{y}_i^{(L)}$ and $\boldsymbol{m}_i^{(L)}$ are the type and mention embeddings in the L-th layer in GNN. $\boldsymbol{W}_4 \in \mathbb{R}^{d_r \times d_r}$ and $\boldsymbol{W}_5 \in \mathbb{R}^{|\mathcal{Y}| \times d_r}$ are learnable parameters. $\boldsymbol{p}_i[k]$ (the k-th element in $\boldsymbol{p}_i$) denotes the predicted probability for type y_k.

4.4 Loss Function

The loss over m_i is computed as:

$$\ell_i = -\sum_{k=1}^{|\mathcal{Y}|} \big(\delta_k^i \log \boldsymbol{p}_i[k] + (1-\delta_k^i) \log(1-\boldsymbol{p}_i[k])\big) \tag{9}$$

where $\delta_k^i \in \{0, 1\}$ indicates whether y_k is the ground-truth type of m_i in the training set. The overall loss is the average over all the mentions, i.e., $\mathcal{L} = \frac{1}{|\mathcal{V}_m|} \sum_i \ell_i$.

4.5 Dropout of $\mathcal{Y}_{m_i}$

The representation $\boldsymbol{m}_i^{(L)}$ incorporates the information from ground-truth type neighbors (Eq.(6)). However, it is then used for predicting the ground-truth types in turn (Eq.(8)). The setting that $\mathcal{Y}_{m_i}$ contains *all* the ground-truth types will inevitably degenerate the model to just focus on the type neighbors while totally ignore the mention neighbors. To overcome this, each neighboring type in $\mathcal{Y}_{m_i}$ is randomly discarded with a certain probability γ. In this way, the prediction of discarded type will force the model to learn from the sibling mentions rather than directly from type neighbors.

5 Scalable Testing

In the following, we describe the prediction process for test mentions.

Step 1: Given a batch of n test mentions, we first obtain their sibling mentions. To be specific, for each test mention m_t, we select a candidate set $\mathcal{V}_m^{'}$ from the training mentions $\mathcal{V}_m$. Then, the cosine similarity is computed between m_t and each m_i in $\mathcal{V}_m^{'}$, based on which the top K mentions are selected as siblings (see Sec 3.2).

Step 2: We add the test mentions as nodes into the mention-type graph $\mathcal{G}$, where the test mentions are connected to their sibling mentions selected at Step 1. Note that, in the new graph, test mentions have no type neighbors since their ground-truth types are not available. Besides, there is no edge between any two test mentions in the new graph.

Step 3: Following Eq.(6), the representations of test mentions $\{m_t\}$ are updated by aggregating the embeddings for their sibling mentions. Note that $\mathcal{Y}_{m_t}$ is empty, so no information from the ground-truth types are involved. Through layers of updates, the final representations $\{\boldsymbol{m}_t^{(L)}\}$ are obtained.

[3] We define that $\mathcal{M}_{m_i}$ contains m_i itself, thus the self-connections are taken into account during graph learning.

Step 4: Based on the mention embedding $\boldsymbol{m}_t^{(L)}$ and the type embeddings $\boldsymbol{Y}^{(L)}$, we predict the type score distribution for m_t by Eq.(8).

We conclude that, (1) *our graph module is scalable* to add arbitrary number of unseen test mentions as new nodes to the existing graph to derive their representations. By contrast, many popular graph settings (Kipf and Welling, 2017; Velickovic et al., 2018; Wang et al., 2019) fail to extend to new nodes. (2) Since the embeddings for sibling mentions have been well learnt during training, the only need is to compute the embeddings for test mentions for prediction, which are derived simultaneously during graph inference with high efficiency.

6 Experiments

6.1 Datasets

We evaluate the proposed model on two widely-used datasets: OntoNotes and BBN.

OntoNotes The original OntoNotes dataset is annotated by distant supervision (Gillick et al., 2014). The training, development and test samples in OntoNotes are about 251K, 2K and 9K, respectively. We also conduct experiments on the augmented version[4] (Choi et al., 2018) with 793K training samples[5]. The above two versions share the same test set and development set, as well as the same type set of size 89.

BBN Different from OntoNotes, BBN is manually annotated (Weischedel and Brunstein, 2005). The training, development and test set contain about 84K, 2K and 14K samples respectively, and the type set contains 47 type in total.

6.2 Experimental Setup

Our model is implemented based on the PyTorch Geometric package (Fey and Lenssen, 2019). In the main experiments (Sec 6.4), we obtain the sibling mentions according to the typing distribution-based metric described in Sec 3.2. We conduct hyper-parameter search on the development set and the optimal settings are presented in Appendix A.

Following the previous works (Ling and Weld, 2012; Ren et al., 2016; Chen et al., 2019), we report the performance in terms of strict accuracy (Acc), macro-average F1 score (Ma-F1) and micro-average F1 score (Mi-F1). To guarantee the reliability, we repeat the experiment three times under each setting, and report the average results.

[4]http://nlp.cs.washington.edu/entity_type

[5]We use the open-sourced version, which is a subset of the dataset reported in Choi et al. (2018).

6.3 Baselines

We compare our proposed model with several state-of-the-art FGET models: (1) AFET (Ren et al., 2016); (2) AAA (Abhishek et al., 2017); (3) NFETC (Xu and Barbosa, 2018); (4) NEURAL (Shimaoka et al., 2017); (5) ACT (Zhang et al., 2018); (6) Lin and Ji (2019); (7) Chen et al. (2020); (8) LABELGCN (Xiong et al., 2019); (9) Choi et al. (2018); (10) Ali et al. (2020). Note that Lin and Ji (2019) is considered as an important baseline in our experiments and is marked with ★ in Table 1-3, since we use it as the base model to derive the prior typing distributions for sibling selection (Sec 3.2).

6.4 Results and Analysis

Table 1, 2 and 3 illustrate the experimental results on the original and the augmented OntoNotes, as well as the BBN dataset.

Analysis The results demonstrate that learning from sibling mentions helps our model outperform most baselines across the benchmarks. The detailed analysis is presented as follows:

(1) We select sibling mentions according to the typing distribution from Lin and Ji (2019). We observe that, after aggregating sibling information through the attentive graph neural module (Sec 4.1), our model significantly outperforms Lin and Ji (2019) on both the original OntoNotes and the BBN dataset. When trained on the augmented OntoNotes of the same size, our model increases the accuracy score by more than 5% over Lin and Ji (2019)★. Compared with Lin and Ji (2019)* which utilizes the full 3M augmented OntoNotes for training, our model still maintains a comparable performance and even improves the accuracy score by about 2%.

(2) Many previous works have demonstrated the effectiveness of modeling type hierarchy for entity typing (Ren et al., 2016; Xu and Barbosa, 2018; Xiong et al., 2019; Chen et al., 2020). As a comparison, our model also considers the hierarchical information of types and incorporates it in a natural way (Sec 4.1). From the results, we conclude that learning jointly from type hierarchy and sibling mentions can remarkably improve the typing performance.

(3) The attention mechanism plays an important role in our graph module and some of the baselines (Ren et al., 2016; Abhishek et al., 2017; Xu and

Barbosa, 2018). It not only helps identify the informative features from neighbors but also helps alleviate noise from the training data constructed by distant supervision (e.g., OntoNotes). The results reveal that our graph-based solution is more effective than the existing solutions.

Model	Acc	Ma-F1	Mi-F1
Ren et al. (2016)	55.1	71.1	64.7
Abhishek et al. (2017)	52.2	68.5	63.3
Shimaoka et al. (2017)	51.7	71.0	64.9
Zhang et al. (2018)	55.5	73.3	67.6
Xu and Barbosa (2018)	54.4	71.5	64.9
Lin and Ji (2019)★	55.4	73.8	68.4
Chen et al. (2020)	58.7	73.0	68.1
Ali et al. (2020)	57.7	74.3	68.5
Our Model	**59.2**	**76.5**	**71.0**

Table 1: Test results on the original OntoNotes dataset.

Model	Acc	Ma-F1	Mi-F1
Choi et al. (2018)*	59.5	76.8	71.8
Lin and Ji (2019)*	63.8	**82.9**	77.3
Xiong et al. (2019)	59.6	77.8	72.2
Lin and Ji (2019)★	60.3	81.6	74.3
Our Model	**65.7**	82.4	**77.4**

Table 2: Test results on the augmented OntoNotes dataset. Note that the baselines with "*" employ the full version (about 3M training samples) of augmented OntoNotes built from the licensed Gigaword (Choi et al., 2018), while the rest only uses the open-sourced subset (i.e., 793K training samples) of it, which may downgrade the performance.

Model	Acc	Ma-F1	Mi-F1
Ren et al. (2016)	67.0	72.7	73.5
Abhishek et al. (2017)	60.4	74.1	75.7
Abhishek et al. (2017)*	**73.3**	79.1	79.2
Shimaoka et al. (2017)	64.7	76.5	71.5
Zhang et al. (2018)	60.1	77.8	76.9
Xu and Barbosa (2018)	72.1	77.1	77.5
Lin and Ji (2019)★	59.9	82.9	81.7
Chen et al. (2020)	55.9	79.3	78.1
Ali et al. (2020)	70.3	81.9	82.3
Our Model	72.2	**85.9**	**86.0**

Table 3: Test results on the BBN dataset. Note that Abhishek et al. (2017)* uses the feature representations learnt from an extra dataset, FIGER (Lin et al., 2012), which results in the higher accuracy score on BBN.

6.5 Ablation Studies

6.5.1 Effect of the sibling selection metrics

In Sec 3.2, we propose two similarity metrics to discover sibling relationships in graph $\mathcal{G}$, and abbreviate them as: "Word-based" and "Typing-based" metrics. Here, we provide two additional metrics for more detailed analysis: the "Gold typing-based" and the "Random-based" metrics, which are two extreme variations of the typing-based metrics. Under the gold typing-based metric, the siblings are selected by the gold typing distribution, where each dimension is 0 or 1 according to the ground-truth types of the mention. In this way, candidate mentions that share more ground-truth types with the target mention will have larger cosine similarity and thus be chosen as the siblings with a higher probability. On the contrary, under the random-based metric, siblings are selected at random. Since the type set is large, the siblings are more likely to be irrelevant with the target mention and may contain much noise.

Measuring sibling quality Intuitively, different similarity metrics will affect the quality of siblings. To quantify this effect, we measure the sibling quality for the test mentions $\mathcal{V}'$ in the original OntoNotes and define the metrics as follows.

For each mention $m_i \in \mathcal{V}'$, denote its ground-truth types as $\mathcal{Y}_{m_i}$ and sibling mentions in graph $\mathcal{G}$ (defined in Sec 3.1) as $\mathcal{M}_{m_i}$. Further, for $\mathcal{M}_{m_i}$, we denote their ground-truth types as $\mathcal{Y}_{\mathcal{M}_i}$, i.e.,

$$\mathcal{Y}_{\mathcal{M}_i} = \bigcup_{m_j \in \mathcal{M}_{m_i} \setminus \{m_i\}} \mathcal{Y}_{m_j}. \tag{10}$$

Similar to the definitions of Precision, Recall and F1, we define *Purity, Coverage* and *Quality* to measure the sibling quality of $\mathcal{V}'$:

$$\begin{aligned} \text{Purity} &= \frac{1}{|\mathcal{V}'|} \sum_{m_i \in \mathcal{V}'} = \frac{|\mathcal{Y}_{m_i} \cap \mathcal{Y}_{\mathcal{M}_i}|}{|\mathcal{Y}_{\mathcal{M}_i}|} \\ \text{Coverage} &= \frac{1}{|\mathcal{V}'|} \sum_{m_i \in \mathcal{V}'} \frac{|\mathcal{Y}_{m_i} \cap \mathcal{Y}_{\mathcal{M}_i}|}{|\mathcal{Y}_{m_i}|} \\ \text{Quality} &= \frac{2 * \text{Coverage} * \text{Purity}}{\text{Coverage} + \text{Purity}} \end{aligned} \tag{11}$$

Results The results are presented in Table 4. In general, the model performance is closely related to the sibling quality. Besides, the typing-based metric performs better than the word-based metric. This indicates that the the continuous type-level probability distribution is more reliable for sibling selection than the discrete word-level distribution. The scores from the gold typing-based and the random-based metrics reveal the upper bound and the lower bound of the scores for the typing-based metric. On the one hand, the quality of the siblings selected by the gold typing-based metric is much higher than those by other methods, with the Coverage up to 97.6%. Meanwhile, its corresponding model also outperforms the other three by a large margin. Note that the typing performance in this scenario is limited by the annotation

Metrics	Purity	Coverage	Quality	Macro			Micro		
				P	R	F1	P	R	F1
Random-based	9.6	71.3	16.9	65.1	56.4	60.1	65.1	43.5	52.2
Word-based	12.2	75.6	21.0	82.7	69.7	75.3	82.1	60.4	69.7
Typing-based	13.1	82.3	22.5	83.3	71.4	76.5	83.3	61.9	71.0
Gold typing-based	21.8	97.6	35.7	96.1	80.0	86.9	94.9	69.2	80.0

Table 4: Comparison among different sibling selection metrics. Note that the sibling quality scores of "Gold typing-based" do not reach 1, since the sibling mentions are only selected from a subset $\mathcal{V}'_m$ (see Sec 3.2) for time efficiency, which are not guaranteed to have exactly the same ground-truth types with the target mention.

K	Purity	Coverage	Quality	Macro			Micro		
				P	R	F1	P	R	F1
0	-	-	-	81.6	68.9	74.4	80.8	60.1	68.9
5	13.1	82.3	22.5	83.3	71.4	76.5	83.3	61.9	71.0
10	10.4	89.5	18.6	82.8	70.2	75.4	81.9	60.5	69.5
15	7.9	94.4	14.5	81.3	68.5	73.8	78.6	61.3	68.8

Table 5: Effect of the sibling size K on sibling quality and typing performance over the original OntoNotes dataset.

quality to some extent. Since OntoNotes is annotated by distant-supervision, the scores for the gold typing-based metric could not reach higher due to the label noise of the siblings. On the other hand, a distinct drop of the scores is observed with the random-based metric. This is reasonable since the randomly selected siblings contain much noisy information, which is helpless and even harmful for typing of the target mention. It can be concluded from the above observations that there is still much room to improve the sibling quality as well as the typing performance of the sibling-enhanced model.

6.5.2 Effect of sibling size K

The model performance is sensitive to the size of selected sibling mentions for a target mention in the graph $\mathcal{G}$. Denote the sibling size as K, following the default hyper-parameter settings, we train our model on the original OntoNotes under different $K \in \{0, 5, 10, 15\}$ using the typing-based sibling selection metric. The corresponding sibling quality and model performance are reported in Table 5. We observe that the best scores are obtained with the top 5 sibling mentions. When $K = 0$, the graph only contains the self-connections from the target mentions to themselves. Without the additional information from siblings, the Macro F1 score decreases by 2.1%, which indicates the effectiveness of sibling mentions for improving the typing performance of our model. When $K \neq 0$, the Coverage score goes up while the Purity and Quality scores go down as K ranges from 5 to 15. Meanwhile, the typing performance decreases as K increases. It suggests that, for OntoNotes, a properly smaller sibling size is a trade-off choice for the model to use siblings with higher quality and thus achieve better typing performance.

6.5.3 Effect of dropout probability γ

We randomly discard some type neighbors with a dropout probability γ during training (Sec 4.5), which forces the model to learn from the sibling mentions other than the ground-truth types. Table 6 shows the results under different values of γ on the original OntoNotes dataset. Generally, the model achieves better performance with larger γ. This indicates discarding a large proportion of ground-truth types is beneficial for learning from sibling mentions. Besides, it also narrows the difference between training and test settings where the test mentions do not have ground-truth types as neighbors. The best performance is achieved when γ equals around 0.7. However, dropping all the type neighbors (i.e., $\gamma = 1$) will block the interaction between the type and mention representations in the graph, which may slightly damage the performance.

γ	Macro			Micro		
	P	R	F1	P	R	F1
0.0	83.1	70.7	76.1	82.1	62.2	70.7
0.3	83.0	70.9	76.1	81.7	62.3	70.7
0.5	83.0	71.3	76.4	81.6	63.0	71.1
0.7	83.3	71.4	76.5	83.3	61.9	71.0
1.0	82.7	71.4	76.3	80.6	63.0	70.8

Table 6: Effect of γ on the performance over the original OntoNotes dataset.

6.6 How sibling mentions work in FGET

6.6.1 Quantifying the hard mentions

To select sibling mentions, we first derive the prior typing distribution from the base model (Lin and Ji, 2019) as described in Sec 3.2. During experiments, we observe that the contextual information

for some mentions are insufficient or too complex, which makes the base model confused on these mentions. Entropy measures the uncertainty of a probability distribution. Thus, we quantify the difficulty of mentions by the entropy of their corresponding prior typing distributions and define the mentions with the top-500 highest entropy values as *hard mentions*, which account for about 5% of the *whole mentions*. Table 7 compares the performance of our model and the base model on both the whole mentions and the hard mentions from the test dataset of the original OntoNotes. we see that both models perform worse on the hard mentions than on the whole mentions. Besides, except for the superiority of our model regarding the Acc, Ma-F1 and Mi-F1 scores, it also achieves a lower entropy value than the base model especially on the hard mentions. This indicates the information from siblings makes the output type distributions more concentrated and therefore increases the confidence for model predictions.

Mention	Model	Ep	Acc	Ma-F1	Mi-F1
whole mentions	Ours	2.1	59.2	76.5	71.0
	Base	2.4	55.4	73.8	68.4
hard mentions	Ours	2.5	57.2	73.6	66.6
	Base	3.3	51.0	65.3	58.9

Table 7: Results on the whole mentions and hard mentions of the original OntoNotes. *Base* denotes the base model from Lin and Ji (2019). *Ep* is short for Entropy.

6.6.2 Case Study

To further provide an intuitive understanding about how our model benefits from sibling mentions, we present an example in Table 8. As expected, the retrieved siblings based on the metric defined in Sec 3.2 share similar ground-truth types with the target mention. This verifies the effectiveness of our sibling selection algorithm. Moreover, we observe that the siblings even help predict the correct but out-of-gold-set types for the target mention in this case. Although the annotated types for the target mention [*GM officials*] only contains */person* in the test set. The sibling mentions still provide a strong evidence for our model to also predict */person/title* as a possible type for the target mention.

Target mention: [*GM officials*] told workers late last week of the following moves: production of full-sized vans will be consolidated into a single plant in Flint, Mich. **Ground-truth**: */person* **Prediction from our model**: */person, /person/title*
Sibling 1: "It's been a steadily improving relationship.", says the [*president*]. **Ground-truth**: */person, /person/title*
Sibling 2: Apart from those two actions, Mr.Sikes and the three other [*commissioners*] said they expect to re-examine how AT&T is regulated since competition has increased. **Ground-truth**: */person, /person/title*
Sibling 3: HUD Secretary [*Jack Kemp*] backed an unsuccessful effort to strike such language last week, but received little support from the White House · · · **Ground-truth**: */person, /person/artist, /person/artist/actor, /person/artist/author, /person/political_figure*

Table 8: An example to illustrate the relationship between the target mention and the sibling mentions from the Original OntoNotes.

7 Related Work

FGET is an important task for the downstream NLP tasks and many efforts have been make in improving its performance (Zhang et al., 2020a; Liu et al., 2021a). Early works in FGET (Ling and Weld, 2012; Shimaoka et al., 2016) mainly focus on feature extraction for mentions, which do not consider label noise introduced by distant supervision (Gillick et al., 2014; Choi et al., 2018; Li et al., 2020). Recent years have witnessed an increasing number of researchers being dedicated to data denoising. A popular solution (Ren et al., 2016; Abhishek et al., 2017; Xu and Barbosa, 2018; Ali et al., 2020) is to design loss functions for the clean and noisy parts of the training data separately. Nevertheless, Zhang et al. (2020c) proposes an automatic relabeling framework to estimate the pseudo-truth label distribution of each sample, which treats the noisy and clean data uniformly. Besides, Chen et al. (2019) groups mentions of the same type into a compact cluster to improve the robustness of the model. Ali et al. (2020) refines noisy representations by corpus-level contextual clues. Onoe and Durrett (2019) introduces two additional models to delete the samples that are too noisy to be useful, and repair noisy labels for the retained examples. In addition, there are some notable work which tries to build FGET with limited resources (Qian et al., 2021).

Modeling the type hierarchy is another important topic in FGET. Prior solutions (Shimaoka et al., 2017) introduce a one-hot matrix to encode the hierarchy. Xu and Barbosa (2018) proposes a hierarchy-aware loss function. Recently, graph-based methods have been proven to be powerful in many NLP tasks (Kipf and Welling, 2017; Liang et al., 2021; Xu et al., 2019; Liang et al., 2022). Using graphs

to model the type hierarchy in FGET is a natural idea. Jin et al. (2019) models the potential type correlations for in-knowledge-base entities via hierarchical multi graph convolutional networks (GCNs). Further, Xiong et al. (2019) extends GCNs to a vast number of free-form types. Chen et al. (2020) designs a multi-level learning-to-rank loss to leverage hierarchical information. Recently, Onoe et al. (2021) models the mention and type representations in a box space instead of the traditional vector space.

8 Conclusion

In this paper, we firstly point out that SOTA typing models suffer from a bottleneck issue, i.e., they perform poorly on a certain number of hard mentions, which leads to their limited overall performance. To this end, we propose to exploit sibling information for mention representation learning and define two metrics for detecting sibling relationship between mentions. Further, we model sibling learning as a graph learning problem. Our model is scalable in that, once trained, it can generate sibling-aware representations for previously unseen mentions efficiently during inference stage. Extensive experiments show that the proposed model indeed handles hard mentions well and thereby yields better overall performance than SOTA baseline models.

9 Acknowledgements

This work was partially supported by the National Natural Science Foundation of China (61876053, 62006062, 62176076), the Shenzhen Foundational Research Funding (JCYJ20200109113441941, JCYJ20210324115614039), Joint Lab of Lab of HITSZ and China Merchants Securities.

References

Abhishek Abhishek, Ashish Anand, and Amit Awekar. 2017. Fine-grained entity type classification by jointly learning representations and label embeddings. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 797–807.

Muhammad Asif Ali, Yifang Sun, Bing Li, and Wei Wang. 2020. Fine-grained named entity typing over distantly supervised data based on refined representations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7391–7398.

Bo Chen, Xiaotao Gu, Yufeng Hu, Siliang Tang, Guoping Hu, Yueting Zhuang, and Xiang Ren. 2019. Improving distantly-supervised entity typing with compact latent space clustering. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2862–2872.

Tongfei Chen, Yunmo Chen, and Benjamin Van Durme. 2020. Hierarchical entity typing via multi-level learning to rank. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8465–8475.

Yi Chen, Haiyun Jiang, Lemao Liu, Shuming Shi, Chuang Fan, Min Yang, and Ruifeng Xu. 2021. An empirical study on multiple information sources for zero-shot fine-grained entity typing. In *EMNLP*.

Jiayang Cheng, Haiyun Jiang, Deqing Yang, and Yanghua Xiao. 2021. A question-answering based framework for relation extraction validation. *arXiv preprint arXiv:2104.02934*.

Eunsol Choi, Omer Levy, Yejin Choi, and Luke Zettlemoyer. 2018. Ultra-fine entity typing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 87–96.

Zhendong Chu, Haiyun Jiang, Y. Xiao, and Wei Wang. 2020. Insrl: A multi-view learning framework fusing multiple information sources for distantly-supervised relation extraction. *ArXiv*, abs/2012.09370.

Matthias Fey and Jan E. Lenssen. 2019. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*.

Jun Gao, Wei Wang, Changlong Yu, Huan Zhao, Wilfred Ng, and Ruifeng Xu. 2022. Improving event representation via simultaneous weakly supervised contrastive learning and clustering. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*.

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

Haiyun Jiang, Qiaoben Bao, Qiao Cheng, Deqing Yang, Li Wang, and Yanghua Xiao. 2020a. Complex relation extraction: Challenges and opportunities. *arXiv preprint arXiv:2012.04821*.

Haiyun Jiang, JunTao Liu, Sheng Zhang, Deqing Yang, Yanghua Xiao, and Wei Wang. 2020b. Surface pattern-enhanced relation extraction with global constraints. *Knowledge and Information Systems*, 62(12):4509–4540.

Haiyun Jiang, Yanghua Xiao, and Wei Wang. 2020c. Explaining a bag of words with hierarchical conceptual labels. *World Wide Web*, 23(3):1693–1713.

Hailong Jin, Lei Hou, Juan-Zi Li, and T. Dong. 2019. Fine-grained entity typing via hierarchical multi graph convolutional networks. In *EMNLP/IJCNLP*.

Thomas Kipf and M. Welling. 2017. Semi-supervised classification with graph convolutional networks. *ArXiv*, abs/1609.02907.

Yangming Li, Lemao Liu, and Shuming Shi. 2020. Empirical analysis of unlabeled entity problem in named entity recognition.

Bin Liang, Yonghao Fu, Lin Gui, Min Yang, Jiachen Du, Yulan He, and Ruifeng Xu. 2021. Target-adaptive graph for cross-target stance detection. In *the Web Conference 2021 (WWW '21)*.

Bin Liang, Hang Su, Lin Gui, Erik Cambria, and Ruifeng Xu. 2022. Aspect-based sentiment analysis via affective knowledge enhanced graph convolutional networks. *Knowledge-Based Systems*, 235:107643.

Thomas Lin, Oren Etzioni, et al. 2012. No noun phrase left behind: detecting and typing unlinkable entities. In *Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning*, pages 893–903.

Ying Lin and Heng Ji. 2019. An attentive fine-grained entity typing model with latent type representation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6198–6203.

Xiao Ling and Daniel Weld. 2012. Fine-grained entity recognition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 26.

Lemao Liu, Haisong Zhang, Haiyun Jiang, Yangming Li, Enbo Zhao, Kun Xu, Linfeng Song, Suncong Zheng, Botong Zhou, Dick Zhu, Xiao Feng, Tao Chen, Tao Yang, Dong Yu, Feng Zhang, ZhanHui Kang, and Shuming Shi. 2021a. TexSmart: A system for enhanced natural language understanding. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations*, pages 1–10, Online. Association for Computational Linguistics.

Lemao Liu, Haisong Zhang, Haiyun Jiang, Yangming Li, Enbo Zhao, Kun Xu, Linfeng Song, Suncong Zheng, Botong Zhou, Dick Zhu, et al. 2021b. Texsmart: A system for enhanced natural language understanding. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations*, pages 1–10.

Yasumasa Onoe, Michael Boratko, and Greg Durrett. 2021. Modeling fine-grained entity types with box embeddings. *arXiv preprint arXiv:2101.00345*.

Yasumasa Onoe and Greg Durrett. 2019. Learning to denoise distantly-labeled data for entity typing. In *NAACL-HLT*.

Yasumasa Onoe and Greg Durrett. 2020. Fine-grained entity typing for domain independent entity linking. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8576–8583.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of NAACL-HLT*, pages 2227–2237.

Jing Qian, Yibin Liu, Lemao Liu, Yangming Li, Haiyun Jiang, Haisong Zhang, and Shuming Shi. 2021. Fine-grained entity typing without knowledge base. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5309–5319, Online and Punta Cana, Dominican Republic. 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 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1369–1378.

Sonse Shimaoka, Pontus Stenetorp, Kentaro Inui, and S. Riedel. 2016. An attentive neural architecture for fine-grained entity type classification. In *AKBC@NAACL-HLT*.

Sonse Shimaoka, Pontus Stenetorp, Kentaro Inui, and Sebastian Riedel. 2017. Neural architectures for fine-grained entity type classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 1271–1280.

Petar Velickovic, Guillem Cucurull, A. Casanova, A. Romero, P. Liò, and Yoshua Bengio. 2018. Graph attention networks. *ArXiv*, abs/1710.10903.

X. Wang, Houye Ji, C. Shi, Bai Wang, Peng Cui, P. Yu, and Yanfang Ye. 2019. Heterogeneous graph attention network. *The World Wide Web Conference*.

Ralph Weischedel and Ada Brunstein. 2005. Bbn pronoun coreference and entity type corpus. *Linguistic Data Consortium, Philadelphia*, 112.

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

Wenhan Xiong, Jiawei Wu, Deren Lei, Mo Yu, Shiyu Chang, Xiaoxiao Guo, and William Yang Wang. 2019. Imposing label-relational inductive bias for extremely fine-grained entity typing. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 773–784.

Keyulu Xu, Weihua Hu, J. Leskovec, and S. Jegelka. 2019. How powerful are graph neural networks? *ArXiv*, abs/1810.00826.

Peng Xu and Denilson Barbosa. 2018. Neural fine-grained entity type classification with hierarchy-aware loss. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 16–25.

Haisong Zhang, Lemao Liu, Haiyun Jiang, Yangming Li, Enbo Zhao, Kun Xu, Linfeng Song, Suncong Zheng, Botong Zhou, Jianchen Zhu, Xiao Feng, Tao Chen, Tao Yang, Dong Yu, Feng Zhang, Zhanhui Kang, and Shuming Shi. 2020a. Texsmart: A text understanding system for fine-grained NER and enhanced semantic analysis. *CoRR*, abs/2012.15639.

Haisong Zhang, Lemao Liu, Haiyun Jiang, Yangming Li, Enbo Zhao, Kun Xu, Linfeng Song, Suncong Zheng, Botong Zhou, Jianchen Zhu, et al. 2020b. Texsmart: A text understanding system for fine-grained ner and enhanced semantic analysis. *arXiv preprint arXiv:2012.15639*.

Haoyu Zhang, Dingkun Long, Guangwei Xu, Muhua Zhu, Pengjun Xie, Fei Huang, and Ji Wang. 2020c. Learning with noise: Improving distantly-supervised fine-grained entity typing via automatic relabeling. pages 3808–3815.

Sheng Zhang, Kevin Duh, and Benjamin Van Durme. 2018. Fine-grained entity typing through increased discourse context and adaptive classification thresholds. In *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics*, pages 173–179.

A Hyperparameter Settings

The default hyperparameters for our model are set as follows, where K is mentioned in Sec 3.2, L, γ and d_r are mentioned in Sec 4.

Hyper-parameter	OntoNotes	BBN
K	5	20
L	2	2
γ	0.7	0.9
d_r	2048	2048

Table 9: The default hyper-parameter settings.