Enhancing Label Representations with Relational Inductive Bias Constraint for Fine-Grained Entity Typing

Jinqing Li[1,2] , **Xiaojun Chen**[1*] , **Dakui Wang**[1] and **Yuwei Li**[1,2]
[1]Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
[2] School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
{lijinqing, chenxiaojun,wangdakui,liyuwei}@iie.ac.cn

Abstract

Fine-Grained Entity Typing (FGET) is a task that aims at classifying an entity mention into a wide range of entity label types. Recent researches improve the task performance by imposing the label-relational inductive bias based on the hierarchy of labels or label co-occurrence graph. However, they usually overlook explicit interactions between instances and labels which may limit the capability of label representations. Therefore, we propose a novel method based on a two-phase graph network for the FGET task to enhance the label representations, via imposing the relational inductive biases of instance-to-label and label-to-label. In the phase I, instance features will be introduced into label representations to make the label representations more representative. In the phase II, interactions of labels will capture dependency relationships among them thus make label representations more smooth. During prediction, we introduce a pseudo-label generator for the construction of the two-phase graph. The input instances differ from batch to batch so that the label representations are dynamic. Experiments on three public datasets verify the effectiveness and stability of our proposed method and achieve state-of-the-art results on their testing sets.

1 Introduction

FGET task, which aims at classifying an entity mention into a wide range of entity label types according to its context, plays an important role in many applications, such as coreference resolution [Durrett and Klein, 2014], relation extraction [Yaghoobzadeh *et al.*, 2016], question answering [Yavuz *et al.*, 2016], knowledge base population [Carlson *et al.*, 2010], etc. Since an entity may contain multiple labels in a certain text, the task can be regarded as a multi-label multi-class problem. For example, given the sentence "*Peter is Tom's father who works as an English teacher*" and the target mention "*Peter*", we can infer it belongs to labels of $\{people, male, teacher\}$.

[*]Contact Author

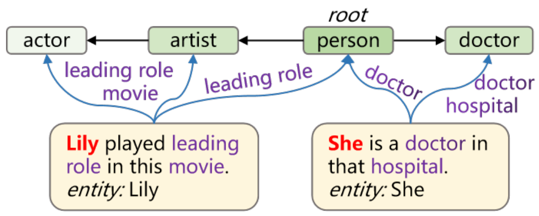

Figure 1: Interactions between instances and labels. The black arrow denotes the label hierarchy and the blue arrow represents the entity contains the labels. The words in purple font directly reflect the characteristics of the corresponding labels.

Recent researches improve the task performance by imposing the label-relational inductive bias based on the hierarchy of labels or label co-occurrence graph, as shown at the top of Figure 1. Some works [Shimaoka *et al.*, 2017; Ren *et al.*, 2016; Xu and Barbosa, 2018] explicitly exploit the inherent hierarchy of labels to share parameters between parent- and sub-labels, or design hierarchy-aware loss functions, while [Chen *et al.*, 2020] employs a coarse-to-fine decoder to search candidate labels on the hierarchy label tree. [Xiong *et al.*, 2019] firstly proposes to build a label co-occurrence graph and then use graph convolution propagation on it, to capture the dependency relationships among labels. Essentially, the graph structure (including hierarchy) is a kind of relational inductive bias in standard deep learning components, which **imposes constraints on relationships and interactions among nodes in a learning process** [Battaglia *et al.*, 2018]. It is beneficial to impose the constraint of label-to-label relational inductive bias to the label optimization representations, as [Xiong *et al.*, 2019] proven. However, the existing works just consider the hierarchy among labels for task optimization, but no literature considers the explicit influence of instance features on label representations. For example, as shown in Figure 1, if we introduce the instance information (like the keyword information in purple font) to label, we can infer the labels of the instance easily. For a label type, its representation should cover as many features of the instances that belong to the label as possible to make the label more representative. Therefore, how to extract the instance features for the label representation is a problem worth considering.

Based on the above observation, we provide a novel method to enhance the label representations by imposing a new relational inductive bias of instance-to-label. Concretely, we implement the above idea via a two-phase graph network with nodes of instances and labels. In phase I, we explicitly introduce the features of instances into the label representations via the instance-label sub-graph and get the instance-aware label representations. Since imposing label relational inductive bias has been proven to be helpful, in phase II, we use the label-label sub-graph to capture dependency relationships among labels and get the dependency-aware label representations. During prediction, the ground truth labels are not available, therefore we design a pseudo-label generator to generate pseudo labels for testing instances. Note that, the input instances differ from batch to batch so that the label representations are dynamic.

Our contributions are summarized as follows:

- We propose to impose graph relational inductive biases of instance-to-label and label-to-label to enhance the label representations. To our best knowledge, we are the first to combine the two biases at the same time.
- We design a two-phase graph network to impose the above relational inductive bias constraints. The graph is divided into two sub-graphs for different representation perspectives: instance-label sub-graph for instance-aware label representations and label-label sub-graph for dependency-aware label representations.
- We demonstrate the effectiveness of the proposed method by comprehensive experiments. The experimental results on three datasets outperform other state-of-the-art methods.

2 Related Work

In this work, we use Graph Neural Networks(GNNs) to enhance label representations under two kinds of graph relational inductive biases for FGET task, so we will introduce the related works of the two aspects.

2.1 Graph Neural Networks

Graphs can be used to represent network structures. [Kipf and Welling, 2017] proposes Graph Convolutional Network(GCN) which performs convolutional operation on graph-structured data. GCN has recently achieved appealing performance in node classification [Kipf and Welling, 2017], recommendation system [Wang *et al.*, 2019b], and so on. It can pass information among neighbor nodes and capture nodes' dependencies. Graph Attention Network(GAT) proposed by [Velickovic *et al.*, 2017] extends the GCN, in which the weights of neighbor nodes are different. [Wang *et al.*, 2019a] recursively propagates the information from neighbors of a node to refine the node's embedding and employs an attention mechanism to discriminate the importance of the neighbors. In our case, different instance features contribute to the label representations differently while GAT can capture the difference.

2.2 Fine-Grained Entity Typing

Entity typing (ET) is a sub-task of name entity recognization [Collins and Singer, 1999; Jiang and Zhai, 2006; Ratinov and Roth, 2009]. The earlier ET task, as a single-label multi-class question, classifies entities roughly into several categories. However, in recent years, finer-grained entity typing datasets are widely raised, which include dozens or hundreds of label types like [Weischedel and Brunstein, 2005; Ling and Weld, 2012], etc. The challenge is how to model the label representations to make similar labels close in their feature space while keeping them away if they are conflicting. [Ren *et al.*, 2016] derives label correlation based on two signals: the given label hierarchy and the shared entities between two labels in KB. [Xu and Barbosa, 2018] proposes a variant of the cross-entropy loss function and introduces hierarchical loss normalization which can understand the label hierarchy and alleviate the negative effect of overly-specific labels, but is difficult to adapt to a non-hierarchical label set. [Chen *et al.*, 2019] applies label propagation (LP) on label graph to estimate their label distribution. [López *et al.*, 2019] exploits hyperbolic embeddings to capture hierarchical relations between mentions in context and their target labels in a shared vector space. [Zhang *et al.*, 2020] exploits probabilistic automatic relabeling module to handling noisy samples in training data based on [Xu and Barbosa, 2018]. [Lin and Ji, 2019] predicts a low-dimensional vector that encodes latent label features and reconstructs the label vector from the latent representation. [Chen *et al.*, 2020] also takes the explicit hierarchy into account, by multi-level learning to rank approach that ranks the candidate labels conditioned on the given entity mention. [Xiong *et al.*, 2019] imposes label-relational inductive bias via GCN propagation on label co-occurrence graph for the first time. Some other works introduce external knowledge like entity linking in [Dai *et al.*, 2019] or use advanced pre-training/fine-tuning methods like [Shi *et al.*, 2020] which proposes a regularization module based on virtual adversarial training, also achieve great improvement. The above methods improve the performance by optimizing the instance representations or label representations separately. Different from them, we will explore the influence of instance features on label representations in this work.

3 Methodology

The proposed method consists of three main components: mention features extractor, instance-aware label encoder, and pseudo-label generator, as shown in Figure 2. The mention features extractor encodes the mention span and its context via attention mechanism to get context-aware mention representation. For the instance-aware label encoder, we design a two-phase heterogeneous graph to extract the learned features from instances to labels and then capture the dependency relationships among labels. During prediction, we employ the pseudo-label generator to generate pseudo labels for the testing instances to assist in constructing the two-phase graph.

3.1 Mention Features Extractor

Given an example $\{m, c, y\}$, where m, c, y denote entity mention span, context, and ground truth labels, the mention features extractor is used to encode the context-aware representation of mention via encoding the mention span and its context respectively. Note that we define $\{m, c\}$ as an instance here.

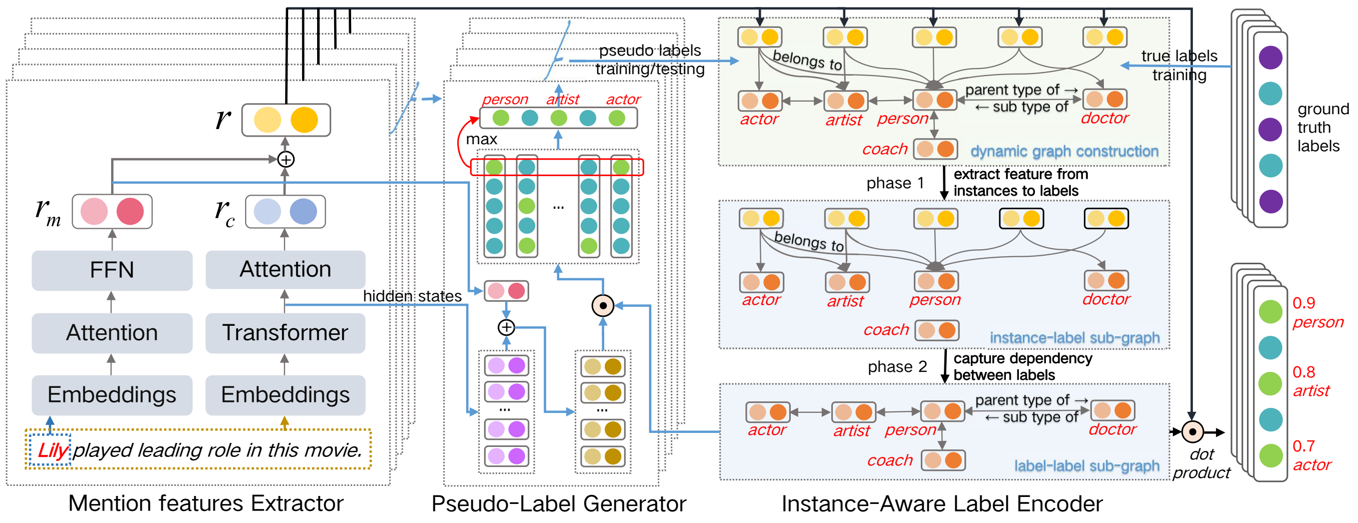

Figure 2: The overview of our proposed method.

Representation of mention. We leverage the attention-weighted sum of mention word embeddings as the semantic representation of the mention span:

$$\boldsymbol{r}_a = Attention(\{\boldsymbol{e}_i\}) \tag{1}$$

$$\boldsymbol{r}_m = tanh(FFN(\boldsymbol{r}_a)) \tag{2}$$

where $Attention(.)$ is an attention-weighted sum operation, $FFN(.)$ is Feed Forward Network, $tanh(.)$ is hyperbolic tangent activation function, and $\boldsymbol{e_i} \in \mathbb{R}^d$ is the embedding of i_{th} word where $i \in [0, 1, ..., L_m]$, L_m is the length of the mention and d is the dimension of word embedding.

Representation of context. To keep the full context information of the mention, we apply a Transformer encoder to encode the whole context sequence. After a feed forward network and activation function, the final representation of context can be calculated by the attention-weighted sum of the hidden states. The forward propagation process is formulated as follows:

$$\{\boldsymbol{h}_i\} = Transformer(\{\boldsymbol{e}_i\}) \tag{3}$$

$$\boldsymbol{h}_i^f = tanh(FFN(\boldsymbol{h}_i)) \tag{4}$$

$$\boldsymbol{r}_c = Attention(\{\boldsymbol{h}_i^f\}) \tag{5}$$

where $Transformer(.)$ is staked Transformer layers, and $\boldsymbol{e}_i \in \mathbb{R}^d$ is the embedding of i_{th} word where $i \in [0, 1, ..., L_c]$, L_c is the length of the context.

The final feature representation of the mention is the weighted sum of r_m and r_c:

$$\boldsymbol{r} = (1 - \alpha) * \boldsymbol{r}_m + \alpha * \boldsymbol{r}_c \tag{6}$$

where α is a hyper-parameter.

3.2 Instance-Aware Label Encoder

Our work focuses on enhancing label representation via imposing relational inductive biases of instance-to-label and label-to-label. We implement that via constructing a two-phase graph dynamically and then separating two sub-graphs to perform graph operations in two phases: instance-label sub-graph for extracting instance-aware label representations, and label-label sub-graph for dependency-aware label representations.

Dynamic graph construction. The two-phase graph contains instance nodes and label nodes. We construct the graph dynamically for every training or prediction batch. Specifically, the two-phase graph are divided into two sub-graphs: the instance-label sub-graph and the label-label sub-graph. For the instance-label sub-graph, we define the adjacent matrix as follows:

$$A_{ij}^1 = \begin{cases} 1, & if \textbf{ instance } i \ belongs \ to \textbf{ label type } j, \\ 0, & otherwise \end{cases} \tag{7}$$

in which $A_{ii}^1 = 1$ for self-connection. For the label-label sub-graph, we use the statistical results from training dataset to build the graph, i.e., the co-occurrence results. The adjacent matrix is defined as:

$$A_{ij}^2 = I(i, j) \tag{8}$$

where $I(.)$ is an indicator function, $I(i, j) = 1$ if label types i and j co-exist in an instance, otherwise 0. Take into account that the transfer probability or dependency level varies from label to label, we also define an edge weight matrix for the sub-graph according to the co-occurrence frequency of them:

$$W_{ij}^l = \frac{count((i, j))}{count(i)} \tag{9}$$

where i, j is label nodes, $count(.)$ counts the cumulative number of occurrence and W_{ij}^l indicates the transfer ratio from label i to j. The W^l is asymmetric as the cumulative numbers of labels i and j are usually different. Note that:

$$\sum_{j=1}^{N_l} W_{ij}^l \geq 1 \tag{10}$$

where the N_l is the size of label set. For example: if the label zoo (label i) appears, the label $location$ (label j) is sure to appear so that $W^l_{ij} = 1$ while the opposite is not necessarily true and $W^l_{ji} < 1$.

Phase I: Instance-aware label representations. A label type usually corresponds to numerous instances whose features reflect those of the label. Therefore, capturing the representation associations between labels and instances can make the label representations more representative. Considering an entity mention in a certain context may belong to several label types, we model the association difference by using GAT propagation with adjacent matrix A^1 on the instance-label sub-graph to get instance-aware label representations. Formally,

$$V = [T_0; B] \tag{11}$$

denotes the initial node features where $T_0 \in \boldsymbol{R}^d$ is randomly initialized label representations, and $B = \{\boldsymbol{r}\}$ is the set of instance features. We follow GAT propagation rule on the instance-label sub-graph as defined in [Velickovic *et al.*, 2017] to get instance-aware label representations $V' = \{\boldsymbol{v}'_i\}$. After GAT propagation, the label representations T_{gat}:

$$T'_{gat} = \{\boldsymbol{v}'_i\}_{i \in [0,\ldots,N_l]} \tag{12}$$

$$T_{gat} = T'_{gat} + T_0 \tag{13}$$

are regards as the initial node features of the next phase. To make better use of historical information and accelerate convergence of network, we introduce the residual connection after the GAT propagation, as formulated in Formula (13).

Phase II: Dependency-aware label representations. The phenomenon of label co-occurrence in the same instance suggests that there exists a certain conceptual similarity or hierarchical dependency (in our case) among labels. To capture the dependency relationships among labels to make label representations more smooth for those low-resource labels, we exploit GCN propagation on the label-label sub-graph. On the basis of statistical results of W^l (after normalization) and A^2, we follow the GCN propagation rule defined in [Xiong *et al.*, 2019] to get new label representations T_{gcn}. Here we also use a residual connection, so the final representations of labels can be formulated as:

$$T = T_{gcn} + T_{gat}. \tag{14}$$

Note that one layer of GCN is enough here as the two- or multi-hops may introduce irrelevant even conflicting prediction, such as $male \leftarrow \boldsymbol{person} \rightarrow female$. If applying two-layer GCNs, the contradictory prediction may be generated with both labels $male$ and $female$. In traditional methods, labels with a small number of instances are updated less frequently and also easy to overfit those instances. With GNNs propagation, the update of adjacent label nodes will also affect the representations of the labels with fewer instances. Therefore, the label representation is no longer bound only by the instances it involves, but also by its neighbors.

We calculate the dot product of instance representation $\boldsymbol{r}$ and T followed by a sigmoid activation function $\sigma(.)$, as the similarity score:

$$\boldsymbol{y'} = \sigma(\boldsymbol{r} \cdot T). \tag{15}$$

Then we apply a threshold thd, which is also a hyper-parameter, to $\boldsymbol{y'}$ to obtain the prediction results of our method.

3.3 Pseudo-Label Generator

For training instances, there are ground truth labels of entity mentions while not for the testing ones. To construct the two-phase graph for prediction, we design a pseudo-label generator to generate pseudo labels for testing instances. As described above, the context representation of a mention is the weighted sum of context hidden states. Some keywords play major roles in the overall representation of context, so we make full use of the words in context to obtain pseudo labels:

$$\boldsymbol{r}^i = (1 - \alpha) * \boldsymbol{r}_m + \alpha * \boldsymbol{h}^i \tag{16}$$

$$\boldsymbol{y}^i_p = \sigma(\boldsymbol{r}^i \cdot T) \tag{17}$$

$$\boldsymbol{y}_p = max\_pooling([\boldsymbol{y}^0_p; \ldots; \boldsymbol{y}^{L_c - 1}_p]) \tag{18}$$

where $\boldsymbol{h}^i$ is the i_{th} hidden state of context and $max\_pooling(.)$ is max pooling operation. Then we can obtain the pseudo label set from $\boldsymbol{y}_p$ where $\boldsymbol{y}_{p,i} > thd$. However, this will lead to inconsistency between training and testing instances because of using ground truth labels for training but pseudo labels for testing. We solve the problem via: i) proportionally replacing (with ratio of β) ground truth labels with pseudo labels during training; ii) adding loss on pseudo and ground truth labels to force the pseudo-label generator to generate labels that match the ground truth ones. Therefore the whole loss can be formulated as:

$$\mathcal{L}_p(\boldsymbol{y}_p | (m, c; \theta), \boldsymbol{y}) = BCE(\boldsymbol{y}_p | (m, c; \theta), \boldsymbol{y}) \tag{19}$$

$$\mathcal{L}_g(\boldsymbol{y'} | (m, c; \theta), \boldsymbol{y}) = BCE(\boldsymbol{y'} | (m, c; \theta), \boldsymbol{y}) \tag{20}$$

$$\mathcal{L} = \mathcal{L}_g + \gamma * \mathcal{L}_p \tag{21}$$

where $\mathcal{L}_p$ is the loss of pseudo labels, $\mathcal{L}_g$ is the loss of prediction result, $\mathcal{L}$ is the overall loss, $\gamma = 1$ is a weight hyper-parameter, and $BCE(.)$ is binary cross entropy loss which is widely used in the multi-label multi-class task.

4 Experiments

4.1 Datasets

We conduct experiments on datasets of OntoNotes [Gillick *et al.*, 2014], FIGER [Ling and Weld, 2012] and BBN [Weischedel and Brunstein, 2005].

OntoNotes. The OntoNotes dataset samples sentences from newswire documents contained in the OntoNotes corpus. The entities are mapped to Freebase types via DBpedia Spotlight. The testing data is manually annotated by [Gillick *et al.*, 2014]. There are 89 labels with 3-level ontology.

FIGER. The FIGER dataset consists of sentences from the Wall Street Journal annotated by [Ling and Weld, 2012]. The training data is annotated using DBpedia Spotlight as well and contains 113 labels with 2-level ontology.

BBN. The BBN dataset is composed of sentences from some of the Penn Treebank corpus of Wall Street Journal texts. It was automatically labeled via distant supervision by mapping the entities to Freebase types. The testing data consists of news reports manually labeled by [Weischedel and Brunstein, 2005]. There are 47 labels with 2-level ontology.

Methods	OntoNotes			FIGER			BBN		
	Mi-F1	Ma-F1	ACC	Mi-F1	Ma-F1	ACC	Mi-F1	Ma-F1	ACC
AttentiveNER[Shimaoka *et al.*, 2016]	0.649	0.710	0.517	0.754	0.790	0.597	-	-	-
AFET[Ren *et al.*, 2016]	0.647	0.711	0.551	-	-	-	0.735	0.727	0.670
AFGET[Lin and Ji, 2019]	0.773	0.829	0. 638	0.781	0.793	0.559	0.781	0.793	0.559
CLCS[Chen *et al.*, 2019]	0.720	0.778	0.628	-	-	-	0.805	0.807	0.747
LabelGCN[Xiong *et al.*, 2019]	0.722	0.778	0.596	0.742*	0.775*	0.623*	0.742*	0.747*	0.584*
FGET-RR[Ali *et al.*, 2020]	0.685	0.743	0.577	0.805	0.847	0.710	0.823	0.819	0.703
HET[Chen *et al.*, 2020]	0.681	0.730	0.587	0.808	0.826	0.691	0.805	0.797	0.752
NFETC-AR[Zhang *et al.*, 2020]	0.730	0.788	0.640	0.801	0.832	0.701	0.815	0.814	**0.767**
Pseudo-label Generator	0.746	0.801	0.618	0.769	0.794	0.632	0.763	0.761	0.596
Our method(*GAT-GCN*)	**0.792**	**0.845**	**0.651**	**0.844**	**0.877**	0.706	**0.860**	**0.876**	0.699
–w/o GCN(Phase I: ***GAT-only***)	0.778	0.830	0.640	0.845	0.867	**0.730**	0.827	0.844	0.644
–w/o GAT(Phase II: ***GCN-only***)	0.761	0.813	0.618	0.774	0.809	0.636	0.779	0.778	0.598

Table 1: Testing results on the OntoNotes/FIGER/BBN testing sets. *GAT-GCN* method denotes the complete method we propose. *GCN-only* method represents we just use GCN propagation on label-label sub-graph while use only GAT propagation on instance-label sub-graph for *GAT-only* method. The testing batch size is set to 300, 400, 200 for OntoNotes, FIGER, and BBN respectively. * denotes that we rerun the publicly available code of the work since it did not evaluate on the two datasets. - represents not run on the specific dataset.

4.2 Baselines

To be fair, we compare our method with only those do not use external information like entity linking or pre-training&fine-tuning, as introduced in Section 2.2: methods using attention mechanism like [Ren *et al.*, 2016; Shimaoka *et al.*, 2016; Lin and Ji, 2019] and methods using graph (or hierarchy) like [Chen *et al.*, 2019; Xiong *et al.*, 2019; Chen *et al.*, 2020; Ali *et al.*, 2020; Zhang *et al.*, 2020].

4.3 Experimental Settings

In our experiments, we use the pre-trained 5.5B ELMo[1] as our word embeddings without fine-tuning. The dimension of word embeddings is 1024. We set the hyper-parameters of the Transformer as follows: encoder of 1 layer, 1024 dimension for the feed-forward network, 8 heads, and 0.1 for dropout. The max lengths are set to 10 for mentions and 50 for contexts of mentions, for all 3 datasets. Then we map the dimension of the representations of mention and context to 200. For the hyper-parameter α, the sum weight of representations of mention and context, is set to $0.3, 0.3, 0.8$ for OntoNotes/FIGER/BBN respectively. For the replacing ratio of pseudo labels β , is set to $0.2, 0.2, 0.075$ for OntoNotes/FIGER/BBN respectively. The train batch size is 200. Maximum-iteration is set to 10000. We explore the best result in the range of [0.5,0.8] with stride 0.1 for thd.

4.4 Result analysis

Following prior works, we evaluate our method using metrics: strict accuracy (ACC), macro-average F1-score (Ma-F1), and micro-average F1-score (Mi-F1).

Overall Results. Table 1 shows the overall results of our method compared to other methods on OntoNotes, FIGER, and BBN respectively. Our method achieves the best performance on most evaluation metrics by a large margin, except for the strict accuracy for BBN. The reason may be that the label set is smaller and the co-occurrence graph is sparser than the other two datasets which weaken the effect of the label-to-label relational bias constraint. The testing result on the BBN dataset demonstrates the necessity of imposing instance-to-label relational inductive bias but also reveals the limitation of this method, that is, the generalization of labels with sparse graph structure is limited.

[1]https://allennlp.org/elmo

Dataset	Methods	Mi-F1	Ma-F1	ACC
OntoNotes	*GCN-only*	**0.761**	**0.813**	**0.618**
	GAT-GCN-static	0.751	0.806	0.612
FIGER	*GCN-only*	**0.774**	**0.809**	**0.636**
	GAT-GCN-static	0.754	0.787	0.601
BBN	*GCN-only*	**0.779**	**0.778**	**0.598**
	GAT-GCN-static	0.761	0.762	0.566

Table 2: Testing results on 3 testing sets comparing *GCN-only* and *GAT-GCN-static* methods. *GAT-GCN-static* denotes that when prediction, we calculate the logit labels with the label representations generated by the last training batch.

Static label representations results. As the instance-aware label representations are dynamic, we explore the *GAT-GCN-static* method for comparison. As shown in Table 2, the *GAT-GCN-static* method even performs worse than the *GCN-only* method. Obviously, more extra parameters introduced in the GAT process do not make the performance better. T-wo reasons here: i) the label representations used are generated by the last training batch, so they overfit the training instances; ii) no testing instance feature is fed into the label representations. The second reason actually matters. Since the training process is the same as before, the method will eventually converge to the same level as before. In the absence of overfitting of the model, the testing performance de-

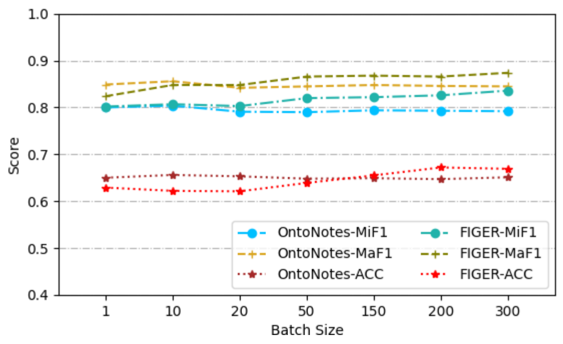

Figure 3: Testing results of different number of involving instances (batch size) for OntoNotes and FIGER.

grades due to the instance-independent label representations.

The feasibility of pseudo-label generator. we also explore pseudo-label generator separately about its feasibility. Note that the training process is consistent with the previous experiments, which means, the loss is still calculated by the normal prediction result as before, but during prediction, we use the generated result from the pseudo-label generator to evaluate all the metrics. The testing results are shown in the fourth row from the bottom of Table 1. As the results suggested, the pseudo-label generator performs worse than the complete method but better than some of the baseline methods which proves the feasibility of the pseudo-label generator. The results also suggest that the introduction of the pseudo-label generator does not offer a direct promotion to the overall performance of the proposed method.

The impact of involving instance number. Because of the dynamic label representations, the number of involving instances may have an impact on the label representations and thus affect the stability of the testing performance. We conduct a set of experiments to test the impact, as shown in Figure 3. The results show that the number of testing instances will lead to slight fluctuations on some metrics, especially for the FIGER testing set. For the training data of FIGER, there are about 36% mentions with noisy labels because of distant supervision annotation while just about 27% for OntoNotes and 24% for BBN. Excessive noise data leads to prediction bias for the model which results in a worse impact on the testing performance because more clean testing instances are needed to correct the label representations. The results on the OntoNotes testing set are almost consistent as the number of instances changes because of the cleaner training data.

Ablation Results. While achieving significant improvements on all datasets, it is still not clear which kind of relational inductive bias constraint is more important. We set up ablation experiments to explore the role of the two constraints. As shown in the last two rows of Table 1, the performance of *GAT-only* method is much better than the GCN-only method especially on the FIGER dataset, which demonstrate our claim again that the instance features reflect those of the labels, i.e., the instance-to-label relational inductive bias is much stronger. So far, our core point has been proved.

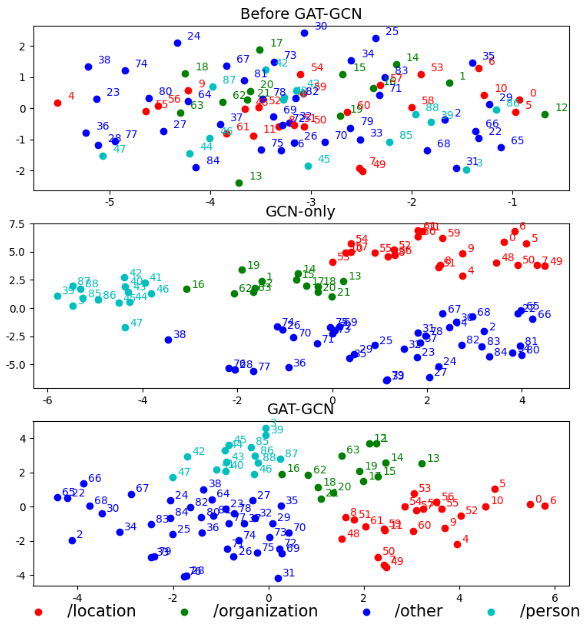

Figure 4: The visualization of label representations with t-SNE for OntoNotes. Note that, *Before GAT-GCN*(the top) visualizes T_0 in *GAT-GCN* method.

Visualization for Label Representations. We visualize the label representations of 3 cases, as shown in Figure 4. The top one shows the representations of the initial T_0 of *GAT-GCN* method, which can only learn something general. *GCN-only*(the middle) and *GAT-GCN*(the bottom) methods significantly separates the labels into four clusters, but the *GAT-GCN* can lead to more compact clusters (like cluster */location*). The effectiveness of our method is proved to some extent.

5 Conclusion

In this work, we propose to enhance the label representations with graph relational inductive bias constraints for the FGET task. We implement this by a two-phase graph network: in the phase I, we use GAT propagation on the instance-label subgraph to capture helpful instance features for the label representations, and in the phase II, we exploit GCN propagation on the label-label sub-graph to capture the dependency relationships among labels. We conduct experiments to evaluate the effectiveness of the proposed method. As an extension, our method can be easily migrated to a non-hierarchical label set as the graph is built according to label co-occurrence. Also, the instance-to-label relational inductive bias constraint is general and can be used in other classification tasks with properly deforming.

Acknowledgments

This work is supported by the Strategic Priority Research Program of Chinese Academy of Sciences, Grant No.XDC02040400. The authors thank Jiawei Sheng from Institute of Information Engineering, Chinese Academy of Sciences, for his insightful and constructive discussions.

References

[Ali *et al.*, 2020] Muhammad Asif Ali, Yifang Sun, Bing Li, and Wei Wang. Fine-grained named entity typing over distantly supervised data based on refined representations *. In *AAAI*, 2020.

[Battaglia *et al.*, 2018] Peter W. Battaglia, Jessica B. Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinícius Flores Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, Çaglar Gülçehre, H. Francis Song, Andrew J. Ballard, Justin Gilmer, George E. Dahl, Ashish Vaswani, Kelsey R. Allen, Charles Nash, Victoria Langston, Chris Dyer, Nicolas Heess, Daan Wierstra, Pushmeet Kohli, Matthew Botvinick, Oriol Vinyals, Yujia Li, and Razvan Pascanu. Relational inductive biases, deep learning, and graph networks. *CoRR*, abs/1806.01261, 2018.

[Carlson *et al.*, 2010] Andrew Carlson, Justin Betteridge, Richard C. Wang, Estevam R. Hruschka, and Tom M. Mitchell. Coupled semi-supervised learning for information extraction. In *WSDM '10*, 2010.

[Chen *et al.*, 2019] Bo Chen, Xiaotao Gu, Yufeng Hu, Siliang Tang, Guoping Hu, Yueting Zhuang, and Xiang Ren. Improving distantly-supervised entity typing with compact latent space clustering. In *NAACL-HLT*, 2019.

[Chen *et al.*, 2020] Tongfei Chen, Yunmo Chen, and Benjamin Van Durme. Hierarchical entity typing via multilevel learning to rank. In *ACL*, 2020.

[Collins and Singer, 1999] Michael Collins and Yoram Singer. Unsupervised models for named entity classification. In *EMNLP*, 1999.

[Dai *et al.*, 2019] Hongliang Dai, Donghong Du, Xin Li, and Yangqiu Song. Improving fine-grained entity typing with entity linking. In *EMNLP/IJCNLP*, 2019.

[Durrett and Klein, 2014] Greg Durrett and Dan Klein. A joint model for entity analysis: Coreference, typing, and linking. *Transactions of the Association for Computational Linguistics*, 2014.

[Gillick *et al.*, 2014] Daniel Gillick, Nevena Lazic, Kuzman Ganchev, Jesse Kirchner, and David Huynh. Context-dependent fine-grained entity type tagging. *ArXiv*, abs/1412.1820, 2014.

[Jiang and Zhai, 2006] Jing Jiang and ChengXiang Zhai. Exploiting domain structure for named entity recognition. In *HLT-NAACL*, 2006.

[Kipf and Welling, 2017] Thomas Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

[Lin and Ji, 2019] Ying Lin and Heng Ji. An attentive fine-grained entity typing model with latent type representation. In *EMNLP-IJCNLP*, 2019.

[Ling and Weld, 2012] Xiao Ling and Daniel S. Weld. Fine-grained entity recognition. In *AAAI*, 2012.

[López *et al.*, 2019] Federico López, Benjamin Heinzerling, and Michael Strube. Fine-grained entity typing in hyperbolic space. In *RepL4NLP@ACL*, 2019.

[Ratinov and Roth, 2009] Lev-Arie Ratinov and Dan Roth. Design challenges and misconceptions in named entity recognition. In *CoNLL*, 2009.

[Ren *et al.*, 2016] Xiang Ren, Wenqi He, Meng Qu, Lifu Huang, Heng Ji, and Jiawei Han. Afet: Automatic fine-grained entity typing by hierarchical partial-label embedding. In *EMNLP*, 2016.

[Shi *et al.*, 2020] Haochen Shi, Siliang Tang, Xiaotao Gu, Bo Chen, Zhigang Chen, Jian Shao, and Xiang Ren. Alleviate dataset shift problem in fine-grained entity typing with virtual adversarial training. In *IJCAI*, 2020.

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

[Shimaoka *et al.*, 2017] Sonse Shimaoka, Pontus Stenetorp, Kentaro Inui, and Sebastian Riedel. Neural architectures for fine-grained entity type classification. In Mirella Lapata, Phil Blunsom, and Alexander Koller, editors, *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 1: Long Papers*, pages 1271–1280. Association for Computational Linguistics, 2017.

[Velickovic *et al.*, 2017] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *ICLR*, 2017.

[Wang *et al.*, 2019a] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. Kgat: Knowledge graph attention network for recommendation. In *KDD*, 2019.

[Wang *et al.*, 2019b] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. Neural graph collaborative filtering. In *SIGIR*, 2019.

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

[Xiong *et al.*, 2019] Wenhan Xiong, Jiawei Wu, Deren Lei, Mo Yu, Shiyu Chang, Xiaoxiao Guo, and William Yang Wang. Imposing label-relational inductive bias for extremely fine-grained entity typing. In *NAACL-HLT*, 2019.

[Xu and Barbosa, 2018] Peng Xu and Denilson Barbosa. Neural fine-grained entity type classification with hierarchy-aware loss. In *NAACL-HLT*, 2018.

[Yaghoobzadeh *et al.*, 2016] Yadollah Yaghoobzadeh, Heike Adel, and Hinrich Schutze. Noise mitigation for neural entity typing and relation extraction. In *EACL*, 2016.

[Yavuz *et al.*, 2016] Semih Yavuz, Izzeddin Gur, Yu Su, Mudhakar Srivatsa, and Xifeng Yan. Improving semantic parsing via answer type inference. In *EMNLP*, 2016.

[Zhang *et al.*, 2020] Haoyu Zhang, Dingkun Long, Guangwei Xu, Muhua Zhu, Pengjun Xie, Fei Huang, and Ji Wang. Learning with noise: Improving distantly-supervised fine-grained entity typing via automatic relabeling. In *IJCAI*, 2020.