MZET: Memory Augmented Zero-Shot Fine-grained Named Entity Typing

Tao Zhang[1], **Congying Xia**[1], **Chun-Ta Lu**[2], **Philip Yu**[1]
[1]University of Illinois at Chicago, Chicago, IL, USA;
[2]Google Research, Mountain view, CA, USA
{tzhang90,cxia8,psyu}@uic.edu; chunta@google.com

Abstract

Named entity typing (NET) is a classification task of assigning an entity mention in the context with given semantic types. However, with the growing size and granularity of the entity types, rare researches in previous concern with newly emerged entity types. In this paper, we propose MZET, a novel memory augmented FNET (Fine-grained NET) model, to tackle the unseen types in a zero-shot manner. MZET incorporates character-level, word-level, and contextural-level information to learn the entity mention representation. Besides, MZET considers the semantic meaning and the hierarchical structure into the entity type representation. Finally, through the memory component which models the relationship between the entity mention and the entity type, MZET transfer the knowledge from seen entity types to the zero-shot ones. Extensive experiments on three public datasets show prominent performance obtained by MZET, which surpasses the state-of-the-art FNET neural network models with up to 7% gain in Micro-F1 and Macro-F1 score.

1 Introduction

Named entity typing (NET) is the task of inferring semantic types for given named entity mentions in the utterances. For instance, given an entity mention "John" in the utterance "John plays piano on the stage", the goal for NET is to infer that "John" is a pianist or a musician, and a person. Standard NET approaches (Chinchor and Robinson, 1997; Sang and De Meulder, 2003; Doddington et al., 2004) only consider a tiny set of coarse-grained types, and discard fine-grained types with a different level of granularity. In recent years, fine-grained named entity typing (FNET) (Ling and Weld, 2012) continues to draw researchers' attention. It can provide more informative named entity types and benefit a lot of downstream tasks like relation extraction(Liu et al., 2014), entity linking(Stern et al., 2012) and question answering (Han et al., 2017).

However, with the ever-growing entity types, it is difficult and expensive to collect a lot of annotation data per category and retrain the whole model. Therefore, a zero-shot paradigm is welcomed in the FNET to handle the increasing unseen types. The task we deal with in this paper is named as Zero-shot fine-grained named entity typing (ZFNET), which is to detect the unseen fine-grained entity types which has no labeled data available.

Learning generalizable representations for entity mentions and types is essential for the ZFNET task. Previous works learn these representations either from hand-crafted features (Ma et al., 2016; Yuan and Downey, 2018), or pre-trained word embeddings (Ren et al., 2016). These methods are insufficient and inefficient when challenging by polysemantic, ambiguity, or even the newly-emerged mentions. There are also works (Obeidat et al., 2019; Zhou et al., 2019) which learn more informative but resource-costing representations by assembling related Wikipedia pages.

With the learned representations for entity mentions and entity types, most of the existing zero-shot FNET methods (Ma et al., 2016; Obeidat et al., 2019) project them into a shared semantic space. The shared space is learned through minimizing the distance between entity mentions and its corresponding seen entity types. In the prediction stage, testing entity mentions are classified to the nearest unseen entity types based on the assumption that the learned distance measurement also works for unseen types. These methods' ability to transfer knowledge from seen types to unseen types is limited since they do not explicitly build connections between seen types and unseen types.

In this work, we propose the memory augmented zero-shot FNET model (MZET) to tackle the problems aforementioned. MZET is designed to automatically extract the multi-information integrated mention representations and structure-aware semantic type representations with large-scale pre-trained language model (Devlin et al., 2018). To effectively transfer knowledge from seen types to unseen types, MZET regards seen types as memory components and explicitly models the relationships between seen types and unseen types. Intuitively, we want to mimic the way how human learn new concepts. Human learn new concepts by comparing the similarities and differences between new concepts and old concepts stored in our memory. In summary, the main contributions of MZET are as follows:

- We propose the memory augmented zero-shot FNET model (MZET) which can be trained in an end-to-end fashion. MZET extracts multi-information integrated mention representations and structure-aware semantic type representations without additional augmented data sources.
- MZET regards seen types as memory components and explicitly models the relationships between seen types and unseen types to effectively transfer knowledge to new concepts.
- MZET outperforms existing zero-shot FNET models significantly on the zero-shot fine-grained, coarse-grained, and hybrid-grained named entity typing over three benchmark datasets.

2 Problem Definition

We begin by formalizing the problem of zero-shot fine-grained named entity typing (ZFNET). For a given entity mention x, the task of named entity typing (NET) is to identify the type y for x. Suppose we have a training label set $\mathcal{Y}_{seen} = \{y_1^s, y_2^s, ..., y_{D_s}^s\}$ with D_s seen labels. There are a large amount of labeled examples available for these seen labels, $\mathcal{D}_{tr} = \{(x_i, y_i), i = 1, 2, ..., |\mathcal{D}_{tr}|\}$ with $y_i \in \mathcal{Y}_{seen}$. The task of ZFNET is to classify a new mention which belongs to one of the unseen fine-grained entity types $\mathcal{Y}_{unseen} = \{y_1^u, y_2^u, ..., y_{D_u}^u\}$, where D_u is the number of unseen fine-grained entity types and $\mathcal{Y}_{seen} \cap \mathcal{Y}_{unseen} = \emptyset$.

sectionThe proposed Model The overview of the proposed MZET framework is illustrated in Figure 1. Specifically, MZET consists of three components: 1) Mention Processor which extracts representation for entity mentions; 2) Label Processor which obtains label representation; 3) Zero-shot Memory Network which identifies labels for the entity mentions.

2.1 Mention Processor

To better understand the entity mention, we not only consider the words contained in the mention, but also the context around it. The Mention Processor has two sub components. A Word Processor is proposed to get the semantic meaning for each word in the entity mention. Another Context Processor is utilized to understand the sequential information together with the context. The final mention representation $\mathbf{m}$ is a concatenation of the word-level representation from the Word Processor and the sequential representation from the Context Processor.

2.1.1 Word Processor

Word Processor is proposed to achieve basic understandings over the words in the entity mentions. Given an input entity mention with K tokens $(t_1, ..., t_K)$, each token t_k is represented as

$$\mathbf{t}_k = [\mathbf{w}_k; \mathbf{c}_k], \qquad (1)$$

using a concatenation of a pre-trained word embedding $\mathbf{w}_k \in \mathbb{R}^{D_w}$ (D_w is the dimension of each pre-trained word embedding) and a character-level embedding $\mathbf{c}_k$ which provides morphological information. The character-level embedding $\mathbf{c}_k \in \mathbb{R}^{2\times D_{h_c}}$ is obtained through a bi-directional LSTM (the hidden state size is D_{h_c}), named as Character Bi-LSTM. Random initialized character embeddings are fed into the Character Bi-LSTM. The final hidden states from the forward and backward LSTM are concatenated as $\mathbf{c}_k$.

Additionally, another bi-directional LSTM, named as Word-Character Bi-LSTM, is utilized to gather the information from all the token embeddings by concatenating the forward and backword hidden states:

$$\mathbf{m}_w = \text{Word-Character}(\mathbf{t}_1, ..., \mathbf{t}_K), \qquad (2)$$

where $\mathbf{m}_w \in \mathbb{R}^{2\times D_h}$, D_h is the dimension of Word-Character Bi-LSTM hidden states. As illustrated in Figure 1, the Word Processor outputs a word-character embedding $\mathbf{m}_w$ for each entity mention.

2.1.2 Context Processor

In the Context Processor, we leverage the powerful pre-trained language model, BERT (Devlin et al., 2018), to incorporate two more parts into the mention representation: (1)$\mathbf{m}_b$, the mention embedding given the context; (2)$\mathbf{m}_c$, the context embedding around the mention.

We first conduct BERT to embed the whole sentence and obtain the BERT embedding for each token. For the tokens contained in the entity mention, named as mention tokens, their BERT embeddings are represented as $\mathbf{b}_1, ..., \mathbf{b}_K$, $\mathbf{b}_i \in \mathbb{R}^{D_b}$, D_b is the BERT embedding dimension. For the tokens in the surrounding context, named as context tokens, we only consider a fixed window for each mention to balance the computational cost. The BERT embeddings for left context tokens are $\mathbf{e}_1^l, ..., \mathbf{e}_n^l$, and those in the right are $\mathbf{e}_1^r, ..., \mathbf{e}_n^r$, $\mathbf{e}_i^j \in \mathbb{R}^{D_b}$ and $j \in \{l, r\}$. n is the window size and we set it as 10.

Then, we utilize Bi-LSTMs (with hidden state size D_h) to aggregate the separated token embeddings to extract $\mathbf{m}_b$ and $\mathbf{m}_c$. $\mathbf{m}_b$ is obtained from the BERT embeddings of mention tokens with a bi-directional LSTM, called as Mention Bi-LSTM:

$$\mathbf{m}_b = \text{Mention}(\mathbf{b}_1, ..., \mathbf{b}_K), \tag{3}$$

where $\mathbf{m}_b \in \mathbb{R}^{2\times D_h}$. $\mathbf{m}_c$ is obtained from the context tokens with a bi-directional LSTM with attention mechanism, called as Attention Bi-LSTM. The hidden states in the bi-directional LSTM for the context tokens are denoted as: $\overrightarrow{\mathbf{h}_1^l}, \overleftarrow{\mathbf{h}_1^l}, ..., \overrightarrow{\mathbf{h}_n^l}, \overleftarrow{\mathbf{h}_n^l}$, and $\overrightarrow{\mathbf{h}_1^r}, \overleftarrow{\mathbf{h}_1^r}, ..., \overrightarrow{\mathbf{h}_n^r}, \overleftarrow{\mathbf{h}_n^r}$.

The attentions over all the context tokens are computed using a 2-layer feed forward neural network:

$$\mathbf{e}_i^j = \tanh(\mathbf{W}_e[\overrightarrow{\mathbf{h}_i^j}; \overleftarrow{\mathbf{h}_i^j}]), \tilde{\mathbf{a}}_i^j = \exp(\mathbf{W}_a \mathbf{e}_i^j), \tag{4}$$

where $\mathbf{W}_e \in \mathbb{R}^{D_h \times 2 \times D_a}$, $\mathbf{W}_a \in \mathbb{R}^{1\times D_a}$, D_a is the attention dimension, $j \in \{l, r\}$. Then we normalize the attentions over all the mention tokens by:

$$\mathbf{a}_i^j = \frac{\tilde{\mathbf{a}}_i^j}{\sum_{i=1}^n \tilde{\mathbf{a}}_i^l + \tilde{\mathbf{a}}_i^r}. \tag{5}$$

The context embedding $\mathbf{m}_c \in \mathbb{R}^{2\times D_h}$ is weighted by the attentions:

$$\mathbf{m}_c = \sum_{i=1}^n (\mathbf{a}_i^l[\overrightarrow{\mathbf{h}_i^l}; \overleftarrow{\mathbf{h}_i^l}] + \mathbf{a}_i^r[\overrightarrow{\mathbf{h}_i^r}; \overleftarrow{\mathbf{h}_i^r}]). \tag{6}$$

2.1.3 Mention Representation

The final entity mention representation with dimension $D_m \in \mathbb{R}^{6\times D_h}$ concatenates the word-character embedding, the mention embedding, and the context embedding as follow:

$$\mathbf{m} = [\mathbf{m}_w; \mathbf{m}_b; \mathbf{m}_c]. \tag{7}$$

2.2 Label Processor

Understanding the label is important in our task, since there is no information other than the label name for the zero-shot entity types. In the Label Processor, we get the semantic embeddings $B^S \in \mathbb{R}^{(D_s+D_u)\times D_b}$ for all the label names, including the seen labels $\mathcal{Y}_{seen}$ and the unseen labels $\mathcal{Y}_{unseen}$, using a pre-trained BERT model.

For a fine-grained entity typing task, their higher level types, i.e. coarse-grained types, are also involved. The fine-grained labels and coarse-grained labels in $\mathcal{Y}_{seen}$ and $\mathcal{Y}_{unseen}$ consist a hierarchical structure naturally. Following (Ma et al., 2016), we utilize a sparse matrix $B^H \in \mathbb{R}^{(D_s+D_u)\times(D_s+D_u)}$ to represent the hierarchical structure in the labels. Each row B_i^H corresponds to a binary hierarchical embedding for label y_i. For each entry in B_i^H, we use 1 to denote the label itself and its parent node, 0 for the rest:

$$B_{ij}^H = \begin{cases} 0, & \text{if } i = j \text{ or } y_j \in Parent(y_i); \\ 1, & \text{otherwise.} \end{cases} \tag{8}$$

In the Label Processor, we integrate the semantic embeddings of the child label and its parent label into a single embedding vector as the fine-grained label representftion. For a label y_i, the final label representation $\mathbf{f} \in \mathbb{R}^{D_b}$ is represented together by the semantic embedding B^S and its hierarchical embedding B_i^H:

$$\mathbf{f} = {B^S}^\top B_i^H. \tag{9}$$

2.3 Zero-Shot Memory Network

In the zero-shot entity typing task, there are no mentions available for these unseen entity types. Without the labeled data, we are not able to model the direct mapping from the new mentions to the new labels. Here, we propose a novel zero-shot memory network which uses the seen entity types to bridge the gap between the new mentions and the zero-shot entity types.

All the seen entity representations are utilized as the memories in the zero-shot memory network.

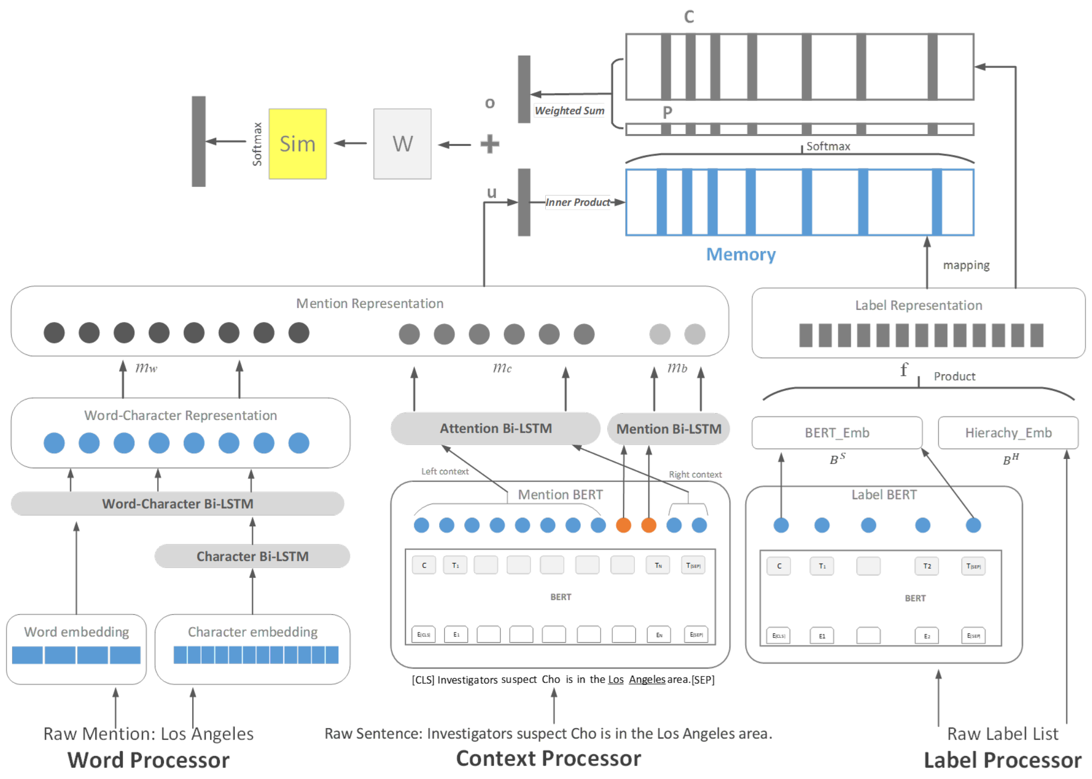

Figure 1: The framework of MZET for zero-shot fine-grained named entity typing. It consist of three main components: mention representation, label representation, memory augmented zero-shot learning.

We propose to use the memory network as a special attention mechanism to model the relationships between the mentions and the seen entity types. Furthermore, we build a zero-shot version memory network which utilizes the label representation similarities to transfer the knowledge from the seen labels to the unseen labels.

We get all the seen label representations from the Label Processor, $\mathbf{F} = (\mathbf{f}_1^s, ..., \mathbf{f}_{D_s}^s) \in \mathbb{R}^{D_s \times D_b}$, where D_s is the number of seen labels and D_b is the dimension of the label representation. The input memory representation $\mathbf{G} \in \mathbb{R}^{D_s \times D_m}$ are converted from $\mathbf{F}$ using an embedding matrix $\mathbf{W}_{f1} \in \mathbb{R}^{D_b \times D_m}$. Then we model the attention between the mention $\mathbf{m}$ and each memory component $\mathbf{g}_i \in \mathbb{R}^{D_m}, i \in \{1, ..., D_s\}$ with:

$$p_i = \text{Softmax}(\mathbf{m}^\top \mathbf{g}_i). \tag{10}$$

Another embedding matrix $\mathbf{W}_{f2} \in \mathbb{R}^{D_b \times D_m}$ are used to get the output memory representation $\mathbf{O} \in \mathbb{R}^{D_s \times D_m}$ from the memory $\mathbf{F}$. An integrated label embedding $\mathbf{q}$ which carries with the attentions between the mention and the seen labels is obtained from:

$$\mathbf{q} = \sum_i p_i \mathbf{o}_i. \tag{11}$$

To extend the memory network into a zero-shot version, we use the similarities between the label representations to transfer knowledge from seen labels to unseen labels. The similarities between label $\mathbf{f}_i$ and $\mathbf{f}_j$ is calculated as:

$$r_{ij} = \frac{\exp\{-d(\mathbf{f}_i, \mathbf{f}_j)\}}{\sum_{j=1}^{D_s} \exp\{-d(\mathbf{f}_i, \mathbf{f}_j)\}}, \tag{12}$$

where $d(\mathbf{f}_i, \mathbf{f}_j)$ is the Euclidean distance between $\mathbf{f}_i$ and $\mathbf{f}_j$. Then we can get a similarity matrix $\mathbf{R} \in \mathbb{R}^{D_u \times D_s}$ for all the unseen labels. We use the attention weighted label embeddings $\mathbf{q}$, the mention embeddings $\mathbf{m}$ and the similarity matrix $\mathbf{R}$ together to classify the zero-shot entity types:

$$y = \text{sigmoid}(\mathbf{R}^\top \mathbf{W_p}(\mathbf{q} + \mathbf{m})), \tag{13}$$

where $\mathbf{W_p} \in \mathbb{R}^{D \times D}$. We can also extend the memory components to handle multiple hop operations by stacking the memories sequentially which leaves for the future work.

2.4 Loss function

We train our model with a multi-label max-margin ranking objective as follows:

$$\mathcal{L} = \sum_{pos \in Y} \sum_{neg \in \overline{Y}} max(0, 1 - y_{pos} + y_{neg}). \tag{14}$$

Given example mention x, Y is the set of correct types assigned to x, y_{pos} is the possibility for such a positive assignment. In contrast, $\overline{Y}$ is the set of incorrect assigned types. y_{neg} is the possibility to assign a false label $neg \in \overline{Y}$ to x.

3 Experiments

3.1 Datasets

We evaluate the performance of our model on three public datasets, which are wildly used in FNET task. Type statistics on three datasets are shown in Table 1.

BBN (Weischedel and Brunstein, 2005) consists of 2,311 WSJ articles that are manually annotated using 93 types in a 2-level hierarchy.

OntoNote (Weischedel et al., 2011) has 13,109 news documents where 77 test documents are manually annotated using 89 types in 3-level hierarchy.

Wiki (Ling and Weld, 2012) consists of 1.5M sentences sampled from 780k Wikipedia articles. 434 news sentences are manually annotated for evaluation. 113 entity types are organized into a 2-level hierarchy.

Dataset	level-1	level-2	level-3
BBN	15	24	–
OntoNote	4	34	21
Wiki	22	28	–

Table 1: Type statistics on three datasets. Level-1, means coarse-grained type, while level-2 and level-3 are the fine-grained.

3.2 Zero-shot Setting

We follow Ma et al. (2016) and Obeidat et al. (2019) to apply the zero-shot setting. Entity types in 3 datasets are hierarchical-structured. So, we let the training set only contain coarse-grained types (level-1), while the testing set includes all fine-grained types (level-2). Especially, from Table 1, we can see that OntoNotes only possesses 4 level-1 types. Hence, we combine the level-1 and level-2 as the coarse-grained typing for training, and level-3 as the fine-grained types for testing.

3.3 Baselines

We compare the proposed method (MZET) and its variants with state-of-the-art FNET neural models. But rare research approach zero-shot FNET without auxiliary resource or hand-crafted features. In such a situation, we select the benchmarks and baselines as follows:

DZET Obeidat et al. (2019) propose a neural structure to extract the mention representations but leverage Wikipedia to augment the label representations. So we only compare with them on the learned mention representation capability, and incorporate our label embedding methods to construct this baseline. Unfortunately, we cannot get their code and preprocessed dataset, so we re-implement their neural architecture as described in the paper, and evaluate it on our datasets.

OTyper Yuan and Downey (2018) devise a neural model for FNET, but still utilize pre-prepared hand-crafted mention features. What's more, It is designed for open entity typing, which means it trains and tests the model on different datasets. Considered most parts of the model are learnable, we employ its results for comparison.

Variants of MZET for ablation study. We show some ablation models to estimate the most important part for MET, the variable aspects range from context representation, word-character representation, memory mechanism to label representation, especially performance over different label embedding methods (Prototype from Ma et al. (2016), averaging word embedding from (Obeidat et al., 2019; Yuan and Downey, 2018; Anand et al., 2017)).

Unfortunately, to the best of our knowledge, the system proposed by Ma et al. (2016) is not available online for empirical comparison.

3.4 Training and Implementation Details

To train the neural network models we optimize the multi-label max-margin loss function over training data concerning all model parameters. We adopt the Adam optimization algorithm with a decreasing learning rate of 0.0005, the decay rate of 0.9. We utilize the pre-trained cased BERT base with the number of transformer blocks is 12, the hidden layer size is 768, the number of self-attention heads is 12. We also choose GloVe pre-training embeddings of size 300 for word-character representation. The hidden state of LSTMs is in size of 200.

3.5 Evaluation Metrics

Following the prior works (Ling and Weld, 2012; Ma et al., 2016; Obeidat et al., 2019), we evaluate our methods and baselines on the metrics: strict accuracy (Acc.), Marco-F1, Micro-F1. Given a

Methods	**Overall**			**Level 1**			**Level 2**		
	Acc	Ma-F1	Mi-F1	Acc	Ma-F1	Mi-F1	Acc	Ma-F1	Mi-F1
DZET + bert	0.214	0.481	0.509	0.517	0.634	0.665	0.207	0.234	0.246
MZET + proto	0.251	0.582	0.631	0.535	0.679	0.680	0.203	0.216	0.249
MZET + avg_emb	0.259	0.602	0.644	0.563	0.689	0.689	0.200	0.213	0.229
MZET + GCN	0.112	0.389	0.332	0.411	0.437	0.437	0.089	0.131	0.131
MZET	**0.293**	**0.606**	**0.687**	**0.701**	**0.713**	**0.713**	**0.289**	**0.301**	**0.319**

Table 2: Fine-grained entity typing evaluation on BBN dataset. DZET+bert, means mention embedding form Obeidat et al. (2019) and label embedding from BERT, which show the best for DZET over the others (proto,avg_emb, GCN). We replace BERT label embedding in MZET with prototype (+proto) from Ma et al. (2016), average GloVe embedding (+avg_emb) from Shimaoka et al. (2016); Anand et al. (2017); Yuan and Downey (2018). Besides, we also attempt GCN (+GCN) to capture the hierarchical information as label embedding.

Methods	**BBN**			**OntoNotes**			**Wiki**		
	Acc	Ma-F1	Mi-F1	Acc	Ma-F1	Mi-F1	Acc	Ma-F1	Mi-F1
DZET+bert	0.214	0.481	0.509	0.231	0.276	0.281	0.285	0.551	0.560
OTYPER$_{Wiki}$	0.270	0.495	0.503	0.316	0.345	0.321	–	–	–
OTYPER$_{BBN}$	–	–	–	0.025	0.051	0.054	0.053	0.115	0.115
OTYPER$_{OntoNotes}$	0.236	0.511	0.479	–	–	–	0.004	0.156	0.168
MZET	**0.293**	**0.606**	**0.687**	**0.337**	**0.423**	**0.437**	**0.319**	**0.555**	**0.579**

Table 3: Evaluation on 3 benchmark datasets. OTYPER is designed for open type typing, which means, training on a dataset, but testing on other datasets. Here OTYPER$_{Wiki}$ is training on Wiki data, but testing on BBN and OntoNotes. OTYPER$_{BBN}$ and OTYPER$_{OntoNotes}$ are similar with OTYPER$_{Wiki}$, but training on BBN and OntoNote, respectively.

collection of mention M, we denote the set of the ground truth and predicted labels of a mention $m \in M$ as Y_m and $\hat{Y}_m$, respectively.

- Strict Accuracy (Acc.): $\frac{\sum_{m \in M} \sigma(Y_m = \hat{Y}_m)}{M}$, where $\sigma(\cdot)$ is an indicator function.
- Macro-F1: based on Macro-Precision(P_{ma}) and Macro-Recall(R_{ma}), where $P_{ma} = \frac{1}{|M|} \sum_{m \in M} \frac{|Y_m \cap \hat{Y}_m|}{\hat{Y}_m}$, $R_{ma} = \frac{1}{|M|} \sum_{m \in M} \frac{|Y_m \cap \hat{Y}_m|}{Y_m}$
- Micro-F1: based on Micro-Precision(P_{mi}) and Micro-Recall(R_{mi}), where $P_{mi} = \frac{\sum_{m \in M} |Y_m \cap \hat{Y}_m|}{\sum_{m \in M} \hat{Y}_m}$, $R_{mi} = \frac{\sum_{m \in M} |Y_m \cap \hat{Y}_m|}{\sum_{m \in M} Y_m}$

3.6 Results and Discussion

Zero-Shot FNET Evaluation We first evaluate our methods for the zero-shot FNET on BBN dataset. Zero-shot setting is that we keep the coarse-grained type for training the models, while testing in 3 ways: (1) Overall, predicting on both coarse-grained and fine-grained testing types; (2) Level 1, predicting only on coarse-grained testing types; (3) Level 2, predicting only on fine-grained testing types which are the unseen before.

Table.2 illustrates the performance of the baselines and the variants of MZET on these 3 aspects. The variants adopt the methods of label embedding designed in previous researches. We see that for the coarse-grained typing (Level 1), MZET earns significant improvements with at least 13% strict accuracy. Obviously, this benefits from both mention and label embedding methods after the comparison over DZET+bert with MZET and the variants of MZET. For the zero-shot setting for fine-grained typing (Level 2), MZET achieves the highest scores and gains up to 9% on strict accuracy. Such considerable improvements mainly appreciate the advantage of BERT label embedding based on the results from variants of MZET. At last, performance on all-grained types indicates the superiority of MZET over the rest, especially for the Micro-F1, which always tells the achievements over infrequent types.

Evaluation on Benchmark Datasets In this part, we display the comparative performances on three benchmark datasets, BBN, OntoNotes, and Wiki. Additionally, we compare with the results from OTyper, which is designed for open entity typing. OTYPER$_{Wiki}$ trains the model on the Wiki dataset, but tests on BBN and OntoNote without separating the data by granularity. OTYPER$_{BBN}$

Methods	Overall			Level 1			Level 2		
	Acc	Ma-F1	Mi-F1	Acc	Ma-F1	Mi-F1	Acc	Ma-F1	Mi-F1
MZET	**29.3**	**60.6**	**68.7**	**70.3**	**71.3**	**71.3**	**23.3**	**30.1**	**31.9**
MZET - Memory	- 3.3	- 2.3	- 3.4	- 1.0	- 1.0	- 1.0	- 2.9	- 2.2	- 3.0
MZET - Cntxt_Attn	- 2.0	- 1.8	- 2.2	- 1.2	- 1.3	- 1.3	- 1.3	- 1.3	- 1.5
MZET - Word_Char	- 3.4	- 2.5	- 3.1	- 2.0	- 2.2	- 2.2	- 2.6	- 2.0	- 2.5
MZET - BERT$_m$	- 1.7	- 1.5	- 1.9	- 1.5	- 1.7	- 1.7	- 1.0	- 1.2	- 1.4

Table 4: Ablation study on BBN dataset. All the results are percentages. The minus number means performance drop after remove or replace the methods. (-Memory) means replacing the memory part (2.3) with regular zero-shot mapping function like the way from Ma et al. (2016). (-Cntxt_Attn) means removing context representation m_c in 2.1.2. (-Word_Char) means removing word and character representation m_w in 2.1.1. (-BERT$_m$) means remove mention embedding m_b by BERT (2.1.2).

and OTYPER$_{OntoNotes}$ are in the same way to set training and testing sets. As Table 3 shows, there are significant outperformances of MZET on small datasets, BBN and OntoNotes. Furthermore, for the large-sized Wiki, MZET also obtains the highest scores for all metrics. Besides, Zero-Shot methods, MZET and DZET, surpass a lot OTYPER. That means open type FNET suffers from the drawbacks of highly dependent on the volume of the corpus in hand for training. But, zero-shot models are bestowed more flexibility and independence to attest their robustness on unseen datasets.

Ablation Study We carry out ablation studies that quantify the contribution of each components in our framework Figure1. As Table 4 shows, the vital parts are the memory network and the word and character representation. Because after removing each of them, the performance declines significantly with over 2.5% loss in strict accuracy for unseen data. The memory network contributes decent augmentations on fine-grained typing, which indicates the noteworthy associations between the seen labels and mentions, seen labels and unseen labels. The word and character representation show its importance on capturing the morphological and semantic information for a single entity mention. The secondary important is the informative context part with attention. It is aggregated into the final representation of the mention to guide the classification. Last, m_b plays a considerable complementary role, as leading the BERT to embed a mention enables the model to gather more contextual information to avoid ambiguity for the polysemantic, like the word "valley" in mention "Silicon Valley".

Error Analysis We also provide insights into specific reasons for the mistakes made by our model.

First, all the datasets follow long-tail frequency distributions. So the examples for each label are significantly imbalanced. Accordingly, the model is prone to assign frequent types for the infrequent ones. For example, the training set processes 719 examples of "/LOCATION" and 6,672 examples of "/GPE" (Geopolitical Entity). The model prefers predicting on the fine-grained type "/GPE/CITY" rather than "/LOCATION/REGION".

Second, types are incorrect tagged in the raw data. To test the ratio for incorrect tagging, we randomly pick out 100 examples in the raw data, including types coming from both training and testing sets. We find there are about 15% for BBN, 11% for OntoNotes, 17% for Wiki with noise, such as mentions with incoherent labels, or missing the correct mention words for the corresponding tagged labels. For example,

"*The harvest arrives in plenty after last year 's drought-ravaged effort : **The** government estimates corn output at 7.45 billion bushels , up 51% from last fall.*"
— labels: ["/ORGANIZATION/CORPORATION", "/ORGANIZATION"]
The correct mention should be the "*The government*", as it exactly matches the assigned labels.

4 Related Work

NET is a long-standing task in Natural Language Processing. For the early development (Chinchor and Robinson, 1997; Sang and De Meulder, 2003; Doddington et al., 2004; Fleischman and Hovy, 2002), NET only concerns with a tiny set of coarse-grained types. Further on, Ling and Weld (2012) derive 113 entity types from Freebase (Bollacker et al., 2008) and boosted the set of entity types in size and level of granularity. Therefore, recent researches underline much the NET for fine-grained types (FNET) (Ling and Weld, 2012; Nakashole et al., 2013; Del Corro et al., 2015; Ren et al., 2016; Ma et al., 2016; Shimaoka et al., 2016; Anand et al., 2017; Huang et al., 2016; Yuan and Downey, 2018; Zhou et al., 2019; Obeidat et al., 2019).

Most of the proposed FNET methods are based on a distant supervisor, but diverse in classification architectures. Ling and Weld (2012) propose multi-label and multi-class multilayer perceptron model assigns each mention of the corresponding label tags. Nakashole et al. (2013) type newly emerging out-of Knowledge Base entities by a fine-grained typing system and harnesses relational paraphrase with type signatures for probabilistic weight computation. Del Corro et al. (2015) designs a system, FINET, for detecting the types of named entities in short inputs concerning WordNet (Miller, 1995) fine-grained type system. Ren et al. (2016) propose AFET for automatic fine-grained entity typing and hierarchical partial-label embedding. A joint optimization framework is designed to learn embeddings for mentions by hand-crafted features and label embedding from the hierarchical type path and iteratively refined until the convergence. Shimaoka et al. (2016) propose an attentive neural network model that uses LSTMs to encode the entity mention and its context, then incorporate an attention mechanism to focus more on the representation over each mention with contextural information. Anand et al. (2017) propose another neural network to obtain the mention representations with contextual information and incorporate label noise information in a variant of the non-parametric hinge loss function. Those methods develope, from hand-crafted features to neural network learned features, to allow fine-grained typing system fancy, automatic and effective. But their architectures can not apply to new and unseen entity types.

To handle unseen types, zero-shot learning is introduced. But hitherto, limited research works are counted for that. One way for unseen entity typing is clustering. Zhou et al. (2019); Huang et al. (2016) cluster mentions, followed by propagating type information from representative mentions to applied on unseen types. Ma et al. (2016) propose a prototype-driven label embedding method for zero-shot FNET. They map the mention and label embedding into a shared latent space, then compute the rating score for each mention-label pair. Even zero-shot setting has been applied, they only utilize hand-crafted mention features and the results focus most on the few-shot setting. They only provide top-k precision for zero-shot, lack of Micro-F1 and Macro-F1 evaluation scores. Yuan and Downey (2018); Obeidat et al. (2019) also harness shared space to allow the correct type close to the mention like what Ma et al. (2016) do. Even they (Yuan and Downey, 2018; Obeidat et al., 2019) construct neural architectures with attention mechanism, but the features of mention are hybrid over learned and hand-crafted. What's more, Obeidat et al. (2019) exploit expanding information for label embedding by Wikipedia to obtain informative label representation. The aforementioned attention mechanism only applied to mention and its context. It ignores the connection between mention and label, which is adopted in our methods and boosts the performance a lot.

5 Conclusions

In this paper, we propose MZET, a zero-shot neural network FNET model to enable FNET on unseen types. It extracts the representations concerning word and character, mention, mention's context, and raw label text without auxiliary information. Then it adopts the memory network to gather the representations for zero-shot paradigm. Extensive experiments on three public datasets show prominent performances obtained by MZET, which surpasses the state-of-the-art neural network models for Zero-Shot FNET.

The contribution of this work is three-fold: First, such a novel neural network model can handle zero-shot FNET problems based on the information of the raw data without the assistant of additional augment resources. Second, we incorporate memory networks to indicate the connections mentions and labels and enable the zero-shot paradigm. Third, the robust performance of MZET attests to its contribution to the zero-shot FNET task.

References

Ashish Anand, Amit Awekar, et al. 2017. Fine-grained entity type classification by jointly learning representations and label embeddings. *arXiv preprint arXiv:1702.06709*.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250.

Nancy Chinchor and Patricia Robinson. 1997. Muc-7 named entity task definition. In *Proceedings of the 7th Conference on Message Understanding*, volume 29, pages 1–21.

Luciano Del Corro, Abdalghani Abujabal, Rainer Gemulla, and Gerhard Weikum. 2015. Finet: Context-aware fine-grained named entity typing. In *EMNLP*, pages 868–878.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

George R Doddington, Alexis Mitchell, Mark A Przybocki, Lance A Ramshaw, Stephanie M Strassel, and Ralph M Weischedel. 2004. The automatic content extraction (ace) program-tasks, data, and evaluation. In *Lrec*, volume 2, page 1. Lisbon.

Michael Fleischman and Eduard Hovy. 2002. Fine grained classification of named entities. In *Proceedings of the 19th international conference on Computational linguistics-Volume 1*, pages 1–7. Association for Computational Linguistics.

Sangdo Han, Soonchoul Kwon, Hwanjo Yu, and Gary Geunbae Lee. 2017. Answer ranking based on named entity types for question answering. In *Proceedings of the 11th International Conference on Ubiquitous Information Management and Communication*, pages 1–4.

Lifu Huang, Jonathan May, Xiaoman Pan, and Heng Ji. 2016. Building a fine-grained entity typing system overnight for a new x (x= language, domain, genre). *arXiv preprint arXiv:1603.03112*.

Xiao Ling and Daniel S Weld. 2012. Fine-grained entity recognition. In *Twenty-Sixth AAAI Conference on Artificial Intelligence*.

Yang Liu, Kang Liu, Liheng Xu, and Jun Zhao. 2014. Exploring fine-grained entity type constraints for distantly supervised relation extraction. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 2107–2116.

Yukun Ma, Erik Cambria, and Sa Gao. 2016. Label embedding for zero-shot fine-grained named entity typing. In *COLING*, pages 171–180.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Ndapandula Nakashole, Tomasz Tylenda, and Gerhard Weikum. 2013. Fine-grained semantic typing of emerging entities. In *ACL*, pages 1488–1497.

Rasha Obeidat, Xiaoli Fern, Hamed Shahbazi, and Prasad Tadepalli. 2019. Description-based zero-shot fine-grained entity typing. In *NAACL*, pages 807–814.

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 *EMNLP*, pages 1369–1378.

Erik F Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. *arXiv preprint cs/0306050*.

Sonse Shimaoka, Pontus Stenetorp, Kentaro Inui, and Sebastian Riedel. 2016. Neural architectures for fine-grained entity type classification. *arXiv preprint arXiv:1606.01341*.

Rosa Stern, Benoît Sagot, and Frédéric Béchet. 2012. A joint named entity recognition and entity linking system. In *Proceedings of the Workshop on Innovative Hybrid Approaches to the Processing of Textual Data*, pages 52–60. Association for Computational Linguistics.

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

Ralph Weischedel, Eduard Hovy, Mitchell Marcus, Martha Palmer, Robert Belvin, Sameer Pradhan, Lance Ramshaw, and Nianwen Xue. 2011. Ontonotes: A large training corpus for enhanced processing. *Handbook of Natural Language Processing and Machine Translation. Springer*, page 59.

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

Ben Zhou, Daniel Khashabi, Chen-Tse Tsai, and Dan Roth. 2019. Zero-shot open entity typing as type-compatible grounding. *arXiv preprint arXiv:1907.03228*.