An Empirical Study on Multiple Information Sources for Zero-Shot Fine-Grained Entity Typing

Yi Chen[1], **Haiyun Jiang**[1], **Lemao Liu**[1], **Shuming Shi**[1], **Chuang Fan**
Min Yang[2], **Ruifeng Xu**[3*]
[1]Tencent AI Lab, Shenzhen, China
[2]Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences
[3]Peng Cheng Laboratory, Shenzhen, China
yichennlp@gmail.com, haiyunjiang@tencent.com
lemaoliu@gmail.com, shumingshi@tencent.com
fanchuanghit@gmail.com, min.yang@siat.ac.cn
xuruifeng.hitsz@gmail.com

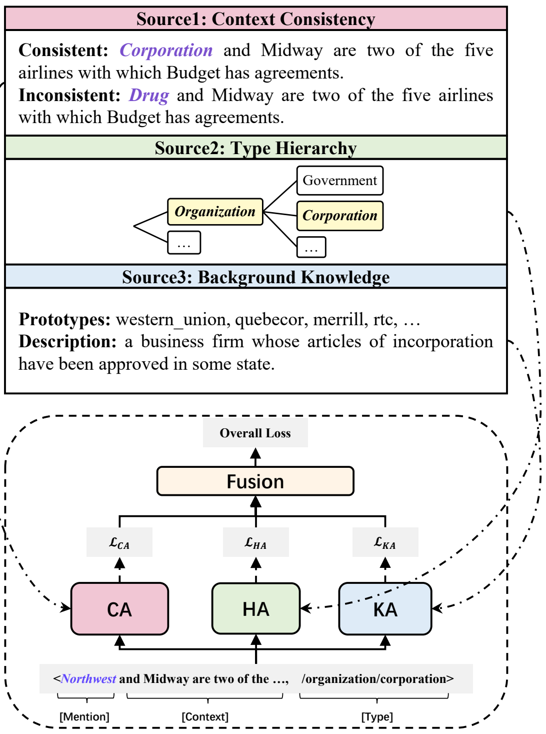
Abstract

Auxiliary information from multiple sources has been demonstrated to be effective in zero-shot fine-grained entity typing (ZFET). However, there lacks a comprehensive understanding about how to make better use of the existing information sources and how they affect the performance of ZFET. In this paper, we empirically study three kinds of auxiliary information: context consistency, type hierarchy and background knowledge (e.g., prototypes and descriptions) of types, and propose a multi-source fusion model (MSF) targeting these sources. The performance obtains up to 11.42% and 22.84% absolute gains over state-of-the-art baselines on BBN and Wiki respectively with regard to macro F1 scores. More importantly, we further discuss the characteristics, merits and demerits of each information source and provide an intuitive understanding of the complementarity among them.

1 Introduction

Fine-grained entity typing (FET) aims to detect the types of an entity mention given its context (Abhishek et al., 2017; Xu and Barbosa, 2018; Jin et al., 2019). The results of FET benefit lots of downstream tasks (Chen et al., 2020; Hu et al., 2019; Zhang et al., 2020a; Liu et al., 2021; Chu et al., 2020). In many scenarios, the type hierarchy is continuously evolving, which requires newly emerged types to be accounted into FET systems. As a result, zero-shot FET (ZFET) is welcomed to handle the new types which are unseen during training stage (Ma et al., 2016; Ren et al., 2020; Zhang et al., 2020b).

The major challenge of ZFET is to build the semantic connections between the *seen* types (during training) and the *unseen* ones (during inference).


Source1: Context Consistency

Consistent: *Corporation* and Midway are two of the five airlines with which Budget has agreements.
Inconsistent: *Drug* and Midway are two of the five airlines with which Budget has agreements.

Source2: Type Hierarchy

Source3: Background Knowledge

Prototypes: western_union, quebecor, merrill, rtc, …
Description: a business firm whose articles of incorporation have been approved in some state.

<*Northwest* and Midway are two of the …, /organization/corporation>

Figure 1: Illustration of the proposed multi-source fusion model (MSF).

Auxiliary information has been proved to be essential in this regard (Xian et al., 2019), with a variety of approaches focused on scattered information (Ma et al., 2016; Zhou et al., 2018; Obeidat et al., 2019; Ren et al., 2020; Zhang et al., 2020b). However, the power of auxiliary information has not been sufficiently exploited in existing solutions. Besides, the effects of each information source also remain to be clearly understood.

In this paper, we propose a **M**ulti-**S**ource **F**usion model (MSF) integrating three kinds of popular auxiliary information for ZFET, i.e., context consistency, type hierarchy, and background knowledge, as illustrated in Figure 1. (i) *Context consistency* means a correct type should be se-

[*]Corresponding Author

mantically consistent with the context if we replace the mention with the type name in the context. Type name is the surface form of a type, which is a word or a phase, e.g., type name of `/organization/corporation` is *corporation*. (ii) *Type hierarchy* is the ontology structure connecting seen and unseen types. (iii) *Background knowledge* provides the external prior information that depicts types in detail, e.g., prototypes (Ma et al., 2016) and descriptions (Obeidat et al., 2019).

MSF is composed of three modules, with each targeting a specific information source. (i) In the CA (**C**ontext-Consistency **A**ware) module, we measure the context consistency by large-scale pre-trained language models, e.g., BERT (Devlin et al., 2019). By masking mentions and predicting the names of ground truth types through finetuning on the data of seen types, CA is expected to measure the context consistency of unseen types more precisely. (ii) In the HA (Type-**H**ierarchy **A**ware) module, we use Transformer encoder (Vaswani et al., 2017) to model the hierarchical dependency among types. There have been substantial works exploring type hierarchy in the supervised typing task (Shimaoka et al., 2017; Xu and Barbosa, 2018; Xiong et al., 2019), but only some preliminary research in ZFET (Ma et al., 2016; Zhang et al., 2020b). (iii) In the KA (Background-**K**nowledge **A**ware) module, we introduce prototypes (Ma et al., 2016) and WordNet descriptions (Miller, 1995) as background knowledge of types. KA is embodied as natural language inference with a translation-based solution to better incorporate knowledge.

Extensive experiments are carried out to verify the effectiveness of the proposed fusion model. We also conduct a deep analysis on the characteristics, merits and demerits of each information source. We find that, similar to type hierarchy, background knowledge also implies some hierarchical information through the shared prototypes and the descriptions semantically similar with their parent types. Besides, the context consistency is an essential clue in handling long-tail unseen types and longer contexts. Moreover, we further discuss the complementarity among different information sources and their contributions to the proposed fusion model.

In summary, our contributions are as follows:

- We propose a multi-source fusion model integrating multiple information sources for ZFET, which achieves new state-of-the-art results on BBN and Wiki.
- We are the first work to conduct a comprehensive study on the strengths and weaknesses of three auxiliary information sources for ZFET. Besides, we also make a deep analysis about how different information sources complement each other and how they contribute to the proposed fusion model.

2 A Multi-Source Fusion Model

2.1 Overview

Zero-shot Fine-grained Entity Typing (ZFET) is defined on a type set $\mathcal{T} = \mathcal{T}_{train} \cup \mathcal{T}_{test}$, which forms a hierarchy. During inference, ZFET aims to identify the correct types for a mention m based on its context c, where the target types are unseen during the training stage, i.e., $\mathcal{T}_{train} \cap \mathcal{T}_{test} = \emptyset$.

As shown in Figure 1, we propose a **M**ulti-**S**ource **F**usion model (MSF) that captures information from these sources and integrates them to make a better prediction under the zero-shot scenario. In the following, we first describe the details of each module (Sec 2.2, 2.3 and 2.4), and then present the joint loss function and inference details (Sec 2.5).

2.2 Context-Consistency-Aware (CA) Module

We base the CA module upon the pre-trained BERT (Devlin et al., 2019) and fine-tune it for assessment of context consistency.

2.2.1 Fine-tuning by Masking Mentions

Vanilla BERT randomly masks some input tokens and then predicts them. Nevertheless, in fine-tuning stage for ZFET, CA only masks the entity mentions and predicts their type names instead. For instance, given the context in Figure 1, we replace the entity mention `Northwest` with a `[MASK]` token and let CA module predict the name `corporation` of the target type `/organization/corporation` with a higher score. In more general cases, the length of a type name may exceed 1. Thus, the number of `[MASK]` tokens for replacement depends on the length of the type name (e.g., for type name `living thing`, we replace the corresponding mention with `[MASK] [MASK]`).

2.2.2 Loss Function for CA Module

For each mention m in the training set, we denote its ground-truth types as $\mathcal{T}_{pos}$. For each type t in $\mathcal{T}_{pos}$, we replace m with l `[MASK]` tokens in the context of m, where l is the length of t's type name. We define the score s_t and loss ℓ_t for type t as

$$s_t = \frac{1}{|n_t|}\sum_{k=1}^{|n_t|} p_{n_{t,k}}, \quad \ell_t = -\frac{1}{|n_t|}\sum_{k=1}^{|n_t|} \log p_{n_{t,k}} \tag{1}$$

where $p_{n_{t,k}}$ is the probability for the k-th token of type name n_t predicted by BERT. Considering all the types in $\mathcal{T}_{pos}$, the overall loss for mention m is:

$$\mathcal{L}_{m,CA} = \sum_{t \in \mathcal{T}_{pos}} \ell_t \tag{2}$$

Note, the vocabulary of BERT contains all the constituent tokens of all the type names in $\mathcal{T}_{train}$ and $p_{n_{t,k}}$ is the output of the Softmax function over the vocabulary, minimizing the loss above will also punish scores of negative types in $\mathcal{T}_{train}$.

2.3 Type Hierarchy-Aware (HA) Module

In HA module, we use Transformer encoder (Vaswani et al., 2017) with mask-self-attention to capture the hierarchical information for better type representations. Besides, we take the encoder from Lin and Ji (2019) to learn the features of mentions and contexts. Then a similarity function is defined to compute the matching score between a mention and a candidate type based on the context.

2.3.1 Mention-Context Encoder

In the mention-context encoder, an entity mention and its context are represented as the weighted sum of their ELMo word representations. Then the mention representation $\boldsymbol{r}_m$ and context representation $\boldsymbol{r}_c$ are concatenated as the final representation: $\boldsymbol{r}_{mc} = \boldsymbol{r}_m \oplus \boldsymbol{r}_c$, where $\boldsymbol{r}_m, \boldsymbol{r}_c \in \mathbb{R}^{d_m}$, $\boldsymbol{r}_{mc} \in \mathbb{R}^{2d_m}$, $\oplus$ denotes concatenation.

2.3.2 Hierarchy-Aware Type Encoder

Given a type set $\mathcal{T} = \mathcal{T}_{train} \cup \mathcal{T}_{test}$ and its hierarchy structure Ψ, we denote the initialized type embeddings[1] as $\boldsymbol{E} = [\boldsymbol{e}_{t_1}, \boldsymbol{e}_{t_2}, ..., \boldsymbol{e}_{t_N}]$, which are the inputs of Transformer encoder, where $\boldsymbol{e}_{t_i}$ is the embedding for the i-th type t_i, N is the size of $\mathcal{T}$. Note that *the positional embeddings are removed since the input type sequence is disordered.* To inject the hierarchical information, we perform the mask-self-attention operation on types. Specifically, during the process of computing self-attention in Transformer encoder, a type only attends to its parent type in the hierarchy and itself, while the attention to the remaining types will be masked. We omit other details and denote the final representation for each type $t \in \mathcal{T}$ as $\boldsymbol{r}_t \in \mathbb{R}^{d_t}$.

[1]The initialization details are presented in Appendix A

2.3.3 Loss Function for HA Module

Given a mention m and a candidate type $t \in \mathcal{T}_{train}$, we first map the mention representation $\boldsymbol{r}_{mc}$ and type representation $\boldsymbol{r}_t$ into a shared space by

$$\begin{aligned} \phi(\boldsymbol{r}_{mc}, \boldsymbol{A}) &\ : \boldsymbol{r}_{mc} \to \boldsymbol{A}\boldsymbol{r}_{mc} \\ \theta(\boldsymbol{r}_t, \boldsymbol{B}) &\ : \boldsymbol{r}_t \to \boldsymbol{B}\boldsymbol{r}_t\ , \end{aligned} \tag{3}$$

where $\boldsymbol{A} \in \mathbb{R}^{d_s \times 2d_m}$ and $\boldsymbol{B} \in \mathbb{R}^{d_s \times d_t}$ are learnable matrices. The matching score is defined as

$$y_t = \phi(\boldsymbol{r}_{mc}, \boldsymbol{A}) \cdot \theta(\boldsymbol{r}_t, \boldsymbol{B}) = (\boldsymbol{A}\boldsymbol{r}_{mc})^\top \boldsymbol{B}\boldsymbol{r}_t \tag{4}$$

During training, we match mention m with all the types in $\mathcal{T}_{train}$, so the loss function for m is:

$$\mathcal{L}_{m,HA} = \text{CrossEntropy}(\boldsymbol{y}, \hat{\boldsymbol{y}})\ , \tag{5}$$

where $\hat{\boldsymbol{y}} \in \mathbb{R}^{|\mathcal{T}_{train}|}$ denotes the binary vector for the ground-truth types of m with 1 for positive and 0 for negative. $|\mathcal{T}_{train}|$ denotes the size of $\mathcal{T}_{train}$. $\boldsymbol{y} \in \mathbb{R}^{|\mathcal{T}_{train}|}$ denotes the predicted score vector.

Although the HA module does not directly learn any knowledge from instances of $\mathcal{T}_{test}$, by encoding the type hierarchy Ψ using mask-self-attention, Transformer encoder will capture the semantic correlation between types in $\mathcal{T}_{train}$ and $\mathcal{T}_{test}$, thus producing reliable representations for types in $\mathcal{T}_{test}$.

2.4 Background Knowledge-Aware (KA) Module

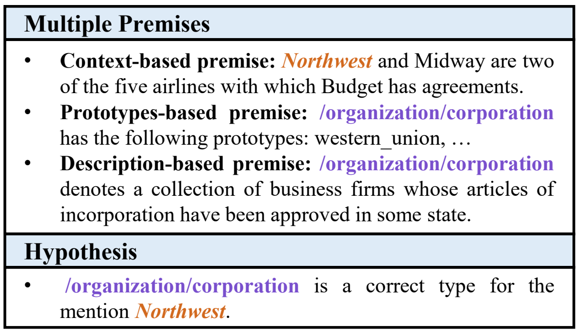
We introduce prototypes and descriptions as two kinds of knowledge in the KA module.

Prototypes refer to the carefully selected mentions for a type based on Normalized Point-wise Mutual Information (NPMI), which provide a mention-level summary for types (Ma et al., 2016).

Descriptions are queried from WordNet glosses (Miller, 1995) by type names, which provide a brief high-level summary for each type.

2.4.1 Inference from Background Knowledge

We hope to infer whether a mention m matches a candidate type t, given the prototypes, type description and the context. In this work, we embody the KA module as natural language inference (NLI) from multiple premises (Lai et al., 2017). An example is presented in Figure 2, with input the same


Multiple Premises

- **Context-based premise:** *Northwest* and Midway are two of the five airlines with which Budget has agreements.
- **Prototypes-based premise:** **/organization/corporation** has the following prototypes: western_union, ...
- **Description-based premise:** **/organization/corporation** denotes a collection of business firms whose articles of incorporation have been approved in some state.

Hypothesis

- **/organization/corporation** is a correct type for the mention *Northwest*.

Figure 2: An example to illustrate the multiple-premises and the hypothesis for KA.

as Figure 1. We construct three premises corresponding to the context, prototypes and description respectively. The target hypothesis encodes that "the type is correct for the mention". For both premises and hypothesis, we organize them into natural language sentences.

We reuse the Mention-Context Encoder in Sec 2.3.1 to obtain representations for the context-based premise, i.e., $\boldsymbol{r}_{mc} = \boldsymbol{r}_m \oplus \boldsymbol{r}_c$, where $\boldsymbol{r}_m$ and $\boldsymbol{r}_c$ represent the mention and context respectively. To encode the prototypes-based and description-based premises, we also use the same encoder, where the type is aligned with the mention while the rest of the sentence is aligned with the context of the mention. We denote the premises based on prototypes and description as $\boldsymbol{r}_{tp} = \boldsymbol{r}_t \oplus \boldsymbol{r}_p$ and $\boldsymbol{r}_{td} = \boldsymbol{r}_t \oplus \boldsymbol{r}_d$, where $\boldsymbol{r}_t, \boldsymbol{r}_p, \boldsymbol{r}_d \in \mathbb{R}^{d_m}$ are considered as the representations for the type, prototypes-based and description-based sentences respectively. Since the hypotheses for the same mention targeting different types have the same word sequences except for the type spans , we simplify the representation of hypothesis as $\boldsymbol{r}_h = \boldsymbol{r}_t \oplus \boldsymbol{r}_m \in \mathbb{R}^{2d_m}$, where $\boldsymbol{r}_t$ and $\boldsymbol{r}_m$ are the type and mention representations directly taken from $\boldsymbol{r}_{mc}$ and $\boldsymbol{r}_{tp}$, In the KA module, the encoders for all the premises and hypothesis share the parameters in ELMo.

Loss Function for KA Module Motivated by TransE (Bordes et al., 2013) and TransR (Lin et al., 2015), we propose a simple translation-based solution for NLI by extending the translation operations over triples to quadruples, i.e., (*context-based premise, prototypes-based premise, description-based premise, hypothesis*).

Given a mention m and a candidate type t, we first use the matrix $\boldsymbol{W}$ to project all the representations to a new space for inference:

$$\{\tilde{\boldsymbol{r}}_{mc}, \tilde{\boldsymbol{r}}_{tp}, \tilde{\boldsymbol{r}}_{td}, \tilde{\boldsymbol{r}}_h\} = \boldsymbol{W}\{\boldsymbol{r}_{mc}, \boldsymbol{r}_{tp}, \boldsymbol{r}_{td}, \boldsymbol{r}_h\} \quad (6)$$

where $\boldsymbol{W} \in \mathbb{R}^{d_w \times 2d_m}$. We hope that $\tilde{\boldsymbol{r}}_{mc} + \tilde{\boldsymbol{r}}_{tp} + \tilde{\boldsymbol{r}}_{td} \approx \tilde{\boldsymbol{r}}_h$ when the hypothesis can be inferred from the premises, i.e., the type t is correct for the mention m under the context c. Thus, we try to minimize their squared euclidean distance

$$\mathcal{D}_t = \|\tilde{\boldsymbol{r}}_{mc} + \tilde{\boldsymbol{r}}_{tp} + \tilde{\boldsymbol{r}}_{td} - \tilde{\boldsymbol{r}}_h\|_2^2 \,, \quad (7)$$

with norm constraints, i.e., $\|\tilde{\boldsymbol{r}}_{mc}\|_2^2 = \|\tilde{\boldsymbol{r}}_{tp}\|_2^2 = \|\tilde{\boldsymbol{r}}_{td}\|_2^2 = \|\tilde{\boldsymbol{r}}_h\|_2^2 = 1$. Then the score for type t is defined as: $p_t = -\mathcal{D}_t$. The closer the distance, the higher the score. Finally, the loss function for mention m is:

$$\mathcal{L}_{m,KA} = \sum_{t \in \mathcal{T}_{pos}, t' \in \mathcal{T}_{neg}} \frac{\max(0, 1 - (p_t - p_{t'}))}{|\mathcal{T}_{\text{pos}}|} \,, \quad (8)$$

where $\mathcal{T}_{pos}$ are the ground-truth types of $\mathcal{T}_{train}$ for m with size $|\mathcal{T}_{pos}|$, while $\mathcal{T}_{neg}$ are the negative types in $\mathcal{T}_{train}$, i,e., $\mathcal{T}_{neg} = \mathcal{T}_{train} \setminus \mathcal{T}_{pos}$.

2.5 Training and Inference

Overall Loss Given a training mention m, we derived the loss from the aforementioned modules. Finally, the overall loss to train the fusion model is:

$$\mathcal{L} = \sum_{m \in \mathcal{M}} \mathcal{L}_{m,CA} + \mathcal{L}_{m,HA} + \mathcal{L}_{m,KA} \,, \quad (9)$$

where $\mathcal{M}$ denotes the training mention set.

Inference Given a test mention m and a candidate type t in $\mathcal{T}_{test}$, we first compute the scores from each module: s_t (by CA module), y_t (by HA module) and p_t (by KA module). Then we normalize them according to

$$x' = \text{sigmoid}(\frac{x - \mu_{\boldsymbol{x}}}{\sigma_{\boldsymbol{x}}}), x \in \{s_t, y_t, p_t\} \,, \quad (10)$$

where $\boldsymbol{x}$ is the score vector from a module for mention m towards all types $t \in \mathcal{T}_{test}$ with x as component. $\mu_{\boldsymbol{x}}$ and $\sigma_{\boldsymbol{x}}$ denote the mean and standard deviation of the vector $\boldsymbol{x}$. The final decision score by our fusion model for type t is:

$$\text{score}_t = \lambda_1 s'_t + \lambda_2 y'_t + \lambda_3 p'_t \,, \quad (11)$$

where $\lambda_1, \lambda_2, \lambda_3 \geq 0$ are hyper-parameters and $\lambda_1 + \lambda_2 + \lambda_3 = 1$.

3 Experimental Setup

3.1 Datasets and Evaluation Metrics

We evaluate our model on two widely-used datasets: BBN (Weischedel and Brunstein, 2005) and Wiki

(Ling and Weld, 2012). The version processed by Ren et al. (2016) is adopted for our experiments. Detailed statistics on two datasets are listed in Table 1. We do not use OntoNotes (Gillick et al., 2014) since it is hard to define the name, description and hierarchy for its special type `/other`. Types of both BBN and Wiki are organized into a 2-level hierarchy. There are 47 types in BBN and 113 types in Wiki. Following Ma et al. (2016); Zhang et al. (2020b), we use the coarse-grained (Level-1) types such as `/organization` for training (denoted as *seen* types), while the fine-grained (Level-2) types such as `/organization/corporation` are reserved for testing (denoted as *unseen* types).

Dataset	**BBN**		**Wiki**	
	train	test	train	test
# sentences	32.7K	6.3K	1.5M	276
# mentions	86.1K	12.3K	2.7M	563

Table 1: Statistics of training and test datasets.

Following prior works (Ling and Weld, 2012; Ma et al., 2016), we report all the popular metrics in our main results for a better comparison, i.e., strict accuracy (Acc), macro-averaged F1 (Ma-F1), micro-averaged F1 (Mi-F1) and micro-averaged precision (Mi-P).

3.2 Comparison Models

We abbreviate our **M**ulti-**S**ource **F**usion model as **MSF**, and compare it with the following baselines: (1) **Proto-HLE** (Ma et al., 2016) which introduces prototype-driven hierarchical label embedding for ZFET; (2) **ZOE** (Zhou et al., 2018) which infers the types of a given mention according to its type-compatible Wikipedia entries; (3) **DZET** (Obeidat et al., 2019) which derives type representations from Wikipedia pages and leverages a context-description matching approach for type inference; (4) **NZFET*** (Ren et al., 2020) which employs entity type attention to make the model focus on information relevant to the entity type; (5) **MZET*** (Zhang et al., 2020b) which adopts a memory network to connect the seen and unseen types.

Specifically, we compare MSF with its single-source modules: the **C**ontext-Consistency-**A**ware module (**CA**), the Type-**H**ierarchy-**A**ware module (**HA**) and the Background-**K**nowledge-**A**ware module (**KA**), as well as the variation **MSF**$_{avg}$ which simply averages scores from single-source modules (i.e., $\lambda_1, \lambda_2, \lambda_3 = 1/3$ in Equation 11).

All the results are reimplemented except the ones indicated by *. The implementation details and hyperparameter settings (e.g., $\lambda_1, \lambda_2, \lambda_3$ for MSF) are presented in Appendix A.

4 Experimental Results

4.1 Main Results

Table 2 and Table 3 present the results on BBN and Wiki, evaluated on both the unseen fine-grained types and the seen coarse-grained types.

Zero-shot Performance From Table 2, we see that our model significantly outperforms the baselines across the metrics. MSF gains up to 11.42% over DZET on BBN and 22.84% over ZOE on Wiki according to Ma-F1. Compared with MSF$_{avg}$, which treats each information source as equally important, MSF considers the importance of each source and achieves better performance on both datasets. Besides, the single-source modules of MSF (i.e., CA, HA and KA) also produce relatively promising results, among which KA yields the best scores. Nevertheless, MSF still surpasses these modules by a large margin, which verifies the necessity of information fusion for the ZFET task.

Supervised Performance Table 3 demonstrates the advantage of MSF in predicting the seen types, with Ma-F1 increased by 2.01% over Proto-HLE on BBN and 2.25% over DZET on Wiki. Besides, CA, HA and KA still maintain a highly competitive performance in this regard. Combined with Table 2, we find that the proposed MSF has a particular superiority on the unseen types, since the auxiliary information from multiple sources tends to be more helpful when short of annotated training samples.

4.2 Ablation Studies

We conduct ablation studies on the single-source modules of MSF. The results are shown in Table 4.

Ablations of CA We observe that the vanilla CA (i.e., the BERT-based CA module without fine-tuning, denoted as "CA w/o finetuning") has reached a certain level of performance. This indicates the potential of BERT for context consistency assessment thanks to its large-scale unsupervised pre-training technique. After fine-tuning with our modified mask mechanism, CA surpasses its vanilla version by 23.13% and 10.28% on BBN and Wiki respectively.

Ablations of HA We show that Transformer-based type encoder greatly contributes to the HA module. To validate it, we replace Transformer

Model	BBN				Wiki			
	Acc (%)	Ma-F1(%)	Mi-F1(%)	Mi-P(%)	Acc(%)	Ma-F1(%)	Mi-F1(%)	Mi-P(%)
Proto-HLE	49.65	49.65	49.65	49.65	23.76	23.76	23.36	23.76
ZOE	58.00	58.95	62.16	<u>65.33</u>	<u>33.67</u>	<u>34.82</u>	<u>34.50</u>	<u>35.03</u>
DZET	<u>62.60</u>	<u>62.60</u>	<u>62.60</u>	62.60	32.67	32.67	32.12	32.67
NZFET*	-	-	-	45.91	-	-	-	24.25
MZET*	28.80	30.10	31.60	-	-	-	-	-
CA	50.36	50.36	50.36	50.36	36.63	37.37	36.98	37.62
HA	62.49	62.49	62.49	62.49	35.15	36.99	36.98	37.62
KA	66.32	66.32	66.32	66.32	41.58	43.80	43.80	44.55
MSF$_{avg}$	70.90	70.90	70.90	70.90	50.00	52.58	52.55	53.47
MSF (ours)	**74.02**	**74.02**	**74.02**	**74.02**	**55.45**	**57.66**	**57.42**	**58.42**

Table 2: Performance on the unseen types. The best scores of baselines and all the models are <u>underlined</u> and **bold-faced** respectively. Since all the test mentions from BBN correspond to only one ground truth seen/unseen type, our implementations simply predict the candidate type with the highest score for BBN. This makes some results of different metrics the same on BBN.

Model	BBN				Wiki			
	Acc (%)	Ma-F1(%)	Mi-F1(%)	Mi-P(%)	Acc(%)	Ma-F1(%)	Mi-F1(%)	Mi-P(%)
Proto-HLE	<u>87.25</u>	<u>87.25</u>	<u>87.25</u>	<u>87.25</u>	68.17	72.37	70.62	73.92
ZOE	58.86	63.06	59.82	66.28	68.30	68.62	71.13	70.24
DZET	86.02	86.02	86.02	86.02	<u>82.73</u>	<u>87.21</u>	<u>84.88</u>	<u>88.85</u>
NZFET*	-	-	-	-	-	-	-	-
MZET*	70.70	71.00	71.00	-	-	-	-	-
CA	80.98	80.98	80.98	80.98	75.90	79.59	77.32	80.94
HA	84.57	84.57	84.57	84.57	83.99	88.46	86.08	90.11
KA	86.05	86.05	86.05	86.05	82.55	87.43	85.22	89.21
MSF$_{avg}$	88.65	88.65	88.65	88.65	84.17	88.91	86.60	90.65
MSF (ours)	**89.26**	**89.26**	**89.26**	**89.26**	**84.71**	**89.46**	**87.11**	**91.19**

Table 3: Performance on the seen types.

Source	Model	BBN	Wiki
context consistency	CA	**50.36**	**37.37**
	CA w/o finetuning	27.23	27.09
type hierarchy	Proto-HLE	49.65	23.76
	MZET*	30.10	-
	HA	**62.49**	**36.99**
	HA-Glove	52.48	18.32
	HA-HierMatrix	58.96	20.67
background knowledge	Proto-HLE	49.65	23.76
	DZET	62.60	32.67
	KA	**66.32**	**43.80**
	KA w/o Description	63.28	32.67
	KA w/o Prototypes	59.54	26.10

Table 4: Ablation results of CA, HA and KA, evaluated on the unseen types of BBN by Ma-F1 (%).

encoder with averaged Glove word embeddings to obtain type representations and denote it as "HA-Glove". Besides, we also implement the variation of HA that removes the Transformer encoder and simply multiplies the type embeddings by a binary hierarchical matrix as (Ma et al., 2016) to model the type hierarchy (denoted as "HA-HierMatrix"). We see that HA greatly advances its counterparts that do not use Transformer encoder. Also notice that HA-HierMatrix performs better than HA-Glove, indicating hierarchical constraint enforced by HierMatrix is also important for type representation learning. In addition, HA also shows a strong advantage over Proto-HLE and MZET* which also take the relationships among types into account.

Ablations of KA We remove either descriptions or prototypes from KA and denote them as "KA w/o Description" and "KA w/o Prototypes". The results reveal that, both descriptions and prototypes consistently contribute to KA, wherein prototypes seem to play a more important role on both datasets. In fact, the prototypes used in KA are carefully selected by Ma et al. (2016) while the descriptions from WordNet only contain the brief high-level summaries of types. Additionally, two baselines (i.e., Proto-HLE and DZET) which also leverages background knowledge are included for a more comprehensive comparison. We notice that KA w/o Prototypes is slightly inferior to DZET which also uses type descriptions by a type-description matching approach. However, when prototypes and descriptions are combined, the superiority of KA with NLI framework is obvious.

4.3 Characteristics, Merits and Demerits of Each Information Source

In this section, we focus on the impact of long-tail types and context length for ZFET. Based on the observations, we discuss the characteristics, merits and demerits of different modules targeting each information source (i.e., CA, HA and KA).

4.3.1 Impact of Long-tail Types

We examine the performance of each module on the test subset of *long-tail* (with less than 200 test cases) unseen types. We compute the precision, recall and F1 value for each type and report the average values over all these types in Table 5. The results show CA obtains the best F1$_{avg}$ score on the long-tail types. In fact, CA is based on the pretrained BERT that contains much implicit information of the unseen long-tail types. Moreover, CA masks the mentions and completely depends on the contexts for prediction. This reduces the risk for CA to remember the mentions for prediction and improves the generalization capability.

KA produces better P$_{avg}$ score than HA, which verifies that background knowledge is helpful in distinguishing among easily confused types. However, KA often makes mistakes on the unseen types that share little knowledge with the seen types, which makes KA perform poorly in R$_{avg}$.

We also notice that the combination of different information sources brings a significant improvement to the performance of MSF regarding P$_{avg}$, but a drop regarding R$_{avg}$ on the contrary. This inspires us to take more advantages of CA while minimizing the disturbance from KA and HA to promote the model's generalization capacity on long-tail types in the future.

Model	P$_{avg}$(%)	R$_{avg}$(%)	F1$_{avg}$(%)
CA	28.03	**35.07**	**25.25**
HA	6.99	14.24	5.92
KA	12.77	8.11	7.06
MSF	**43.72**	19.34	21.16

Table 5: The results on long-tail unseen types in BBN.

4.3.2 Impact of Context Length

We separate the test samples into three groups by the context length, and compare the Ma-F1 scores in each group, as shown in Figure 3. We see that CA, HA, KA and MSF all perform better on the mentions with longer contexts, since longer contexts tend to be more informative than the shorter ones. MSF outperforms the single-source modules CA, HA and KA in both the situations with short and median contexts. Nevertheless, the performance of MSF is poorer than CA in the long-context scenario. This indicates that the information from context consistency is with higher confidence in handling longer contexts. Whereas introducing HA and KA modules may prevent the performance growth compared with only using CA module in this case. Conversely, a distinct drop appears when CA is evaluated on the mentions with short contexts.

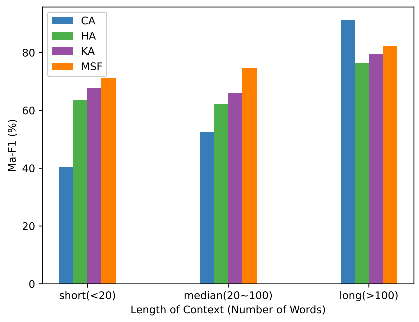


Figure 3: Performance on the unseen types of BBN relative to the context length.

4.4 Complementarity among Different Information Sources

We present the overlaps and disjoint parts of the true cases predicted by the single-source modules in Figure 4. About 31.33% of the test mentions are successfully categorized by all the three modules, while the rest are misidentified by at least one module. We notice that HA and KA share the most true cases (up to 61.04%, i.e., $31.33\% + 29.71\%$) among the pairwise intersections. A possible reason is that HA and KA use the same mention-context encoder based on ELMo. Another reason is that the premises and hypothesis constructed by KA implicitly encode some hierarchical information like HA. For example, part of the prototypes are shared between the parent and child types.

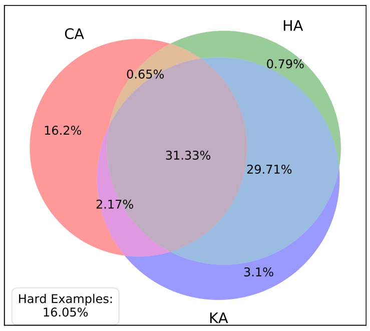


Figure 4: Venn diagram of the true test cases of unseen types correctly predicted by CA, HA and KA on BBN. The annotated percentages (Acc) are proportional to the entire test set.

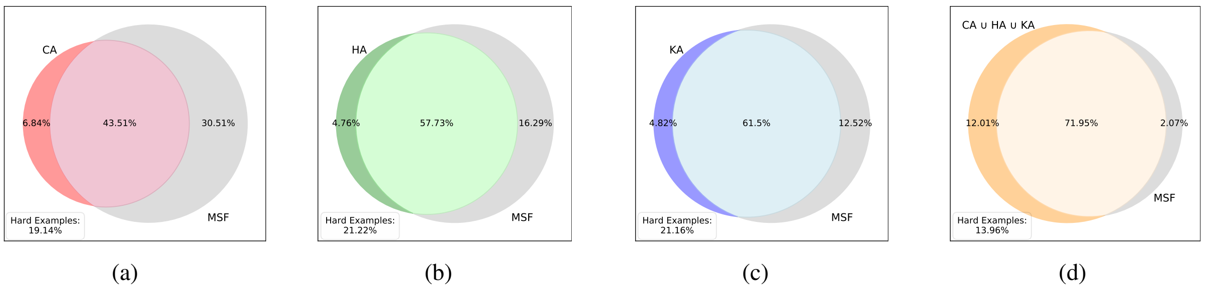


Figure 5: The intersections and differences between the true case sets of unseen types predicted by MSF and CA (a), HA (b), KA (c) or CA ∪ HA ∪ KA (d) on BBN. CA ∪ HA ∪ KA denotes the union of true cases correctly predicted by CA, HA or KA.

KA demonstrates greater capacity than HA with 5.27% (i.e., 2.17% + 3.1%) additional true cases that HA fails to recognize, since background knowledge helps to distinguish among the confusing sibling types sharing the same parent type. However, there still exist 1.44% (i.e., 0.65% + 0.79%) cases where HA does better than KA. This is because the hierarchy-wise information incorporated to KA is less obvious than that inside HA. Meanwhile, KA also suffers from the problem of low recall in long-tail types as discussed in Sec 4.3.1.

Another noticeable observation is that quite a proportion of cases (16.2%) are difficult for HA and KA to recognize, but easy for CA. This indicates the consistency between type names and contexts is a nonnegligible clue for the improvement of performance in ZFET.

4.5 Contributions of Multiple Information Sources to MSF

We also look into the intersections and differences between the true case sets of MSF and CA/HA/KA, as well as their union in Figure 5. We see that MSF takes more advantage of HA and KA, with 57.73% and 61.5% overlaps, respectively. Although CA provides lots of auxiliary information for MSF, there still exist 6.84% true cases of CA wrongly predicted by MSF after fusion. Besides, the 4.76% missing part of HA and the 4.82% of KA also remain to be more fully exploited. Thus, it is worth exploring deeply to make the best of each information source during model fusion. In addition, Figure 5(d) shows that 2.07% complex examples are correctly predicted by MSF while are mistaken by all the three single-source modules. 12.01% samples are correctly identified by at least one of the modules but are mistaken by MSF. Besides, there are 13.96% hard examples misidentified by both the single-source modules (i.e., CA ∪ HA ∪ KA) and the multi-source fusion model (i.e., MSF).

5 Related Work

As a zero-shot paradigm of FET, ZFET suffers from a huge information gap between the seen and unseen types due to the lack of annotated data. In spite of simply computing type representations by averaging the embeddings of words comprising their names (Yuan and Downey, 2018), a variety of auxiliary information has been explored to fill this gap. Huang et al. (2016) proposes a hierarchical clustering model with domain-specific knowledge base for unsupervised entity typing. Ma et al. (2016) first introduces prototypical information to learn type embeddings and encodes type hierarchy by multiplying the type embeddings with a binary hierarchical matrix. Zhou et al. (2018) matches the entity mention with a set of Wikipedia entries and classifies the mention based on the Freebase types of its type-compatible entries. Obeidat et al. (2019) leverages Wikipedia descriptions of types and designs a context-description matching model. Ren et al. (2020) employs entity type attention to make the model focus on context semantically relevant to the type. Zhang et al. (2020b) transfers the knowledge from seen types to the unseen ones through memory network. As for context consistency, Xin et al. (2018) first takes the language models as constraint in supervised typing tasks. Recently, Qian et al. (2021) studies unsupervised entity typing without using knowledge base, where pseudo data with fine-grained labels are automatically created from large unlabeled dataset.

6 Conclusion

In this paper, we explored multiple information sources for ZFET. We proposed a multi-source fusion model to better integrate these sources, which has achieved state-of-the-art performance in ZFET.

Besides, we conducted a deep analysis about the characteristics, merits and demerits of each information source, and discussed the complementarity among different sources. In particular, the context consistency information from the pre-trained language model is relatively useful in complex scenarios with long-tail types or long contexts. Along this way, we will conduct more in-depth research to take full advantage of context consistency. Besides, we will also explore more reasonable methods for information fusion in ZFET.

References

Abhishek, Ashish Anand, and Amit Awekar. 2017. Fine-grained entity type classification by jointly learning representations and label embeddings. In *EACL*.

Antoine Bordes, Nicolas Usunier, Alberto García-Durán, J. Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *NIPS*.

Shuang Chen, Jinpeng Wang, Feng Jiang, and Chin-Yew Lin. 2020. Improving entity linking by modeling latent entity type information. In *AAAI*.

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

J. Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*.

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

Linmei Hu, T. Yang, C. Shi, Houye Ji, and Xiaoli Li. 2019. Heterogeneous graph attention networks for semi-supervised short text classification. In *EMNLP/IJCNLP*.

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*, abs/1603.03112.

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

A. Lai, Yonatan Bisk, and J. Hockenmaier. 2017. Natural language inference from multiple premises. In *IJCNLP*.

Yankai Lin, Zhiyuan Liu, M. Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *AAAI*.

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

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

Lemao Liu, Haisong Zhang, Haiyun Jiang, Yangming Li, Enbo Zhao, Kun Xu, Linfeng Song, Suncong Zheng, Botong Zhou, Jianchen Zhu, et al. 2021. Texsmart: A system for enhanced natural language understanding.

Yukun Ma, E. Cambria, and Sa Gao. 2016. Label embedding for zero-shot fine-grained named entity typing. In *COLING*.

G. Miller. 1995. Wordnet: a lexical database for english. *Commun. ACM*, 38:39–41.

Rasha Obeidat, Xiaoli Z. Fern, Hamed Shahbazi, and P. Tadepalli. 2019. Description-based zero-shot fine-grained entity typing. In *NAACL-HLT*.

Jing Qian, yibin liu, Lemao Liu, Yangming Li, Haiyun Jiang, Haisong Zhang, and Shuming Shi. 2021. Fine-grained entity typing without knowledge base. *EMNLP*.

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

Yankun Ren, J. Lin, and Jun Zhou. 2020. Neural zero-shot fine-grained entity typing. *Companion Proceedings of WWW Conference 2020*.

Sonse Shimaoka, Pontus Stenetorp, Kentaro Inui, and S. Riedel. 2017. Neural architectures for fine-grained entity type classification. *ArXiv*, abs/1606.01341.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, \Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*, pages 5998–6008.

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

Yongqin Xian, Christoph H. Lampert, B. Schiele, and Zeynep Akata. 2019. Zero-shot learning—a comprehensive evaluation of the good, the bad and the ugly. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41:2251–2265.

J. Xin, Hao Zhu, Xu Han, Zhiyuan Liu, and M. Sun. 2018. Put it back: Entity typing with language model enhancement. In *EMNLP*.

Wenhan Xiong, Jiawei Wu, Deren Lei, Mo Yu, S. Chang, Xiaoxiao Guo, and William Yang Wang. 2019. Imposing label-relational inductive bias for extremely fine-grained entity typing. *ArXiv*, abs/1903.02591.

Peng Xu and Denilson Barbosa. 2018. Neural fine-grained entity type classification with hierarchy-aware loss. In *NAACL-HLT*, pages 16–25.

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

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

T. Zhang, Congying Xia, Chun-Ta Lu, and Philip S. Yu. 2020b. Mzet: Memory augmented zero-shot fine-grained named entity typing. *COLING*.

Ben Zhou, Daniel Khashabi, Chen-Tse Tsai, and D. Roth. 2018. Zero-shot open entity typing as type-compatible grounding. In *EMNLP*.

A Implementation Details

For Proto-HLE and DZET we employ their type representation methods but reuse our ELMo-based mention-context encoder for representations of mentions and contexts. In ZOE, we remove the test mentions of target dataset from the Wikipedia entry source and report the performance under our zero-shot setting.

For **CA**, our implementation is based on the pre-trained BERT (BERT-base, uncased) available in the HuggingFace Library[2]. For **HA**, we adopt GloVe 200-dimensional word embeddings for the initialization of type embeddings. The type embeddings are frozen during training. The Transformer encoder is trained from scratch with 4 heads and 2 layers with hidden dimension of 2048. For **KA**, the numbers of prototypes used for BBN and Wiki are 5 and 30 respectively. For the fusion of CA, HA and KA, λ_1, λ_2, λ_3 are tuned according to the performance on the development set by Macro F1, and their values are as follows.

Dataset	λ_1	λ_2	λ_3
BBN	0.393	0.041	0.566
Wiki	0.348	0.424	0.228

Table 6: Values of λ_1, λ_2, λ_3 for MSF on BBN and Wiki.

[2]https://github.com/huggingface