Recall, Expand and Multi-Candidate Cross-Encode: Fast and Accurate Ultra-Fine Entity Typing

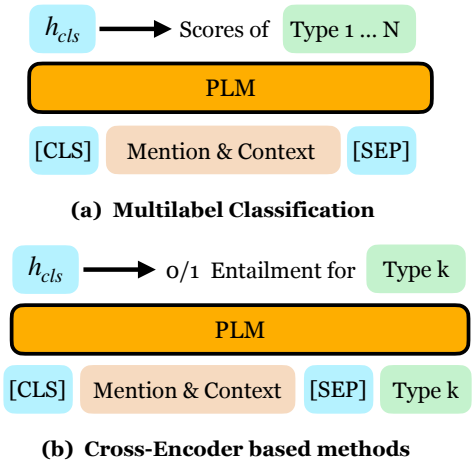
Chengyue Jiang[◇♠], **Wenyang Hui**[◇♠], **Yong Jiang**[♣], **Xiaobin Wang**[♣], **Pengjun Xie**[♣], **Kewei Tu**[♠*]
[♠]School of Information Science and Technology, ShanghaiTech University
Shanghai Engineering Research Center of Intelligent Vision and Imaging
[♣]DAMO Academy, Alibaba Group, China
{jiangchy,huiwy,tukw}@shanghaitech.edu.cn;
{yongjiang.jy,xuanjie.wxb,chengchen.xpj}@alibaba-inc.com

Abstract

Ultra-fine entity typing (UFET) predicts extremely free-formed types (e.g., *president, politician*) of a given entity mention (e.g., *Joe Biden*) in context. State-of-the-art (SOTA) methods use the cross-encoder (CE) based architecture. CE concatenates a mention (and its context) with each type and feeds the pair into a pretrained language model (PLM) to score their relevance. It brings deeper interaction between the mention and the type to reach better performance but has to perform N (the type set size) forward passes to infer all the types of a single mention. CE is therefore very slow in inference when the type set is large (e.g., $N = 10k$ for UFET). To this end, we propose to perform entity typing in a recall-expand-filter manner. The recall and expansion stages prune the large type set and generate K (typically much smaller than N) most relevant type candidates for each mention. At the filter stage, we use a novel model called MCCE to concurrently encode and score all these K candidates in only one forward pass to obtain the final type prediction. We investigate different model options for each stage and conduct extensive experiments to compare each option, experiments show that our method reaches SOTA performance on UFET and is thousands of times faster than the CE-based architecture. We also found our method is very effective in fine-grained (130 types) and coarse-grained (9 types) entity typing. Our code is available at http://github.com/modelscope/AdaSeq/tree/master/examples/MCCE.

1 Introduction

Ultra-fine entity typing (UFET) (Choi et al., 2018) aims to predict extremely fine-grained types (*e.g., president, politician*) of a given entity mention within its context. It provides detailed semantic understandings of entity mentions and is a fundamental step in fine-grained named entity recognition (Ling and Weld, 2012). It can also be utilized to assist various downstream tasks such as relation extraction (Han et al., 2018), keyword extraction (Huang et al., 2020) and content recommendation (Upadhyay et al., 2021). Most recently, the cross-encoder (CE) based method (Li et al., 2022) achieves the SOTA performance in UFET. Specifically, Li et al. (2022) proposes to treat the mention with its context as a premise, and each ultra-fine-grained type as a hypothesis. They then concatenate them together as input and feed it into a pretrained language model (PLM) (e.g., RoBERTa (Liu et al., 2019)) to score the entailment between the mention-type pair as illustrated in Figure 1(b). Compared with the traditional multi-label classification method (shown in Figure 1(a)) that simultaneously scores all types using the same mention representation, CE has the advantage of incorporating type semantics in the encoding and inference process (by taking words in type labels as input) and enabling deeper interactions between each type and the mention via cross-encoding. However, the CE-based method is slow in inference because it



(a) Multilabel Classification

(b) Cross-Encoder based methods

Figure 1: Cross-Encoder and multi-label classification.

* Kewei Tu is the corresponding author.
◇ Equal Contribution.

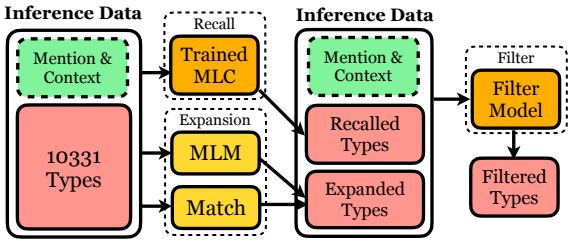


Figure 2: Recall-expand-filter paradigm.

has to enumerate all the types (up to $10k$ types in UFET) and score entailment for each of them given the mention as a premise. There is also no direct interaction between types in CE for modeling correlations between types (e.g., one has to be a person if he or she is categorized as a politician), which has been proven to be useful in previous works (Xiong et al., 2019; Jiang et al., 2022).

To this end, we propose a recall-expand-filter paradigm for UFET (illustrated in Figure 2) for faster and more accurate ultra-fine entity typing. As the name suggests, we first train a multi-label classification (MLC) model to efficiently **recall** top candidate types, which reduces the number of potential types from 10k to hundreds. As the MLC model recalls candidates based on representations learned from the training data, it may not be able to recall candidates that are scarce or unseen in the training set. Consequently, we apply a type candidate **expansion** step utilizing lexical information and weak supervision from masked language models (Dai et al., 2021) to improve the recall rate of the candidate set. Finally, we propose a novel method called multi-candidate cross-encoder (**MCCE**) to concurrently encode and **filter** the expanded type candidate set. Different from CE, **MCCE** concatenates all the recalled type candidates with the mention and its context. The concatenated input is then fed into a PLM to obtain candidate representations and candidate scores. The **MCCE** allows us to simultaneously encode and infer all the types from the candidate set and is thus much faster than the CE-based method, but it still preserves the advantages of CE in modeling interactions between the types and the mention. Concatenating all the candidates also enables **MCCE** to implicitly learn correlations between types. The advantages of **MCCE** over existing methods are summarized in Figure 3.

Experiments on two UFET datasets show that our recall-expand-filter paradigm reaches SOTA performance and **MCCE** is thousands of times faster than the previous SOTA CE-based method. We also comprehensively investigate the performance and efficiency of **MCCE** with different input formats and attention mechanisms. We found **MCCE** is effective in fine-grained (130 types) and coarse-grained (9 types) entity typing. Our code is available at `http://github.com/modelscope/AdaSeq/tree/master/examples/MCCE`.

	Fast Infer	Interact M&C - T	Interact T - T	Semantics of T
MLC	✓			
CE		✓		✓
MCCE	✓	✓	✓	✓

Figure 3: Comparison of different models. M, C, and T are abbreviations of mention, context, and type.

dataset	$\lvert\mathcal{Y}\rvert$	avg($\lvert\boldsymbol{y}^g\rvert$)	train/dev/test	language
UFET	10331	5.4	2k/2k/2k	English
CFET	1299	3.5	3k/1k/1k	Chinese

Table 1: Statistics of UFET datasets. avg($\lvert\boldsymbol{y}^g\rvert$) denotes the average number of gold types per instance.

2 Background

2.1 Problem Definition

Given an entity mention m within its context sentence c, ultra-fine entity typing (UFET) aims to predict its correct types $\boldsymbol{y}^g \subset \mathcal{Y}$ ($\lvert\mathcal{Y}\rvert$ can be larger than $10k$). As $\lvert\boldsymbol{y}^g\rvert > 1$ in most cases, UFET is a multi-label classification problem. We show statistics of two UFET datasets, **UFET** (Choi et al., 2018) and **CFET**[1] (Lee et al., 2020), in Table 1.

2.2 Multi-label Classification Model for UFET

Multi-label classification (MLC) models are widely adopted as backbones for UFET (Choi et al., 2018; Onoe and Durrett, 2019; Onoe et al., 2021). They use an encoder to obtain the mention representation and use a decoder (e.g., MLP) to score types simultaneously. Figure 1(a) shows a representative MLC model adopted by recent methods (Dai et al., 2021; Jiang et al., 2022). The contextualized mention representation is obtained by feeding c and m into a pretrained language model (PLM) and taking the last hidden state of `[CLS]`, h_{cls}. The mention representation is then fed into an MLP layer to concurrently obtain all type scores $s_1, \cdots s_N (N = \lvert\mathcal{Y}\rvert)$.

[1] As there is no official split available for **CFET**, we split it by ourselves and will release our split in our code.

MLC Inference Types with a probability higher than a threshold τ are predicted: $\boldsymbol{y}^p = \{y_j | \sigma(s_j) > \tau, 1 \leq j \leq N\}$, where σ is the sigmoid function. τ is tuned on the development set.

MLC Training The binary cross-entropy (BCE) loss over the predicted label probabilities and the gold types is used to train the MLC model. MLC is very efficient in inference. However, the interactions between mention and types in MLC are weak, and the correlations between types are ignored (Onoe et al., 2021; Xiong et al., 2019; Jiang et al., 2022). In addition, MLC has difficulty in integrating type semantics (Li et al., 2022).

2.3 Vanilla Cross-Encoders for UFET

Li et al. (2022) first proposed to use Cross-Encoder (CE) for UFET. As shown in Figure 1(b), CE concatenates m, c together with a type $y_j \in \mathcal{Y}$ and feeds them into a PLM to obtain the `[CLS]` embedding. Then an MLP layer is used to obtain the score of y_j given m, c.

$$\boldsymbol{h}_{cls} = \texttt{PLM}(\texttt{[CLS]}\ c\ \texttt{[SEP]}\ m\ \texttt{[SEP]}\ y_j\ ) \quad (1)$$

$$s_j = \texttt{MLP}(\boldsymbol{h}_{cls}) \quad (2)$$

The concatenation allows deeper interaction between the mention, context, and type (via the multi-head self-attention in PLMs), and also incorporates type semantics.

CE Inference Similar to MLC, types that have a higher probability than a threshold are predicted. To compute the probabilities, CE requires N forward passes to infer types of a single mention, so its inference is very slow when N is large.

CE Training CE is typically trained with the marginal ranking loss (Li et al., 2022). A positive type $y_+ \in \boldsymbol{y}^g$ and a negative type $y_- \notin \boldsymbol{y}^g$ are sampled from $\mathcal{Y}$ for each training sample (m, c). The loss is computed as:

$$L = \max(\sigma(s_-) - \sigma(s_+) + \delta, 0)$$

where s_+, s_- are scores of the sampled positive and negative types, and δ is the margin tuned on the development set to determine how far positive and negative samples should be separated.

3 Method

Inspired by techniques in information retrieval (Larson, 2010) and entity linking (Ledell et al., 2020), we decompose the inference of UFET into three stages as illustrated in Figure 2: (1) A recall stage to reduce the type candidate number (e.g., from $N = 10k$ to $K = 100$) while maintaining a good recall rate using an efficient MLC model. (2) An expansion stage to improve the recall rate by incorporating lexical information using exact matching and weak supervision (Dai et al., 2021) from large pretrained language models such as BERT-Large (Devlin et al., 2019). (3) A filter stage to filter the expanded type candidates to obtain the final prediction. For the filter stage, we propose an efficient model, Multi-Candidate Cross-Encoder (MCCE), to concurrently encode and filter type candidates of a given mention with only a single forward pass.

3.1 Recall Stage

To prune the type candidate set, we train a MLC model introduced in Sec. 2.2 on the training set and tune it based on the recall rate (e.g., recall@64) on the development set. Then we use it to infer the top K_1 (typically less than 256) candidates $\mathcal{C}_R$ for each data point (m, c). We find that MLC significantly outperforms BM25 (Robertson and Zaragoza, 2009) as a recall model (see Sec. 5.1.1).

3.2 Expansion Stage

In UFET, the number of training data per type is small, especially for fine-grained and ultra-fine-grained types. 30% of the types in the development set of **UFET** dataset is unseen during training. Consequently, we find the MLC used in the recall stage easily overfits the train set and has difficulty in predicting types that only appear in the development or test set. Therefore, we utilize two methods, exact match and masked language models (MLM), to expand the recalled candidates. Both exact match and MLM are able to recall unseen type candidates without any training.

Exact Match MLC recalls candidates using dense representations. They are known to be weak at identifying and utilizing lexical matching information between the input and types (Tran et al., 2019; Khattab and Zaharia, 2020). However, types are extremely fine-grained in UFET (e.g., *son, child*) and are very likely to appear in the context or mention (e.g., mention "*He*" in context "*He is the **son** and **child** of ...*"). To this end, we first identify and normalize all nouns in the context and mention using NLTK[2], and then recall types that exactly

[2]nltk.tag package `https://www.nltk.org`

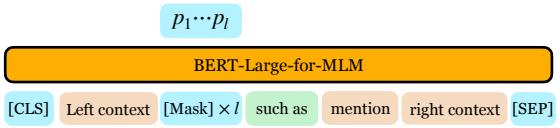


Figure 4: Recall from MLM using prompts.

match these nouns. We denote the recalled type set as $\mathcal{C}_{EM}$.

MLM Inspired by recent prompt-based methods for entity typing (Ding et al., 2021; Pan et al., 2022), we recall candidates by asking PLMs to fill masks in prompts. Suppose a type $y \in \mathcal{Y}$ is tokenized to l subwords $w_1, \cdots w_l$. To score y given m, c, we first formulate the input as in Figure 4. We use 'such as' as the template to induce types. The input is then fed into BERT-large-uncased[3] to obtain the probabilities of the subwords. The score of y is calculated by $s^{MLM} = (\sum_{n=1}^{l} \log p_n)/l$, where p_n denotes the probability of subword w_n predicted by the PLM. We rank all types in descending order by their scores s^{MLM}.

We expand K_2 candidates by the following strategy: First, expand all candidates recalled by exact match $\mathcal{C}_{EM} \setminus \mathcal{C}_R$. Then expand $K_2 - |\mathcal{C}_{EM} \setminus \mathcal{C}_R|$ candidates using MLM based on their scores. After expansion, we obtain $K = K_1 + K_2$ type candidates for each data point.

3.3 Filter Stage

The filter stage infers types from the candidate pool $\mathcal{C}$ generated by the recall and expansion stages. Let $K = |\mathcal{C}| = K_1 + K_2$ and K is typically less than 128. A trivial choice of the filter model is the CE model introduced in Sec. 2.3. We can score these candidates $\mathcal{C}$ using CE by K forward passes as introduced before. For training, the positive type y^+ and negative type y^- are sampled from $\mathcal{C}$ instead of $\mathcal{Y}$ and are then used for calculating the marginal ranking loss. As K is much smaller than $|\mathcal{Y}|$, the inference speed with CE under our Recall-Expand-Filter paradigm is much faster than that of vanilla CE. However, it is still inefficient compared with the MLC model that concurrently predicts scores of all types in a single forward pass. For faster inference and training, we propose multi-candidate cross-encoders (**MCCE**) in the next section.

[3]We use the PLM from https://huggingface.co

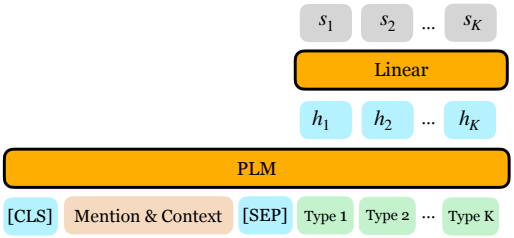


Figure 5: Multi-candidate cross-encoder (MCCE).

4 Multi-Candidate Cross-Encoder

4.1 Overview

As shown in Figure 5, compared with CE that concatenates one candidate at a time, **MCCE** concatenates all the candidates in $\mathcal{C}$ with the mention and context. The concatenated input is then fed into the PLM to obtain the hidden state of each candidate. Finally, we apply an MLP over the hidden states to concurrently score all the candidates. **MCCE** models use only one forward pass to infer types from candidates.

$$\begin{aligned} \boldsymbol{h}_{1:K} &= \texttt{PLM}(\texttt{[CLS]}\ c\ \texttt{[SEP]}\ m\ \texttt{[SEP]}\ y_{1:K}) \\ \boldsymbol{s}_{1:K} &= \texttt{Linear}(\boldsymbol{h}_{1:K}) \end{aligned} \tag{3}$$

where $y_{1:K}$ is short for $y_1, \ldots, y_K \in \mathcal{C}$, and similarly, $\boldsymbol{h}_{1:K}$ and $\boldsymbol{s}_{1:K}$ denote hidden states and scores of all the candidates respectively.

Similar to MLC training and inference discussed in Sec. 2.2, we use the binary cross-entropy loss as the training objective and tune a probability threshold on the development set for inference. We find that during training, all positive types are ranked very high in the candidate set at the first stage, which is however not the case for the development and test data. To prevent the filter model from overfitting the order of training candidates and only learning to predict the highest-ranked candidates, we keep permuting candidates during training.

4.2 Input Format of Candidates

We show two kinds of candidates representations in this section.

Average of type sub-tokens We treat each possible type $y \in \mathcal{C}$ as a new token u and add it to the vocabulary of the PLM. The static embedding (layer 0 embedding of the PLM) of u is initialized with the average static embedding of all the sub-tokens in type y. The advantages of this method include: (1) Compressing types into single tokens allows us to consider more candidates; (2) Types

in **UFET** are tokenized into only 2.1 sub-tokens on average (by RoBERTa's tokenizer), so averaging sub-token embeddings does not lose too much semantic information of the sub-tokens.

Fixed-size sub-token block To preserve more type semantics, we represent each candidate type with its sub-tokens. We pad or truncate the sub-tokens to a fixed-sized block to facilitate parallel implementation of the attention mechanisms that we will introduce next. We use the PLM hidden state of the first sub-token in the block as the output representation of each candidate.

4.3 Attention in MCCE

There are four kinds of attention in **MCCE** as shown in Figure 6: sentence-to-sentence (S2S), sentence-to-candidates (S2C), candidate-to-sentence (C2S), and candidate-to-candidate (C2C). Since we score candidates based on the mention and its context, the attention from candidates to the sentence (C2S) is necessary. On the other hand, the C2C, S2S, and S2C attention are optional. We empirically find that S2C is important, S2S is useful, and C2C is only useful in some settings (see Sec. 6). Considering that C2C is computationally expensive, we propose a variant of **MCCE** in which C2C attention is discarded in computation (not by masking), as shown in the right part of Figure 6. Removing C2C attention significantly reduces the time complexity of attention from $O(D(L_S + L_C)^2)$ to $O(D(L_S^2 + 2L_S L_C + B^2 L_C))$, where L_S and L_C be the number of sub-tokens used by the sentence and candidates respectively, $L_C > L_S$ in most cases. D is the embedding dimension, and B is the block size ($B = 1$ when we use the averaged sub-tokens to represent a candidate). The detailed computation procedure after removing C2C attention is shown in Appendix A.

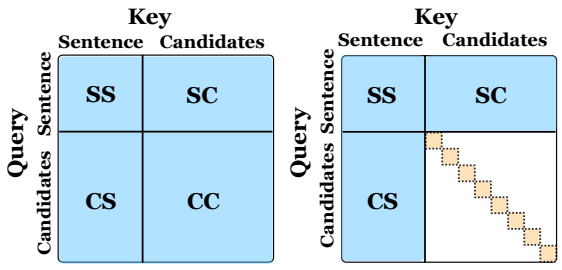


Figure 6: Attention in **MCCE** (left) and in **MCCE** without candidate-to-candidate (C2C) attention (right).

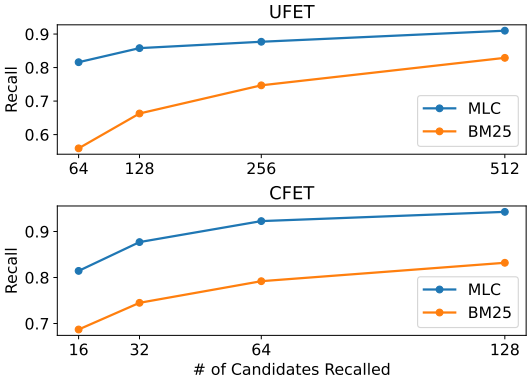


Figure 7: Recall@K of MLC and BM25.

5 Experiments

We conduct experiments on two ultra-fine entity typing datasets, **UFET** (English) and **CFET** (Chinese). Their statistics are shown in Table 1. We mainly report macro-averaged recall at the recall and expansion stages and macro-F1 of the final prediction. We also evaluate the **MCCE** models on fine-grained (130 types) and coarse-grained (9 types) entity typing.

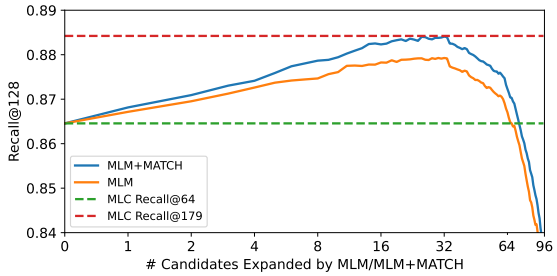
5.1 UFET and CFET

5.1.1 Recall Stage

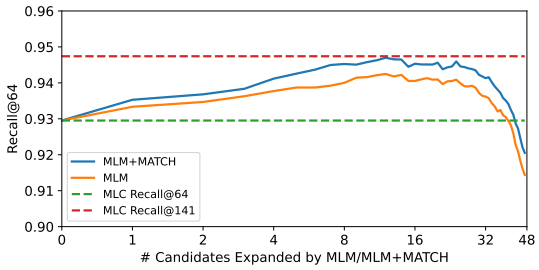
We compare recall@K on the test sets of **UFET** and **CFET** between our MLC model and a traditional BM25 model (Robertson and Zaragoza, 2009) in Figure 7. The MLC model uses RoBERTa-large as the backbone and is tuned based on recall@128 on the development set. We use the AdamW optimizer with a learning rate of 2×10^{-5}. Results show that MLC is a strong recall model, consistently outperforming BM25 on both **UFET** and **CFET** datasets. Its recall@128 reaches over 85% on **UFET** and over 94% on **CFET**.

5.2 Expansion Stage

We show the improvement of recall from using candidate expansion in Figure 8. On the **UFET** dataset, the recall of expanding $K_2 = 32$ additional candidates based on $K_1 = 96$ MLC candidates is 2% (absolute value) higher than the recall of $K_1 = 128, K_2 = 0$, and is comparable to the recall of $K_1 = 179, K_2 = 0$ (the red dotted line in Figure 8). Similarly in **CFET**, expanding 10 candidates based on 54 MLC candidates is comparable to recalling 141 candidates using MLC alone in the recall. In subsequent experiments, we ex-



(a) Recall@128 on **UFET** by including different numbers of expanded candidates.



(b) Recall@64 on **CFET** by including different numbers of expanded candidates.

Figure 8: The effect of the expansion stage.

pand 48 and 10 candidates for **UFET** and **CFET** respectively for the filter stage.

5.3 Filter Stage and Final Results.

We report the performance of **MCCE** variants as the filter model and compare them with various strong baselines. We treat the number of candidates K_1 and K_2 recalled and expanded by the first two stages as a hyper-parameter and tune it on the development set. For a fair comparison with baselines, we conduct experiments of **MCCE** using different backbone PLMs. For all **MCCE** models, we use the AdamW optimizer with a learning rate tuned between 5×10^{-6} and 2×10^{-5}. The batch size we use is 4 and we train the models for at most 50 epochs with early stopping.

Baselines The **MLC** model we use for the recall stage and the cross-encoder (**CE**) we introduced in Sec. 2.3 are natural baselines. We also compare our methods with recent PLM-based methods. We introduce them in Appendix B.

Naming Conventions **MCCE-S** denotes the **MCCE** model using the average of sub-tokens as candidates' input, and **MCCE-B** denotes the model representing candidates as fixed-sized blocks. The **MCCE** model without **C2C** attention (mentioned in Sec. 4.3) is denoted as **MCCE-B w/o C2C**. For PLM backbones used in **UFET**, we use **B**, **R**, **RM** to denote BERT-base-cased (Devlin et al., 2019), RoBERTa (Liu et al., 2019), and RoBERTa-MNLI (Liu et al., 2019) respectively. For **CFET**, we adopt two widely-used Chinese PLM, BERT-base-Chinese (**C**) and NEZHA-base (**N**) (Wei et al., 2019) . We call 12-layer PLMs base models and 24-layer PLMs large models.

Base Models on UFET	**P**	**R**	**F1**
MLC-like models			
B **BOX4TYPES**(Onoe et al., 2021)	52.8	38.8	44.8
B **LDET**† (Onoe and Durrett, 2019)	51.5	33.0	40.1
B **MLMET**† (Dai et al., 2021)	53.6	45.3	49.1
B **PL** (Ding et al., 2021)	57.8	40.7	47.7
B **DFET** (Pan et al., 2022)	55.6	44.7	49.5
B **MLC** (reimplemented by us)	46.5	34.9	39.9
R **MLC** (reimplemented by us)	42.2	44.9	43.5
Seq2seq based models			
B **LRN** (Liu et al., 2021)	54.5	38.9	45.4
Filter models under our recall-expand-filter paradigm			
B **CE**$_{128}$	47.2	48.5	47.8
B **MCCE-S**$_{128}$ (Ours)	53.2	48.3	**50.6**
B **MCCE-S**$_{128}$ **W/O C2C** (Ours)	52.3	48.3	50.2
B **MCCE-B**$_{128}$ (Ours)	49.9	50.0	49.9
B **MCCE-B**$_{128}$ **W/O C2C** (Ours)	49.9	48.2	49.0
R **CE**$_{128}$	49.6	49.0	49.3
R **MCCE-S**$_{128}$ (Ours)	53.3	47.3	50.1
R **MCCE-S**$_{128}$ **W/O C2C** (Ours)	53.2	46.6	49.7
R **MCCE-B**$_{128}$ (Ours)	52.5	47.9	50.1
R **MCCE-B**$_{128}$ **W/O C2C** (Ours)	52.7	46.4	49.3
Large Models on UFET	**P**	**R**	**F1**
MLC-like models			
R **MLC** (Jiang et al., 2022)	47.8	40.4	43.8
R **MLC-NPCRF** (Jiang et al., 2022)	48.7	45.5	47.0
R **MLC-GCN** (Xiong et al., 2019)	51.2	41.0	45.5
R **PL-NPCRF** (Jiang et al., 2022)	49.9	46.9	48.4
B **PL** (Ding et al., 2021)	59.3	42.6	49.6
B **PL-NPCRF** (Jiang et al., 2022)	55.3	46.7	50.6
Cross-encoder based models and **MCCE***s*			
R **LITE+L** (Li et al., 2022)	48.7	45.8	47.2
RM **LITE+NLI+L** (Li et al., 2022)	52.4	48.9	50.6
Filter models under our recall-expand-filter paradigm			
B **CE**$_{128}$	50.3	49.6	49.9
B **MCCE-S**$_{128}$ (Ours)	52.5	49.1	50.8
B **MCCE-S**$_{128}$ **W/O C2C** (Ours)	54.1	47.1	50.4
B **MCCE-B**$_{128}$ (Ours)	54.0	48.6	51.2
B **MCCE-B**$_{128}$ **W/O C2C** (Ours)	52.8	48.3	50.4
R **CE**$_{128}$	54.5	49.3	51.8
R **MCCE-S**$_{128}$ (Ours)	50.8	49.8	50.3
R **MCCE-S**$_{128}$ **W/O C2C** (Ours)	51.5	48.8	50.1
R **MCCE-B**$_{128}$ (Ours)	51.9	50.8	51.4
R **MCCE-B**$_{128}$ **W/O C2C** (Ours)	51.6	51.6	51.6
RM **MCCE-B**$_{128}$ **W/O C2C** (Ours)	56.3	48.5	**52.1**

Table 2: Macro-averaged UFET result. **LITE+L** is LITE without NLI pretraining, **LITE+L+NLI** is the full LITE model. Methods marked by † additionally utilize either distantly supervised or augmented data for training. **MCCE-S**$_{128}$ denotes we use 128 candidates recalled and expanded from the first two stages.

Models on CFET	P	R	F1
MLC-like models			
N **MLC**	55.8	58.6	57.1
N **MLC-NPCRF** (Jiang et al., 2022)	57.0	60.5	58.7
N **MLC-GCN** (Xiong et al., 2019)	51.6	63.2	56.8
C **MLC**	54.0	59.5	56.6
C **MLC-NPCRF** (Jiang et al., 2022)	54.0	61.6	57.3
C **PL-NPCRF** (Xiong et al., 2019)	52.4	64.1	57.7
C **MLC-GCN** (Xiong et al., 2019)	56.4	58.6	57.5
Filter models under our recall-expand-filter paradigm			
N **CE**$_{64}$	57.6	64.3	60.7
C **CE**$_{64}$	54.0	63.3	58.3
N **MCCE-S**$_{64}$ (Ours)	58.4	62.1	60.2
N **MCCE-S**$_{64}$ **W/O C2C** (Ours)	59.1	61.5	60.3
N **MCCE-B**$_{64}$ (Ours)	56.7	66.1	61.1
N **MCCE-B**$_{64}$ **W/O C2C** (Ours)	58.8	64.1	**61.4**
C **MCCE-S**$_{64}$ (Ours)	55.5	62.6	58.8
C **MCCE-S**$_{64}$ **W/O C2C** (Ours)	54.0	63.4	58.3
C **MCCE-B**$_{64}$ (Ours)	55.0	63.5	59.0
C **MCCE-B**$_{64}$ **W/O C2C** (Ours)	57.3	61.3	59.3

Table 3: Macro-averaged CFET result.

UFET Results We show the results on the **UFET** dataset in Table 2 and make the following observations. (1) The recall-expand-filter paradigm is effective. Our methods outperform all the baselines without the paradigm by a large margin. The CE under our paradigm reaches 51.8 F1, while **LITE**, a more complicated CE, achieves only 50.6 F1. (2) **MCCE**s are strong filter models and reach SOTA performances. **MCCE-S**$_{128}$ with BERT-base performs best and reaches **50.6** F1 score among base models, which is comparable with previous SOTA performance of large models such as **LITE+NLI+L** and **PL+NPCRF**. Among large models, **MCCE-B**$_{128}$ **W/O C2C** also reaches SOTA performance with **52.1** F1 score. (3) **C2C** attention is not necessary on large models, but is useful in base models. (4) Large models can utilize type semantics better. We find **MCCE-B** outperforms **MCCE-S** on large models, but underperforms **MCCE-S** on base models. (5) The choice of the backbone PLM matters. We find the performance of **CE** under our paradigm is largely affected by the PLM it uses. It reaches 47.8 F1 with BERT-base and 51.8 F1 with RoBERTa-large. We also find that BERT is more suitable for **MCCE-S** compared to RoBERTa, and RoBERTa is more suitable for **MCCE-B** as the backbone.

CFET Results On **CFET**, we compare **MCCE** models with several strong baselines: **NPCRF** and **GCN** with an MLC-like architecture, and **CE** under our paradigm which is shown to be better than **LITE** on **UFET**. The results are shown in Table 3.

MODEL	# FP	SENTS/S	F1
MLC	1	58.8	43.8
LITE+NLI+L (CE)	N	0.02	50.6
filter stage inference speed.			
CE$_{128}$	128	1.64	51.8
MCCE-S$_{128}$	1	60.8	50.1
MCCE-B$_{128}$	1	22.3	51.4
MCCE-B$_{128}$ **W/O C2C**	1	25.2	**52.1**

Table 4: Inference speed comparison of models. **# FP** means the number of PLM forward passes required by a single inference. We also report the practical inference speed **SENTS/S** and the **F1** scores on **UFET**.

Models	P	R	F1
coarse (9 types) Open Entity			
R **MLC**	76.8	78.5	77.6
R **CE**$_{9}$	82.3	81.0	81.6
R **MCCE-S**$_{9}$	77.0	87.7	**82.0**
R **MCCE-B**$_{9}$ **W/O C2C**	77.2	85.4	81.1
fine (130 types)			
R **MLC**	70.4	63.7	66.9
R **CE**$_{130}$	67.9	66.4	67.1
R **MCCE-S**$_{130}$	65.8	71.8	**68.7**
R **MCCE-B**$_{130}$ **W/O C2C**	64.1	70.5	67.1

Table 5: Micro-averaged results for fine and coarse-grained entity typing on **UFET.**

Similar to the results on **UFET**, filter models under our paradigm significantly outperform MLC-like baselines, +2.0 F1 for NEZHA-base and +1.8 F1 for BERT-base-Chinese. **MCCE-B** is significantly better than **MCCE-S** on both NEZHA-base and BERT-base-Chinese, indicating the importance of type semantics in the Chinese language. We also find that **MCCE w/o C2C** is generally better than **MCCE w/ C2C**, possibly because C2C attention distracts the candidates from attending to the mention and contexts.

Speed Comparison Table 4 shows the number of PLM forward passes and the empirical inference speed of different RoBERTa-Large models on **UFET**. We conduct the speed test using NVIDIA TITAN RTX for all the models and the inference batch size is 4. At the filter stage, the inference speed of **MCCE-S** is 40 times faster than **CE**$_{128}$ and thousands of times faster than **LITE**. Surprisingly, **MCCE-B W/O C2C** is not significantly faster than **MCCE-B**. It is possibly because the computation (Appendix A) related to the block attention is not fully optimized by the deep learning framework we use. However, we expect the speed advantage of **MCCE-B W/O C2C** over **MCCE-**

Ablation of expansion stage	P	R	F1
UFET MCCE WITH C2C BERT-LARGE			
B MCCE-S_{128} (Ours)	52.5	49.1	50.8
B MCCE-S_{128} W/O EXPAND (Ours)	52.7	48.1	50.2
CFET MCCE WITH C2C BERT-BASE-CHINESE			
C MCCE-S_{64} (Ours)	55.5	62.6	58.8
C MCCE-S_{64} W/O EXPAND (Ours)	55.4	60.4	57.8

Table 6: Ablation study of the expansion stage.

Analysis about attention on UFET	P	R	F1
MCCE-S USING BERT-BASE			
B MCCE-S_{128} FULL	53.2	48.3	50.6
B MCCE-S_{128} W/O C2C	52.3	48.3	50.2
B MCCE-S_{128} W/O S2S	50.6	48.4	49.4
B MCCE-S_{128} W/O S2C	48.7	47.1	47.9
B MCCE-S_{128} W/O C2S	19.7	17.4	18.7
B MCCE-S_{128} W/O S2S,C2C	50.2	47.3	48.8

Table 7: Ablation of different types of attention.

B would become greater with more candidates.

5.4 Fine and Coarse-grained Entity Typing

We also conduct experiments on fine-grained (130-class) and coarse-grained (9-class) entity typing, and the results are shown in Table 5. Since the type candidate set is already small, it is not necessary to apply the recall and expand stage to further prune the type set. Then, we only evaluate different model options for the filter stage. Results show that **MCCE** models are still better than **MLC** and **CE**, and **MCCE-S** is better than **MCCE-B** on the coarser-grained setting possibly because the coarser-grained types are simpler in surface-forms and **MCCE-S** does not lose much type semantics.

6 Analysis

6.1 Importance of expansion stage

We perform an ablation study on the importance of the expansion stage by comparing the results of **MCCE-S** with and without the expansion stage in Table 6. It can be seen that the expansion stage has a positive effect, improving the final recall by $+1.0$ and $+2.2$ on **UFET** and **CFET** respectively without harming the precision.

6.2 Attention

We conduct an ablation study on S2S, C2S, S2C, and C2C attention introduced in Sec. 4.3 and show the results in Table 7. According to the results, we find that C2C is useful but not necessary on base models, **MCCE-S** using BERT-base reaches 50.2 without C2C on UFET. Removing S2S has a non-negligible negative effect but surprisingly, it will not destroy the model. A possible reason is the interaction between sub-tokens in the sentence can be achieved indirectly by first attending to the candidates and then being attended back by the candidates in the next layer. We also find that C2S is necessary for the task (18.7 F1 w/o C2S) because we rely on the mention and context to encode and classify candidates. Furthermore, it is important for sentences to attend to all the candidates (S2C), possibly because certain candidate types may help highlight informative words in the sentence.

7 Related Work

While writing this paper, we noticed that a paper (Du et al., 2022) that has similar ideas to our work was submitted to arXiv. They target the task of selecting from multiple options, of which UFET is a special case. Their second model, Parallel-TE, is similar to our **MCCE-B**. In addition, when applying their model to UFET in their experiments, they prune types in a similar manner to our recall stage. Below we summarize the differences between our method and theirs when applied to UFET. (1) Difference in the paradigms. Our paradigm has an additional expansion stage which improves the quality of recalled candidates, as shown in Sec. 6. (2) There are many differences in model details. For example, in the PLM input, our **MCCE-S** learns a single token for each type and our **MCCE-B** uses fixed-sized blocks without SEP tokens in-between, while they use full text of types separated by SEP tokens. We also propose a new model variant with C2C attention removed. (3) We conduct more comprehensive experiments on UFET, covering two languages and three settings (ultra-fine-grained, fine-grained, and coarse-grained), as well as comparing and analyzing different options such as PLM backbones and types of attention. More related works about entity typing are shown in Appendix B.

8 Conclusion

We propose a recall-expand-filter paradigm for ultra-fine entity typing. We train a recall model to generate candidates, use MLM and exact match to improve the quality of recalled candidates, and finally use filter models to obtain final type predictions. We propose a filter model called multi-candidate cross-encoder (**MCCE**) to concurrently encode and filter all candidates, and investigate

different input formats and attention mechanisms. Extensive experiments on entity typing show that our paradigm is effective and the **MCCE** models under our paradigm reach SOTA performances on both English and Chinese UFET datasets and are also very effective on fine and coarse-grained entity typing. Further, **MCCE** models have comparable inference speed to simple (**MLC**) models and are thousands of times faster than previous SOTA cross-encoder-based methods.

Limitation

One limitation of the **MCCE** models is that the number of candidates during training and inference should be the same, otherwise, the performance drops severely. One simple potential solution is to divide or pad the candidates during inference to match the number of candidates during training. For example, divide 128 candidates into two sets with 64 candidates and apply twice forward passes of a filter model if it is trained on 64 candidates and required to filter 128 candidates during inference. We don't fully explore the solutions to this limitation and leave it as future work.

Acknowledgement

This work was supported by the National Natural Science Foundation of China (61976139) and by Alibaba Group through Alibaba Innovative Research Program.

References

Eunsol Choi, Omer Levy, Yejin Choi, and Zettlemoyer. 2018. Ultra-fine entity typing. In *Proceedings of the ACL*. Association for Computational Linguistics.

Hongliang Dai, Yangqiu Song, and Haixun Wang. 2021. Ultra-fine entity typing with weak supervision from a masked language model. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1790–1799, Online. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. *ArXiv*, abs/1810.04805.

Ning Ding, Yulin Chen, Xu Han, Guangwei Xu, Pengjun Xie, Hai-Tao Zheng, Zhiyuan Liu, Juanzi Li, and Hong-Gee Kim. 2021. Prompt-learning for fine-grained entity typing. *arXiv preprint arXiv:2108.10604*.

Jiangshu Du, Wenpeng Yin, Congying Xia, and Philip Yu. 2022. Learning to select from multiple options. *ArXiv*, abs/2212.00301.

Xu Han, Hao Zhu, Pengfei Yu, Ziyun Wang, Y. Yao, Zhiyuan Liu, and Maosong Sun. 2018. Fewrel: A large-scale supervised few-shot relation classification dataset with state-of-the-art evaluation. In *Conference on Empirical Methods in Natural Language Processing*.

Han Huang, Xiaoguang Wang, and Hongyu Wang. 2020. Ner-rake: An improved rapid automatic keyword extraction method for scientific literatures based on named entity recognition. *Proceedings of the Association for Information Science and Technology*, 57(1):e374.

Chengyue Jiang, Yong Jiang, Weiqi Wu, Pengjun Xie, and Kewei Tu. 2022. Modeling label correlations for ultra-fine entity typing with neural pairwise conditional random field. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

O. Khattab and Matei A. Zaharia. 2020. Colbert: Efficient and effective passage search via contextualized late interaction over bert. *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Ray R. Larson. 2010. Introduction to information retrieval. *J. Assoc. Inf. Sci. Technol.*, 61:852–853.

Wu Ledell, Petroni Fabio, Josifoski Martin, Riedel Sebastian, and Zettlemoyer Luke. 2020. Zero-shot entity linking with dense entity retrieval. In *EMNLP*.

Chin Lee, Hongliang Dai, Yangqiu Song, and Xin Li. 2020. A Chinese corpus for fine-grained entity typing. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 4451–4457, Marseille, France. European Language Resources Association.

Bangzheng Li, Wenpeng Yin, and Muhao Chen. 2022. Ultra-fine entity typing with indirect supervision from natural language inference. *arXiv preprint arXiv:2202.06167*.

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

Qing Liu, Hongyu Lin, Xinyan Xiao, Xianpei Han, Le Sun, and Hua Wu. 2021. Fine-grained entity typing via label reasoning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4611–4622, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019.

Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Yasumasa Onoe, Michael Boratko, Andrew McCallum, and Greg Durrett. 2021. Modeling fine-grained entity types with box embeddings. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2051–2064, Online. Association for Computational Linguistics.

Yasumasa Onoe and Greg Durrett. 2019. Learning to denoise distantly-labeled data for entity typing. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2407–2417, Minneapolis, Minnesota. Association for Computational Linguistics.

Weiran Pan, Wei Wei, and Feida Zhu. 2022. Automatic noisy label correction for fine-grained entity typing. *arXiv preprint arXiv:2205.03011*.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Stephen E. Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: Bm25 and beyond. *Found. Trends Inf. Retr.*, 3:333–389.

Vu Mai Tran, Minh Le Nguyen, and Ken Satoh. 2019. Building legal case retrieval systems with lexical matching and summarization using a pre-trained phrase scoring model. *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Law*.

Chirayu Upadhyay, Hasan Abu-Rasheed, Christian Weber, and Madjid Fathi. 2021. Explainable job-posting recommendations using knowledge graphs and named entity recognition. In *2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 3291–3296. IEEE.

Junqiu Wei, Xiaozhe Ren, Xiaoguang Li, Wenyong Huang, Yi Liao, Yasheng Wang, Jiashu Lin, Xin Jiang, Xiao Chen, and Qun Liu. 2019. NEZHA: neural contextualized representation for chinese language understanding. *CoRR*, abs/1909.00204.

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

A Removing C2C Attention

Let L_S and L_C be the number of sub-tokens used by the sentence and candidates respectively. We can formulate the attention query of the sentence as $\boldsymbol{Q}_S = [\boldsymbol{q}_1^s; \cdots; \boldsymbol{q}_{L_S}^s] \in \mathbb{R}^{L_S \times D}$, where $\boldsymbol{q}_i^s$ is the query vector of the i-th sub-token in the sentence, and D is the embedding dimension. Similarly, the query of candidates is formulated as $\boldsymbol{Q}_C = [\boldsymbol{q}_1^c; \cdots; \boldsymbol{q}_{L_C}^c] \in \mathbb{R}^{L_C \times D}$. When we treat candidates as average of sub-tokens, $\boldsymbol{q}_i^c$ is a D-dimensional vector, and when we use fixed-sized blocks to place candidates, $\boldsymbol{q}_i^c \in \mathbb{R}^{B \times D}$ is the concatenation of the query vectors in the i-th candidate block and B is the number of sub-tokens in a block. The keys and values are defined similarly as $\boldsymbol{K}_C, \boldsymbol{V}_C, \boldsymbol{O}_C \in \mathbb{R}^{L_C \times D}, \boldsymbol{K}_S, \boldsymbol{V}_S, \boldsymbol{O}_S \in \mathbb{R}^{L_S \times D}$. The attention outputs are computed as:

$$\boldsymbol{O}_S = \text{Softmax}\big(\frac{\boldsymbol{Q}_S[\boldsymbol{K}_S; \boldsymbol{K}_C]^T}{\sqrt{D}}\big) \cdot [\boldsymbol{V}_S; \boldsymbol{V}_C] \quad (4)$$

$$[\boldsymbol{A}_{CS}; \boldsymbol{A}_{CC}] = \text{Softmax}\big(\frac{[\boldsymbol{Q}_C \boldsymbol{K}_S^T; \boldsymbol{M}_C^T]}{\sqrt{D}}\big) \quad (5)$$

$$\boldsymbol{M}_C = [\boldsymbol{q}_1^{cT} \boldsymbol{k}_1^c; \cdots; \boldsymbol{q}_{L_C}^{c\ T} \boldsymbol{k}_{L_C}^c] \quad (6)$$

$$\boldsymbol{A}_{CC} = [\boldsymbol{a}_1^c; \cdots; \boldsymbol{a}_{L_C}^c] \quad (7)$$

$$\boldsymbol{O}_C = \boldsymbol{A}_{CS} \boldsymbol{V}_S + \sum_{j=1}^{L_C} \boldsymbol{a}_j \boldsymbol{v}_j^c \quad (8)$$

where $\boldsymbol{A}_{CC}$ is the intra-candidate or intra-block attention, and $\boldsymbol{a}_j^c$ is a scaler when we treat candidates as the average of sub-tokens and is a $B \times B$ matrix when we represent candidates as blocks. The last step (Eq. 8) can be parallelly implemented by Einstein summation.

B Baselines

We introduce recent PLM-based methods for UFET that we compare in Sec. 5.3 here. **LDET** (Onoe and Durrett, 2019) is an MLC with Bert-base-uncased and ELMo (Peters et al., 2018) trained on 727k examples automatically denoised from the distantly labeled UFET. **GCN** (Xiong et al., 2019) uses GCN to model type correlations and obtain type embeddings. Types are scored by dot-product of mention and type embeddings. The original paper uses BiLSTM as the mention encoder, but we report the results of the re-implementation by Jiang

et al. (2022) using RoBERTa-large. **BOX4TYPE** (Onoe et al., 2021) uses Bert-large as the backbone and uses box embedding to encode mentions and types for training and inference. **LRN** (Liu et al., 2021) use Bert-base as the encoder and an LSTM decoder to generate types in a seq2seq manner. **MLMET** (Dai et al., 2021) is an **MLC** with Bert-base, first pretrained by distantly-labeled data augmented with masked word prediction, and then fine-tuned and self-trained on the 2k human-annotated data. **PL** (Ding et al., 2021) uses prompt learning for entity typing. **DFET** (Pan et al., 2022) uses **PL** as the backbone and is a multi-round automatic denoising method on the 2k labeled data. **LITE** (Li et al., 2022) is the previous SOTA system that formulates entity typing as textual inference. **LITE** uses RoBERTa-large-MNLI as the backbone and is a cross-encoder (introduced in Sec. 2.3) with designed templates and a hierarchical loss. Jiang et al. (2022) proposes **NPCRF** to enhance backbones such as **PL** and **MLC** by modeling type correlations, reaching performance comparable to **LITE**.