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

Chengyue Jiang<sup>†</sup>, Wenyang Hui<sup>†</sup>, Yong Jiang, Xiaobin Wang, Pengjun Xie, Kewei Tu<sup>‡\*</sup>

School of Information Science and Technology, Shanghai Tech University
Shanghai Engineering Research Center of Intelligent Vision and Imaging
{jiangchy, huiwy, tukw}@shanghaitech, edu.cn;
{jiangyong.ml,xpjandy}@gmail.com;
czwangxiaobin@foxmail.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 the mention (and its context) with each type and feeds the pairs into a pretrained language model (PLM) to score their relevance. It brings deeper interaction between mention and types to reach better performance but has to perform N (type set size) forward passes to infer types of a single mention. CE is therefore very slow in inference when the type set is large (e.g., N = 10kfor UFET). To this end, we propose to perform entity typing in a recall-expand-filter manner. The recall and expand stages prune the large type set and generate K (K is typically less than 256) most relevant type candidates for each mention. At the filter stage, we use a novel model called MCCE to concurrently encode and score these K candidates in only one forward pass to obtain the final type prediction. We investigate different variants of MCCE and extensive experiments show that MCCE under our paradigm reaches SOTA performance on ultra-fine entity typing and is thousands of times faster than the cross-encoder. We also found MCCE 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 mention and is a funda-

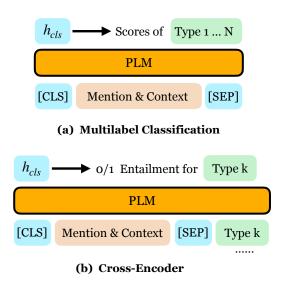


Figure 1: Cross-Encoder and multi-label classification.

mental step in fine-grained named entity recognition (Ling and Weld, 2012), and can 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) proposed 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 of mention-type pair as illustrated in Figure 1(b). Compared to the traditional multilabel classification method (shown in Figure 1(a)) that simultaneously scores all types using the mention representation, CE incorporates type semantics in the inference process and enables deeper interactions between types and mention to achieve better performance. However, the CE architecture is slow

<sup>\*</sup> Kewei Tu is the corresponding author.

Equal Contribution.

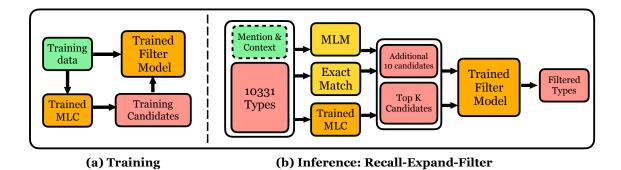


Figure 2: Training and inference of the recall-expand-filter pradigm.

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

To this end, we propose a recall-expand-filter Paradigm for UFET (illustrated in Figure 2) and a novel model called MCCE 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 K candidate types which reduce the number of potential types from thousands to hundreds. As the MLC model recalls candidates based on representations learned from the training data, it's hard to recall candidates that are scarce or unseen in the training set. To this end, we apply a multi-way 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. Last but not least, we propose a backbone called multi-candidate cross-encoder (MCCE) to concurrently encode and filter the expanded type candidate set. Different from CE, (MCCE) concatenates all recalled type candidates to the mention and its context. The concatenated input is then fed into a PLM to obtain candidate representations and candidate scores. The MCCE architecture allows us to infer types simultaneously from the candidate set while preserving the advantages of CE. Concatenating all candidates also enables MCCE implicitly learns the correlation between types. The advantages of MCCE over existing

		Interact M, C & T		Semantic of T
MLC	>			
CE		<b>~</b>		<b>~</b>
MCCE	>	<b>~</b>	<b>~</b>	<b>~</b>

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

architectures are shown in Figure 3. We also comprehensively investigate the performance and efficiency of **MCCE** with different input formats and attention mechanisms.

Experiments on two UFET datasets show that MCCE and its variants under our recall-expand-filter paradigm reach SOTA performance and are thousands of times faster than the CE-based previous SOTA method. We also found MCCE is still 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.

#### 2 Background

#### 2.1 Problem Definition

Given an entity mention  $m_i$  within its context sentence  $c_i$ , ultra-fine entity typing (UFET) aims to predict its correct types  $y_i^g \subset \mathcal{Y}$ , where  $y_i^g$  is the gold types of the *i*-th mention and is a subset of a large type set  $\mathcal{Y}(|\mathcal{Y}|)$  can be larger than 10k). As  $|y_i| > 1$  in most cases, UFET can be categorized as a multi-label classification problem. We show statistics of two UFET datasets: **UFET** (Choi et al.,

dataset	$ \mathcal{Y} $	$\operatorname{avg}( y_i^g )$	train/dev/test	Lang
UFET	10331	5.4	2k/2k/2k	EN
<b>CFET</b>	1299	3.5	3k/1k/1k	ZH

Table 1:  $avg(|y_i^g|)$  denotes the average number of gold types per instance, ZH for Chinese.

2018) and **CFET**<sup>1</sup> (Lee et al., 2020) in Table 1.

## 2.2 Multi-label Classification Model for UFET

Multi-label classification 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 multi-label classification model adopted by recent methods (Jiang et al., 2022; Dai et al., 2021). The contextualized mention representation is obtained by feeding  $c_i$  and  $m_i$  into the pretrained language models (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  $s_1, \dots s_N, N = |\mathcal{Y}|$ . We call this model MLC and describe its inference and training below.

**MLC Inference** For inference, types with probability higher than a threshold  $\tau$  are predicted:  $\mathcal{Y}_i = \{y_j | \sigma(s_j) > \tau\}$ ,  $\sigma$  is the sigmoid function. The threshold is tuned on the development set.

MLC Training Binary Cross-Entropy (BCE) loss between the predicted scores and the gold types are used to train the MLC model:  $\mathcal{L}_i = -\frac{1}{N}\sum_{j=1}^N \alpha \cdot I_j \log \sigma(s_j) + (1-I_j) \log(1-\sigma(s_j))$ , where  $I_j$  is the indicator of  $y_j$  being one of the gold types  $(y_j \in y_i^g)$ , and  $\alpha$  is a hyper-parameter balancing the loss of positive and negative types. 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). MLC also 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  $m_i$ ,  $c_i$ ,  $y_j$  together 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  $m_i, c_i$ .

$$m{h}_{cls,i} = extstyle{ t PLM([CLS]} \ c_i \ [SEP] \ m_i \ [SEP] \ y_j \ ) \ (1)$$

$$s_i = \mathsf{MLP}(\boldsymbol{h}_{cls,i}) \tag{2}$$

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

**CE Inference** CE predicts types of a single input  $(m_i, c_i)$  by concatenating the input with all possible types  $y_j \in \mathcal{Y}$  one by one to predict the scores  $s_1, \dots, s_j$  for each type. Similar to MLC, types that have a higher probability than a threshold are predicted  $\mathcal{Y}_i = \{y_j | \sigma(s_j) > \tau\}$ . CE requires N forward passes to infer types of a single mention, its inference speed is very slow when N is large.

**CE Training** CE is typically trained with marginal ranking loss (Li et al., 2022). A positive type  $y_+ \in y_i^g$  and a negative type  $y_- \notin y_i^g$  are sampled from  $\mathcal{Y}$  for each data point  $(m_i, c_i)$ . The loss is computed as:

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

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

#### 3 Methodology

Inspired by techniques in information retrieval (Larson, 2010) and entity linking (Ledell et al., 2020), we decompose the training and inference of UFET into three stages as illustrated in Figure 2: (1) Recall stage to reduce the type candidate size (e.g., from N = 10k to K = 100) while guaranteeing the recall rate by an efficient MLC model. (2) Expand stage to incorporate 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) to improve recall rate. (3) Filter stage to filter the expanded type candidates to obtain 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.

<sup>&</sup>lt;sup>1</sup>As there is no official split available for **CFET**, we split it by ourselves and will release our split in our code.

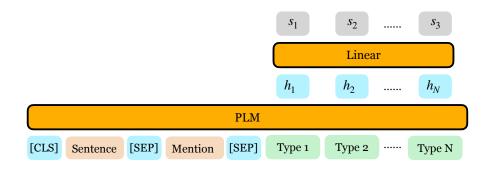


Figure 4: Multi-candidate cross-encoder (MCCE).

### 3.1 Recall Stage

To prune the type candidates set, we train a very efficient MLC model introduced in Sec. 2.2 and select the model 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  $\mathcal{C}_i^R$  for each data point  $(m_i, c_i)$  for train, development, and test set. We compare MLC with a widely-used baseline model BM25 (Robertson and Zaragoza, 2009) and show its advantages in Sec. 5.1.1.

#### 3.2 Expand Stage

Due to the lack of training data per type, we found that the MLC we used in the recall stage easily overfits the train set, and is hard to predict the types that only appear in the development and test set. In **UFET** dataset, 30% of the types in the development set are unseen. To this end, we utilize lexical information using exact match and weak supervision from the masked language model (MLM) to expand the recalled candidates. Both exact match and MLM are able to recall unseen type candidates without any training.

**Exact Match** MLC and Bi-Encoder recall candidates by dense representations. They are known weak at identifying and utilizing the lexical matching information between the input and types (Tran et al., 2019; Khattab and Zaharia, 2020). However, types are free-formed in UFET (e.g., president, businessman), and are very likely to appear in the context or mention (e.g., the mention is 'the president Joe Biden'). To this end, we first find all nouns in the context and mention by NLTK<sup>2</sup> POS tagger and normalize their forms, then we recall types that exactly matched with these nouns.

Weak Supervision from MLM Inspired by recently 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  $y_j \in \mathcal{Y}$  can be tokenized into l subwords  $w_1, \dots w_l$ . To score  $y_j$  given  $m_i, c_i$ , we first formulate the input as in Figure 5. where

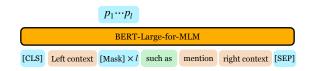


Figure 5: Recall from MLM using prompts.

 $c_i^l, c_i^r$  are left and right context of  $m_i$ , and 'such as' is the template we use to induce types. The input is then fed into BERT-large-uncased<sup>3</sup> for masked language model to obtain the probabilities of subwords, the score of  $y_j$  is calculated by  $s_j^{MLM} = (\sum_{n=1}^l \log p_n)/l$ , where  $p_l$  denotes the probability of subword  $w_l$  predicted by the PLM. We rank all types by enumerating all possible l and recall  $K_2$  additional candidates that haven't been recalled by the recall stage and exact match. We found that the expand stage improves recalls and contributes to the performance in Sec. 5.2.

## 3.3 Filter Stage

In the filter stage, we use the recall and expand method introduced above to efficiently generate type candidates  $C_i$  for data in the train, development, and test set. For training,  $C_i$  is used to produce positive and hard negative type candidates. For inference,  $C_i$  is the candidate pool for the trained filter models. Let  $|C_i| = K$  and K is typically less than 256.

<sup>&</sup>lt;sup>2</sup>nltk.tag package https://www.nltk.org

<sup>&</sup>lt;sup>3</sup>We use the PLM from https://huggingface.co

#### 3.3.1 CE

A trivial idea is to train a CE model introduced in Sec. 2.3 to filter  $C_i$  instead of filtering the whole type set  $\mathcal{Y}$ . The positive type  $y^+$  and negative  $y^-$  type are both sampled from  $C_i$  and are used for calculating marginal ranking loss. To infer types, we also recall and expand K candidates and score these candidates by K forward passes to predict types. As  $K << |\mathcal{Y}|$ , CE with our Recall-Expand-Filter paradigm is much faster than vanilla CE. However, it's still inefficient compared to MLC-like models that concurrently predict scores of all types in a single forward pass. For faster inference and training, we propose multi-candidate crossencoders (MCCE) and introduce them in the next section.

# 4 Multi-Candidate Cross-Encoder (MCCE)

In this section, we introduce **MCCE** for filtering candidates in one forward pass and propose several variants.

#### 4.1 Overall Introduction of MCCE

As shown in Figure 4, compared to CE that concatenates one candidate at a time, **MCCE** models concatenate all candidates in  $C_i$  with the mention and context. The input is then fed into the PLM to obtain the hidden states of each candidate as their representation. Finally, we use an MLP to concurrently score all candidates.

$$m{h}_{1:K} = extstyle{ t PLM([CLS]}\ c_i\ [SEP]\ m_i\ [SEP]\ t_{1:K}\ )$$
  $m{s}_{1:K} = extstyle{ t Linear}(m{h}_{1:K})$ 

where  $t_{1:K}$  is the short for  $t_1, \dots, t_K$ , and  $t_j \in C_i$ . Similarly,  $h_{1:K}$  and  $s_{1:K}$  are hidden representations and scores of corresponding candidates respectively.

**Training and Inference** For training, we found that all positive types are ranked very high in the training candidates, which is 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 first several candidates, we keep permuting type candidates during training. Same as the MLC model mentioned in Sec. 2.2, we use the Binary Cross-Entropy loss as the training objective and tune a threshold of probabilities on the development set for inference.

In the next subsection, we discuss different model configurations of MCCE regarding the input formats of candidates and attention mechanisms.

## 4.2 Different Input Formats of Candidates

Average of type sub-tokens We treat each type  $t_j \in \mathcal{Y}$  as a new token  $u_j$  and add it to the vocabulary of PLM. The static embedding (layer 0 embedding of PLM) of  $u_j$  is properly initialized by the average static embedding of  $t_j$ 's sub-tokens. As type candidates are capsuled into single tokens, the candidate representation  $t_j$  is simply the last hidden state of  $u_j$ . The reasons for representing each type as a single token is (1) The max sequence length allowed by most PLMs is limited to 512, compressing types into single tokens is position saving. (2) Types in **UFET** are tokenized into 2.1 sub-tokens in average (by RoBERTa's tokenizer). Compressing types will not lose too many type semantics.

**Fixed-size sub-token block** To preserve more type semantics, we place each candidate into a fixed-sized block as shown in Figure 6. We found the fixed block size makes PLM easier to enable the parallel implementation of different attention mechanisms that we will introduce next. We use the first hidden state in the block as the candidate representation.

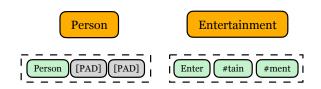


Figure 6: Illustration of candidate block.

### 4.3 Attentions in MCCE

There are four kinds of attention in MCCE as shown in Figure 7, sentence to sentence (S2S), sentence to candidates (S2C), candidate to sentence (C2S), and candidate to candidate (C2C). As we score candidates based on mention and its context, attention from candidates to the sentence (C2S) is necessary. However, the necessity of C2C, S2S, and S2C is questionable. As our analytical experiment in Sec. 6 shows, it is important for words in the sentence to attend to all candidates (S2C), and is useful to have self-attention in the sentence (S2S), but the attentions

(C2C) between different candidates are unnecessary. Based on these findings, we propose a new variant of MCCE that the C2C attention is discarded in computation as shown in the right part of Figure 7. 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  $Q_S = [q_1^s; \cdots; q_{L_S}^s] \in \mathbb{R}^{L_S \times D}$ , where  $q_i^s$  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  $Q_C = [q_1^c; \cdots; q_{L_C}^c] \in \mathbb{R}^{L_C \times D}$ . When we treat candidates as average of sub-tokens,  $oldsymbol{q}_i^c$  is a D-dimensional vector, and when we use fixedsized blocks to place candidates,  $\mathbf{q}_i^c \in \mathbb{R}^{B \times D}$  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  $K_C, V_C, O_C \in \mathbb{R}^{L_C \times D}, K_S, V_S, O_S \in \mathbb{R}^{L_S \times D}$ . The attention outputs are computed as:

$$O_S = \operatorname{Softmax}\left(\frac{Q_S[K_S; K_C]^T}{\sqrt{D}}\right) \cdot [V_S; V_C]$$
 (4)

$$[\boldsymbol{A}_{CS}; \boldsymbol{A}_{CC}] = \operatorname{Softmax}\left(\frac{[\boldsymbol{Q}_{C}\boldsymbol{K}_{S}^{T}; \boldsymbol{M}_{C}^{T}]}{\sqrt{D}}\right)$$
 (5)

$$\boldsymbol{M}_{C} = [\boldsymbol{q}_{1}^{cT} \boldsymbol{k}_{1}^{c}; \cdots; \boldsymbol{q}_{L_{C}}^{c} {}^{T} \boldsymbol{k}_{L_{C}}^{c}]$$
 (6)

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

$$O_C = A_{CS}V_S + \sum_{j=1}^{L_C} a_j v_j^c$$
 (8)

where  $A_{CC}$  is the intra-candidate or intra-block attention, and  $a_i^c$  is a scaler when we treat candidates as 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. In most cases, candidate length  $L_C$  is significantly larger than sentence length  $L_S$ . As a result, by ignoring the C2C attention, the inference speed is further improved because the time complexity of the attention is significantly reduced from  $O(D(L_S + L_C)^2)$  to  $O(D(L_S^2 + 2L_SL_C + B^2L_C))$ . More importantly, the space complexity in attention also gets reduced from  $O((L_S + L_C)^2)$  to  $O(L_S^2 + 2L_SL_C))$ , which allows us to filter more candidates concurrently. The improvement in space and time complexity by discarding C2C attention is more obvious when the number of candidates becomes larger.

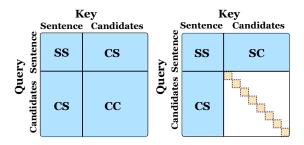


Figure 7: Attentions in MCCE (left), and MCCE without candidate-to-candidate (C2C) attention (right).

## 5 Experiments

We conduct experiments on two ultra-fine entity typing datasets, **UFET** (English) and **CFET** (Chinese). Their data statistics are shown in Table 1. We mainly focus on and report the macro-averaged recall at the recall and expand stage, and concern mainly on the macro-F1 of the final prediction at the filter stage. We also evaluate the **MCCE** models on the fine-grained (130 types) and coarse-grained (9 types) settings of entity typing without the recall and expand stage.

#### 5.1 UFET and CFET

#### 5.1.1 Recall Stage

We compare the recall@K on the test sets of UFET and CFET between the trained MLC model (introduced in 2.2) and a traditional BM25 model (Robertson and Zaragoza, 2009) in Figure 8. The MLC model uses the RoBERTa-large as backbone and is tuned based on the recall@128 on the development set. We use AdamW optimizer with a learning rate of  $2 \times 10^{-5}$ . Results show that MLC is a strong recall model, it consistently has better recall compared to BM25 on both UFET and CFET dataset, and the recall@128 reaches over 85% on UFET, and over 94% on CFET.

## 5.2 Expand Stage

In Table 2, we evaluate the F1 scores of all candidates expanded by exact match, and top-10 candidates expanded by the MLM using Bert-large. We also demonstrate the improvement of recall by using candidate expansion in Figure 9. On **UFET** dataset, expanding around 32 additional candidates based on 112 MLC candidates results in 2% higher recall compared to recalling all 128 candidates by MLC. The recall of 128 candidates after the expansion is comparable to the recall of 180 candidates

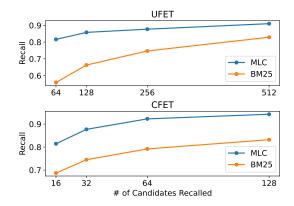


Figure 8: Recall@K of MLC and BM25.

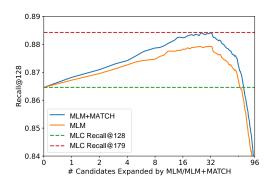
DATASET	EXPAND	P	R	F1	Avg # Expanded
UFET	MATCH MLM				5.23 10
CFET	MATCH MLM				4.57 10

Table 2: Evaluation of the recalled candidates.

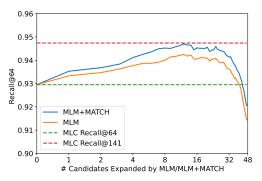
recalled from MLC. Similarly, expanding 10 candidates is comparable to additionally recalling 80 candidates using MLC. In our experiments, we replace the last 48 candidates recalled by MLC with the candidates recalled by MLM and Exact match for **UFET** and 10 for **CFET**. We found the expand stage has a positive effect on the final performance of **MCCE**s, and helps them reach SOTA performance (analyze in Sec. 6).

#### 5.3 Filter Stage and Final Results.

In this section, we report the performance of MCCE variants as the filter models and compare them with various strong baselines that we will introduce later. We also compare the inference speed of different models in this section. For filter models, we treat the number of candidates Krecalled and expanded by the first two stages as hyper-parameters, and tune it on the development set. We found the choice of PLM backbones has a non-negligible effect on the performance, and the PLM backbone of previous methods varies. Therefore for fairer comparisons to baselines, we conduct experiments of MCCE using different backbone PLMs for our MCCE models and report the results. For all MCCE models, we use AdamW optimizer with a learning rate tuned between  $5 \times 10^{-6}$ and  $2 \times 10^{-5}$ . The batch size we use is 4 and we train the models for at most 50 epochs with early



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



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

Figure 9: Demonstration of the effect of expand stage. x-axis represents the number of candidates expanded by MLM/MLM+MATCH among these 128 candidates.

stopping. **UFET** also provides a large dataset obtained from distant supervision such as entity linking, we do not use it and only train and evaluate our models on human-labeled data.

Baselines The MLC model we used 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. 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 and we use the results reimplemented by Jiang et al. (2022) using RoBERTalarge. BOX4TYPE (Onoe et al., 2021) uses Bertlarge 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

Bas	e Models on UFET	P	R	F1
ML	C-like models			
В	BOX4TYPES(Onoe et al., 2021)	52.8	38.8	44.8
В	LDET <sup>†</sup> (Onoe and Durrett, 2019)	51.5	33.0	40.1
В	<b>MLMET</b> <sup>†</sup> (Dai et al., 2021)	53.6	45.3	49.1
В	<b>PL</b> (Ding et al., 2021)	57.8	40.7	47.7
В	<b>DFET</b> (Pan et al., 2022)	55.6	44.7	49.5
В	MLC (reimplemented by us)	46.5	34.9	39.9
R	MLC (reimplemented by us)	42.2	44.9	43.5
Seq	2seq based models			
В	<b>LRN</b> (Liu et al., 2021)	54.5	38.9	45.4
Filte	er models under our recall-expand-filte	er para	digm	
В	VANILLA CE <sub>128</sub>	47.2	48.5	47.8
В	MCCE-S <sub>128</sub> (Ours)	53.2	48.3	50.6
В	MCCE-S <sub>128</sub> W/o C2C (Ours)	52.3	48.3	50.2
В	MCCE-B <sub>128</sub> (Ours)	49.9	50.0	49.9
В	MCCE-B <sub>128</sub> w/o C2C (Ours)	49.9	48.2	49.0
R	VANILLA CE <sub>128</sub>	49.6	49.0	49.3
R	MCCE-S <sub>128</sub> (Ours)	53.3	47.3	50.1
R	MCCE-S <sub>128</sub> w/o C2C (Ours)	53.2	46.6	49.7
R	MCCE-B <sub>128</sub> (Ours)	52.5	47.9	50.1
R	MCCE-B <sub>128</sub> w/o C2C (Ours)	52.7	46.4	49.3
-		P	n	T14
Lar	ge Models on UFET	r	R	F1
	C-like models	r	K	F1
	C-like models	47.8	40.4	43.8
ML	C-like models  MLC (Jiang et al., 2022)			
ML R	C-like models	47.8 48.7	40.4	43.8
ML R R	C-like models  MLC (Jiang et al., 2022)  MLC-NPCRF (Jiang et al., 2022)  MLC-GCN (Xiong et al., 2019)	47.8 48.7 51.2	40.4 45.5	43.8 47.0
ML R R R	C-like models  MLC (Jiang et al., 2022)  MLC-NPCRF (Jiang et al., 2022)  MLC-GCN (Xiong et al., 2019)  PL (Ding et al., 2021)	47.8 48.7	40.4 45.5 41.0	43.8 47.0 45.5
MLO R R R R B	C-like models  MLC (Jiang et al., 2022)  MLC-NPCRF (Jiang et al., 2022)  MLC-GCN (Xiong et al., 2019)  PL (Ding et al., 2021)  PL-NPCRF (Jiang et al., 2022)	47.8 48.7 51.2 59.3 55.3	40.4 45.5 41.0 42.6	43.8 47.0 45.5 49.6
MLO R R R R B	C-like models  MLC (Jiang et al., 2022)  MLC-NPCRF (Jiang et al., 2022)  MLC-GCN (Xiong et al., 2019)  PL (Ding et al., 2021)  PL-NPCRF (Jiang et al., 2022)  sss-encoder based models and MCCE	47.8 48.7 51.2 59.3 55.3	40.4 45.5 41.0 42.6	43.8 47.0 45.5 49.6
MLC R R R B B	C-like models  MLC (Jiang et al., 2022)  MLC-NPCRF (Jiang et al., 2022)  MLC-GCN (Xiong et al., 2019)  PL (Ding et al., 2021)  PL-NPCRF (Jiang et al., 2022)	47.8 48.7 51.2 59.3 55.3	40.4 45.5 41.0 42.6 46.7	43.8 47.0 45.5 49.6 50.6
MLO R R R B B Cro	C-like models  MLC (Jiang et al., 2022)  MLC-NPCRF (Jiang et al., 2022)  MLC-GCN (Xiong et al., 2019)  PL (Ding et al., 2021)  PL-NPCRF (Jiang et al., 2022)  sss-encoder based models and MCCE  LITE+L (Li et al., 2022)  LITE+NLI+L (Li et al., 2022)	47.8 48.7 51.2 59.3 55.3 s 48.7 52.4	40.4 45.5 41.0 42.6 46.7 45.8 48.9	43.8 47.0 45.5 49.6 50.6
MLO R R R B B Cro	C-like models  MLC (Jiang et al., 2022)  MLC-NPCRF (Jiang et al., 2022)  MLC-GCN (Xiong et al., 2019)  PL (Ding et al., 2021)  PL-NPCRF (Jiang et al., 2022)  sss-encoder based models and MCCE  LITE+L (Li et al., 2022)  LITE+NLI+L (Li et al., 2022)  er models under our recall-expand-filte	47.8 48.7 51.2 59.3 55.3 s 48.7 52.4	40.4 45.5 41.0 42.6 46.7 45.8 48.9	43.8 47.0 45.5 49.6 50.6
MLO R R R B B Cro R RM	C-like models  MLC (Jiang et al., 2022)  MLC-NPCRF (Jiang et al., 2022)  MLC-GCN (Xiong et al., 2019)  PL (Ding et al., 2021)  PL-NPCRF (Jiang et al., 2022)  sss-encoder based models and MCCE  LITE+L (Li et al., 2022)  LITE+NLI+L (Li et al., 2022)	47.8 48.7 51.2 59.3 55.3 s 48.7 52.4 er para	40.4 45.5 41.0 42.6 46.7 45.8 48.9 digm	43.8 47.0 45.5 49.6 50.6 47.2 50.6
MLO R R R B B Croo R RM Filto	C-like models  MLC (Jiang et al., 2022)  MLC-NPCRF (Jiang et al., 2022)  MLC-GCN (Xiong et al., 2019)  PL (Ding et al., 2021)  PL-NPCRF (Jiang et al., 2022)  ss-encoder based models and MCCE  LITE+L (Li et al., 2022)  LITE+NLI+L (Li et al., 2022)  er models under our recall-expand-filte  VANILLA CE <sub>128</sub> MCCE-S <sub>128</sub> (Ours)	47.8 48.7 51.2 59.3 55.3 s 48.7 52.4 er para 50.3	40.4 45.5 41.0 42.6 46.7 45.8 48.9 digm 49.6	43.8 47.0 45.5 49.6 50.6 47.2 50.6
MLO R R R B B Cro R RM Filte B B	C-like models  MLC (Jiang et al., 2022)  MLC-NPCRF (Jiang et al., 2022)  MLC-GCN (Xiong et al., 2019)  PL (Ding et al., 2021)  PL-NPCRF (Jiang et al., 2022)  ss-encoder based models and MCCE  LITE+L (Li et al., 2022)  LITE+NLI+L (Li et al., 2022)  er models under our recall-expand-filte  VANILLA CE <sub>128</sub> MCCE-S <sub>128</sub> (Ours)  MCCE-S <sub>128</sub> W/O C2C (Ours)	47.8 48.7 51.2 59.3 55.3 8 48.7 52.4 er para 50.3 52.5	40.4 45.5 41.0 42.6 46.7 45.8 48.9 digm 49.6 49.1	43.8 47.0 45.5 49.6 50.6 47.2 50.6 49.9 50.8
MLC R R B B Crook R RM Filto B B B B	C-like models  MLC (Jiang et al., 2022)  MLC-NPCRF (Jiang et al., 2022)  MLC-GCN (Xiong et al., 2019)  PL (Ding et al., 2021)  PL-NPCRF (Jiang et al., 2022)  sss-encoder based models and MCCE  LITE+L (Li et al., 2022)  LITE+NLI+L (Li et al., 2022)  er models under our recall-expand-filte  VANILLA CE <sub>128</sub> MCCE-S <sub>128</sub> (Ours)  MCCE-S <sub>128</sub> (Ours)  MCCE-B <sub>128</sub> (Ours)	47.8 48.7 51.2 59.3 55.3 8 48.7 52.4 er para 50.3 52.5 54.1 54.0	40.4 45.5 41.0 42.6 46.7 45.8 48.9 digm 49.6 49.1 47.1	43.8 47.0 45.5 49.6 50.6 47.2 50.6 49.9 50.8 50.4 51.2
MLC R R R B B Cro R RM Filta B B B B	C-like models  MLC (Jiang et al., 2022)  MLC-NPCRF (Jiang et al., 2022)  MLC-GCN (Xiong et al., 2019)  PL (Ding et al., 2021)  PL-NPCRF (Jiang et al., 2022)  ss-encoder based models and MCCE  LITE+L (Li et al., 2022)  LITE+NLI+L (Li et al., 2022)  er models under our recall-expand-filte  VANILLA CE <sub>128</sub> MCCE-S <sub>128</sub> (Ours)  MCCE-S <sub>128</sub> (Ours)  MCCE-B <sub>128</sub> (Ours)  MCCE-B <sub>128</sub> (Ours)	47.8 48.7 51.2 59.3 55.3 8 48.7 52.4 er para 50.3 52.5 54.1	40.4 45.5 41.0 42.6 46.7 45.8 48.9 digm 49.6 49.1 47.1 48.6	43.8 47.0 45.5 49.6 50.6 47.2 50.6 49.9 50.8 50.4
MLC R R R B B B Cro R RM Filte B B B B B B	C-like models  MLC (Jiang et al., 2022)  MLC-NPCRF (Jiang et al., 2022)  MLC-GCN (Xiong et al., 2019)  PL (Ding et al., 2021)  PL-NPCRF (Jiang et al., 2022)  Ses-encoder based models and MCCE  LITE+L (Li et al., 2022)  LITE+NLI+L (Li et al., 2022)  ET models under our recall-expand-filte  VANILLA CE <sub>128</sub> MCCE-S <sub>128</sub> (Ours)  MCCE-S <sub>128</sub> (Ours)  MCCE-B <sub>128</sub> (Ours)  MCCE-B <sub>128</sub> (WO C2C (Ours)  VANILLA CE <sub>128</sub>	47.8 48.7 51.2 59.3 55.3 8 48.7 52.4 er para 50.3 52.5 54.1 54.0 52.8	40.4 45.5 41.0 42.6 46.7 45.8 48.9 digm 49.6 49.1 47.1 48.6 48.3 49.3	43.8 47.0 45.5 49.6 50.6 47.2 50.6 49.9 50.8 50.4 51.2 50.4
MLC R R R B B Croo R RM Filte B B B B B R	C-like models  MLC (Jiang et al., 2022)  MLC-NPCRF (Jiang et al., 2022)  MLC-GCN (Xiong et al., 2019)  PL (Ding et al., 2021)  PL-NPCRF (Jiang et al., 2022)  sss-encoder based models and MCCE  LITE+L (Li et al., 2022)  LITE+NLI+L (Li et al., 2022)  er models under our recall-expand-filte  VANILLA CE <sub>128</sub> MCCE-S <sub>128</sub> (Ours)  MCCE-S <sub>128</sub> (Ours)  MCCE-B <sub>128</sub> (Ours)  MCCE-B <sub>128</sub> (WO C2C (Ours)  VANILLA CE <sub>128</sub> MCCE-S <sub>128</sub> (Ours)  MCCE-B <sub>128</sub> (Ours)	47.8 48.7 51.2 59.3 55.3 8 48.7 52.4 er para. 50.3 52.5 54.1 54.0 52.8 54.5 50.8	40.4 45.5 41.0 42.6 46.7 45.8 48.9 digm 49.6 49.1 47.1 48.6 48.3	43.8 47.0 45.5 49.6 50.6 47.2 50.6 49.9 50.8 50.4 51.2 50.4 51.8 50.3
MLC R R R R B B Cro R R RM Filta B B B B R R	C-like models  MLC (Jiang et al., 2022)  MLC-NPCRF (Jiang et al., 2022)  MLC-GCN (Xiong et al., 2019)  PL (Ding et al., 2021)  PL-NPCRF (Jiang et al., 2022)  sss-encoder based models and MCCE  LITE+L (Li et al., 2022)  LITE+NLI+L (Li et al., 2022)  er models under our recall-expand-filte  VANILLA CE <sub>128</sub> MCCE-S <sub>128</sub> (Ours)  MCCE-S <sub>128</sub> (Ours)  MCCE-B <sub>128</sub> (Ours)  MCCE-B <sub>128</sub> (Ours)  VANILLA CE <sub>128</sub> MCCE-S <sub>128</sub> (Ours)  MCCE-S <sub>128</sub> (Ours)  MCCE-S <sub>128</sub> (Ours)  WCCE-S <sub>128</sub> (Ours)  VANILLA CE <sub>128</sub> MCCE-S <sub>128</sub> (Ours)  MCCE-S <sub>128</sub> (Ours)  MCCE-S <sub>128</sub> (Ours)	47.8 48.7 51.2 59.3 55.3 8 48.7 52.4 er para 50.3 52.5 54.1 54.0 52.8 54.5	40.4 45.5 41.0 42.6 46.7 45.8 48.9 digm 49.6 49.1 47.1 48.6 48.3 49.8	43.8 47.0 45.5 49.6 50.6 47.2 50.6 49.9 50.8 50.4 51.2 50.4 51.8
MLC R R R R B B Cro R R RM Filta B B B B R R	C-like models  MLC (Jiang et al., 2022)  MLC-NPCRF (Jiang et al., 2022)  MLC-GCN (Xiong et al., 2019)  PL (Ding et al., 2021)  PL-NPCRF (Jiang et al., 2022)  sss-encoder based models and MCCE  LITE+L (Li et al., 2022)  LITE+NLI+L (Li et al., 2022)  er models under our recall-expand-filte  VANILLA CE <sub>128</sub> MCCE-S <sub>128</sub> (Ours)  MCCE-B <sub>128</sub> (Ours)  MCCE-B <sub>128</sub> (Ours)  WCCE-B <sub>128</sub> (Ours)  VANILLA CE <sub>128</sub> MCCE-S <sub>128</sub> (Ours)  MCCE-S <sub>128</sub> (Ours)  MCCE-B <sub>128</sub> (Ours)  WCCE-S <sub>128</sub> (Ours)  MCCE-S <sub>128</sub> (Ours)  MCCE-S <sub>128</sub> (Ours)  MCCE-S <sub>128</sub> (Ours)  MCCE-S <sub>128</sub> (Ours)	47.8 48.7 51.2 59.3 55.3 8 48.7 52.4 er para 50.3 52.5 54.1 54.0 52.8 54.5 50.8 51.5	40.4 45.5 41.0 42.6 46.7 45.8 48.9 digm 49.6 49.1 47.1 48.6 48.3 49.8 48.8	43.8 47.0 45.5 49.6 50.6 47.2 50.6 49.9 50.8 50.4 51.2 50.4 51.8 50.3 50.1 51.4
MLC R R R R B B B B B B B R R R R	C-like models  MLC (Jiang et al., 2022)  MLC-NPCRF (Jiang et al., 2022)  MLC-GCN (Xiong et al., 2019)  PL (Ding et al., 2021)  PL-NPCRF (Jiang et al., 2022)  sss-encoder based models and MCCE  LITE+L (Li et al., 2022)  LITE+NLI+L (Li et al., 2022)  er models under our recall-expand-filte  VANILLA CE <sub>128</sub> MCCE-S <sub>128</sub> (Ours)  MCCE-S <sub>128</sub> (Ours)  MCCE-B <sub>128</sub> (Ours)  MCCE-B <sub>128</sub> (Ours)  VANILLA CE <sub>128</sub> MCCE-S <sub>128</sub> (Ours)  MCCE-S <sub>128</sub> (Ours)  MCCE-S <sub>128</sub> (Ours)  WCCE-S <sub>128</sub> (Ours)  VANILLA CE <sub>128</sub> MCCE-S <sub>128</sub> (Ours)  MCCE-S <sub>128</sub> (Ours)  MCCE-S <sub>128</sub> (Ours)	47.8 48.7 51.2 59.3 55.3 8 48.7 52.4 er para 50.3 52.5 54.1 54.0 52.8 54.5 50.8 51.5	40.4 45.5 41.0 42.6 46.7 45.8 48.9 digm 49.6 49.1 47.1 48.6 48.3 49.8 48.8 50.8	43.8 47.0 45.5 49.6 50.6 47.2 50.6 49.9 50.8 50.4 51.2 50.4 51.8 50.3 50.1

Table 3: Macro-averaged UFET result. LITE+L is LITE without NLI pretraining, LITE+L+NLI is the full LITE model. Methods marked by † utilize either distantly supervised or augmented data for training. MCCE-S<sub>128</sub> denotes we use 128 candidates recalled and expanded from the first two stages.

encoder and an LSTM decoder to generate types in a seq2seq manner. MLMET (Dai et al., 2021) is a MLC with Bert-base, but first pretrained by the distantly-labeled data augmented by masked word prediction, then finetuned 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 backbone and is a multiround automatic denoising method for 2k labeled data. LITE (Li et al., 2022) is the previous SOTA

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 MLC-GCN (Xiong et al., 2019)	56.4	58.6	57.5
Filter models under our recall-expand-fi	lter par	adigm	
N VANILLA CE	57.6	64.3	60.7
C VANILLA CE	54.0	63.3	58.3
N MCCE-S <sub>64</sub> (Ours)	58.4	62.1	60.2
N MCCE-S <sub>64</sub> w/o C2C (Ours)	59.1	61.5	60.3
N MCCE-B <sub>64</sub> (Ours)	56.7	66.1	61.1
N MCCE- $\mathbf{B}_{64}$ w/o C2C (Ours)	58.8	64.1	61.4
C MCCE-S <sub>64</sub> (Ours)	55.5	62.6	58.8
C MCCE-S <sub>64</sub> w/o C2C (Ours)	54.0	63.4	58.3
C MCCE-B <sub>64</sub> (Ours)	55.0	63.5	59.0
C MCCE-B <sub>64</sub> w/o C2C (Ours)	57.3	61.3	59.3

Table 4: Macro-averaged CFET result.

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, and reach performance comparable to **LITE**.

Naming Conventions Let MCCE-S be the MCCE model that treats candidates as sub-tokens, and MCCE-B be the model representing candidates as fixed-size 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 and NeZha-base-Chinese, and denote them as C and N respectively.

UFET Results We show the results of UFET dataset in Table 3. The results show that: (1) The recall-expand-filter paradigm is effective. Filter models outperform all baselines without the paradigm by a large margin. The vanilla CE under our paradigm reaches 51.8 F1 compared to more complexed CE LITE with 50.6 F1 (2) MCCE models reach SOTA performances. MCCE-S<sub>128</sub> with BERT-base performs best and reaches 50.6 F1 score, which is comparable to previous SOTA performance of large models such as LITE+NLI+L

and PL+NPCRF. Among large models, MCCE-B<sub>128</sub> 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 found MCCE-B outperforms MCCE-S on large models, but underperforms MCCE-S on base models. (5) Backbone PLM matters. We found the performance of VANNILA CE under our paradigm is largely affected by the PLM it used. It reaches 47.8 F1 with BERT-base and 51.8 F1 with RoBERTa-large. For MCCE models, we found MCCE performs better than MCCE-B with BERT, and worse than MCCE-B with RoBERTa.

CFET Results We conduct experiments on CFET and compare MCCE models with several strong baselines: NPCRF and GCN with MLC-like architecture, and VANILLA CE under out paradigm which is proved to be better than **LITE** on **UFET**. The results are shown in Table 4. Similar to results in **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. In CFET, MCCE-B is significantly better than MCCE-S, on both Nezhabase and BERT-base-Chinese, indicating the importance of type semantics in Chinese language. We also find that MCCE w/o C2C is generally better than MCCE w/ C2C, it is possibly because the C2C attention distracts the candidates from attending to mention and contexts.

**Speed Comparison** Table 5 shows the theoretical inference complexity (number of PLM forward passes, and attention complexity), and practical inference speed (number of sentences inferred per second) of different models. We conduct the speed test using NVIDIA TITAN RTX for all models, and the inference batch size is 4. At the filter stage, the inference speed of MCCE-S is on par with MLC (even slightly faster because we don't need to score all types), and is about 40 times faster than VANNILA CE and thousands of times faster than LITE. MCCE-B w/o C2C is not significantly faster than MCCE-B as expected. It's possibly because the computation related to the block attention is not fully optimized by existing deep learning frameworks. The speed advantage of MCCE-B w/o C2C over MCCE-B will be greater with more candidates.

## 5.4 Fine-grained and Coarse-grained Entity Typing

We also conduct experiments on Fine-grained (130-class) and Coarse-grained (9-class, also known as "Open Entity") entity typing, and the results are shown in Table 6. As the type candidate set is much smaller in these settings, we skip the recall and expand stages and directly run the filter models and compare them to baselines. Results show that both MCCE-S and MCCE-B are still better than MLC and VANILLA CE, and MCCE-S is better than MCCE-B on coarser-grained cases possibly because the coarser-grained types are simpler in surface-forms and MCCE-S will not lose many type semantics.

### 6 Analysis

#### 6.1 Importance of Expand Stage

We perform the ablation study on the importance of the expand stage and show the results in Table 7. We compare the performances of MCCE-S using the expanded or the not expanded candidate sets on UFET and CFET. We replace the last 48 candidates recalled by MLC with candidates expanded by MLM and exact matching for UFET, and 10 candidates for CFET. Results show that expand stage has a positive effect on performance, it improves the final recall by +1.0 and +2.2 on UFET and CFET without harming the precision.

#### **6.2** Attentions

We conduct an ablation study on S2S, C2S, S2C, and C2C attention introduced in Sec. 4.3 and show the results in Table 8. From the results, we are surprised to find that removing C2C and S2S doesn't have a big negative impact on performance. The MCCE-S using BERT-base reaches 48.8 F1 even without both C2C and S2S attention. One possible reason is that 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 candidate in the next layer. We also find that the C2S is necessary for the task (18.7 F1 without C2S) because we rely on the mention and its context to encode and classify candidates. Furthermore, we found that it is important for sentences to attend to all candidates (S2C), indicating that the interaction between the sentence and different types is crucial for the task.

MODEL	# FP	ATTN	SENTS/SEC	F1
MLC	1	$L_S^2D$	58.8	43.8
LITE+NLI+L (CE)	N	$L_S^2D$	0.02	50.6
filter stage inference speed	il.			
VANILLA $\mathbf{CE}_{128}$	128	$L_S^2 D$ $(L_S + 128)^2 D$	1.64	51.8
$\mathbf{MCCE} extbf{-}\mathbf{S}_{128}$	1		60.8	50.1
MCCE-B <sub>128</sub>	1	$ (L_S + 128B)^2 D  (L_S^2 + 256L_SB + 128B^2)D $	22.3	51.4
MCCE-B <sub>128</sub> w/o C2C	1		25.2	<b>52.1</b>

Table 5: Inference speed comparison of models. # FP means the number of PLM forward passes required by a single inference. ATTN column lists the theoretical attention complexity. We also report the practical inference speed SENTS/SEC and the F1 scores on UFET with RoBERTa-large architecture.

Mo	Models		R	F1
co	arse (9 types) Open Entity			
R	MLC	76.8	78.5	77.6
R	VANILLA CE <sub>9</sub>	82.3	81.0	81.6
R	$MCCE-S_9$	77.0	87.7	82.0
R	MCCE-B <sub>9</sub> w/o C2C	77.2	85.4	81.1
fin	e (130 types)			
R	MLC	70.4	63.7	66.9
R	VANILLA $CE_{130}$	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 6: Micro-averaged results on UFET fine and coarse.

Ablation of Expand Stage		R	F1
UFET MCCE WITH C2C BERT-LAI	RGE		
<b>B</b> MCCE- $S_{128}$ (Ours)	52.5	49.1	50.8
<b>B</b> MCCE-S <sub>128</sub> w/o Expand (Ours)	52.7	48.1	50.3
CFET MCCE WITH C2C BERT-BAS	Е-Сні	NESE	
C MCCE-S <sub>64</sub> (Ours)	55.5	62.6	58.8
C MCCE-S <sub>64</sub> w/o Expand (Ours)	55.4	60.4	57.8

Table 7: Ablation study of expand stage.

#### 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 the arXiv. They propose a two-stage paradigm for selecting from multiple questions. They also propose a network similar to our MCCE-B to select from multiple options in parallel. We summarize the differences between their work and ours as follows: (1) Different in paradigm. We have an expand stage to further improve the quality of recalled candidates (2) Different in models. MCCE-S and MCCE-B are both different from theirs in both input format and scoring. We additionally propose to discard the C2C attention and study the effect of removing different parts of

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

Table 8: Attention analysis.

attention. (3) We focus more on entity typing and conduct extensive experiments covering two languages and three settings (ultra-fine-grained, finegrained, and coarse-grained). We analyze the effect of using different PLM backbones for a fairer and more comprehensive comparison. The paradigm of our work is also inspired by works in entity linking and information retrieval. Ledell et al. (2020) uses a retrieval and rerank paradigm for entity linking, they first generate entity candidates using a bi-encoder and rerank them using a vanilla crossencoder. Our paradigm with an additional expand stage and our proposed MCCE models are also potentially useful for entity linking. We leave it for future work. Zhang et al. (2022) represents the query document and candidate documents as vectors and proposed to use a transformer to rerank all candidate documents in parallel for passage retrieval. Compared to them, we tackle entity typing and preserve all information of mention and context rather than represent them as a single vector, the paradigm, model architecture, and training objective are also different.

### 8 Conclusion

In conclusion, we propose a recall-expand-filter paradigm for ultra-fine entity typing. We train a recall model to generate candidates and use MLM and exact match to improve the quality of recalled candidates, then use filter models to obtain final type predictions. We also propose a filter model called multi-candidate cross-encoder (MCCE) to concurrently encode and filter all candidates and study the influences of 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. MCCE models have comparable inference speed to simple (MCCE) models and are thousands of times faster than previous SOTA cross-encoders.

#### 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.
- 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.
- Yanzhao Zhang, Dingkun Long, Guangwei Xu, and Pengjun Xie. 2022. HLATR: enhance multi-stage text retrieval with hybrid list aware transformer reranking. *CoRR*, abs/2205.10569.