Denoising Enhanced Distantly Supervised Ultrafine Entity Typing

Yue Zhang, Hongliang Fei, Ping Li

Cognitive Computing Lab
Baidu Research
10900 NE 8th St. Bellevue, WA 98004, USA
{yuezhang030, feihongliang0, pingli98}@gmail.com

Abstract

Recently, the task of distantly supervised (DS) ultra-fine entity typing has received significant attention. However, DS data is noisy and often suffers from missing or wrong labeling issues resulting in low precision and low recall. This paper proposes a novel ultra-fine entity typing model with denoising capability. Specifically, we build a noise model to estimate the unknown labeling noise distribution over input contexts and noisy type labels. With the noise model, more trustworthy labels can be recovered by subtracting the estimated noise from the input. Furthermore, we propose an entity typing model, which adopts a bi-encoder architecture, is trained on the denoised data. Finally, the noise model and entity typing model are trained iteratively to enhance each other. We conduct extensive experiments on the Ultra-Fine entity typing dataset as well as OntoNotes dataset and demonstrate that our approach significantly outperforms other baseline methods.

1 Introduction

Entity typing is the task of identifying specific semantic types of entity mentions in given contexts. Recently, more and more research has focused on ultra-fine entity typing (Choi et al., 2018; Onoe and Durrett, 2019; Dai et al., 2021). Comparing to traditional entity typing tasks (Ren et al., 2016a,b; Xu and Barbosa, 2018; Ling and Weld, 2012; Yosef et al., 2013; Abhishek et al., 2017; Shimaoka et al., 2017; Xin et al., 2018; Dai et al., 2019; Zhang et al., 2022), the type set in ultra-fine entity typing is not restricted by KB schema, but includes a vast number of free-form types.

To automatically annotate the large-scale ultra-fine entity typing data, Choi et al. (2018) utilized different sources for distant supervision (DS), including: 1) entity linking, where they mine entity mentions that were linked to Wikipedia in HTML, and extract relevant types from their encyclopedic definitions, and 2) head words, where they automatically extracted nominal head words from raw text as types. However, distant supervision often suffers from the low-precision and low-recall problems (Ren et al., 2016b), where recall can suffer from KB or Wikipedia incompleteness, and precision can suffer when the selected types do not fit the context.

Instance	DS label
S1: On her first match on grass at the AEGON International in Eastbourne, Lisicki lost to [Samantha Stosur] in the first round.	actor, athlete, person
S2: [The film] was adapted by Hugh Walpole, Howard Estabrook and Lenore J. Coffee from the Dickens novel, and directed by George Cukor.	film, movie, show, art, entertainment, creation

Table 1: Examples selected from the Ultra-Fine Entity Typing dataset in Choi et al. (2018). Labels in red font indicate wrong labels, while labels in grey indicate missed labels.

Table 1 shows two examples from these datasets (Choi et al., 2018) to illustrate the challenges in automatic annotation using distant supervision. Sentence S1 is incorrectly annotated as actor through entity linking, which is beyond the given context. Sentence S2 shows that simply treating the head word film as the type label, while correct in this case, but misses many other valid types: movie, show, art, etc.

To address the noisy labeling problem in distantly supervised entity typing, researchers devoted much effort to denoising. Xiong et al. (2019) learns the hierarchical correlations between different

Findings of the Association for Computational Linguistics: ACL 2023, pages 9880–9892

types by injecting type co-occurrence Graph. Onoe et al. (2021) considers box embedding, which is more robust to data noise. While these methods implicitly learn to denoise data noise, it is difficult for humans to interpret their denoising capacity. Onoe and Durrett (2019) proposed an explicit denoising method, where they learn a filtering function and a relabeling function to denoise DS data and then train an entity typing model on the denoised DS dataset. However, they only utilized a small scale gold data to learn the filtering and relabeling function. Besides, their model did not model the dependency between context and entity phrases.

In this paper, we aim to develop an explicit denoising method for distantly supervised ultra-fine entity typing. Our framework mainly consists of two modules: a noise modeling component and an entity typing model. The noise model estimates the unknown labeling noise distribution over input contexts and observed (noisy) type labels. However, noise modeling is challenging because the noise information in the DS data is often unavailable, and noise can vary with different distant labeling techniques. To model the noise, we perturb the small-scale gold-labeled dataset's labels to mimic the DS's noise. Additionally, we utilize the L_1 norm regularization on the large-scale DS data to pursue the sparseness of labeling noise. Our noise model conditions on the input context sentence and its noisy labels to measure the underlying noise, where the denoised data can be recovered from DS data by subtracting the noise. For the entity typing model, we adopt a bi-encoder architecture to match input context and type phrases and train the entity typing model on gold labeled and denoised data. Finally, we design an iterative training (Tanaka et al., 2018; Xie et al., 2020) procedure to train the noise model and entity typing model iteratively to enhance each other.

We summarize our **contributions** as follows:

(i) We propose a denoising enhanced ultra-fine entity typing model under the distant supervised setting, including noise modeling and entity typing modeling. Unlike previous denoising work (Onoe and Durrett, 2019) to filter low-quality samples, our noise model directly measures underlying labeling noise, regardless of DS techniques.

(ii) Onoe and Durrett (2019) learns a relabel function to directly relabel samples, while, we model the labeling noise. iii) We evaluate our model on both the Ultra-Fine entity typing (UFET) and OntoNotes datasets, which are benchmarks for distantly supervised ultra-fine entity typing and fine-grained entity typing tasks. We show that our model can effectively denoise the DS data and learn a superior entity typing model through detailed comparison, analysis, and case study.

2 Related Works

2.1 Ultra-Fine Entity Typing

Entity typing is one of the information extraction tasks (Sun et al., 2018; Liu et al., 2020b,a; Zhang et al., 2021). The ultra-fine entity typing task was first proposed by Choi et al. (2018). They considered a multitask objective, where they divide labels into three bins (general, fine, and ultra-fine), and update labels only in a bin containing at least one positive label. To further reduce the distant supervision noise, Xiong et al. (2019) introduces a graph propagation layer to impose a label-relational bias on entity typing models to implicitly capture type dependencies. Onoe et al. (2021) uses box embedding to capture latent type hierarchies, which is more robust to the labeling noise comparing to vector embedding. Dai et al. (2021) proposes to obtain more weakly supervised training data by prompting weak labels from language models. Zhang et al. (2022) leverages retrieval augmentation to resolve the distant supervision noise.

Among the previous works, Onoe and Durrett (2019) is the most similar one to ours, where the filtering function is used to discard useless instances, and relabeling function is used to relabel an instance. Through filtering and relabeling, Onoe and Durrett (2019) explicitly denoise the distant supervision data. However, their denoising procedure is trained only on a small-scale gold-labeled data, while ignoring the large-scale data with distant supervision labels. In addition, our denoising method directly models the underlying label noise instead of brutally filtering all the samples with partial wrong labels.

2.2 Learning from Noisy Labeled Datasets

We briefly review the broad techniques for learning from noisy labeled datasets. Traditionally, regularization is an efficient method to deal with the issue of DNNs easily fitting noisy labels, including weight decay, dropout and multi-view consistency penalty (Fei and Li, 2020). Besides, a few studies achieve noise-robust classification using

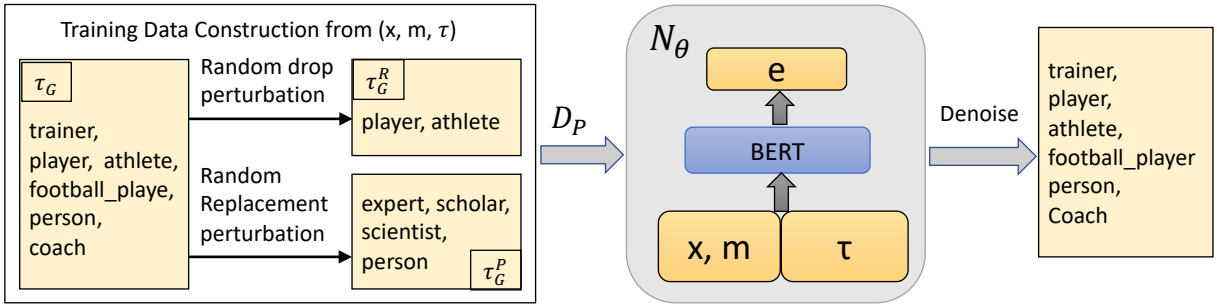

Figure 1: The procedure of training our noise model using one example. We use an instance from gold dataset "But Laporte, who names his World Cup squad on Wednesday, feels [he] has now found a World Cup fly..." as example, where we perturb the gold type set τ_G into low-recall set τ_G^R and low-precision set τ_G^P, separately. The noise model $\mathcal{N}_\theta$ takes the perturbed data as input, and outputs the estimated noise.

noise-tolerant loss functions, such as mean square error and mean absolute error (Ghosh et al., 2017). Recently, self-training (Xie et al., 2020) first uses labeled data to train a good teacher model, then uses the teacher model to label unlabeled data, and finally uses the labeled data and unlabeled data to jointly train a student model. Furthermore, various noise modeling methods are developed, including normalizing flows based methods (Abdelhamed et al., 2019), and GAN based methods (Chen et al., 2018). However, these noise modeling methods cannot be directly adapted to NLP tasks because of the differentiability issues.

3 Methodology

3.1 Problem Setup

Given l gold labeled triplets (context, mention, label) $\mathcal{D}_{\text{G}} = \{(x_{\text{G}}^{(i)}, m_{\text{G}}^{(i)}, \mathbf{Y}_{\text{G}}^{(i)})\}_{i=1}^{l}$ and n noisily labeled triplets $\mathcal{D}_{\text{N}} = \{(x_{\text{N}}^{(i)}, m_{\text{N}}^{(i)}, \mathbf{Y}_{\text{N}}^{(i)})\}_{i=1}^{n}$, where both $\mathbf{Y}_{\text{G}}^{(i)}$ and $\mathbf{Y}_{\text{N}}^{(i)} \in \{0,1\}^T$, and T is the total number of different types, our task aims to build a multi-label classification model to predict correct entity types for input contexts and mentions. For simplicity of notation, we define the complete entity type set as $\mathcal{T} = \{t_i\}_{i=1}^{T}$, where each t_i is a type represented as a phrase, e.g., "basketball player". Therefore, each label vector $\mathbf{Y}_{\text{G}}^{(i)}$ has a corresponding type set $\tau_{\text{G}} = \{t_j | Y_{\text{G}}^{(i,j)} = 1, j = 1, \cdots, T\}$, similarly for $\mathbf{Y}_{\text{N}}^{(i)}$ with τ_{N}.

3.2 Model Architecture

Our distantly supervised approach consists of two major components: a denoising module and an entity typing module. The denoising module models label noise based on the perturbed gold labeled data and existing noisy labeled data from distant supervision. In particular, we characterize two kinds of entity typing noise: i) low coverage (low recall), and ii) wrong labeling (low precision). Using a unified noise modeling mechanism, we build a connection between ground truth labels, observed labels, and noise. With reliable noise modeling, we can recover high-quality labels for noisy data and further train a more accurate entity typing model. Below we provide details of each component.

3.3 Noise Modeling

Given a certain context and mention pair (x, m), we assume the relation among gold label $\mathbf{y}_{\text{G}}$ and observed (noisy) label $\mathbf{y}_{\text{N}}$ is given by:

$$\mathbf{y}_{\text{G}} = [\min(\mathbf{y}_{\text{N}} - \mathbf{e}, 1)]_+ \quad (1)$$

where $[x]_+ = \max(x, 0)$, $\mathbf{e} \in \{-1, 0, 1\}^T$ is the noise term, including causes to both false positive and false negative errors. For gold labeled data $\mathcal{D}_{\text{G}}$, $\mathbf{y}_{\text{G}} = \mathbf{Y}_{\text{G}}^{(i)}$ and $\mathbf{e} = 0$. For noisily labeled data $\mathcal{D}_{\text{N}}$, $\mathbf{y}_{\text{G}}$ and $\mathbf{e}$ are unknown. Our denoising aims at recovering a more trustworthy label $\mathbf{y}_{\text{G}}$ from its noisy observation $\mathbf{y}_{\text{N}}$ by subtracting $\mathbf{e}$.

Figure 1 illustrates the workflow of our denoising model. The noise model $\mathcal{N}_\theta(x, m, \tau)$ is a neural network model parameterized by θ, which takes the query sentence x with the target entity mention m as well as the current assigned (noisy) type set τ as input, and outputs the noise measure $\mathbf{e} = \mathcal{N}_\theta(x, m, \tau)$. By Eq (1), it is relatively easy to conclude that $e_i \to 0$ indicates no change in the corresponding type assignment for type t_i. Similarly, $e_i \to 1$ indicates changing the type assignment towards negative, and when $e_i \to -1$ means changing the type assignment towards positive.

We use BERT (Devlin et al., 2019) model to build $\mathcal{N}_\theta(.)$. Specifically, BERT jointly encodes

input context, target mention as well as current assigned entity type set to d dimensional vector for each token and we extract the vector corresponding to the first token [CLS] as a pooled representation of the input as $\text{Embed}(x, m, \tau) = \text{BERT}_{\text{CLS}}(\text{Joint}(x, m, \tau))$.

To joint the context x, mention m and current assigned entity types τ in an entity-aware manner, we first utilize the special tokens preserved in BERT to indicate the positions of the target entity mention in x. Specifically, we insert [E0]/[/E0] at the beginning/ending of the target mention m. Following the BERT convention, we add special tokens [CLS] and [SEP] into the spans of context text and the entity type text spans. To encode the assigned type set τ, we concatenate the type's plain text after query x. Since there is no sequence order between types, for type phrases, the position ids of all the tokens in type phrase spans are set to be the length of encoded x. Hence $\text{Joint}(x, m, \tau)$ is defined as:

$$\begin{aligned}\text{Joint}(x, m, \tau) =&[\text{CLS}]w_1, ..., [\text{E0}]w_p, ..., w_q[\text{/E0}],\\ &..., w_n[\text{SEP}]t_i, ..., t_j[\text{SEP}],\end{aligned}$$

where $w_p, \cdots, w_q$ represents the tokens of mention m, and $t_i, \cdots, t_j$ are concatenated type phrases.

The estimated noise $\mathbf{e}$ is calculated by appending a linear layer with $\tanh$ activation on $\text{Embed}(.)$:

$$\mathbf{e} = \tanh\left(\mathbf{W} * \text{Embed}(x, m, \tau) + \mathbf{b}\right), \quad (2)$$

where $\mathbf{W} \in \mathcal{R}^{d \times T}$ and $\mathbf{b} \in \mathcal{R}^{T}$ are trainable parameters.

3.3.1 Training Data for Noise Modeling

We utilize the both available small-scale gold data and large-scale distant supervision data to train our noise model. Below we use ultra-fine dataset (Choi et al., 2018) as the example. Other datasets can be processed similarly.

Utilize Gold Labeled Data $\mathcal{D}_{\text{G}}$. We perturb the labels of $\mathcal{D}_{\text{G}}$ to mimic the low-recall and low-precision issues under distant supervision. First we analyze the average number of types in $\mathcal{D}_{\text{G}}$ and $\mathcal{D}_{\text{N}}$, respectively. In ultra-fine dataset (Choi et al., 2018), there are 5.4 and 1.5 types per gold example and DS example, respectively.

To mimic the low-recall issue, for each instance $(x_{\text{G}}, m_{\text{G}}, \tau_{\text{G}})$ from $\mathcal{D}_{\text{G}}$, we randomly drop each type with a fixed rate 0.7 independent of other types to produce a corrupted type set τ_{G}^{R}. We denote the corrupted gold data with randomly dropped types as $\mathcal{D}_{\text{G}}^{\text{R}}$. Meanwhile, to mimic the low-precision issue, for each instance $(x_{\text{G}}, m_{\text{G}}, \tau_{\text{G}})$, we also randomly replace its gold entity type set τ_{G} to a random set τ_{G}^{P}, where τ_{G}^{P} is randomly sampled from $\mathcal{D}_{\text{N}}$. Note that τ_{G} and τ_{G}^{P} may or may not have overlapping entity types. The non-overlapping replacement leads to a totally corrupted DS instance. The overlapping replacement represents the partially correct labeled instance. We denote the corrupted gold data with randomly replaced labels as $\mathcal{D}_{\text{G}}^{\text{P}}$. Given the complete entity set $\mathcal{T}$, τ_{G} and $\tau_{\text{P}} \in \{\tau_{\text{G}}^{P}, \tau_{\text{G}}^{R}\}$, it is straightforward to construct multi-hot vector representations $\mathbf{y}_{\text{G}}$ (i.e., $\mathbf{y}_{\text{G}} = \mathbf{Y}_{\text{G}}^{(i)}$) and $\mathbf{y}_{\text{P}} \in \{0, 1\}^{T}$. Finally, we collect the combined perturbation dataset $\mathcal{D}_{\text{P}} = \mathcal{D}_{\text{G}}^{\text{P}} \cup \mathcal{D}_{\text{G}}^{\text{R}}$.

Utilize Distant Supervision Data $\mathcal{D}_{\text{N}}$. Although the perturbed dataset $\mathcal{D}_{\text{P}}$ could be large, the gold labeled dataset $\mathcal{D}_{\text{G}}$ per se is still small, which means the number of different query sentences in $\mathcal{D}_{\text{P}}$ is limited. Hence training the noise model $\mathcal{N}_{\theta}(.)$ only on $\mathcal{D}_{\text{P}}$ may be insufficient for satisfactory performance.

Although distant supervision datasets are noisy and the noise is unknown, they still can provide weak supervision. Hence we use the available large-scale DS dataset $\mathcal{D}_{\text{N}}$ to better train $\mathcal{N}_{\theta}(.)$. Our motivation grounds on the study in Choi et al. (2018) showing that removing any source of distant supervision data from the whole training set results in a significant performance drop of the entity typing model. In other words, DS data contains a significant amount of correctly assigned entity types. Inspired by the analysis in Choi et al. (2018), we argue that the estimated noise $\mathbf{e}$ on DS data should be sparse. The sparsity enables us to design a suitable loss function to use $\mathcal{D}_{\text{N}}$ in training $\mathcal{N}_{\theta}(.)$.

3.3.2 Objective Function for Noise Modeling

Training the noise model $\mathcal{N}_{\theta}(.)$ on $\mathcal{D}_{\text{P}}$ is a supervised learning procedure, and we apply the binary cross-entropy loss on each entity type. We consider below loss function for one corrupted input $((x, m, \mathbf{y}_{\text{P}}, \tau_{\text{P}}), \mathbf{y}_{\text{G}})$ from $\mathcal{D}_{\text{P}}$:

$$\begin{aligned}J_{\mathcal{D}_{\text{P}}} = -\sum_{t=1}^{T}[&\mathbf{y}_{\text{G}}^{(t)} \cdot \log \hat{y}_t + \\ &(1 - \mathbf{y}_{\text{G}}^{(t)}) \cdot \log(1 - \hat{y}_t)]\end{aligned} \quad (3)$$

$$\hat{y}_t = [(\min(\mathbf{y}_{\text{P}}^{(t)} - \mathcal{N}_{\theta}^{(t)}(x, m, \tau_{\text{P}}), 1)]_{+} \quad (4)$$

To utilize $\mathcal{D}_{\text{N}}$ on training $\mathcal{N}_{\theta}(.)$, we use L_1 norm

regularization on the difference between predicted labels and observed (noisy) labels. L_1 norm enforces sparseness, which leads to zero noise on a certain entity types. Such a procedure makes our prediction partially consistent with observed noisy labels, which is reasonable since $\mathcal{D}_{\text{N}}$ contains a significant amount of correct labels (Choi et al., 2018). The loss function on one instance $((x, m, \tau_{\text{N}}), \mathbf{Y}_{\text{N}}^{(i)})$ from $\mathcal{D}_{\text{N}}$ is as follows:

$$J_{\mathcal{D}_{\text{N}}} = ||\hat{\mathbf{y}} - \mathbf{Y}_{\text{N}}^{(i)}||_1, \tag{5}$$

where $\hat{\mathbf{y}} = [(\min(\mathbf{Y}_{\text{N}}^{(i)} - \mathcal{N}_\theta(x, m, \tau_{\text{N}}), 1)]_+$. The overall objective function becomes:

$$J_{\text{denoising}} = J_{\mathcal{D}_{\text{P}}} + \alpha * J_{\mathcal{D}_{\text{N}}} \tag{6}$$

where $\alpha \geq 0$ is the regularization parameter, which is set as a small value so that distant supervision data can provide weak supervision but without overwhelming the training procedure.

3.4 Entity Typing Model

After training the noise model, we apply the learned model $\mathcal{N}_\theta(.)$ on DS data $\mathcal{D}_{\text{N}}$ to get the denoised dataset $\mathcal{D}_{\text{D}}$. We then use both $\mathcal{D}_{\text{G}}$ and $\mathcal{D}_{\text{D}}$ to train our entity typing model $\mathcal{M}_\phi(.)$ parameterized by ϕ. Our entity typing model adopts the two-tower architecture, including the context tower and type candidate tower, as shown in Figure 2.

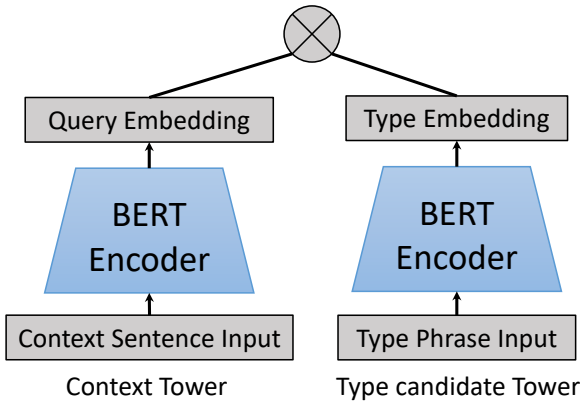

Figure 2: The architecture of entity typing model.

In particular, the context tower takes the context sentence as input. We encode the sentence in an entity-aware manner using BERT model:

$$\text{Joint}_{\text{context}}(x) = [\text{CLS}]w_1, ..., [\text{E0}]w_p, ..., w_q[\text{/E0}]...$$
$$\text{Embed}_{\text{context}}(x) = \text{BERT}_{\text{CLS}}(\text{Joint}_{\text{context}}(x))$$

The candidate tower takes one entity type phrase as input. Again, we use another BERT model to encode the type phrase:

$$\text{Joint}_{\text{candidate}}(t) = [\text{CLS}]w_1, ..., w_n[\text{SEP}]$$
$$\text{Embed}_{\text{candidate}}(t) = \text{BERT}_{\text{CLS}}(\text{Joint}_{\text{candidate}}(t))$$

where $w_1, ..., w_n$ represents tokens of one type t.

The final matching score $s(x, t)$ is computed as the inner product of the query embedding and the type embedding followed by a sigmoid activation:

$$s(x, t) = \sigma(\text{Embed}_{\text{context}}(x)^T \text{Embed}_{\text{candidate}}(t))$$

where $\sigma(.)$ is the sigmoid function, which maps the value into 0 to 1. In our entity typing model $\mathcal{M}_\phi$, we independently compute the matching score for each candidate type t.

Objective function. Previous works Choi et al. (2018); Xiong et al. (2019); Onoe and Durrett (2019); Onoe et al. (2021); Dai et al. (2021) all adopt multi-task learning to handle the labeling noise, where they partition the labels into general, fine, and ultra-fine classes, and only treat an instance as an example for types of the class in question if it contains a label for that class. The multi-task objective avoids penalizing false negative types and can achieve higher recalls. In our work, since we already denoise and re-label the distant supervision data using our learned model $\mathcal{N}_\theta(.)$, we directly train the entity typing model using cross entropy loss without multi-task learning:

$$J_{\text{typing}} = -\sum_{t=1}^{T}[y_t \cdot \log \hat{y}_t + (1 - y_t) \cdot \log(1 - \hat{y}_t)]$$
$$\hat{y}_t = \mathcal{M}_\phi(x, t) \tag{7}$$

3.5 Iterative Training

In our framework, the noise model $\mathcal{N}_\theta(.)$ and the entity typing model $\mathcal{M}_\phi$ are iterative trained as shown in Figure 3. We describe one training iteration for $\mathcal{N}_\theta(.)$ and $\mathcal{M}_\phi$ in the following:

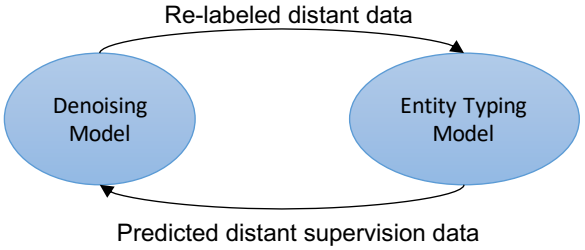

Figure 3: Illustration of the iterative training.

Updating $\mathcal{N}_\theta(.)$: We train the noise model $\mathcal{N}_\theta(.)$ by Eq (6) using the perturbed gold labeled dataset $\mathcal{D}_\text{P}$ and noisy dataset $\mathcal{D}'$. At the first iteration, $\mathcal{D}'$ is from the original distant supervision, $\mathcal{D}' = \mathcal{D}_\text{N}$. After the first iteration, labels in $\mathcal{D}'$ are re-calculated by applying current entity typing model $\mathcal{M}_\phi(.)$. Also, after each iteration, we increase the value of the weight α in Eq (6). After noise modeling, we get the denoised dataset $\mathcal{D}_\text{D}$ by removing the noise calculated from applying $\mathcal{N}_\theta(.)$ on $\mathcal{D}'$.

Updating $\mathcal{M}_\phi(.)$: We train the entity typing model $\mathcal{M}_\phi(.)$ by Eq (7) using gold dataset $\mathcal{D}_\text{G}$ and the latest denoised dataset $\mathcal{D}_\text{D}$. After training current $\mathcal{M}_\phi(.)$, we re-calculate the labels of distant supervision data, and get the updated DS dataset $\mathcal{D}'$. We pass $\mathcal{D}'$ to the next noise modeling iteration.

4 Experiments

4.1 Experimental Setup

Datasets Our experiments mainly focus on the Ultra-Fine entity typing (UFET) dataset, which has 10,331 labels. The distant supervision training set is annotated with heterogeneous supervisions based on KB, Wikipedia, and headwords, resulting in about 25.2M training samples. This dataset also includes around 6,000 crowdsourced samples equally split into training, validation, and test set.

In addition, we investigate on OntoNotes dataset, which is a widely used benchmark for fine-grained entity typing systems. The initial training, development, and test splits contain 250K, 2K, and 9K examples, respectively. Choi et al. (2018) augmented the training set to include 3.4M distant supervision examples. To train our noise model, we further augment the training data using the 2,000 training crowdsourced samples from the UFET dataset. We map the labels from ultra-fine types to OntoNotes types. Most OntoNote's types can directly correspond to UFET's types (e.g., "doctor" to "/person/doctor"). We then expand these labels according to the ontology to include their hypernyms (e.g., "/person/doctor" will also generate "person").

Baselines. For the UFET dataset, we compare with

1) AttentiveNER (Shimaoka et al., 2016);
2) Multi-task model (Choi et al., 2018), which is proposed together with the UFET data;
3) LabelGCN (Xiong et al., 2019);
4) BERT (Onoe and Durrett, 2019), which was first introduced as a baseline;
5) Filter+Relabel (Onoe and Durrett, 2019);
6) Vector Embedding (Onoe et al., 2021);
7) Box Embedding (Onoe et al., 2021);
8) MLMET (Dai et al., 2021).

For experiments on OntoNotes, additionally, we compare with AFET (Ren et al., 2016a), LNR (Ren et al., 2016b), and NFETC (Xu and Barbosa, 2018).

Evaluation Metrics. For the UFET dataset, we report the mean reciprocal rank (MRR), macro precision(P), recall (R), and F_1. As P, R and F_1 all depend on a chosen threshold on probabilities, we tune the threshold on the validation set from 50 equal-interval thresholds between 0 and 1 and choose the optimal threshold which can lead to the best F1 score. Then, we use the found optimal threshold for the test set. Also, we plot the precision-recall curves, which are the more transparent comparison. For the OntoNotes dataset, we report the standard metrics used by baseline models: accuracy, macro, and micro F1 scores.

Implementation Details. To train models on UFET dataset, all the baselines adopt the multi-task loss proposed in Choi et al. (2018). For our model, we use the standard binary cross-entropy (BCE) losses in Eq (6, 7). We carefully tune α from $[0.05, 0.1, 0.25, 0.5, 0.75, 1]$ and set it to 0.25 based on validation set. We use "BERT-base-uncased" to initialize Bert encoder weights, and set dropout rate to 0.1. We use Adam optimizer (Kingma and Ba, 2015) with learning rate $3e-5$. We repeat our experiments five times and report the average metrics on the test set.

MLMET results are reproduced using the public released code and data, not directly taken from their paper. For all our ultra-fine entity typing experiments, we consider 25.2M distant supervision training samples and 6,000 crowdsourced samples equally split into training, validation, and test set. While, the original MLMET also consider additional 3.7M pronoun mentions dataset from EN Gigaword.

4.2 Evaluation Results

Evaluation on UFET Dataset. We report the comparison results on UFET in Table 2. MRR score is independent with threshold choices. For F1 score, we apply threshold-tuning, which further improves the F1 score on both the development and test sets. In terms of MRR and F1, our model outperforms baseline methods by a large margin, especially on

Model	Dev				Test			
	MRR	P	R	F1	MRR	P	R	F1
AttentiveNER	22.1	53.7	15.0	23.5	22.3	54.2	15.2	23.7
Multi-task	22.9	48.1	23.2	31.3	23.4	47.1	24.2	32.0
LabelGCN	25.0	**55.6**	25.4	35.0	25.3	**54.8**	25.9	35.1
BERT	-	51.6	32.8	40.1	-	51.6	33.0	40.2
Filter+Relabel	-	50.7	33.1	40.1	-	51.5	33.0	40.2
VectorEmb	-	53.3	36.7	43.5	-	53.0	36.3	43.1
BoxEmb	-	52.9	39.1	45.0	-	52.8	38.8	44.8
MLMET*	29.0	53.6	39.4	45.4	29.2	53.4	40.5	46.1
Ours	**30.3**	52.8	41.7	46.6	**30.9**	53.4	41.9	47.0
Ours+ *thresholding*	**30.3**	50.8	**43.7**	**47.0**	**30.9**	51.2	**43.7**	**47.3**

Table 2: Comparison with baseline models on the UFET dataset. All the baseline results are from their papers. "-" means no report. Best results with statistical significance are marked in bold (one-sample t-test with $p < 0.05$). "*" means we reproduced the results based on the public released code and dataset.

the test set. We can see that recall is usually lagging behind precision by a large margin for most baseline models. It is because that these baselines easily correctly predict the nine general types but have difficulty predicting the large number of fine-grained and ultra-fine types correctly. On the other hand, our model can balance the precision and recall scores well even without threshold-tuning. The "thresholding" sacrifices the precision and tunes towards recall to lead to a higher F1 score.

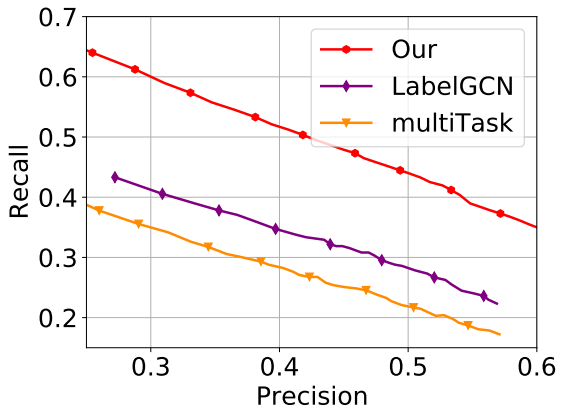

Figure 4: Precision-recall curves on UFET dev set.

For a more transparent comparison, we show the precision-recall curves in Figure 4. These data points are based on the performance on the development set given by 50 equal-interval thresholds between 0 and 1. We can see there is a clear margin between our model v.s. LabelGCN and the multi-task model (Choi et al., 2018). With higher recalls or more retrieved types, achieving high precision requires being accurate on fine-grained and ultra-fine types, which are often harder to predict.

Model	Acc	Mac-F1	Mic-F1
AttentiveNER	51.7	71.0	64.9
AFET	55.1	71.1	64.7
LNR	57.2	71.5	66.1
NFETC	60.2	76.4	70.2
Multi-task	59.5	76.8	71.8
LabelGCN	59.6	77.8	72.2
BERT	51.8	76.6	69.1
Filter+Relabel	64.9	84.5	79.2
MLMET	67.4	**85.4**	80.4
Ours	**70.0**	**85.4**	**80.9**

Table 3: Comparison results on OntoNotes. Best results with statistical significance are marked in bold.

Evaluation on OntoNotes Dataset. We report the comparison results on OntoNotes in Table 3. Baseline models including AttentiveNER, AFET, LNR, and NFETC explicitly use the hierarchical type structures provided by the OntoNotes ontology. While other baselines and ours do not consider the type hierarchy and treat each type as a free-form phrase. From Table 3, we can see that our model significantly outperforms other baselines on all the metrics, especially on the accuracy metric.

4.3 Analysis and Ablation Study

4.3.1 Utility of Iterative Training

First, we analyze the effectiveness of iterative training. We report the test results of our model and

Filter+Relabel model under different iteration steps. In the original Filter+Relabel model, the filter and relabel functions are trained on the gold data only. To make the Filter+Relabel model take advantage of the iterative training, we relabel all the DS data by the trained entity typing model after each iteration. Then we leverage the current filtering function to evaluate the relabeled DS samples and filter out the high-quality DS samples. Then we joint the high-quality DS samples with gold data to train filter and relabel functions in the next iteration.

Iteration	Model	P	R	F1
1	Ours	46.6	46.4	46.5
	Filter+Relabel	50.7	33.1	40.1
2	Ours	48.0	46.1	47.0
	Filter+Relabel	52.3	35.0	41.9
3	Ours	51.2	43.7	47.3
	Filter+Relabel	52.9	35.1	42.2

Table 4: Test results of our model and Filter+Relabel on UFET under different iteration training steps.

Table 4 shows that with the step of iteration increasing, the overall performance of both models get better. This proves the significance of iterative training. With the iteration step increasing, the DS data becomes less noisy. Also, we can see at every iteration step, our model is much better than Filter+Relabel, which proves that modeling the noise instead of the label is a better choice.

4.3.2 Effectiveness of Noise Modeling

Since noise modeling's output directly impacts the final entity typing performance, we also quantify the performance of trained $\mathcal{N}_\theta(.)$ on held-out gold-labeled dev set and perturbed gold dev set to mimic the low-recall and low-precision scenarios.

Data	Filter+Relabel	Ours
Gold Dev	87.9	**94.2**
Low-recall set	54.5	**55.8**
Low-precision set	33.0	**35.5**

Table 5: F1 scores of noise modeling on three datasets.

We report the F_1 score on these three datasets in Table 5. To fairly compare with Filter+Relabel, we re-implement it using BERT as the backbone. Our denoising module generates more accurate prediction and is more robust to noise.

4.3.3 Consistency Analysis

We investigate whether our model can predict type relations in a consistent manner. Following the evaluation in Onoe et al. (2021), we conduct the analysis on the UFET dev set. We count the number of occurrences for all subtypes in 30 (supertype, subtype) pairs listed in Onoe et al. (2021). Then, for each subtype, we count how many times its corresponding supertype is also predicted. Finally, the accuracy (acc) is the ratio of predicting the corresponding supertype when the subtype is exhibited.

Model	# (sup, sub)	# sub	acc
VectorEmb	1451	1631	89.0
BoxEmb	1851	1997	92.7
Ours	**2514**	**2674**	**94.0**

Table 6: Consistency: accuracy evaluated on the 30 (supertype, subtypes) pairs.

Table 6 reports the count and accuracy of the 30 (supertype, subtype) pairs, where the #(sup, sub) column shows the number of pairs found in the predictions, # sub column shows the number of subtypes found in the predictions. Our model achieves a higher count and accuracy. Intuitively, a higher count indicates a higher recall of the model. A higher accuracy proves that although the supertype-subtype relations are not strictly defined in the training data, our model still captures the correlations.

4.3.4 Ablation Study

Model	MRR	P	R	F1
Full model	**30.3**	**50.7**	**43.5**	**46.8**
w/o denoise	27.2	43.7	39.2	41.3
w/o $\mathcal{D}_\text{N}$	28.3	45.3	39.7	42.3
w/o cross-attn.	29.9	47.1	43.9	45.4

Table 7: Ablation study on the UFET test set.

To prove the effectiveness of our denoising mechanism for the entity typing task, we conduct an ablation study on the UFET dev set and show the results in Table 7. We study three model variants, including i) full model w/o denoising, where we train entity typing model on gold data and DS data; ii) full model w/o denoising or DS data, where we

train entity typing model only on gold data; and iii) full model w/o cross-attention between the input context and assigned entity type phrases when modeling noise.

From Table 7, we first observe that directly training an entity typing model without our denoising mechanism results in a significant performance drop. Second, we see that introducing more distant supervision data $\mathcal{D}_{\text{N}}$ can improve the overall entity typing performance. Finally, joining the context and assigned types and introducing self-attention (Vaswani et al., 2017) further improves F_1 score by 1.6%. Therefore, when designing the denoising model, it is necessary to fully explore the dependency between the context sentence and the assigned type set instead of a simple sum-pooling in Filter+Relabel (Onoe and Durrett, 2019).

4.4 Case Study

To better explore the effectiveness of our denoising model, we show two case studies from the DS development set. We show all types with at least one score over the threshold of 0.5, or is annotated true by distant supervision in Table 8 and Table 9. The target entity mentions are underlined within brackets.

Case Study S1: For the context "My grandfather joined an [artillery regiment] with the Canadian Expeditionary Force and then set off to fight...", as shown in Table 8, DS label misses the type "group", our denoising mechanism successfully identify the missing type, but Filter+Relabel (Onoe and Durrett, 2019) fails.

Type	DS	Filter+Relabel	Ours
group	0	0.0	0.56
artillery	1	0.67	0.83
regiment	1	0.19	0.73

Table 8: Case study on DS instance S1, which is labeled by head words.

Type	DS	Filter+Relabel	Ours
person	1	0.00	0.05
engineer	1	0.54	0.46
writing	1	0.92	0.88

Table 9: Case study on DS instance S2, which is labeled by entity linking.

Case Study S2: For the context "Richard Schickel, writing in [Time magazine] gave a mixed review, ...", from Table 9, we see DS wrongly assigned "person" and "engineer" to the entity "Time magazine". Both Filter+Relabel and our work successfully lower the probability scores of the wrong types.

5 Conclusion

In this paper, we aim to improve the performance of ultra-fine entity typing by explicitly denoising the DS data. Noise modeling is our key component to denoise, where the model fully explores the correlation between the query context and assigned noisy type set, and outputs the estimated noise. To train the noise model, we perturb on the small-scale gold dataset to mimic the noise distribution on DS instances. Furthermore, we utilize the large-scale DS data as weak supervision to train our noise model. The entity typing model is then trained on the gold data set and denoised DS dataset. Experimental results empirically prove the effectiveness of our method on handling distantly supervised ultra-fine entity typing.

References

Abdelrahman Abdelhamed, Marcus A. Brubaker, and Michael S. Brown. 2019. Noise flow: Noise modeling with conditional normalizing flows. In *Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3165–3173, Seoul, Korea.

Abhishek, Ashish Anand, and Amit Awekar. 2017. Fine-grained entity type classification by jointly learning representations and label embeddings. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 797–807, Valencia, Spain.

Jingwen Chen, Jiawei Chen, Hongyang Chao, and Ming Yang. 2018. Image blind denoising with generative adversarial network based noise modeling. In *Proceedings of the 2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3155–3164, Salt Lake City, UT.

Eunsol Choi, Omer Levy, Yejin Choi, and Luke Zettlemoyer. 2018. Ultra-fine entity typing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 87–96, Melbourne, Australia.

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 (ACL/IJCNLP), pages 1790–1799, Virtual Event.

Zeyu Dai, Hongliang Fei, and Ping Li. 2019. Coreference aware representation learning for neural named entity recognition. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 4946–4953, Macao, China.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 4171–4186, Minneapolis, MN.

Hongliang Fei and Ping Li. 2020. Cross-lingual unsupervised sentiment classification with multi-view transfer learning. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 5759–5771, Online.

Aritra Ghosh, Himanshu Kumar, and P. S. Sastry. 2017. Robust loss functions under label noise for deep neural networks. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI)*, pages 1919–1925, San Francisco, CA.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, San Diego, CA.

Xiao Ling and Daniel S. Weld. 2012. Fine-grained entity recognition. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence (AAAI)*, Toronto, Canada.

Guiliang Liu, Xu Li, Mingming Sun, and Ping Li. 2020a. An advantage actor-critic algorithm with confidence exploration for open information extraction. In *Proceedings of the 2020 SIAM International Conference on Data Mining (SDM)*, pages 217–225, Cincinnati, OH.

Guiliang Liu, Xu Li, Jiakang Wang, Mingming Sun, and Ping Li. 2020b. Extracting knowledge from web text with monte carlo tree search. In *Proceedings of the Web Conference (WWW)*, pages 2585–2591, Taipei.

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 (ACL/IJCNLP)*, pages 2051–2064, Virtual Event.

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 (NAACL-HLT)*, pages 2407–2417, Minneapolis, MN.

Xiang Ren, Wenqi He, Meng Qu, Lifu Huang, Heng Ji, and Jiawei Han. 2016a. AFET: automatic fine-grained entity typing by hierarchical partial-label embedding. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1369–1378, Austin, TX.

Xiang Ren, Wenqi He, Meng Qu, Clare R. Voss, Heng Ji, and Jiawei Han. 2016b. Label noise reduction in entity typing by heterogeneous partial-label embedding. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 1825–1834, San Francisco, CA.

Sonse Shimaoka, Pontus Stenetorp, Kentaro Inui, and Sebastian Riedel. 2016. An attentive neural architecture for fine-grained entity type classification. In *Proceedings of the 5th Workshop on Automated Knowledge Base Construction (AKBC@NAACL-HLT)*, pages 69–74, San Diego, CA.

Sonse Shimaoka, Pontus Stenetorp, Kentaro Inui, and Sebastian Riedel. 2017. Neural architectures for fine-grained entity type classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 1271–1280, Valencia, Spain.

Mingming Sun, Xu Li, Xin Wang, Miao Fan, Yue Feng, and Ping Li. 2018. Logician: A unified end-to-end neural approach for open-domain information extraction. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining (WSDM)*, pages 556–564, Marina Del Rey, CA.

Daiki Tanaka, Daiki Ikami, Toshihiko Yamasaki, and Kiyoharu Aizawa. 2018. Joint optimization framework for learning with noisy labels. In *Proceedings of the 2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5552–5560, Salt Lake City, UT.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems (NIPS)*, pages 5998–6008, Long Beach, CA.

Qizhe Xie, Minh-Thang Luong, Eduard H. Hovy, and Quoc V. Le. 2020. Self-training with noisy student improves imagenet classification. In *Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10684–10695, Seattle, WA.

Ji Xin, Hao Zhu, Xu Han, Zhiyuan Liu, and Maosong Sun. 2018. Put it back: Entity typing with language model enhancement. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 993–998, Brussels, Belgium.

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 (NAACL-HLT)*, pages 773–784, Minneapolis, MN.

Peng Xu and Denilson Barbosa. 2018. Neural fine-grained entity type classification with hierarchy-aware loss. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 16–25, New Orleans, LA.

Mohamed Amir Yosef, Sandro Bauer, Johannes Hoffart, Marc Spaniol, and Gerhard Weikum. 2013. HYENA-live: Fine-grained online entity type classification from natural-language text. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL Demonstrations)*, pages 133–138, Sofia, Bulgaria.

Yue Zhang, Hongliang Fei, and Ping Li. 2021. Readsre: Retrieval-augmented distantly supervised relation extraction. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 2257–2262, Virtual Event, Canada.

Yue Zhang, Hongliang Fei, and Ping Li. 2022. End-to-end distantly supervised information extraction with retrieval augmentation. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 2449–2455, Madrid, Spain.

ACL 2023 Responsible NLP Checklist

A For every submission:

☐ A1. Did you describe the limitations of your work?
Left blank.

☐ A2. Did you discuss any potential risks of your work?
Left blank.

☐ A3. Do the abstract and introduction summarize the paper's main claims?
Left blank.

☐ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☐ Did you use or create scientific artifacts?

Left blank.

☐ B1. Did you cite the creators of artifacts you used?
Left blank.

☐ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
Left blank.

☐ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
Left blank.

☐ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
Left blank.

☐ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
Left blank.

☐ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
Left blank.

C ☐ Did you run computational experiments?

Left blank.

☐ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
Left blank.

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☐ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
Left blank.

☐ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
Left blank.

☐ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
Left blank.

D ☐ Did you use human annotators (e.g., crowdworkers) or research with human participants?

Left blank.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
Left blank.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
Left blank.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
Left blank.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
Left blank.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
Left blank.