Generative Entity Typing with Curriculum Learning

Siyu Yuan[♡1], **Deqing Yang**[♡2*], **Jiaqing Liang**[♡2], **Zhixu Li**[♠2] **Jinxi Liu**[♡1], **Jingyue Huang**[♡2], **Yanghua Xiao**[♠♣2*]

[♡]School of Data Science, Fudan University, Shanghai, China

[♠]Shanghai Key Laboratory of Data Science, School of Computer Science, Fudan University

[♣]Fudan-Aishu Cognitive Intelligence Joint Research Center

[1]{syyuan21,jxliu22}@m.fudan.edu.cn,

[2]{yangdeqing,liangjiaqing,zhixuli,jingyuehuang18,shawyh}@fudan.edu.cn

Abstract

Entity typing aims to assign types to the entity mentions in given texts. The traditional classification-based entity typing paradigm has two unignorable drawbacks: 1) it fails to assign an entity to the types beyond the predefined type set, and 2) it can hardly handle few-shot and zero-shot situations where many long-tail types only have few or even no training instances. To overcome these drawbacks, we propose a novel generative entity typing (GET) paradigm: given a text with an entity mention, the multiple types for the role that the entity plays in the text are generated with a pre-trained language model (PLM). However, PLMs tend to generate coarse-grained types after fine-tuning upon the entity typing dataset. In addition, only the heterogeneous training data consisting of a small portion of human-annotated data and a large portion of auto-generated but low-quality data are provided for model training. To tackle these problems, we employ curriculum learning (CL) to train our GET model on heterogeneous data, where the curriculum could be self-adjusted with the self-paced learning according to its comprehension of the type granularity and data heterogeneity. Our extensive experiments upon the datasets of different languages and downstream tasks justify the superiority of our GET model over the state-of-the-art entity typing models. The code has been released on https://github.com/siyuyuan/GET.

1 Introduction

Entity typing aims to assign types to mentions of entities from a predefined type set, which enables machines to better understand natural languages and benefit many downstream tasks, such as entity linking (Yang et al., 2019) and text classification (Chen et al., 2019b). Traditional entity typing approaches follow the classification paradigm to classify (assign) the entity into a predefined set of types, which have the following two unignorable drawbacks.

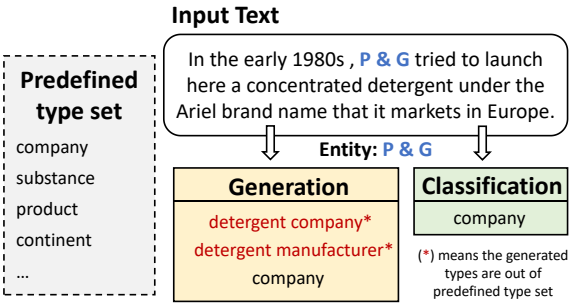

Figure 1: A toy example of entity typing through generation and classification paradigm, respectively.

1) *Closed Type Set*: The classification-based approaches fail to assign the entity to the types out of the predefined set. 2) *Few-shot Dilemma for Long-tail Types*: Although fine-grained entity typing (FET) and ultra-fine entity typing approaches can classify entities into fine-grained types, they can hardly handle few-shot and zero-shot issues. In fact, there are many long-tail types only having few or even no training instances in the datasets. For example, more than 80% types have less than 5 instances and 25% types even never appear in the training data from the ultra-fine dataset (Choi et al., 2018).

To address these drawbacks, in this paper, we propose a novel *generative entity typing* (GET) paradigm: given a text with an entity mention, the multiple types for the role that the entity plays in the text are generated by a pre-trained language model (PLM). Compared to traditional classification-based entity typing methods, PLM-based GET has two advantages. First, instead of a predefined closed type set, PLMs can generate more open types for entity mentions due to their strong generation capabilities. For example, in Figure 1, fine-grained types such as “large detergent company" and “large detergent manufacturer" can be generated by PLMs for entity *P&G*, which contain richer semantics but are seldom included by

*Corresponding authors.

Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, pages 3061 - 3073
December 7-11, 2022

a predefined type set. Second, PLMs are capable of conceptual reasoning and handling the few-shot and zero-shot dilemma (Hwang et al., 2021), since massive knowledge has been learned during their pre-training.

However, it is nontrivial to realize PLM-based GET due to the following challenges: 1) Entity typing usually requires generating fine-grained types with more semantics, which are more beneficial to downstream tasks. However, PLMs are biased to generate high-frequency vocabulary in the corpus due to their primary learning principle based on statistical associations. As a result, a typical PLM tends to generate high-frequent but coarse-grained types even if we carefully finetune the PLM on the fine-grained entity typing dataset (refer to Figure 5 in Section 4). Therefore, how to guide a PLM to generate high-quality and fine-grained types for entities is crucial. 2) It is costly for humans to annotate a great number of samples with fine-grained types. Therefore, most existing works adopt heterogeneous data consisting of a small portion (less than 10%) of human-annotated data and a large portion (more than 90%) of auto-generated low-quality data (e.g., by distant supervision), which greatly hurts the performance of entity typing models (Gong et al., 2021). How to train a PLM to generate desirable types on these low-quality heterogeneous data is also challenging.

The difficulty of using PLMs to generate high-quality fine-grained types based on the low-quality heterogeneous training data motivates us to leverage the idea from curriculum learning (CL) (Bengio et al., 2009), which better learns heterogeneous data by ordering the training samples based on their quality and difficulty (Kumar et al., 2019). In this paper, we propose a CL-based strategy to train our GET model. Specifically, we first define a fixed *curriculum instruction* and partition the training data into several subsets according to the granularity and heterogeneity of samples for model training. Based on the curriculum instruction, CL can control the order of using these training subsets from coarse-grained and lower-quality ones to fine-grained and higher-quality ones. However, a fixed curriculum ignores the feedback from the training process. Thus, we combine the predetermined curriculum with self-paced learning (SPL) (Kumar et al., 2010), which can enforce the model dynamically self-adjusting to the actual learning order according to the training loss. In this way, our CL-based GET model can make the learning process move towards a better global optimum upon the heterogeneous data to generate high-quality and fine-grained types. Our contributions in this paper are summarized as follows:

- To the best of our knowledge, our work is the first to propose the paradigm of generative entity typing (GET).
- We propose to leverage curriculum learning to train our GET model upon heterogeneous data, where the curriculum can be self-adjusted with self-paced learning.
- Our extensive experiments on the data of different languages and downstream tasks justify the superiority of our GET model.

2 Related Work

Classification-based Entity Typing The traditional classification-based entity typing methods can be categorized into three classes. 1) *Coarse-grained entity typing* methods (Weischedel and Brunstein, 2005; Tokarchuk et al., 2021) assign mentions to a small set of coarse types; 2) *Fine-grained entity typing (FET)* methods (Yuan and Downey, 2018; Onoe et al., 2021) classify mentions into more diverse and semantically richer ontologies; 3) *Ultra-fine entity typing* methods (Choi et al., 2018; Ding et al., 2021; Dai et al., 2021) use a large open type vocabulary to predict a set of natural-language phrases as entity types based on texts. However, FET and ultra-fine entity typing methods hardly perform satisfactorily due to the huge predefined type set. They also hardly handle few-shot and zero-shot issues. Comparatively, our GET model can generate high-quality multi-granularity types even beyond the predefined set for the given entity mentions.

Concept Acquisition Concept acquisition is very related to entity typing which also aims to obtain the types for the given entities, since entity types are often recognized as concepts. Concept acquisition can be categorized into the extraction-based or generation-based scheme. The extraction scheme cannot acquire concepts not existing in the given text (Chen et al., 2019a; Yang et al., 2020). The existing approaches of concept generation (Zeng et al., 2021) focus on utilizing the existing concept taxonomy or knowledge bases to generate concepts but neglect to utilize the large corpus. Our GET model can also achieve text-based concept genera-

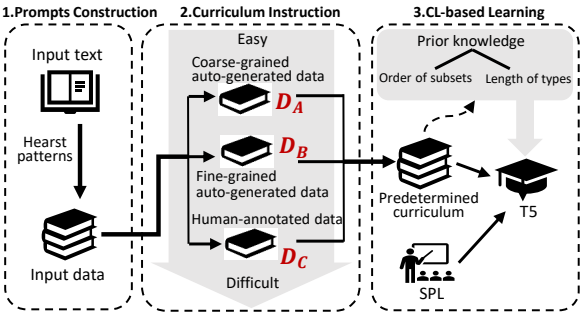

Figure 2: Our PLM-based GET framework trained with curriculum learning.

tion.

Curriculum Learning According to the curriculum learning (CL) paradigm, a model is first trained with the easier subsets or subtasks, and then the training difficulty is gradually increased (Bengio et al., 2009) to improve model performance in difficult target tasks, such as domain adaption (Wang et al., 2021) and training generalization (Huang and Du, 2019). The existing CL methods can be divided into predefined CL (PCL) (Bengio et al., 2009) and automatic CL (ACL) (Kumar et al., 2010). PCL divides the training data by the difficulty level with prior knowledge, while ACL, such as self-paced learning (SPL), measures the difficulty according to its losses or other models.

3 Methodology

In this section, we first formalize our task in this paper and overview the framework of our GET model. Then, we introduce the details of model implementation.

3.1 Task Formalization

Given a piece of text X and an entity mention M within it, the task of **generative entity typing (GET)** is to generate multiple types $TS = \{T_1, T_2, ..., T_K\}$, where each $T_k (1 \leq k \leq K)$ is a type for M w.r.t. the context of X.

3.2 Framework

As most of the previous entity typing models (Choi et al., 2018; Lee et al., 2020; Gong et al., 2021), our GET model is also trained upon the heterogeneous data consisting of a small portion of human-annotated data and a large portion of auto-generated data, due to the difficulty and high cost of human annotation. We will introduce how to obtain our auto-generated data in Section 4.1. The framework of our model learning includes the following three steps, as shown in Figure 2.

1. *Prompt Construction*: To better leverage the knowledge obtained from the pre-training of PLM, we employ the prompt mechanism (Liu et al., 2021a) to guide the learning of our PLM-based GET model;

2. *Curriculum Instruction*: As a key component of CL, the curriculum instruction is responsible for measuring the difficulty of each sample in the heterogeneous training data, and then designing a suitable curriculum for the model training process;

3. *CL-based Learning*: In this step, our PLM-based GET model is trained with the designed curriculum, which is capable of adjusting its learning progress dynamically through self-paced learning (SPL).

3.3 Prompt Construction

To generate the types of given entities by a PLM, we construct the prompts in cloze format from the Hearst patterns listed in Table 1. Specifically, each input text X including an entity mention M is concatenated with a cloze prompt constructed with M, and the PLM is asked to fill the blank within the cloze prompt. Recall the example in Figure 1, the original text "*In the early 1980s, P & G tried to launch here a concentrated detergent under the Ariel brand name that it markets in Europe*" can be concatenated with a cloze prompt such as "P & G is a ____" to construct an input prompt for the PLM, which predicts "large detergent company", "large detergent manufacturer" and "company" as the types for *P & G* to fill the blank.

M is a ____	____ such as M
M is one of ____	____ especially M
M refers to ____	____, including M
M is a member of ____	

Table 1: Prompts constructed from Hearst patterns.

3.4 Curriculum Instruction

Curriculum instruction is the core issue of CL, which requires estimating the difficulty of samples in terms of model learning, to decide the order of using samples for model training.

For our specific PLM-based GET model, we argue that the difficulty of a sample in terms of

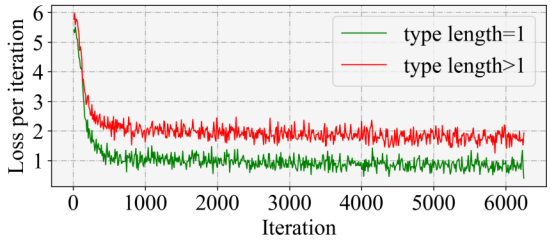

Figure 3: The landscapes of CE loss comparison for the two training subsets with type length=1 and type length≥2.

model learning greatly depends on the granularity of its type, which could be roughly measured by the length of the type term. To prove this, we generate two subsets of training samples in the same size from the auto-generated data according to their type length: one subset with type length=1, and the other subset with type length≥2[1]. Then, we train a typical PLM, T5 (Raffel et al., 2019) for one epoch in the two subsets and record the landscapes of loss. As observed in Figure 3, the length≥2 subset has lower converge and higher cross-entropy (CE) losses than the length=1 subset when training converges, which shows that it is more difficult for the PLM to fit the training samples of fine-grained types.

Based on this observation, we partition the auto-generated data into two subsets, i.e., the subset with one-word types (denoted as D_A), and the subset with types of more than one word (denoted as D_B), and D_A is used for model training earlier than D_B. The human-annotated data (denoted as D_C) is finally used, as it usually contains many ultra fine-grained types annotated by human annotators, which is harder for model learning. For easier presentation later, we denote the whole training data as $D = D_A \bigcup D_B \bigcup D_C = \{< X^{(i)}, M^{(i)}, TS^{(i)} >\}_{i=1}^{N}$, where N is the training sample size and a sample is denoted as $D_k^{(i)} = < X^{(i)}, M^{(i)}, T_k^{(i)} >$. Based on the fixed curriculum instruction, CL can control the order in which data are used for model training, i.e., from D_A to D_B to D_C.

3.5 CL-based Learning

T5 Backbone To meet the cloze format and generate more fine-grained types, we choose T5 (Raffel et al., 2019) as the backbone PLM of our GET model. T5 is an encoder-decoder pre-trained model, which can focus on the entire text and allow multiple words to fill in the blanks. To train the T5, in our settings, we define the loss function of sample $D_k^{(i)}$ as,

$$L\big(D_k^{(i)}\big) = L_{CE}\big(T_k^{(i)}, f(X^{(i)}, \boldsymbol{\theta}, M^{(i)})\big) \quad (1)$$

where L_{CE} is a CE loss function that calculates the cost between the ground truth type $T_k^{(i)}$ and the predicted type $f(X^{(i)}, \boldsymbol{\theta}, M^{(i)})$. $\boldsymbol{\theta}$ denote the model parameters.

SPL-based Training Process Our model training can be performed according to the above predefined curriculum, but the feedback from the learning process is inevitably ignored, which may lead to divergent solutions (Jiang et al., 2015). As an alternative, we adopt self-paced learning (SPL) to enforce the model to self-adjust the curriculum according to the feedback from the training loss. Formally, we define the objective of our CL as,

$$\min_{\boldsymbol{\theta},\boldsymbol{v}} E(\boldsymbol{\theta}, \boldsymbol{v}; \lambda) = \sum_{i=1}^{N}\sum_{k=1}^{K^{(i)}} v_k^{(i)} L\big(D_k^{(i)}\big) + g(\boldsymbol{v}; \lambda) \quad (2)$$

where the binary variable $v_k^{(i)} \in [0, 1]$ indicates whether sample $D_k^{(i)}$ should be incorporated into the calculation of the objective. Specifically,

$$v_k^{(i)} = \begin{cases} 1 & L\big(D_k^{(i)}\big) < \lambda \\ 0 & L\big(D_k^{(i)}\big) \geq \lambda \end{cases} \quad (3)$$

and

$$g(\boldsymbol{v}; \lambda) = -\lambda \sum_{i=1}^{N}\sum_{k=1}^{K^{(i)}} v_k^{(i)}, \quad (4)$$

where $K^{(i)}$ is the number of types for $< X^{(i)}, M^{(i)}, TS^{(i)} >$, and λ is the "age" of SPL to control the learning pace. The regularizer $g(;)$ is the binary self-paced function used to avoid overfitting (Jiang et al., 2014).

In the training process, "easy" samples with small losses are used first for training. We update $\lambda = \mu\lambda$ to increase λ gradually, where $\mu > 1$ is the step size. With the growth of λ, more samples with larger losses are gradually incorporated into model training to obtain a more "mature" model.

Prior Knowledge to Optimize SPL As mentioned in Section 3.4, we expect that the model is trained orderly by the three subsets according to the predetermined curriculum and generates more

[1]Here we chose length=1 and length≥2 to divide the training data since most of the types are no longer than 2

fine-grained types. However, the order of using the data for model training totally depends on the loss function (Eq. 1) in SPL. Thus, SPL is limited in incorporating predetermined curriculum and the type granularity into learning. Therefore, we treat the predetermined curriculum and the type granularity as prior knowledge to optimize SPL. Specifically, we increase the weight of the samples with fine-grained types to let the model pay more attention to these data and assign each subset with different weights to ensure that the training process is executed according to the predetermined curriculum. In particular, given the sample $D_k^{(i)}$, we define its weight as

$$w(D_k^{(i)}) = length(T_k^{(i)}) * \gamma(D_k^{(i)}) \quad (5)$$

where

$$\gamma(D_k^{(i)}) = \begin{cases} 1 & \text{if } D_k^{(i)} \in D_A, \\ 2 & \text{if } D_k^{(i)} \in D_B, \\ 3 & \text{if } D_k^{(i)} \in D_C. \end{cases} \quad (6)$$

Then, the loss function L in Eq. 1 is updated as

$$L_{CE}\big(T_k^{(i)}, f(X^{(i)}, \boldsymbol{\theta}, M^{(i)})\big) * w\big(D_k^{(i)}\big), \quad (7)$$

which indicates that a sample with a large weight (more difficult) can be incorporated later into the training process since its $v_k^{(i)}$ is more likely to be 0 according to Eq. 3. We adopt an alternative convex search (ACS) to realize SPL, of which the algorithm is shown in Appendix A. We use Adam (Kingma and Ba, 2015) to update the model parameters.

3.6 Type Generation

When the PLM in our GET model has been trained through the aforementioned CL-based learning process, we use it to generate the types for the given entity mention. Specifically, we let the PLM fill in the blank of the input text. To obtain more diverse candidate types, we apply the beam search (Reddy et al., 1977) with beam size as b, and select the b most probable candidates. Then, we reserve the types with confidence scores bigger than 0.5.

4 Experiments

In this section, we verify the advantages of our GET model over the classification-based entity typing models through our experiments. We also explore the role of CL in guiding PLM-based type generation with different language data. We further display the effectiveness of our generated entity types in two downstream tasks.

4.1 Datasets

As we mentioned before, due to the expensive manual labeling of fine-grained types, the training dataset consists of a large portion of auto-generated data and a small portion of human-annotated data. Since PLMs have more difficulty fitting the training samples of fine-grained types (elaborated in Figure 3 in Section 3.4), we partition the auto-generated data into two subsets, and denote the subset with one-word types as D_A while the other as D_B. Furthermore, human-annotated data with ultra fine-grained types is denoted as D_C.

Auto-generated Data The auto-generated data used in our model is obtained from the abstracts of entities on Wikipedia (Vrandečić and Krötzsch, 2014). Specifically, we collect the abstract texts and their hyperlinks pointing to the web pages of mentioned entities, from which the type labels of these mentioned entities can be obtained. In this way, the obtained type labels are more consistent with the contexts of entities, and thus of much higher quality than those auto-generated with distant supervision (Gong et al., 2021).

To construct D_A and D_B from the auto-generated data, we collect 100,000 Chinese and English abstracts from Wikipedia, from which we randomly select 500 samples as our test set, and the rest are used as the training set. Then we split the training set into two subsets D_A and D_B, according to the length of the types as mentioned in Section 3.4.

Human-annotated Data To demonstrate that GET is superior to the classification-based approaches, we collect the human-annotated data from four different entity type datasets. The statistics of them are listed in Table 2. We compare our model with baselines on BNN, FIGER and Ultra-fine to demonstrate the superior performance of GET on entity typing. GT dataset is used to evaluate the effectiveness of CL upon the texts of different languages and heterogeneous data. Please note that we do sample the test set to reduce the cost of manual evaluation on assessing whether the newly-generated types are correct. However, the results from the baselines and our model are evaluated in the same test instances.

Dataset	Type	Lang.	Size of D3	Size of test set
BNN (Weischedel and Brunstein, 2005)	Coarse-grained	EN	10,000	500
FIGER (Shimaoka et al., 2016)	Fine-grained	EN	10,000	278
Ultra-Fine (Choi et al., 2018)	Ultra fine-grained	EN	5,500	500
GT (Lee et al., 2020)	Multilingual	EN	4,750	250
		ZH	4,750	250

Table 2: The statistic of different entity typing datasets.

Model	BNN				FIGER			
	CT #	Prec.	R-Recall	R-F1	CT #	Prec.	R-Recall	R-F1
Zhang et al. (2018)	555	58.10%	50.49%	54.03%	348	62.00%	49.85%	55.26%
Lin and Ji (2019)	534	55.90%	48.58%	51.98%	353	62.90%	50.57%	56.07%
Xiong et al. (2019)	558	58.40%	50.75%	54.31%	350	62.30%	50.09%	55.53%
Ali et al. (2020)	697	73.00%	63.43%	67.88%	399	**71.00%**	57.08%	63.29%
Chen et al. (2020)	718	75.20%	65.35%	69.93%	388	69.10%	55.56%	61.59%
Zhang et al. (2021)	732	76.70%	66.65%	71.32%	394	70.10%	56.36%	62.48%
Li et al. (2021)	668	69.90%	60.74%	65.00%	397	70.60%	56.76%	62.93%
Ours	**875**	**82.30%**	**79.62%**	**80.94%**	**444**	66.20%	**63.52%**	**64.83%**

Table 3: Comparison results of different approaches on the sample test set in coarse-grained and fine-grained entity typing dataset.

4.2 Evaluation Metrics

The detailed information about the baselines and some experiment settings is shown in Appendix B. Please note that the baselines on BNN and FIGER are different from those on Ultra-Fine.

For BNN and FIGER, we only reserve the type in the predefined type set with the highest probability predicted by the model since there is only one golden label in the dataset. Ultra-fine entity typing aims to predict a set of natural-language phrases that describe the types of entity mentions based on texts. Therefore, we reserve the types with the probability bigger than 0.5 for all models. Please note that previous work adopts a classification paradigm, while our GET model can generate new types not existing in the predefined type set. Therefore, annotators are invited to assess whether the generated types are correct. The annotation detail is shown in Appendix B.4.

We record the number of correct types assigned to the entity mentions (CT #) and strict macro-averaged precision (Prec.). Obviously, it is impossible to know all correct types generated based on the input text in advance due to incalculable search space. Therefore, we report the relative recall. Specifically, suppose CTs # is the total number of new types obtained by all models. Then, the relative recall (R-Recall) is calculated as CT # divided by CTs #. Accordingly, the relative F1 (R-F1) can also be calculated with Prec. and R-Recall. In addition, we also record the average length of types (Len.) under different training strategies to investigate the effectiveness of CL and prior knowledge.

4.3 Overall Comparison Results

The comparison results on traditional entity typing and Ultra-fine entity typing are shown in Table 3 and Table 4 [2]. The tables show that our model (Ours) achieves consistent performance improvements on these datasets. For BNN, our model significantly improves Prec. and covers more entity types. For FIGER, our model generates more types than other baselines. For Ultra-Fine, the existing models based on the classification paradigm are extremely difficult to select the appropriate types from the large predefined type set. Comparatively, our GET model has no classification constraint since it transforms multi-classification into a generation paradigm that is more suitable for PLMs. Therefore, our model greatly improves the precision of Ultra-Fine and covers more types of entities.

We further display the capability of our model to generate new types beyond the predefined type set, as shown in Table 5. In the table, MaNew (Macro-

[2]The statistical significance test and data size analysis are provided in Appendix B.5

Model	Ultra-Fine			
	CT #	Prec.	R-Recall	R-F1
Xiong et al. (2019)	782	50.30%	24.28%	32.75%
Onoe and Durrett (2019) ELMo	884	51.50%	27.44%	35.81%
Onoe and Durrett (2019) BERT	884	51.60%	27.44%	35.83%
López and Strube (2020)	915	43.40%	28.41%	34.34%
Onoe et al. (2021)	1039	52.80%	32.26%	40.05%
Liu et al. (2021b)	1042	54.50%	32.35%	40.60%
Dai et al. (2021)	1213	53.60%	37.66%	44.24%
Ours	**1275**	**87.10%**	**39.58%**	**54.43%**

Table 4: Comparison results of different approaches on the sample test set in Ultra-fine entity typing dataset.

Dataset	MaNew	MiNew	R.New
BNN	4	100	11.61%
FIGER	25	137	26.81%
Ultra-Fine	73	543	42.14%

Table 5: The number and ratio of new types generated by our model on different datasets.

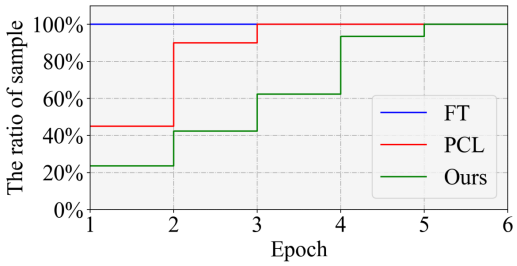

Figure 4: The ratio of training samples of different learning strategies in each epoch.

New) is the total number of generated types beyond the predefined type set, MiNew (Micro-New) is the total number of generated types beyond the human-annotated type set (i.e., golden labeled set TS) of each instance, and R.New is the ratio of new generated types per sample. The listed results are counted upon the test sets in different datasets, from which we find that our model can generate abundant types that are not in the golden label set, thereby increasing the diversity of entity types.

4.4 Effectiveness of CL

We further compare our model with the following ablated variants to verify the effectiveness of CL. **FT** is fine-tuned directly with training data without CL; **PCL** adopts Baby Step (Bengio et al., 2009) instead of SPL, which inputs the subsets into the model in turn according to a fixed curriculum, but ignores the feedback from the learning process;

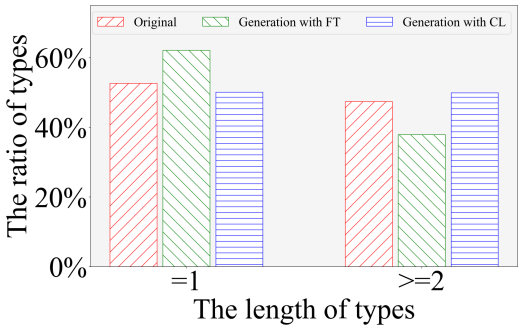

Figure 5: The ratio of the types of length=1 and length $\geq$ 2 in the original training dataset, and the types generated by our GET model with fine-tuning (FT) or curriculum learning (CL).

SPL w/o PK adopts SPL but ignores prior knowledge in training.

In order to demonstrate the performance of the compared models on Chinese and English data, we only consider D_C of GT in our ablation studies. Since the models are designed toward heterogeneous training data, we investigate their performance on both the auto-generated test set and human-annotated test set (in GT), which is displayed in Table 6. Please note that we only report the Prec. and R-F1 due to the limited space. Based on the results, the superiority of PCL and SPL over FT verifies the advantage of CL over the general training strategy, while the superiority of SPL w/o PK over PCL verifies the effectiveness of SPL. Furthermore, our GET model performs well on type generation upon abstract texts (auto-generated data) and common free texts (human-annotated data).

To explore the reason for the advantage of SPL, we record the ratio of incorporated training samples in each epoch. As shown in Figure 4, SPL gradually incorporates the whole training data to train the model. The training on the former subsets can

Model	Dataset	Chinese				English			
		CT #	Prec.	R-F1	Len.	CT #	Prec.	R-F1	Len.
FT	Auto-generated data	690	84.46%	70.81%	2.80	870	75.85%	52.87%	1.48
PCL		646	91.76%	70.37%	2.75	864	85.97%	54.87%	1.32
SPL w/o PK		672	**92.18%**	72.22%	2.75	900	87.12%	56.66%	1.54
Ours		**714**	90.04%	**74.18%**	**2.86**	**928**	**87.14%**	**57.84%**	**1.62**
FT	Human-annotated data	383	72.54%	53.98%	**2.65**	352	84.82%	48.82%	1.72
PCL		370	77.24%	54.01%	2.64	**375**	88.03%	51.62%	1.69
SPL w/o PK		383	78.64%	55.59%	2.61	370	90.46%	51.53%	1.74
Ours		**409**	**83.64%**	**59.28%**	2.63	373	**90.75%**	**51.88%**	**1.82**

Table 6: Performance comparisons of our model and its variants on the auto-generated and human-annotated test set.

be regarded as a pre-training process that helps model optimization and regularizes the training on the latter subsets. Thus, SPL can better guide the model to find a global minimum loss and make it more generalizable.

4.5 Effectiveness of Prior Knowledge

From the Table 6, we also find that Ours can generate more fine-grained types than SPL w/o PK. Without the prior knowledge, SPL only relies on the self-judgment of the model and treats all the selected samples equally, which ignores the data heterogeneity and type granularity during training and harms the model performance.

As shown in Figure 5, compared to the original training dataset, there are more coarse-grained types (length=1) than fine-grained types (length$\geq$2) generated by the directly fine-tuned GET model (Generation with FT), while the GET model with CL can generate more fine-grained types of the almost same ratio as the coarse-grained types. It is because that the prior knowledge about the type length is considered to re-weight the importance of samples, making the model pay more attention to fine-grained types. Thus, more fine-grained and high-quality types are generated. Based on these results, we believe that combining prior knowledge with SPL is an excellent way to optimize CL.

We also explore the influence of different λ and μ which is shown in Appendix B.6.

4.6 Applications

We further conduct experiments on the task of short text classification and entity linking to prove that the types generated by our model can promote the downstream tasks.

Short Text Classification. Existing short text classification approaches (Chen et al., 2019b) directly use KG as external knowledge to improve model performance. However, how to choose the context-consistent types for the roles that the entities play in the text relation is still a problem, which may lead to unsatisfactory results. GET can generate context-consistent types for entities, and thus it can be adopted to promote the classification performance. We conduct our experiments in the NLPCC2017 dataset [3], the Chinese news title dataset with 18 classes (e.g., entertainment, game, food). We first use an NER model to identify entities in the text and directly apply our model upon NLPCC2017 dataset to generate types for the entities. Then we choose Bi-LSTM to achieve classification. we also collect corresponding types of entities in the representative KG *CN-DBpedia* for comparison. The results in Table 7 show that external knowledge enhances the classification performance, and the types generated by our GET model

Method	Prec.	Recall	F1
No type	72.92%	72.70%	72.47%
types (KG)	73.99%	73.17%	73.30%
types (Gen.)	**74.51%**	**73.41%**	**73.53%**

Table 7: Performance of short text classification based on Bi-LSTM without/with different external knowledge on NLPCC2017 dataset.

Dataset	Method	F1
AIDA CoNLL-YAGO	triples (KG.)	94.58%
	triples (Gen.)	**94.92%**
ACE 2014	triples (KG)	89.74%
	triples (Gen.)	**90.54%**

Table 8: Performance of entity linking model DCA-SL with different external knowledge.

[3] http://tcci.ccf.org.cn/conference/2017/taskdata.php

Input Text	Golden	Generated
He was capped 42 times and scored 8 goals for **Sweden**, and he played at the 2002 FIFA World Cup	nation	nation, nordic country, scandinavian country
The Audit Bureau of Circulations was formed in 1914 to verify publication circulation figures and track media rates	administration, organization	organization, government agency, investigative service
Chapman retired from playing **hockey** after the 1943 to 1944 hockey season	tournament, event,contest, game,activity, sport,hockey	sport, contact sport, team sport, winter sport

Table 9: Ultra-fine entity typing examples with the corresponding golden labels and generated types. Entity mentions are in bold and underlined.

are more effective than those directly obtained from KG.

Entity Linking. The representative entity linking model DCA-SL (Yang et al., 2019) adopts the entity description and triples in KG as an external knowledge to enhance the performance of the model in the entity linking task. To prove that the types generated by our model are of high quality, we first adopt our model to generate types for entities based on texts. Then we replace the types in the original triples in KGs with the types we generated. From Table 8 we find that the generated types by our model can improve the entity lining performance of DCA-SL effectively, indicating that the generated types are of high quality and meaningful.

5 Conclusion

In this paper, we propose a novel generative paradigm for entity typing, which employs a generative PLM trained with curriculum learning. Compared with the traditional classification-based entity typing methods, our generative entity typing (GET) model can generate new types beyond the predefined type set for given entity mentions. Our extensive experiments on several benchmark datasets justify that the curriculum learning with SPL and the prior knowledge of type length and subset order help our model generate more high-quality fine-grained types.

6 Limitations

Although we have proven that our GET model can generate high-quality and new types beyond the predefined type set for given entity mentions, it also has some limitations. In this section, we analyze these limitations and hopefully advance future work in GET.

6.1 Uncontrolled Generation

To delve into the model performance, we compare the types generated by our approach with the golden type labels in Ultra-fine entity typing datasets. Table 9 lists three examples with the correct types generated by our model and the golden labeled type set of the entities in the Ultra-Fine dataset. The first example shows that our model can generate more fine-grained types which may not appear in the golden labeled set. The second and third examples demonstrate that although our model can generate new concepts, it may ignore some correct types in the golden label set, e.g., “administration". However, enforcing the model by constraint decoding to generate the types in the predefined type set may compromise the flexibility of our model to generate new concepts. Therefore, we hope that future work can handle this dilemma with better methods.

6.2 Type Selection

As mentioned in Sec. 3.6, T5 adopts beam search to generate the b most probable types with confidence scores. Then we reserve the types with confidence scores larger than the selection threshold. However, it can hardly achieve a satisfactory balance to reserve the types by choosing a specific threshold to directly truncate the output of T5. If we select a relatively big threshold, we can get more accurate types but may lose some correct types. If the recall is preferred, precision might be hurt. Therefore, we suggest that future work consider how to achieve a better trade-off between precision and recall.

Acknowledgement

We would like to thank the anonymous reviewers for their valuable comments and suggestions for this work. This work is supported by the National Natural Science Foundation of China (No.62072323), Shanghai Science, the Science and Technology Commission of Shanghai Municipality Grant (No. 22511105902) and Technology Innovation Action Plan (No.21511100401& 22511104700).

References

Muhammad Asif Ali, Yifang Sun, Bing Li, and Wei Wang. 2020. Fine-grained named entity typing over distantly supervised data based on refined representations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7391–7398.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48.

Jiangjie Chen, Ao Wang, Haiyun Jiang, Suo Feng, Chenguang Li, and Yanghua Xiao. 2019a. Ensuring readability and data-fidelity using head-modifier templates in deep type description generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2036–2046, Florence, Italy. Association for Computational Linguistics.

Jindong Chen, Yizhou Hu, Jingping Liu, Yanghua Xiao, and Haiyun Jiang. 2019b. Deep short text classification with knowledge powered attention. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6252–6259.

Tongfei Chen, Yunmo Chen, and Benjamin Van Durme. 2020. Hierarchical entity typing via multi-level learning to rank. *arXiv preprint arXiv:2004.02286*.

Eunsol Choi, Omer Levy, Yejin Choi, and Luke Zettlemoyer. 2018. Ultra-fine entity typing. *arXiv preprint arXiv:1807.04905*.

Hongliang Dai, Yangqiu Song, and Haixun Wang. 2021. Ultra-fine entity typing with weak supervision from a masked language model. *arXiv preprint arXiv:2106.04098*.

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*.

Rotem Dror, Gili Baumer, Segev Shlomov, and Roi Reichart. 2018. The hitchhiker's guide to testing statistical significance in natural language processing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1383–1392, Melbourne, Australia. Association for Computational Linguistics.

Hongyu Gong, Alberto Valido, Katherine M Ingram, Giulia Fanti, Suma Bhat, and Dorothy L Espelage. 2021. Abusive language detection in heterogeneous contexts: Dataset collection and the role of supervised attention. *arXiv preprint arXiv:2105.11119*.

Yuyun Huang and Jinhua Du. 2019. Self-attention enhanced cnns and collaborative curriculum learning for distantly supervised relation extraction. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pages 389–398.

Jena D. Hwang, Chandra Bhagavatula, Ronan Le Bras, Jeff Da, Keisuke Sakaguchi, Antoine Bosselut, and Yejin Choi. 2021. Comet-atomic 2020: On symbolic and neural commonsense knowledge graphs. In *AAAI*.

Lu Jiang, Deyu Meng, Teruko Mitamura, and Alexander G Hauptmann. 2014. Easy samples first: Self-paced reranking for zero-example multimedia search. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 547–556.

Lu Jiang, Deyu Meng, Qian Zhao, Shiguang Shan, and Alexander G Hauptmann. 2015. Self-paced curriculum learning. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Gaurav Kumar, George Foster, Colin Cherry, and Maxim Krikun. 2019. Reinforcement learning based curriculum optimization for neural machine translation. *arXiv preprint arXiv:1903.00041*.

M Pawan Kumar, Benjamin Packer, and Daphne Koller. 2010. Self-paced learning for latent variable models. In *NIPS*, volume 1, page 2.

Chin Lee, Hongliang Dai, Yangqiu Song, and Xin Li. 2020. A chinese corpus for fine-grained entity typing. *arXiv preprint arXiv:2004.08825*.

Jinqing Li, Xiaojun Chen, Dakui Wang, and Yuwei Li. 2021. Enhancing label representations with relational inductive bias constraint for fine-grained entity typing.

Ying Lin and Heng Ji. 2019. An attentive fine-grained entity typing model with latent type representation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6197–6202.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2021a. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *arXiv preprint arXiv:2107.13586*.

Qing Liu, Hongyu Lin, Xinyan Xiao, Xianpei Han, Le Sun, and Hua Wu. 2021b. Fine-grained entity typing via label reasoning. *arXiv preprint arXiv:2109.05744*.

F López and M. Strube. 2020. A fully hyperbolic neural model for hierarchical multi-class classification.

Yasumasa Onoe, Michael Boratko, Andrew McCallum, and Greg Durrett. 2021. Modeling fine-grained entity types with box embeddings. *arXiv preprint arXiv:2101.00345*.

Yasumasa Onoe and Greg Durrett. 2019. Learning to denoise distantly-labeled data for entity typing. *arXiv preprint arXiv:1905.01566*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.

D Raj Reddy et al. 1977. Speech understanding systems: A summary of results of the five-year research effort. department of computer science.

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

Evgeniia Tokarchuk, David Thulke, Weiyue Wang, Christian Dugast, and Hermann Ney. 2021. Investigation on data adaptation techniques for neural named entity recognition. *arXiv preprint arXiv:2110.05892*.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85.

X. Wang, Y. Chen, and W. Zhu. 2021. A survey on curriculum learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PP(99).

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

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. *arXiv preprint arXiv:1903.02591*.

Xi Yang, Jiang Bian, William R Hogan, and Yonghui Wu. 2020. Clinical concept extraction using transformers. *Journal of the American Medical Informatics Association*, 27(12):1935–1942.

Xiyuan Yang, Xiaotao Gu, Sheng Lin, Siliang Tang, Yueting Zhuang, Fei Wu, Zhigang Chen, Guoping Hu, and Xiang Ren. 2019. Learning dynamic context augmentation for global entity linking. *arXiv preprint arXiv:1909.02117*.

Zheng Yuan and Doug Downey. 2018. Otyper: A neural architecture for open named entity typing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Qingkai Zeng, Jinfeng Lin, Wenhao Yu, Jane Cleland-Huang, and Meng Jiang. 2021. Enhancing taxonomy completion with concept generation via fusing relational representations. *arXiv preprint arXiv:2106.02974*.

Haoyu Zhang, Dingkun Long, Guangwei Xu, Muhua Zhu, Pengjun Xie, Fei Huang, and Ji Wang. 2021. Learning with noise: improving distantly-supervised fine-grained entity typing via automatic relabeling. In *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, pages 3808–3815.

Sheng Zhang, Kevin Duh, and Benjamin Van Durme. 2018. Fine-grained entity typing through increased discourse context and adaptive classification thresholds. *arXiv preprint arXiv:1804.08000*.

A CL Algorithm

We adopt an alternative convex search (ACS) to realize the SPL of our model training. As shown in Algorithm 1, we use Adam to update the model parameters.

Algorithm 1 Self-paced Learning with prior knowledge for types generation.

Input: Input training data D, predetermined curriculum γ, self-paced function g, step size μ, pre-training learning parameters $\boldsymbol{\theta}$, maximum number of iterations T and the number of training epoch EP

Output: Model parameters $\hat{\boldsymbol{\theta}}$

Initialize $\boldsymbol{v}$ and λ;
Derive prior knowledge according to γ and type length;
while $ep < EP$ **do**
  while $t < T$ **do**
    Compute the objective function $E(\boldsymbol{\theta_t}, \boldsymbol{v}; \lambda)$ with prior knowledge;
    Update $\boldsymbol{\theta_{t+1}} = Adam(E(\boldsymbol{\theta_t}, \boldsymbol{v}; \lambda))$;
    $t = t + 1$;
  end while
  Record $\hat{\boldsymbol{\theta}} = \boldsymbol{\theta_T}$
  Record $\boldsymbol{v} = argmin_{\boldsymbol{v}}(E(\hat{\boldsymbol{\theta}}, \boldsymbol{v}; \lambda))$
  $\lambda = \mu\lambda$;
  $ep = ep + 1$;
end while
return $\hat{\boldsymbol{\theta}}$

B Experiment Detail

B.1 Baselines of Traditional Entity Typing

Upon traditional entity typing dataset, namely FIGER and BNN, we compare our model with following baselines:

- Zhang et al. (2018): This approach uses a neural architecture to learn a distributional semantic representation to classify.
- Lin and Ji (2019): This approach proposes a two-step mention-aware attention mechanism to enable the model to focus on the important words in mentions and contexts to improve type classification performance.
- Xiong et al. (2019): This approach utilizes a graph propagation layer to capture label correlations for type classification.
- Ali et al. (2020): This method adopts edge-weighted attentive graph convolution network to refine the noisy mention representations.
- Chen et al. (2020): Under the undefined case, this approach does not modify the labels in the dataset.
- Zhang et al. (2021): This approach utilizes a probabilistic automatic relabeling method that treats all training samples uniformly to handle noisy samples.
- Li et al. (2021): This approach proposes a novel method based on a two-phase graph network for the Fine-Grained Entity Typing task to enhance the label representations via imposing the relational inductive biases of instance-to-label and label-to-label.

B.2 Baselines of Ultra-Fine Entity Typing

For Ultra-Fine dataset, we compare our model with the following baselines:

- Onoe and Durrett (2019): This approach adopts ELMo and BERT as the encoder to fine-tune on the crowdsourced train split or raw and denoised distantly-labeled data.
- López and Strube (2020): This approach proposes a fully hyperbolic model for multi-class multi-label classification, which performs all operations in hyperbolic space.
- Onoe et al. (2021): This approach adopts a BERT-based model with box embeddings to capture latent type hierarchies for type classification.
- Liu et al. (2021b): This approach discovers and exploits label dependencies knowledge entailed in the data to sequentially reason fine-grained entity labels for type classification.
- Dai et al. (2021): This approach uses a BERT Masked Language Model to generate weak labels for ultra-fine entity typing to improve the performance of type classification.

B.3 Experiment Settings

Our experiments are conducted on a workstation of dual GeForce GTX 1080 Ti with 32G memory, and the environment of torch 1.7.1. We adopted a T5-base with 12 layers and 12 self-attention heads for the English dataset and mT5-small with 8 layers and 6 self-attention heads for the Chinese dataset. The hyperparameter settings of training our PLM-based GET are: $\lambda = 0.5$, $\mu = 2$. The beam size b is 8. The coefficient weight α in the loss function is 4.

Data Size	CT #.	Prec.	Recall	F1
50%	1040	81.95%	32.29%	46.32%
100%	**1275**	**87.10%**	**39.58%**	**54.43%**

Table 10: Model Performance with different sizes of training data in Ultra-fine entity typing task. 50% and 100% are the proportions of auto-generated data used to train the model.

μ	λ	CT #	Prec.	Recall	F1
2	0.5	**373**	**90.75%**	**30.52%**	**45.68%**
2	0.1	330	80.10%	27.00%	40.39%
2	1	359	86.51%	29.38%	43.86%
4	0.5	328	82.21%	26.84%	40.47%
4	0.1	361	82.42%	29.54%	43.49%
4	1	348	81.88%	28.48%	42.26%

Table 11: Our model's performance with different hyper-parameter settings.

B.4 Human Assessment

It is impossible to know all newly-generated types apriori. Thus human annotators are needed to assess whether the generated types are correct. We employ two annotators to ensure the quality of the assessment. Each predicated type is labeled with 0 or 1 by two annotators, where 0 means a wrong type for the given entity and 1 represents the right type for the given entity. If the results from the two annotators are different, the third annotator will be hired for a final check.

B.5 Result Confidence

We also conduct a statistical significance test (Dror et al., 2018) to show our experiment results are convincing. Specifically, we run our method on the test set of the Ultra-fine entity typing dataset twice with different random seeds. Then we implement a t-test on the two results with a 0.05 significance level. The result is not significant (p-value: 0.208) and thus we can not reject the null hypothesis (H0: result1-result2=0, where result$_i$=(CT#, Prec., R-recall, R-F1)). Based on the above hypothesis test, we believe that our experiment results are confident and reproducible.

Besides, we do a run with 50% auto-generated training data for the Ultra-fine entity typing task and the results are shown in Table 10. We find that our method suffers from a slight performance drop but still outperforms the baselines, which shows the effectiveness of auto-generated data.

B.6 Parameter Tuning Results

We explore the influence of different λ and μ on the performance of our model, as shown in Table 11.