Language Model Pre-Training with Sparse Latent Typing

Liliang Ren[1*], **Zixuan Zhang**[1*], **Han Wang**[2], **Clare R. Voss**[3], **Chengxiang Zhai**[1], **Heng Ji**[1]
[1]University of Illinois at Urbana-Champaign, [2]Amazon Alexa,
[3]US Army Research Laboratory
`{liliang3, zixuan11, czhai, hengji}@illinois.edu`
`wnghn@amazon.com, clare.r.voss.civ@army.mil`

Abstract

Modern large-scale Pre-trained Language Models (PLMs) have achieved tremendous success on a wide range of downstream tasks. However, most of the LM pre-training objectives only focus on text reconstruction, but have not sought to learn latent-level interpretable representations of sentences. In this paper, we manage to push the language models to obtain a deeper understanding of sentences by proposing a new pre-training objective, *Sparse Latent Typing*, which enables the model to sparsely extract sentence-level keywords with diverse latent types. Experimental results show that our model is able to learn interpretable latent type categories in a self-supervised manner without using any external knowledge. Besides, the language model pre-trained with such an objective also significantly improves Information Extraction related downstream tasks in both supervised and few-shot settings. Our code is publicly available at `https://github.com/renll/SparseLT`.

1 Introduction

Transformer-based Pre-trained Language Models (PLMs) have achieved significant success on a wide range of NLP tasks. However, typical pre-training objectives for PLMs only focus on teaching the model to directly reconstruct text-level words or sentences, but have not sought to obtain deeper sentence understanding by learning latent-level interpretable representations. For example, transformer-decoder models like the OpenAI GPT series (Radford et al., 2018, 2019; Brown et al., 2020) adopt

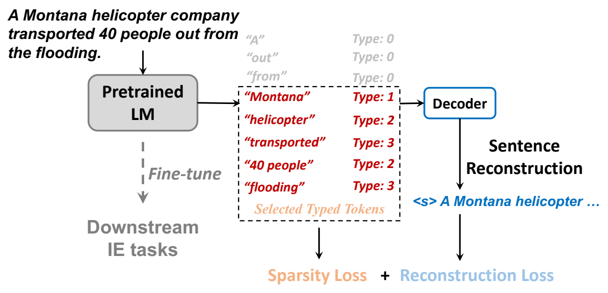

Figure 1: A general illustration of our approach to teach pre-trained language model to extract sentence-level keywords with latent type representations in a completely self-supervised manner.

the task of language modeling for pre-training, and transformer-encoder models like BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019) are trained by predicting the masked tokens within a sentence. Both of these training objectives merely train the models to recover the masked tokens or predict the next words or sentences, while ignoring to learn latent-level representations of sentences that could be potentially useful for both better language understanding and downstream tasks.

Pre-training a language model to learn latent representations is extremely hard: First, there are no ground-truth labels for the latent representations that could be used for reliable supervised learning. During pre-training, the model is only given an unlabeled text corpus over which to identify latent representations such as sentence-level keywords and structures. This means the training process must be strictly self-supervised (Rush et al., 2018). Furthermore, to be interpretable, the latent representations for natural language texts are supposed to be discrete, which further complicates the design of a completely differentiable training framework.

To push the language models to learn deeper understandings of sentences, in this paper, we propose a novel pre-training framework, *Sparse Latent Typing*, that enables the language model to sparsely extract sentence-level keywords with

[*] Equal contribution. Listing order is random. Liliang proposed and implemented the architecture designs and the training objectives of Sparse Latent Typing (SLT), and he also conducted extensive experiments for pre-training, few-shot evaluation and the analyses. Zixuan designed the language model pre-training pipeline for SLT, built the initial training codebase and conducted experiments for pre-training and supervised evaluation. Both of the authors initially came up with the same project goal of encouraging the model to sparsely select sentence-level key words during pre-training.

meaningful latent types. We have tackled all above-mentioned challenges and our framework is fully differentiable and completely self-supervised. As shown in Figure 1, given an input sentence from the pre-training corpus, we introduce a latent typing mechanism to jointly selects and classifies the keywords from the sentence into a category of randomly initialized latent types. We implement such a latent classification model based on Gumbel Sampling (Jang et al., 2017) to make sure the overall pre-training framework is differentiable. Since there are no ground-truth labels available for the selected keywords and latent types, we incorporate an one-layer transformer decoder into the training pipeline to map the fused token and latent type representations back to the original sentence, and use the sentence reconstruction loss to control for adequate usefulness of the latent representations. Our approach provides the decoder model with a shortcut to directly access the encoded token representations, so that the latent representation for each of the input tokens can be learned as an auxiliary type representation. For pre-training objectives, in addition to minimizing the sentence reconstruction error, we also introduce a novel typing sparsity loss to minimize the number of token representation selected for latent typing. A KL-divergence based diversity loss is also proposed to encourage a diverse selection of the latent types. Experimental results show that our model is able to learn interpretable latent type categories in a self-supervised manner without using any external knowledge. Besides, the language model pre-trained with such an objective also significantly improves Information Extraction related downstream tasks in both supervised and few-shot settings.

In summary, our contributions are three-fold:

- We propose a fully differentiable language model pre-training framework that enables the model to sparsely extract sentence-level keywords with latent types in a completely self-supervised manner.
- We provide comprehensive analysis and interpretation for our experimental results showing that the pre-trained model is able to extract meaningful latent type representations.
- Extensive experiments on IE-related downstream tasks demonstrate that our proposed pre-training framework can significantly advance state-of-the-art.

2 Related Work

Knowledge-Enhanced Language Models As pretrained language models (Radford et al., 2018; Devlin et al., 2019; Liu et al., 2019; Radford et al., 2019; Brown et al., 2020; Lewis et al., 2020a; Raffel et al., 2020) are achieving great success on downstream NLP tasks, many research studies focus on how to make these PLMs more knowledgeable. Previous studies (Peters et al., 2019; Zhang et al., 2019; Xiong et al., 2020; He et al., 2020; Yamada et al., 2020; Qin et al., 2021; Wang et al., 2021) either focus on designing entity-relation-aware pre-training objectives, or modifying the model architecture to make it capable of fusing both text and entity information. However, all of these previous approaches utilize large-scale, human-annotated, semi-structured external resources (e.g., Wikipedia). In comparison, our method is completely self-supervised and only needs a text corpus for pre-training, which focuses more on encouraging the model to learn knowledge clusters at a latent level.

Latent Structure Learning There are also several studies (Liu et al., 2021; Subramani et al., 2022) that incorporate latent structure learning into language model pre-training. Particularly, Montero et al. (2021) also proposes to use a transformer decoder layer to reconstruct the original sentence to provide training signals. However, instead of learning coarse-grained sentence representations, we focus on learning fine-grained latent type representation that are interpretable and useful at the token level. To meet this end, we propose a series of novel training objectives and architecture designs to facilitate a sparse selection and typing of the token representations in the latent space.

Information Extraction Our approach to detect sentence-level keywords with latent types is inspired by Information Extraction (IE) (Cowie and Lehnert, 1996), an essential NLP task that aims to extract knowledge from texts. Although IE includes a wide range of tasks varying in *what to extract* (entities, relations, events) and *where to extract from* (sentences, documents, corpora), typical IE frameworks usually include two essential steps: 1) *Selection*: selecting the most task-relevant units from the inputs, 2) *Classification*: assigning each of these a correct type label. Such a select-and-classify framework is common to several IE tasks, including entity extraction, event detection and event argument extraction. Accordingly, in our approach, we follows a similar *Selection-Classification* approach to incorporate word selections and latent typing in pre-training.

3 Problem Formulation

Given a text corpus $\mathcal{D}$ composed of text sentences $\mathcal{S}$, we use $\mathbf{s} = \{w_1, \cdots, w_N\}, \mathbf{s} \sim \mathcal{S}$ to represent a sentence consisting of N tokens. Assuming a text encoder $f : \mathcal{S} \mapsto \mathcal{X}$ that takes the sentence $\mathbf{s}$ as input and outputs the token representations $\mathbf{x}_1, \cdots, \mathbf{x}_N$, our latent type classifier $h : \mathcal{X} \mapsto (\mathcal{Z}, \hat{\mathcal{X}})$ then selects a subset of token representations $\hat{\mathbf{x}}_1, \cdots, \hat{\mathbf{x}}_T, T \leq N$ and classifies them into latent type representations $\mathbf{z}_1, \cdots, \mathbf{z}_T$. Each of the token types $\mathbf{z}_i$ is selected from a latent embedding space $\mathcal{C}$ consisting of $V_c = |\mathcal{C}|$ different latent vectors. The text decoder $g : (\mathcal{Z}, \hat{\mathcal{X}}) \mapsto \mathcal{S}$ then reconstructs the original sentence $\mathbf{s}$ through the pair of latent types and selected token representations $(\mathcal{Z}, \hat{\mathcal{X}})$.

The objective of sparse latent typing is to find pairs of latent types and token representations that are as compact as possible but still contain the necessary information for reconstructing the original input sentences. Formally, we want to minimize the following joint objective,

$$\min_{\theta_f, \theta_h, \theta_g} T, \mathcal{L}_{\text{rec}}(f, h, g), \mathcal{L}_{\text{KL}}(f, h) \qquad (1)$$

with:

$$\begin{aligned} \mathcal{L}_{\text{rec}}(f, h, g) &= \mathbb{E}_{\mathbf{s} \sim \mathcal{S}}[-\log p_g(\mathbf{s}|h(f(\mathbf{s})))], \\ \mathcal{L}_{\text{KL}}(f, h) &= D_{\text{KL}}(p_h(\mathbf{z}|f(\mathbf{s}))||p(\mathbf{z})), \end{aligned}$$

where T is the number of the selected token representations, $p(\mathbf{z})$ is a prior distribution of the latent types, and $D_{\text{KL}}(\cdot||\cdot)$ is the Kullback–Leibler (KL) divergence. The reconstruction loss and the KL term in our formulation follows the classical VAE (Kingma and Welling, 2013), but there are two key differences: (1) The latent variables $\mathbf{z}$ are discrete categorical variables, (2) Instead of only taking the latent representation z, the decoder takes both the token vectors and the corresponding latent vectors for sentence reconstruction. Since the discrete version of VAE is well studied by the previous efforts such as VQ-VAE (Van Den Oord et al., 2017) and Gumbel-Softmax (Jang et al., 2017), the optimization problem remains as how to minimize the non-differentiable term T to encourage the sparse selection of the token representations.

4 Learning Sparse Latent Types

To tackle the non-differentiable problem of the size of the selected typing pairs $T = |(\mathcal{Z}, \hat{\mathcal{X}})|$, we first take a closer look at the latent type classifier h which decides the latent type $\mathbf{z}_i$ of each token representation $\mathbf{x}_i$. Our insight is that we can regard the action of not selecting a token representation as a frozen zero type vector $\mathbf{c}_1 = \mathbf{0} \in \mathcal{C}$. We then do an element-wise multiplication between $\mathbf{z}_i$ and $\mathbf{x}_i$ to obtain the representations $\bar{\mathbf{x}}_i = \mathbf{x}_i \otimes \mathbf{z}_i$ that are to be fed into the text decoder g. The advantages of this approach are that (1) the element-wise multiplication naturally prevents the gradient from being propagated to the token representations that are classified as the zero type vector $\mathbf{c}_1$, (2) the element-wise multiplication directly modulates the gradients of the token representations with the latent type vectors. This can in principle provide better guidance to the text encoder with the information of the latent vectors than can be provided by other vector fusion operators such as element-wise addition or vector concatenation. Based on this framework, we developed a novel typing sparsity loss in Section 4.2 to approximately minimize the typing pairs size T. While our approach is generally applicable for any text encoder and decoder, specific neural architectures used in this work are discussed in Section 5.1.

In our framework, the latent type classifier h is simplified as a mapping $h' : \mathcal{X} \mapsto \mathcal{Z}$ that only outputs the latent types $\mathbf{z}_i$ for each token representation. The simplified text decoder $g' : \bar{\mathcal{X}} \mapsto \mathcal{S}$ then only needs to model the fused representation space $\bar{\mathcal{X}} = \mathcal{Z} \otimes \mathcal{X}$ for sentence reconstruction. $\otimes$ is the vector fusion operator and should be interpreted as element-wise multiplication in this work. The proposed architecture for sparse latent typing is illustrated in Figure 2, which is further explained in the following subsections.

4.1 Gumbel Latent Typing

Given the token representations generated from the text encoder, $X = \{\mathbf{x}_1, \cdots, \mathbf{x}_N\} \in \mathbb{R}^{N \times d_m}$, where N is the number of input tokens, and d_m is the length of the token representation vectors, our Gumbel latent type classifier first maps X into logits $L \in \mathbb{R}^{N \times V_c}$ with a weight matrix $W \in \mathbb{R}^{d_m \times V_c}$, and then outputs the probabilities $P_{i,v}$ of choosing the v-th latent type for each token representation

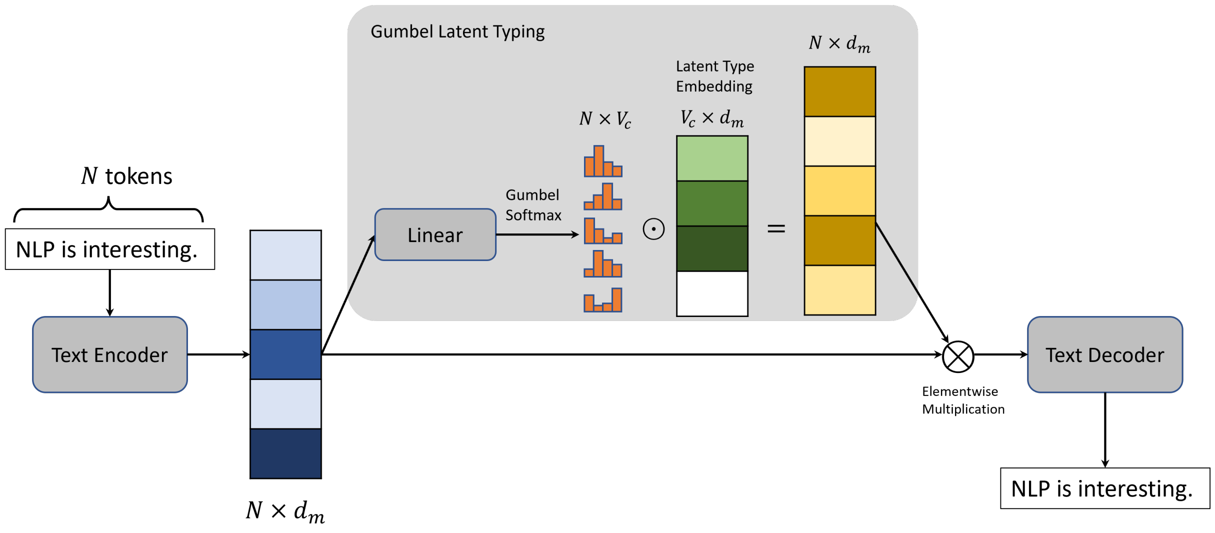

Figure 2: The proposed architecture for pre-training a language model with Gumbel Latent Typing, where d_m is the length of the token representation vectors, V_c is the pre-defined size of the latent types, and $\odot$ means the matrix multiplication. The white block in the latent type embedding is the zero type vector.

$\mathbf{x}_i$,

$$P_{i,v} = \frac{e^{(L_{i,v}+G_{i,v})/\tau}}{\sum_{k=1}^{V_c} e^{(L_{i,k}+G_{i,k})/\tau}},$$
$$L = XW,$$

where $G_{i,v} \sim \text{Gumbel}(0,1)$ is the Gumbel noise sampled from a standard Gumbel distribution and τ is the non-negative temperature, following the previous efforts on the Gumbel softmax operation (Jang et al., 2017; Maddison et al., 2016). The reason why we are using Gumbel softmax for our latent type classifier is that it enables choosing a latent type representation in a fully differentiable way, and thus can further facilitate our design of the sparsity loss to do an approximate minimization of the size of the typing pairs T.

With the Gumbel decision probability $P \in \mathbb{R}^{N\times V_c}$, the latent type representations $Z \in \mathbb{R}^{N\times d_m}$ are obtained through a marginalization of P over the latent type embeddings $C \in \mathbb{R}^{V_c\times d_m}$,

$$\mathbf{c}_1 = \mathbf{0},$$
$$Z = \{\mathbf{z}_1, \cdots, \mathbf{z}_N\},$$
$$\mathbf{z}_i = \sum_{k=1}^{V_c} P_{i,k}\mathbf{c}_k,$$

where $\mathbf{c}_1$ is the zero type vector. The final fused representation $\bar{X}$ is obtained by an element-wise multiplication,

$$\bar{X} = X \otimes Z.$$

Intuitively, if $P_{i,k}$ is entirely concentrated on c_1 as $\tau \to 0$, i.e., $P_{i,1} = 1$, we effectively eliminate token w_i (or its representation x_i).

During the evaluation stage of our latent type classifier, the latent type embedding with the largest logit score is selected as the latent type representation for each of the token vectors $\mathbf{x}_i$,

$$k^* = \arg\max_k L_{i,k},$$
$$\mathbf{z}_i = \mathbf{c}_{k^*}.$$

To alleviate the discrepancy between the training and the evaluation, we adopt the temperature annealing (Jang et al., 2017) trick to the Gumbel-Softmax for a better differentiable approximation of the argmax operator.

4.2 Training Objectives

Based on our problem formulation, we adopt three types of training loss for the end-to-end training of our model: (1) *Typing Sparsity Loss* that encourages the latent type classifier to choose more zero types, (2) *KL-Divergence* with respect to a uniform prior distribution to encourage the diverse selection of the latent types, (3) *Reconstruction Loss* that ensures the latent representation maintains essential information of the input text.

Typing Sparsity Loss An important property of Gumbel-Softmax is that when the temperature $\tau \to 0$, the decision probability $P_i \in \mathbb{R}^{V_c}$ will tend to be an one-hot index vector sampled from the underlying categorical distribution,

$$\hat{P}_i(C = v) = \frac{e^{L_{i,v}}}{\sum_{k=1}^{V_c} e^{L_{i,k}}},$$

where $L \in \mathbb{R}^{N \times V_c}$ is the logits before doing Gumbel-Softmax. This means that we can control the decision behavior of the model through modulating the shape of this categorical distribution. Therefore, our typing sparsity loss is designed as the negative log-likelihood of the global averaged probability of choosing the zero type $\mathbf{c}_1$,

$$\mathcal{L}_s = -\log \frac{1}{N} \sum_{i=1}^{N} \hat{P}_i(C = 1),$$

where N is the number of tokens in the input sentence. Intuitively, if $\hat{P}_i$ converges to an one-hot vector, then $\hat{P}_i(C = 1) \in \{1, 0\}$, and $\sum_{i=1}^{N} \hat{P}_i(C = 1) = N - T$ becomes the number of the tokens that are not selected for typing, which is equivalent to what we want to maximize in the problem formulation of Equation (1).

KL-Divergence To encourage a diverse selection of the latent types, we assume a uniform prior distribution of the latent type representations $p(\mathbf{z}) = 1/V_c$. The KL-divergence term is calculated between the global averaged probability $\bar{P}_v$ and the uniform prior, *i.e.*,

$$\mathcal{L}_{\text{KL}} = \frac{1}{V_c} \sum_{v=1}^{V_c} \bar{P}_v \log(\bar{P}_v),$$
$$\bar{P}_v = \frac{1}{N} \sum_{i=1}^{N} \hat{P}_{i,v},$$

where V_c is the number of the latent types.

Reconstruction Loss Our reconstruction loss directly follows our problem formulation, *i.e.*,

$$\mathcal{L}_{\text{rec}} = -\frac{1}{B} \sum_{i=1}^{B} \log p_g(\mathbf{s}_i | h(f(\mathbf{s}_i))),$$

where B is the batch size of the sampled text data. When pre-training a masked language model, we also include a Masked Language Model loss following BERT (Devlin et al., 2019),

$$\mathcal{L}_{\text{MLM}} = -\frac{1}{B} \sum_{i=1}^{B} \log p_f(\mathbf{s}_i | \tilde{\mathbf{s}}_i), \quad (2)$$

where f is the text encoder, and $\tilde{\mathbf{s}}_i$ is the corrupted sequence.

The total loss function is a weighted sum of the above four losses,

$$\mathcal{L} = \mathcal{L}_{\text{MLM}} + \alpha \mathcal{L}_{\text{rec}} + \beta \mathcal{L}_s + \gamma \mathcal{L}_{\text{KL}}, \quad (3)$$

where $\alpha, \beta, \gamma \in \mathbb{R}_{\geq 0}$ are weighting factors.

5 Experiments

In our experiments, we first conduct intrinsic evaluation to investigate whether the model can successfully learn word selections with meaningful latent types during pre-training. Then, we apply our model on both supervised and few-shot IE tasks to evaluate the effectiveness of our pre-training framework on downstream tasks.

5.1 Sparse Latent Type Learning

Pre-training Setup We adopt the VOA corpus constructed by (Li et al., 2020) for sparse latent type pre-training, which was extracted from 108,693 multimedia news articles openly available on the Voice of America website between 2006 and 2017. We use the *bert-base-uncased* version of the BERT (Devlin et al., 2019) model as our encoder, and a single transformer decoder layer to reconstruct the sentence, following Kasai et al. (2020); Montero et al. (2021). While our approach is generally applicable for both encoder-only Masked Language Model (MLM) and the encoder-decoder Denoising Language Model (e.g. BART (Lewis et al., 2020b)), we focus on MLM because MLM is more widely used in the downstream information extraction tasks. The implementation details can be found in Appendix A.

Downstream Evaluation To evaluate the validity of our latent typing approach, we apply our pre-trained model by fine-tuning on downstream tasks. We focus on Information Extraction specifically, and adopt *Supervised Joint Information Extraction* (Lin et al., 2020) and *Few-shot Named Entity Recognition* (Ding et al., 2021) as two typical IE tasks to evaluate our model on both supervised and few-shot IE settings. We initialize the BERT model with *bert-base-uncased* weights and continue pre-training 100,000 steps using the combined loss defined in (3) on the VOA corpus. Since we only focus on evaluating our model on the IE tasks, only the pre-trained text-encoder is used for fine-tuning on the downstream tasks. More details can be found in Appendix A.

5.2 Supervised Information Extraction

Datasets We evaluate our pre-trained model on the English subset of ACE-2005 dataset[1] and the ERE dataset, which are the most widely-used event-centric IE dataset containing annotations for extracting entities, events, and relations. Following the preprocessing steps and dataset splits in (Lin et al., 2020), we keep 7 entity types, 6 relation types, 33 event types, and 22 argument roles for the ACE-2005 dataset, and 7 entity types, 5 relation types, 38 event types, and 20 argument roles for the ERE dataset. More detailed dataset statistics are shown in Table 1.

Dataset	Split	#Sents	#Ents	#Events	#Rels
ACE-05	Train	17,172	29,006	4,202	4,664
	Dev	923	2,451	450	560
	Test	832	3,017	403	636
ERE	Train	14,736	39,501	6,208	5,054
	Dev	1,209	3,369	525	408
	Test	1,163	3,295	551	466

Table 1: Dataset statistics for supervised IE.

Baselines We compare the performances of fine-tuning BERT on supervised IE with the following pre-training approaches: 1) ***BERT-Vanilla***: we directly use the *bert-base-uncased* checkpoint to fine-tune on the supervised IE tasks, which is also the same as what the baseline models do in OneIE (Lin et al., 2020). 2) ***BERT-MLM***: we initialize the BERT model with the *bert-base-uncased* checkpoint and then fine-tune on the VOA corpus for 100,000 steps only using the masked language modeling loss $\mathcal{L}_{\text{MLM}}$ defined in (2). 3) ***BERT-SparseLT***: our proposed approach. We pretrain the BERT model from the *bert-base-uncased* checkpoint on the VOA corpus for 100,000 steps by encouraging the model to learn sparse latent types using the loss function defined in (3), with the hyper-parameters $\alpha = 0.05, \beta = 0.05, \gamma = 0.1$. We did not compare our model with knowledge-enhanced pretrained language models like ERNIE (Zhang et al., 2019) and ERICA (Qin et al., 2021) because they use external knowledge resources aligned with the pre-training corpus, while our methods are completely self-supervised.

Results We report the F1 scores for four different IE subtasks on both ACE-2005 and ERE datasets: *Entity Extraction*, *Relation Extraction*, *Event Detection* and *Event Argument Role Labeling*, and the results are shown in Table 2. In general, our *BERT-SparseLT* model has the best performance among all model competitors, even better than the OneIE model which uses global features for fine-tuning. In particular, our proposed method greatly improves the entity extraction performance (an absolute improvement of 7.59% on the ERE-Entity subtask), which follows our intuition since sparse latent typing can make the model more sensitive about important entity mentions in the sentence. We can also see that *BERT-MLM* outperforms *BERT-Vanilla*, which comes as no surprise since further pre-training the models on additional corpora usually leads to better performances. The key observation that demonstrates the effectiveness of our approach is *BERT-SparseLT* outperforms *BERT-MLM* significantly, where our model uses exactly the same pre-training corpus without any additional information.

5.3 Few-shot Named Entity Recognition

Dataset We use the most recent FewNERD (Ding et al., 2021) dataset to evaluate the performance of our proposed model on few-shot IE settings. The FewNERD dataset includes 8 coarse-grained and 66 fine-grained entity types, which has two experimental settings: 1) *Inter*: The training and testing entity types are divided only based on 66 fine-grained entity types. 2) *Intra*: A more challenging setting where the training and testing entity types strictly belong to different coarse-grained entity types. We evaluate the few-shot performance of our models in both settings and the results are shown in Table 3.

Baselines and Results We compare our model with two competitive baselines *StructShot* (Yang and Katiyar, 2020) and *CONTaiNER-Viterbi* (Das et al., 2022). We replace the BERT text encoder from the state-of-the-art model *CONTaiNER-Viterbi* with our text encoder pre-trained with sparse latent typing, and denote it as *CONTaiNER-Viterbi + BERT-SparseLT*. We report the F1 score of the fewshot Named Entity Recognition tasks in both *intra* and *inter* evaluation settings. In general, our proposed framework greatly ourperforms previous models in both settings by 6.24% and 3.75% respectively while creating a new state-of-the-art of this benchmark.

[1]https://catalog.ldc.upenn.edu/LDC2006T06

Dataset	*ACE-2005*				*ERE*			
IE subtasks	Entity	Trigger	Argument	Relation	Entity	Trigger	Argument	Relation
BERT-Vanilla	75.34	66.40	44.90	41.35	79.39	55.58	36.76	31.05
OneIE-bert-base (Lin et al., 2020)	77.86	66.91	48.01	47.88	79.84	54.33	37.85	33.39
BERT-MLM	78.30	68.66	48.53	51.72	79.54	55.39	38.35	33.78
BERT-SparseLT	**81.10**	**70.95**	**50.87**	**52.80**	**87.13**	**55.76**	**39.64**	**37.12**

Table 2: Overall test F1-scores (%) of Supervised Joint Information Extraction.

Model	***5-way***		***10-way***		***Average***
	1 ∼ 2 shot	5 ∼ 10 shot	1 ∼ 2 shot	5 ∼ 10 shot	
INTRA					
StructShot (Yang and Katiyar, 2020)	30.21	38.00	21.03	26.42	28.92
CONTaiNER-Viterbi + BERT (Das et al., 2022)	40.40	53.71	33.82	47.51	43.86
CONTaiNER-Viterbi + BERT-SparseLT (Ours)	**47.20**	**59.67**	**40.48**	**53.04**	**50.10**
INTER					
StructShot (Yang and Katiyar, 2020)	51.88	57.32	43.34	49.57	50.52
CONTaiNER-Viterbi + BERT (Das et al., 2022)	56.10	61.90	48.36	57.13	55.87
CONTaiNER-Viterbi + BERT-SparseLT (Ours)	**57.14**	**66.17**	**52.75**	**62.43**	**59.62**

Table 3: Overall test F1-scores (%) of Few-shot Named Entity Recognition.

6 Analysis

In this section, we address the following research questions on Sparse Latent Typing.

How are the latent types distributed over the encoded token representations? We draw a t-SNE (van der Maaten and Hinton, 2008) plot of the encoded token representations $\mathbf{x}$ of 1,000 sentences (30792 tokens in total) randomly sampled from the pre-training corpus in Figure 3. The token representations are colored with their corresponding latent type indices. From the figure, we can observe that the token representations begin to be distributed as individual islands with the same colors after 300k steps of pre-training from scratch. This implies that our sparse latent typing objective can effectively encourage the clustering of the token representations in a small latent space defined by 64 randomly initialized type embeddings. We can also observe a similar trend of latent clustering for the *BERT-SparseLT* model, which is illustrated in Appendix B, Figure 4.

Can Typing Sparsity Loss effectively control the sparsity of token selections? We pre-trained three *BERT-SparseLT* models with different weighting factors β of the Typing Sparsity Loss to investigate its influence on latent type selections and sentence reconstruction. (Additional samples for $\beta = 0.05$ are included in Appendix B, Table 14) From Table 4, we can observe that as the β increases the model will select fewer tokens with non-zero types. The corresponding sentence reconstruction also degenerates significantly as fewer tokens are selected by our Gumbel latent type classifier. This means that our proposed typing sparsity loss can effectively control the number of the typed tokens and thus affect the quality of reconstructed sentences. We also did the same experiments for the models pre-trained from scratch (Appendix B, Table 11) to illustrate that such sparse selection behavior is independent of the initialization of the model parameters.

To what extent are the learned latent types interpretable? In Table 5, we calculate the frequencies for the top-10 most selected latent types v and the probability $P(x|z = v)$ of the top-5 frequent tokens tagged by the type v. The statistics are computed over 100 randomly sampled sentences from the pre-training VOA corpus with the *BERT-SparseLT* model. We can observe that the zero type (index 0) is mostly associated with less meaningful tokens such as ",", "CLS","SEP", which are used for delimiting the semantics. This means that the model can effectively learn to select more meaningful tokens through our sparse latent typing objective. We can also observe that the type 61 seems more correlated with the physical locations and the type 50 is mostly related to functional words. We also do the same analysis for the model pre-trained from scratch in Appendix B, Table 12. The results

β	Latent Typing of the Input Tokens	Reconstructed Sentence
0.05	she(58), murdered(24), her(1), new(8), york(42), office(61), just(34), days(11), learning(4), waitress(62), accepted(60), sundance(9), film(63), festival(50),	she was murdered in her new york office, just days after learning that waitress was accepted in her the sundance film festival.
0.06	murdered(47),learning(22), waitress(42), accepted(51), sundance(47),	the government was murdered in the philippines, and a man after learning that waitress was accepted in the sundance festival.
0.07	$\emptyset$	the u. s. officials have been critical of the country's.

Table 4: The latent typing and the reconstruction results of the input tokens, "She was murdered in her New York office, just days after learning that Waitress had been accepted into the Sundance Film Festival.", with different BERT-SparseLT models continually pre-trained with different values of the weighting factors β. The corresponding latent type indices for each of the tokens are noted in the parentheses.

Top-10 Frequent Type Index v	Top-5 Frequent Tokens Tagged by the Type v
0 (37.86%)	, (12.2%) the (12.1%) . (11.6%) CLS (8.2%) SEP (8.2%)
61 (3.82%)	##ta (2.4%) abdullah (2.4%) kabul (2.4%) afghan (2.4%) ##hi (1.6%)
48 (3.29%)	identified (2.8%) told (2.8%) ' (1.9%) ##j (1.9%) calls (1.9%)
50 (2.82%)	" (5.5%) mm (5.5%) " (4.4%) which (3.3%) we (2.2%)
20 (2.30%)	but (8.1%) abortion (8.1%) " (5.4%) that (4.1%) dr (4.1%)
53 (2.27%)	carrier (2.7%) ka (2.7%) fraud (2.7%) claims (2.7%) harder (2.7%)
8 (2.02%)	also (4.6%) 2017 (3.1%) among (3.1%) commission (3.1%) department (3.1%)
38 (1.83%)	never (3.4%) story (3.4%) afghanistan (3.4%) could (3.4%) rate (3.4%)
62 (1.80%)	abdullah (5.2%) indigo (3.4%) allegations (3.4%) graduating (1.7%) innovative (1.7%)
29 (1.77%)	low (5.3%) cost (5.3%) me (5.3%) may (3.5%) gee (3.5%)

Table 5: The frequencies of the top-10 most selected latent types v and the corresponding top-5 frequent tokens tagged by the type v. The frequencies are computed over 100 randomly sampled sentences from the pre-training corpus and are noted in the parentheses after the tokens. The model is continually pretrained from a *bert-base-uncased* checkpoint.

appear to be more interpretable due to more consistent training dynamics than continual pre-training from a checkpoint.

How could different loss combinations affect the model performance? We conduct an ablation study of the loss weighting factors of the *BERT-SparseLT* model to illustrate the influence of different loss combinations on the few-shot NER performance in Table 7. Including all the four losses produces the best test performance on the 5-way 1$\sim$2 shot evaluation of the *Intra* setting of the FewNERD benchmark. This proves that all the training objectives are necessary for improving the generalizability of the token representations learned by the encoder. We also include the reconstruction and the latent typing results of the models trained with $\alpha = 0.05$ in Appendix B, Table 13 for further qualitative analyses.

Can sparse latent typing improve the sentence-level Natural Language Understanding (NLU) ability? We evaluate *BERT-Vanilla*, *BERT-MLM* and the *BERT-SparseLT* models on the General Language Understanding Evaluation (GLUE) benchmark to demonstrate the influence of sparse latent typing on NLU, and the results are shown in Table 6. The finetuning hyperparameters are shown in Appendix B, Table 9. Our *BERT-SparseLT* model obtains slightly worse results than the vanilla BERT, but still has marginal improvement over the MLM baseline that excludes the sparse latent typing objectives. The inferior results are as expected for two reasons: 1) The evaluation of the GLUE benchmark heavily relies on the [CLS] token which is always latent typed as a zero-type by the *BERT-SparseLT* model and thus lacks the enough training signals for fine-grained clustering in the latent space. 2) The VOA corpus for continual pretraining is in the specific news domain, which may not be beneficial for general NLU. We hypothesize that large-scale pretraining from scratch of an encoder-decoder model should overcome these limitations, and we leave it as the future work due to the limitation of the computation resources.

7 Conclusion

In this paper, we propose a novel language model pre-training framework that encourages the model

System	MNLI-(m/mm)	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	**Average**
BERT$_{BASE}$-Vanilla	84.7/84.8	88.5	91.5	92.4	58.1	88.7	90.8	70.4	83.3
BERT$_{BASE}$-MLM-VOA	84.5/84.8	88.6	91.4	92.1	59.4	88.5	89.3	64.6	82.6
BERT$_{BASE}$-SparseLT-VOA	84.5/84.8	88.6	91.4	92.1	59.4	89.1	90.7	66.8	83.0

Table 6: The evaluation results on the development sets of the GLUE benchmark. Following Devlin et al. (2019), we report the F1 scores for QQP and MRPC, Spearman correlations are reported for STS-B, and accuracy scores for the other tasks. A fixed random seed is applied for all the experiments for fair comparisons.

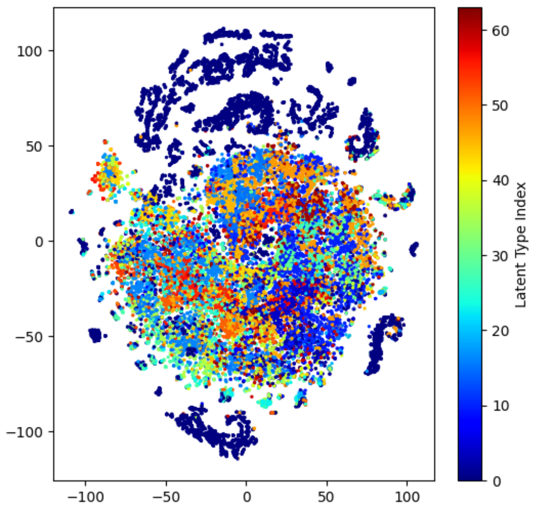

Figure 3: The t-SNE visualization of the encoded token representations $\mathbf{x}$ with the corresponding latent type indices after 300k steps of pre-training with the sparse latent typing objective. The pre-training process is conducted on the VOA-corpus with randomly initialized parameters.

α	β	γ	5-way 1∼2 shot
0	0	0	46.10
0	0	0.1	45.87
0	0.05	0	45.72
0.05	0.05	0	46.72
0.05	0	0	46.48
0.05	0	0.1	46.99
0	0.05	0.1	45.94
0.05	0.05	0.1	**47.20**

Table 7: The test F1 scores on the *INTRA* 5-way 1∼2 shot setting of the FewNERD dataset for different *BERT-SparseLT* models continually pre-trained with different loss weighting factors, α, β, γ.

to sparsely extract sentence-level keywords with meaningful latent types in a completely self-supervised manner. Experimental results and analysis demonstrate that incorporating sparse latent type learning early in the pre-training stage will not only facilitate the model to learn sentence-level keyword selections with interpretable latent types, but also improves downstream Information Extraction tasks in both supervised and few-shot settings.

8 Limitations

One primary limitation of our framework is that the language model pretrained with sparse latent typing might only improve performance on Information Extraction tasks. Although this is intuitive since IE shares essential similarity with latent typing, it is exciting to see whether our model can improve other downstream tasks such as natural language generation and abstractive summarization.

Another limitation of our work is that, due to the lack of the computation resources, we did not conduct experiments of large-scale pretraining from scratch for a comprehensive examination of our framework's ability on improving the general NLU performance. For future works, it is also worth exploring on whether the sparse latent typing objective can improve the machine translation performance by regularizing a sparse and unified latent space for cross-lingual meaning representations.

Finally, our model is only capable of extracting sentence-level keywords with latent types, but is not designed to learn a comprehensive graph structure for each input sentence. Although it is debatable whether a more complex latent graph representation is better than concise latent types, it is still worth adding this into future work plans.

Acknowledgement

We thank the anonymous reviewers helpful suggestions. This material is based upon work supported in part by the IBM-Illinois Discovery Accelerator Institute and by the National Science Foundation under Grant No. 1801652. This research is also based upon work supported by U.S. DARPA KAIROS Program No. FA8750-19-2-1004 and U.S. DARPA AIDA Program No. FA8750-18-2-0014. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of DARPA, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation therein. We also thank Suyu Ge for early discussions on this project.

References

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901.

Jim Cowie and Wendy Lehnert. 1996. Information extraction. *Communications of the ACM*, 39(1):80–91.

Sarkar Snigdha Sarathi Das, Arzoo Katiyar, Rebecca J Passonneau, and Rui Zhang. 2022. Container: Few-shot named entity recognition via contrastive learning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6338–6353.

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

Ning Ding, Guangwei Xu, Yulin Chen, Xiaobin Wang, Xu Han, Pengjun Xie, Haitao Zheng, and Zhiyuan Liu. 2021. Few-nerd: A few-shot named entity recognition dataset. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3198–3213.

Bin He, Di Zhou, Jinghui Xiao, Xin Jiang, Qun Liu, Nicholas Jing Yuan, and Tong Xu. 2020. BERT-MK: Integrating graph contextualized knowledge into pre-trained language models. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2281–2290, Online. Association for Computational Linguistics.

Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical reparameterization with gumbel-softmax. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Jungo Kasai, Nikolaos Pappas, Hao Peng, James Cross, and Noah Smith. 2020. Deep encoder, shallow decoder: Reevaluating non-autoregressive machine translation. In *International Conference on Learning Representations*.

Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020a. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020b. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Manling Li, Alireza Zareian, Qi Zeng, Spencer Whitehead, Di Lu, Heng Ji, and Shih-Fu Chang. 2020. Cross-media structured common space for multimedia event extraction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2557–2568, Online. Association for Computational Linguistics.

Ying Lin, Heng Ji, Fei Huang, and Lingfei Wu. 2020. A joint neural model for information extraction with global features. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7999–8009, Online. Association for Computational Linguistics.

Qian Liu, Dejian Yang, Jiahui Zhang, Jiaqi Guo, Bin Zhou, and Jian-Guang Lou. 2021. Awakening latent grounding from pretrained language models for semantic parsing. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1174–1189, Online. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. In *ArXiv: abs/1907.11692*.

Chris J Maddison, Andriy Mnih, and Yee Whye Teh. 2016. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*.

Ivan Montero, Nikolaos Pappas, and Noah A. Smith. 2021. Sentence bottleneck autoencoders from transformer language models. In *Proceedings of the 2021*

Conference on Empirical Methods in Natural Language Processing, pages 1822–1831, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Matthew E. Peters, Mark Neumann, Robert Logan, Roy Schwartz, Vidur Joshi, Sameer Singh, and Noah A. Smith. 2019. Knowledge enhanced contextual word representations. 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 43–54, Hong Kong, China. Association for Computational Linguistics.

Yujia Qin, Yankai Lin, Ryuichi Takanobu, Zhiyuan Liu, Peng Li, Heng Ji, Minlie Huang, Maosong Sun, and Jie Zhou. 2021. ERICA: Improving entity and relation understanding for pre-trained language models via contrastive learning. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3350–3363, Online. Association for Computational Linguistics.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J Liu, et al. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(140):1–67.

Alexander Rush, Yoon Kim, and Sam Wiseman. 2018. Deep latent variable models of natural language. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: Tutorial Abstracts*, Melbourne, Australia. Association for Computational Linguistics.

Nishant Subramani, Nivedita Suresh, and Matthew Peters. 2022. Extracting latent steering vectors from pretrained language models. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 566–581, Dublin, Ireland. Association for Computational Linguistics.

Aaron Van Den Oord, Oriol Vinyals, et al. 2017. Neural discrete representation learning. *Advances in neural information processing systems*, 30.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605.

Xiaozhi Wang, Tianyu Gao, Zhaocheng Zhu, Zhengyan Zhang, Zhiyuan Liu, Juanzi Li, and Jian Tang. 2021. KEPLER: A unified model for knowledge embedding and pre-trained language representation. *Transactions of the Association for Computational Linguistics*, 9:176–194.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Wenhan Xiong, Jingfei Du, William Yang Wang, and Veselin Stoyanov. 2020. Pretrained encyclopedia: Weakly supervised knowledge-pretrained language model. In *International Conference on Learning Representations*.

Ikuya Yamada, Akari Asai, Hiroyuki Shindo, Hideaki Takeda, and Yuji Matsumoto. 2020. LUKE: Deep contextualized entity representations with entity-aware self-attention. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6442–6454, Online. Association for Computational Linguistics.

Yi Yang and Arzoo Katiyar. 2020. Simple and effective few-shot named entity recognition with structured nearest neighbor learning. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6365–6375, Online. Association for Computational Linguistics.

Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. 2019. ERNIE: Enhanced language representation with informative entities. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1441–1451, Florence, Italy. Association for Computational Linguistics.

A Implementation Details

Our transformer decoder layer follows the same architecture as the BART model (Lewis et al., 2020a). The word embeddings for both the text encoder and decoder are tied together. Our implementation is based on the Transformers codebase [2]. (Wolf et al., 2020).

We train our model on 4 NVIDIA Tesla V100 GPUs with 16GB memory. The pre-training time for *bert-base-uncased* model on the VOA-corpus is about 12 hours. A linear learning rate scheduling

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

with warm-up is adopted. For Gumbel latent typing, we adopt the following temperature annealing schedule,

$$\tau = \max(5 \times 0.99997^T, 0.5)$$

where T is the number of training steps. We continually pretrain a BERT-base model from the *bert-base-uncased* checkpoint for 100k steps. For the experiment of pre-training from scratch, we only train a RoBERTa-large (Liu et al., 2019) model on the VOA-corpus for 300k steps, given the limitation of our computation resources. The detailed hyper-parameters are summarized in Table 8 and Table 10.

For fine-tuning on downstream tasks, we replace the BERT model used in the state-of-the-arts with our pre-trained BERT-SparseLT model and follow the same hyper-parameter settings. Specifically, for supervised IE, we use the codebase from OneIE (Lin et al., 2020) [3], and for few-shot IE, the codebase from CONTaiNER (Das et al., 2022) is adopted.[4].

Hyper-parameters	Values
Weighting factor for Reconstruction loss, α	0.05
Weighting factor for Typing Sparsity loss, β	0.05
Weighting factor for KL-divergence, γ	0.1
Model dimension, d_m	768
Number of latent types, V_c	64
Batch size	32
Learning rate warm-up steps	300
Max trainig steps	100,000
Adam β_1	0.9
Adam β_2	0.999
Adam ϵ	1e-8
Weight decay	0.01
Learning rate	1e-5
Gradient clipping norm	1.0
MLM masking probability	0.15
Number of decoder layers	1
Dropout rate	0.1
Activation function	*GELU*

Table 8: Detailed settings for model hyper-parameters when pretraining from a *bert-base-uncased* checkpoint.

B Additional Analysis

This section includes all the additional figures and the tables mentioned in section 6.

[3] http://blender.cs.illinois.edu/software/oneie/

[4] https://github.com/psunlpgroup/CONTaiNER

Hyper-parameters	Values
Learning Rate	{1e-5, 2e-5, 5e-5}
Batch size	32
Learning rate warm-up ratio	0.06
Max trainig epochs	10
Weight decay	0.1
Learning rate decay	Linear

Table 9: Hyperparameters for finetuning the *BERT-SparseLT* model on the GLUE benchmark. We follow the settings in Liu et al. (2019).

Hyper-parameters	Values
Weighting factor for Reconstruction loss, α	1.3
Weighting factor for Typing Sparsity loss, β	0.2
Weighting factor for KL-divergence, γ	0.1
Model dimension, d_m	1024
Number of latent types, V_c	64
Batch size	32
Learning rate warm-up steps	300
Max trainig steps	300,000
Adam β_1	0.9
Adam β_2	0.999
Adam ϵ	1e-8
Weight decay	0.01
Learning rate	1e-5
Gradient clipping norm	1.0
MLM masking probability	0.15
Number of decoder layers	1
Dropout rate	0.1
Activation function	*GELU*

Table 10: Detailed settings for model hyper-parameters when pretraining from scratch.

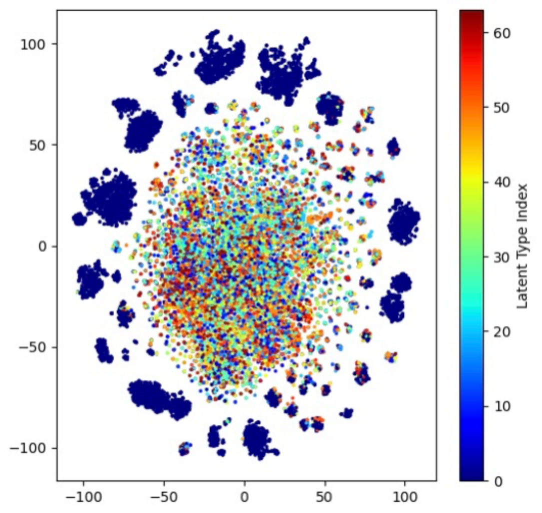

Figure 4: The t-SNE visualization of the encoded token representations $\mathbf{x}$ with the corresponding latent type indices for the *BERT-SparseLT* model. The model is continually pretrained from a *bert-base-uncased* checkpoint.

β	Latent Typing of the Input Tokens	Reconstructed Sentence
0.2	She(36), murdered(20), her(57), New(17), office(1), just(51), days(33), after(23), learning(62), that(39), Wait(10), ress(10), had(59), accepted(11), into(59), Sund(10), ance(10), Film(10), Festival(10)	She said murdered of her New York office, just days after learning that wanderingress had been accepted into the Sundance Film Festival.
0.3	murdered(59), her(63), office(7), just(63), days(49), learning(49), Wait(7), ress(43), had(24), accepted(24), Sund(43), ance(43), Film(43), Festival(43)	The U.S. officials have just just days of learning the countryress had been accepted had been a Sundance Film Festival, the
0.4	murdered(9), learning(63), Wait(9), ress(50), accepted(63), Sund(63), ance(63), Film(9), Festival(9)	The U.S. President Barack Obama, the U.S. akaress, and the accepted the Sundance Film Festival.

Table 11: The latent typing and the reconstruction results of the input tokens, "She was murdered in her New York office, just days after learning that Waitress had been accepted into the Sundance Film Festival.", with different models pre-trained from scratch with different values of the weighting factors β. The corresponding latent type indices for each of the tokens are noted in the parentheses.

Top-10 Frequent Type Index v	Top-5 Frequent Tokens Tagged by the Type v
0 (30.55%)	, (14.1%) the (13.8%) . (13.1%) CLS (10.3%) SEP (10.3%)
47 (5.86%)	in (3.2%) and (3.2%) said (1.6%) as (1.1%) out (1.1%)
10 (5.73%)	abortion (2.7%) cost (2.2%) Indigo (1.6%) eta (1.6%) innovative (1.1%)
16 (5.14%)	eta (1.8%) parallel (1.8%) autism (1.8%) MMR (1.8%) Smart (1.2%)
61 (3.2%)	that (7.8%) was (5.9%) a (3.9%) has (2.9%) would (2.9%)
46 (2.76%)	Abdullah (5.7%) Ge (2.3%) Jones (2.3%) measles (2.3%) Minnesota (2.3%)
36 (2.73%)	was (3.4%) that (3.4%) on (3.4%)) (2.3%) she (2.3%)
15 (2.70%)	," (3.5%) told (2.3%) could (2.3%) statistics (2.3%) ((2.3%)
45 (2.54%)	and (3.7%) under (2.5%) Afghan (2.5%) new (2.5%) vaccine (2.5%)
2 (2.23%)	cost (2.8%) ., (1.4%) hopes (1.4%) challenges (1.4%) os (1.4%)

Table 12: The frequencies of the top-10 most selected latent types v and the corresponding top-5 frequent tokens tagged by the type v. The frequencies are computed over 100 randomly sampled sentences from the pre-training corpus and are noted in the parentheses after the tokens. The model is pretrained from scratch on the VOA corpus. We can observe that the zero type (index 0) is mostly associated with less meaningful tokens. We can also observe that the type 46 seems more correlated with the physical locations and the types 47 and 36 are mostly related to functional words. Type 16 is also meaningful as it appears to be related to the discussion of the potential linking of MMR vaccines to autism in children.

α	β	γ	Latent Typing	Reconstructed Sentence
0.05	0.05	0.1	she(58), murdered(24), her(1), new(8), york(42), office(61), just(34), days(11), learning(4), waitress(62), accepted(60), sundance(9), film(63), festival(50)	she was murdered in her new york office, just days after learning that waitress was accepted in her the sundance film festival.
0.05	0	0.1	CLS(46), she(54), was(13), murdered(35), in(47), her(28), new(28), york(32), office(25), ,(26), just(46), days(14), after(31), learning(44), that(4), waitress(44), had(41), been(27), accepted(58), into(41), the(30), sundance(61), film(5), festival(36), .(1)	she was murdered in her new york office, just days after learning that waitress had been accepted into the sundance film festival.
0.05	0.05	0	$\emptyset$	the u. s. military chief said.
0.05	0.0	0	CLS(38), she(4), was(4), murdered(4), in(4), her(4), new(4), york(4), office(4), ,(4), just(4), days(4), after(4), learning(4), that(4), waitress(4), had(4), been(4), accepted(4), into(4), the(4), sundance(4), film(4), festival(4), .(4), SEP(48)	she was murdered in her new york office, just days after learning that waitress had been accepted into the sundance film festival.

Table 13: The latent typing and the reconstruction results of various *BERT-SparseLT* models continually pretrained with different α, β and γ values. The input sentence is "She was murdered in her New York office, just days after learning that Waitress had been accepted into the Sundance Film Festival.". From the table, we can see that removing the typing sparsity loss ($\beta = 0$) will result the model to select all the input tokens, and removing the KL-divergence term ($\gamma = 0$) will cause the model to either not select any tokens (assigning all the tokens as the zero type) or have almost the same latent type (e.g. type 4) for all the input tokens. The corresponding latent type indices for each of the tokens are noted in the parentheses.

Input Tokens	Latent Typing	Reconstructed Sentence
	In-domain Sentences	
A 45-year-old man who was tackled down and arrested by police after allegedly attacking officers with a knife in South Roxana, Illinois, has been charged by the local prosecutors.	45(36), year(48), old(16), man(35), who(8), was(52), tackled(37), down(59), arrested(19), police(36), after(28), allegedly(14), attacking(48), officers(60), with(5), knife(7), sou th(4), ro(62), ##xa(48), ##na(48), illinois(20), has(50), charged(49), local(39), prosecutors(38),	a 45 - year - old man who was tackled down and arrested by police after allegedly attacking officers with a knife in south roxana, illinois, has been charged at the local prosecutors.
Natural language processing (NLP) is a subfield of linguistics, computer science, and artificial intelligence concerned with the interactions between computers and human language, in particular how to program computers to process and analyze large amounts of natural language data.	natural(20), language(24), processing(54), nl(29), ##p(42),)(43), sub(34), ##field(61), linguistics(29), computer(29), science(15), artificial(59), intelligence(59), concerned(12), interactions(19), between(34), computers(20), human(28), language(21), particular(15), how(38), program(6), computers(61), process(32), analyze(13), large(53), amounts(48), natural(25), language(10), data(21),	natural language processing (nlp) is a subfield of linguistics, computer science and artificial intelligence concerned about the interactions between computers and human language, in particular how to program computers to process to process and analyze large amounts of natural language data.
	Out-of-domain Sentences	
The objective of sparse latent typing is to find pairs of latent types and token representations that are as compact as possible but still contain the necessary information for reconstructing the original input sentences.	objective(30), sparse(27), late(48), ##nt(42), typing(23), find(60), pairs(63), late(29), ##nt(4), types(53), token(53), representations(6), as(49), compact(59), as(58), possible(20), but(58), still(20), contain(48), necessary(20), information(35), rec(36), ##ons(42), ##tructing(44), original(6), input(27), sentences(44)	the objective of sparse latent typing is to find pairs of latent types and token representations, as compact as possible but still contain the necessary information for reconstructing the original input sentences.
Our approach provides the decoder model with a shortcut to directly access the encoded token representations, so that the latent representation for each of the input tokens can be learned as an auxiliary type representation.	our(20), approach(20), provides(48), deco(19), ##der(13), model(27), with(16), short(49), ##cut(61), directly(18), access(48), encoded(25), token(53), representations(6), so(2), that(59), late(49), ##nt(4), representation(22), each(26), input(25), token(53), can(41), learned(38), as(58), auxiliary(32), type(30), representation(53)	our approach provides the decoder model with a shortcut to directly access the encoded token representations, so that the latent representation of each of the input tokens can be learned as an auxiliary type representation representation.

Table 14: Sample latent typing and sentence reconstruction results of the continually pretrained *BERT-SparseLT* model for both the in-domain and the out-of-domain sentences. The in-domain sentences are sampled from the VOA corpus and the Wikipedia, while the out-of-domain sentences are from the main content of this paper.