Joint Entity and Relation Extraction Based on Table Labeling Using Convolutional Neural Networks

Youmi Ma **Tatsuya Hiraoka** **Naoaki Okazaki**
Tokyo Institute of Technology
{youmi.ma, tatsuya.hiraoka}@nlp.c.titech.ac.jp
okazaki@c.titech.ac.jp

Abstract

This study introduces a novel approach to the joint extraction of entities and relations by stacking convolutional neural networks (CNNs) on pretrained language models. We adopt table representations to model the entities and relations, casting the entity and relation extraction as a table-labeling problem. Regarding each table as an image and each cell in a table as an image pixel, we apply two-dimensional CNNs to the tables to capture local dependencies and predict the cell labels. The experimental results showed that the performance of the proposed method is comparable to those of current state-of-art systems on the CoNLL04, ACE05, and ADE datasets. Even when freezing pretrained language model parameters, the proposed method showed a stable performance, whereas the compared methods suffered from significant decreases in performance. This observation indicates that the parameters of the pretrained encoder may incorporate dependencies among the entity and relation labels during fine-tuning.

1 Introduction

The purpose of a joint entity and relation extraction is to recognize entities and relations in a text. A task can be decomposed into two subtasks: named entity recognition (NER) and relation extraction (RE). In recent years, several researchers have built high-performance NER and RE systems based on contextualized representations (Yan et al., 2021; Zhong and Chen, 2021; Wang and Lu, 2020; Eberts and Ulges, 2020; Lin et al., 2020). These contextualized representations obtained from pretrained language models, such as bidirectional encoder representations from transformers (BERT) Devlin et al., 2019, have significantly improved the performance for various NLP tasks. As a result, studies on NER and RE have focused on the design of task-specific layers stacked on top of pretrained language models.

A common idea is to formulate NER and RE as table-filling problems (Miwa and Sasaki, 2014). The core concept is to extract entities and relations by filling a table with entity labels in the diagonal cells and relation labels in the off-diagonal cells. Based on this concept, Ma et al. (2022) proposed TablERT, which is a combined system of NER and RE based on a pretrained BERT. TablERT predicts the diagonal cells sequentially and off-diagonal cells simultaneously. Although the system is simple and effective, it ignores the dependencies among predicted relation labels. As noted in Ma et al. (2022), this does not improve the performance with label dependencies incorporated through refined decoding orders.

We propose TablERT-CNN, a novel NER and RE system that encodes the dependencies among the cells within the table. Our method employs two-dimensional convolutional neural networks (2D-CNNs), which are widely used neural architectures for object detection (Krizhevsky et al., 2012). We considered each table as a 2D image and each cell as a pixel, transforming the task into a table-labeling problem at the cell level. By applying 2D-CNNs to the output of BERT, the system is expected to implicitly perceive local information and label dependencies from neighboring cells. Notably, the range of cells to be processed is expandable by stacking multiple CNN layers, we model the dependencies among distant cells.

We evaluated TablERT-CNN based on multiple benchmarks: CoNLL04 (Roth and Yih, 2004), ACE05 (Walker et al., 2006), and ADE (Gurulingappa et al., 2012). The experimental results showed that the performance of the proposed method is on par with those of current state-of-art systems. We hypothesized that parameter updates during fine-tuning helped the BERT encoder capture the necessary dependencies for label predictions; thus, incorporating dependencies using the CNN became less helpful. To verify this hy-

Proceedings of the Sixth Workshop on Structured Prediction for NLP, pages 11 - 21

pothesis, we compared the performance of several NER and RE systems while keeping the BERT parameters frozen and updating them during fine tuning. In addition, we used different layers from which the prediction model extracts token embeddings to analyze how parameter updates within each layer contribute to the performance. As a result, TablERT-CNN still performed well while keeping the BERT parameters unchanged, whereas the performance of the other systems significantly decreased. This observation indicates the ability of the BERT architecture to consider token- and label-wise dependencies during task-specific fine tuning. The source code for the proposed system is publicly available at https://github.com/YoumiMa/TablERT-CNN.

2 Related Work

2.1 NER and RE Using Contextualized Representations

BERT and its variants have recently achieved significant performance improvements on various NLP tasks (Devlin et al., 2019; Lan et al., 2020). These transformer-based (Vaswani et al., 2017) encoders learn syntactic and semantic languages, generating a contextualized representation of each input token (Jawahar et al., 2019; Rogers et al., 2020). Owing to the superiority of BERT encoders, recent studies on NER and RE have tended to focus on the design of a good prediction model that fully utilizes BERT embeddings to further improve the performance.

Promising and straightforward prediction models for NER and RE have been developed. Eberts and Ulges (2020) proposed SpERT, which employs span-level representations obtained from BERT encoders for linear classification based on a negative sampling strategy during training. In addition, Zhong and Chen (2021) introduced a pipelined system, which performs span-based NER similarly to that of SpERT but re-encodes the input sentence using BERT to perform RE. In the RE model, the context and predicted entity labels are jointly encoded, enabling the computation of token-label attention. These approaches rely mainly on parameter updates in the BERT encoder during fine-tuning, where the encoder learns to capture task-specific dependencies. This study compares our system with SpERT to distinguish the dependencies captured by the encoder from those captured by the prediction model.

Some studies have used NER and RE for generative NLP tasks. Li et al. (2019) cast NER and RE as a multiturn question-answering problem. They designed natural language question templates whose answers specify the entities and relations within each sentence. In addition, Paolini et al. (2021) tackled structured language prediction tasks as sequence-to-sequence translations between augmented languages. Structural information can be extracted by postprocessing the target augmented language. Huguet Cabot and Navigli (2021) followed their idea and built a translation system that auto-regressively generates linearized relation triplets, considering an input text. These approaches utilize the attention mechanism within the transformer to capture long-range dependencies; however, they tend to be computationally burdensome. Inspired by their study, we have explored ways to incorporate token and label dependencies into the prediction model. However, our goal is to develop a mechanism that is more explainable and computationally efficient.

Another common approach is building a directed graph, modelling entity with spans as nodes and relations as arcs. Luan et al. (2019) and Wadden et al. (2019) focused on information propagation among span pairs to obtain effective span representations for a prediction. Based on their study, Lin et al. (2020) explicitly modeled cross-task and cross-instance dependencies by introducing a predefined set of global features. Instead of manually defining the global features, Ren et al. (2021) introduced a text-to-graph extraction model that automatically captures global features based on the auto-regressive generation process of a graph. These approaches are delicately designed to involve graph propagation and beam search strategies, resulting in a relatively high complexity.

Formulating the task as a table-filling problem is also a common idea (Miwa and Sasaki, 2014; Gupta et al., 2016; Zhang et al., 2017). Efforts have recently been made to incorporate BERT into this framework. Wang and Lu (2020) designed separate encoders for entities and relations. To use word-word interactions captured within the BERT model, the authors leveraged the attention weights computed from BERT into a relation encoder. Yan et al. (2021) applied a partition filter to divide neurons into multiple partitions and generated task-specific features based on a linear combination of these partitions. Moreover,Ma et al. (2022) built

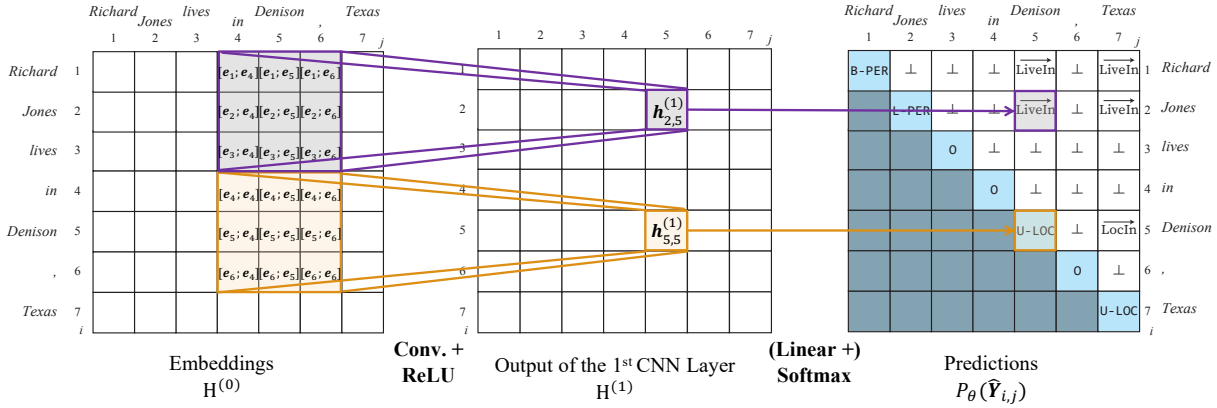


Figure 1: Overview of the proposed TablERT-CNN method when setting the number of CNN layers to 1. An example of a table-filling representation is shown on the right side. We employ the entire table to represent the features and the upper triangular part to represent the labels.

the straightforward TablERT system by predicting entities sequentially and relations simultaneously. Although the model exhibited state-of-art performance when published, it was unable to leverage the interactions among the table cells, especially for RE. This study applies their system as a strong baseline and explores the effect of incorporating local dependencies at the top of BERT. Because the proposed system is an extension of TablERT, we recap this approach (Ma et al., 2022) in the following subsection.

2.2 TablERT

TablERT (Ma et al., 2022) is a simple and effective method for a combined system applying both NER and RE. As shown on the right side of Figure 1, the method uses the upper triangular part of a table to represent the label spaces of NER and RE. The diagonal entries of the table are filled by entity labels, adopting the BILOU scheme to identify the **b**eginning, **i**nside, **l**ast words of multi-word and **u**nit-length spans (Ratinov and Roth, 2009). The off-diagonal entries in the table are filled with relation labels, with directions hard-coded onto each label. For each entity, the corresponding relation is annotated for all component words. For the sentence shown in Figure 1, the relation "LiveIn" pointing from "Richard Jones" to "Denison" is labeled as $\overrightarrow{\text{LIVEIN}}$ for the entries $(i = 1, j = 5)$ and $(i = 2, j = 5)$, corresponding to (*Richard*, *Denison*) and (*Jones*, *Denison*), respectively.

Ma et al. (2022) designed two separate prediction models for NER and RE. For NER, they sequentially assign a label to each word using features at the current and previous timesteps. For RE, they concatenate word embeddings with their corresponding entity label embeddings as relation embeddings. The relation scores of each word pair are computed based on a matrix multiplication of the linearly transformed relation embeddings.

Despite its simplicity, TablERT has shown promising performance. However, the system predicts the relation labels simultaneously, discarding label dependencies between the table cells. It has been reported that the performance of TablERT has shown little improvement, even when the off-diagonal cells are decoded individually following a predefined order Ma et al. (2022). In this study, we are interested in the effect of incorporating label dependencies at the top of contextualized representations. In contrast to Ma et al. (2022), we address this problem using 2D-CNNs.

3 Proposed Method

3.1 Problem Formulation

The goal of NER and RE systems is to extract entities and relations between pairs of entities, based on word sequences. Specifically, we consider a sentence $w_1, \cdots, w_N$. NER aims to identify every word span $s_i = w_b, \cdots, w_e$ that forms an entity with entity type $t_i \in \mathcal{E}$. By contrast, RE aims to extract every relation triple $(s_0\langle t_0\rangle, r, s_1\langle t_1\rangle)$, where $r \in \mathcal{R}$ represents the relation type between s_0 and s_1. Here, $\mathcal{E}$ and $\mathcal{R}$ represent the label sets of entities and relations, respectively.

3.2 TablERT-CNN

We propose TablERT-CNN as an extension of TablERT (Ma et al., 2022), considering the dependencies among labels by applying 2D-CNNs. Fig-

ure 1 shows an overview of TablERT-CNN under a setting in which the prediction model contains only one CNN layer. Based on existing studies (Miwa and Sasaki, 2014; Gupta et al., 2016; Zhang et al., 2017; Ma et al., 2022), we use the upper triangular part of a table to represent the entity and relation labels. The table representation is formally defined as follows:

Table Representation We define a matrix $\boldsymbol{Y} \in \mathbb{R}^{N \times N}$ and use the upper triangular part to represent the label space of NER and RE. A diagonal entry $\boldsymbol{Y}_{i,i}$ represents the entity label of word w_i, and an off-diagonal entry $\boldsymbol{Y}_{i,j} (j > i)$ represents the relation label of the word pair (w_i, w_j). We adopt the labeling rules of NER and RE, as in Ma et al. (2022); i.e., we annotate an entity using the BILOU notation and annotate a relation to every composing word of an entity span, with the direction hard-encoded into the label.

Word Embeddings We obtain word embeddings from contextualized representations generated from the pretrained BERT model (Devlin et al., 2019). Based on the existing study, we compute the embedding of each word by max-pooling its composing sub-words (Liu et al., 2019; Eberts and Ulges, 2020; Ma et al., 2022). Specifically, for word w_i composed of subwords $\text{start}(i), \cdots, \text{end}(i)$, the embedding of $\boldsymbol{e}_i$ is computed as follows:

$$\boldsymbol{e}_i := \max(\boldsymbol{x}^{(l)}_{\text{start}(i)}, \cdots, \boldsymbol{x}^{(l)}_{\text{end}(i)}). \quad (1)$$

Here, $\boldsymbol{x}^{(l)} \in \mathbb{R}^{d_{\text{emb}}}$ is the output of the pretrained BERT model, where l is the layer index[1], d_{emb} is the dimension size, and $\max(\cdot)$ is the max-pooling function. Therefore, we obtain $\boldsymbol{e}_i \in \mathbb{R}^{d_{\text{emb}}}$.

Prediction Model We adopt a 2D-CNN to capture the local dependencies among neighboring cells. 2D-CNNs are widely used for extracting image-classification and object-detection features (Krizhevsky et al., 2012). To apply a 2D-CNN to jointly extract entities and their relations, we treat the 2D table as an image and each cell within the table as a pixel. We then employ the 2D-CNN to encode the representation of each cell, as shown in Figure 1. The convolution network enables the model to capture local dependencies, and for each cell, a 2D-CNN layer yields a weighted linear combination among all surrounding cells within the convolutional window. The dependency range can be extended by stacking multiple 2D-CNN layers.

Specifically, for each word pair (w_i, w_j), we concatenate the word embeddings $\boldsymbol{e}_i, \boldsymbol{e}_j$, and construct the bottom layer $\mathsf{H}^{(0)} \in \mathbb{R}^{N \times N \times 2d_{\text{emb}}}$ (i.e., layer 0) of the 2D-CNN.

$$\mathsf{H}^{(0)}_{i,j,:} = \boldsymbol{h}^{(0)}_{i,j} := [\boldsymbol{e}_i;\ \boldsymbol{e}_j]. \quad (2)$$

Here, $[\cdot;\cdot]$ represents the concatenation of two vectors. Hence, the representation of each cell is a vector of dimension $2 \times d_{\text{emb}}$, which is denoted as d_0. Similarly, we denote the dimension number of the vector representation for each cell in layer l as d_l.

We then compute the output of the first 2D-CNN layer $\mathsf{H}^{(1)}$ based on the output of the bottom layer $\mathsf{H}^{(0)}$. Analogously, the output of any layer l can be computed by applying convolutions to the output of the previous layer, $l-1$. For any word pair (w_i, w_j), we obtain its corresponding output at the lth layer $\mathsf{H}^{(l)}_{i,j,:} = \boldsymbol{h}^{(l)}_{i,j} \in \mathbb{R}^{d_l}$ as follows:

$$\mathsf{H}^{(l)}_{i,j,:} = \boldsymbol{h}^{(l)}_{i,j} := \boldsymbol{b}^{(l)} + \sum_{c=0}^{d_{l-1}} (\mathsf{K}^{(l)}_{c,:,:} * \mathsf{H}^{(l-1)}_{:,:,c})_{i,j}, \quad (3)$$

where $\mathsf{H}^{(l-1)} \in \mathbb{R}^{N \times N \times d_{l-1}}$ is the output of layer $l-1$, $\mathsf{K}^{(l)} \in \mathbb{R}^{d_l \times d_h \times d_w}$ is a convolution kernel with window size $d_h \times d_w$, and $\boldsymbol{b}^{(l)} \in \mathbb{R}^{(l)}$ is the bias. Thus, for any dimension (i.e., channel) c, we have $\mathsf{K}^{(l)}_{c,:,:} \in \mathbb{R}^{d_h \times d_w}$ and $\mathsf{H}^{(l-1)}_{:,:,c} \in \mathbb{R}^{N \times N}$. $\boldsymbol{A} * \boldsymbol{B}$ represents the operation of computing 2D correlations. Given that $\boldsymbol{A} \in \mathbb{R}^{(2\alpha+1) \times (2\beta+1)}$, the computation is defined as follows:

$$(\boldsymbol{A} * \boldsymbol{B})_{m,n} := \sum_{h=-\alpha}^{\alpha} \sum_{w=-\beta}^{\beta} A_{\alpha+h,\beta+w} B_{m+h,n+w}. \quad (4)$$

The last layer of the 2D-CNN is a convolutional classifier for RE. That is, for the last layer L, we set its output dimension number to be the same as the number of relation labels; i.e., $d_L := |\mathcal{R}|$. Thus, for each word pair (w_i, w_j) where $i \neq j$, we obtain the relation label distribution $P_\theta(\hat{\boldsymbol{Y}}_{i,j})$ by applying a softmax function to $\mathsf{H}^{(L)}_{i,j,:}$:

$$P_\theta(\hat{\boldsymbol{Y}}_{i,j}) := \text{softmax}(\mathsf{H}^{(L)}_{i,j,:}), \quad (5)$$

[1]Previous studies have adopted the top layer (Li et al., 2019; Wadden et al., 2019; Eberts and Ulges, 2020) or the average of the top three layers (Wang and Lu, 2020) to generate word representations.

where P is the estimated probability function, and θ represents the model parameters.

For NER, we linearly transform the representations of the diagonal cells at layer L to compute the entity label distribution of each word w_i.

$$P_\theta(\hat{\boldsymbol{Y}}_{i,i}) := \mathrm{softmax}(\boldsymbol{W} \cdot \mathsf{H}_{i,i,:}^{(L)} + \boldsymbol{b}), \quad (6)$$

where $\boldsymbol{W} \in \mathbb{R}^{|\mathcal{E}| \times |\mathcal{R}|}$ and $\boldsymbol{b} \in \mathbb{R}^{|\mathcal{E}|}$ are the trainable weight matrix and the bias vector, respectively.

Training and Prediction During training, we use the sum of cross-entropy losses of NER and RE as the objective function. Given the ground-truth label matrix of table $\boldsymbol{Y} \in \mathbb{R}^{N \times N}$, we compute the cross-entropy loss for NER ($\mathcal{L}^{\mathrm{NER}}$) and RE ($\mathcal{L}^{\mathrm{RE}}$).

$$\mathcal{L}^{\mathrm{NER}} = -\sum_{1 \le i \le N} \log P_\theta(\hat{\boldsymbol{Y}}_{i,i} = Y_{i,i}), \quad (7)$$

$$\mathcal{L}^{\mathrm{RE}} = -\sum_{\substack{1 \le i \le N \\ i < j \le N}} \log P_\theta(\hat{\boldsymbol{Y}}_{i,j} = Y_{i,j}). \quad (8)$$

We minimize $\mathcal{L}^{\mathrm{NER}} + \mathcal{L}^{\mathrm{RE}}$ to update the model parameters θ.

To predict the entity label of each word w_i, we select the label yielding the highest probability from $P_\theta(\hat{\boldsymbol{Y}}_{i,i})$ as the predicted result. When a conflict occurs with regard to the entity type within an entity span, we select the entity type labeled to the last word as the final prediction. To predict the relation label for each entity pair (s_i, s_j), we select the last words of both entity spans to represent the corresponding span s_i, s_j. For example, supposing the last word of entity span s_i, s_j is indexed as $\mathrm{end}(i), \mathrm{end}(j)$, the predicted relation label for entity pair (s_i, s_j) is determined as the label yielding the highest probability from $P_\theta(\hat{\boldsymbol{Y}}_{\mathrm{end}(i),\mathrm{end}(j)})$.

4 Experiments

4.1 Datasets

We evaluated the performance of our proposed system on CoNLL04 (Roth and Yih, 2004), ACE05 (Walker et al., 2006), and ADE (Gurulingappa et al., 2012), the statistics of which are listed in Table 1. Based on the conventional evaluation scheme for CoNLL04 and ACE05, we measured the micro F1-scores, and for ADE, we measured the macro F1-scores.

CoNLL04 is an annotated corpus collected from newswires. We processed the data released by Eberts and Ulges (2020)[2] to obtain the BILOU notations of the entities. Thus, our data split is the same as that in Eberts and Ulges (2020).

Dataset	# Sentences			$\lvert\mathcal{E}\rvert$	$\lvert\mathcal{R}\rvert$
	train	dev	test		
CoNLL04	922	231	288	4	5
ACE05	10,051	2,424	2,050	7	6
ADE	4,272 (10-fold)			2	1

Table 1: Statistics of each dataset used in this study.

ACE05 is an annotated corpus collected from various sources, including newswires and online forums. We used the data preprocessing scripts provided by Wadden et al. (2019)[3] and Luan et al. (2019), which inherits that of Miwa and Bansal (2016)[4]. After preprocessing, an entity is regarded as correct if its label and head region are identical to the ground truth.

Adverse **D**rug **E**ffect (**ADE**, Gurulingappa et al., 2012) is a corpus constructed based on the medical reports of drug usages and their adverse effects. Based on existing studies (Eberts and Ulges, 2020; Wang and Lu, 2020), we removed overlapping entities from the dataset, which comprises only 2.8% of the total number of entities.

4.2 Experimental Settings

We implemented the proposed system using PyTorch (Li et al., 2020) and applied the pretrained BERT model provided by the Huggingface libraries (Wolf et al., 2020). Except for those within the pretrained BERT model, the parameters were randomly initialized. During training, we adopted the AdamW algorithm (Loshchilov and Hutter, 2019) for parameter updates. The details of hyperparameters are listed in Appendix A.

All experiments were conducted on a single GPU of an NVIDIA Tesla V100 (16 GiB). Throughout this study, we report the average values of 5 runs with different random seeds for all evaluation metrics.

4.3 Main Results

The main results of the proposed method are presented in Table 2. We adopted TablERT (Ma et al., 2022) for the primary comparison and trained the system from scratch with different pretrained en-

[2] https://github.com/lavis-nlp/spert
[3] https://github.com/dwadden/dygiepp
[4] https://github.com/tticoin/LSTM-ER

Dataset	Method	Encoder	NER	RE	RE+
CoNLL04△	SpERT (Eberts and Ulges, 2020)	$\text{BERT}_{\text{BASE}}$	88.9	-	71.5
	Table-Sequence (Wang and Lu, 2020)	$\text{ALBERT}_{\text{XXLARGE}}$	90.1	**73.8**	73.6
	TablERT (Ma et al., 2022)	$\text{BERT}_{\text{BASE}}$	90.2	72.8	72.6
	TablERT (Ma et al., 2022)	$\text{BERT}_{\text{LARGE}}$	**90.5**	**73.8**	**73.8**
	TablERT-CNN (Ours)	$\text{BERT}_{\text{BASE}}$	**90.5**	73.2	73.2
ADE▲	SpERT (Eberts and Ulges, 2020)	$\text{BERT}_{\text{BASE}}$	89.3	-	79.2
	Table-Sequence (Wang and Lu, 2020)	$\text{ALBERT}_{\text{XXLARGE}}$	89.7	80.1	80.1
	PFN (Yan et al., 2021)	$\text{BERT}_{\text{BASE}}$	89.6	-	80.0
	PFN (Yan et al., 2021)	$\text{BERT}_{\text{LARGE}}$	**91.3**	–	**83.2**
	TablERT(Ma et al., 2022)	$\text{BERT}_{\text{BASE}}$	89.9	**80.6**	80.6
	TablERT-CNN (Ours)	$\text{BERT}_{\text{BASE}}$	89.7	80.5	80.5
ACE05△	DyGIE++ (Wadden et al., 2019)	$\text{BERT}_{\text{BASE}}$	88.6	63.4	-
	Table-Sequence (Wang and Lu, 2020)	$\text{BERT}_{\text{LARGE}}$	88.2	67.4	-
	Table-Sequence (Wang and Lu, 2020)	$\text{ALBERT}_{\text{XXLARGE}}$	89.5	67.6	64.3
	PURE (Zhong and Chen, 2021)	$\text{BERT}_{\text{BASE}}$	88.7	66.7	63.9
	PURE (Zhong and Chen, 2021)	$\text{BERT}_{\text{XXLARGE}}$	89.7	**69.0**	65.6
	PFN (Yan et al., 2021)	$\text{ALBERT}_{\text{XXLARGE}}$	89.0	-	**66.8**
	TablERT (Ma et al., 2022)	$\text{BERT}_{\text{BASE}}$	87.6	66.2	62.6
	TablERT (Ma et al., 2022)	$\text{BERT}_{\text{LARGE}}$	88.4	67.5	64.6
	TablERT (Ma et al., 2022)	$\text{ALBERT}_{\text{XXLARGE}}$	**89.8**	67.7	65.2
	TablERT-CNN (Ours)	$\text{BERT}_{\text{BASE}}$	87.8	65.0	61.8

Table 2: Comparison between the existing and the proposed method (TablERT-CNN). Here, △ and ▲ denote the use of micro-and macro-average F1 scores for evaluation, respectively. The results of TablERT are our replications, and the results of the others are reported values from the original papers. To ensure a fair comparison, the reported values of PURE follow the single-sentence setting.

coders[5].

We evaluated the RE performance based on two criteria: `RE` and `RE+`. Specifically, `RE`regards each predicted relation triple as correct if the relation label and spans of both entities are identical to the ground truth, whereas `RE+` requires the labels of both entities to be correct. Because comparing systems using different encoders is unfair, we discuss the condition in which the encoders are aligned.

With regard to the CoNLL04 and ADE datasets, we observed that TablERT-CNN achieved high and stable performance on all datasets, on par with that of TablERT. In particular, for CoNLL04, the performance of the proposed method surpassed TablERT for both NER and RE. One possible explanation for this performance gain is that CoNLL04 is a relatively small dataset, as listed in Table 1. Such a low-resource setting possibly brought out the advantage of TablERT-CNN, as the CNN layers helped to utilize rich information about dependencies among entities and relations.

However, regarding the ACE05 dataset, we did not observe any performance gain by stacking the CNN layers. As listed in Table 2, TablERT-CNN lagged its competitor TablERT for around 1.0 point on the F1 score of RE. The reason for this can be multifactorial, and the nature of the ACE05 dataset might provide an answer. The dataset contains entities that do not contribute to any relation triple, which significantly confuses the model during the RE.

5 Analysis

Although our system exhibited a good performance based on multiple datasets, no significant improvement was observed against TablERT (Ma et al., 2022). We hypothesize that the reason for this is the parameter updates within the BERT encoder during fine-tuning, which overshadowed the ability of the CNNs in the prediction model. Self-attention modules within BERT potentially learn to encode the dependencies between word pairs during fine-tuning, overlapping with those captured by the CNNs.

Experiments were conducted to verify this hypothesis. Specifically, we trained multiple BERT-based NER and RE systems (i.e., systems using a pretrained BERT as the encoder) while freezing the BERT parameters (§ 5.1). In this manner, we prevented the encoder from obtaining task-specific features during fine-tuning. The performance of these encoder-frozen systems was compared with that of their counterparts, whose encoder parameters were updated during fine-tuning. Based on this comparison, we investigated the extent to which the parameter updates within BERT contribute to

[5]The code is available at `https://github.com/YoumiMa/TablERT`.

Method	Parameter Updates	Encoder Layer 0	1	2	4	6	8	10	12
SpERT (Eberts and Ulges, 2020)	No	27.4	30.9	32.1	36.5	41.0	40.6	37.2	8.0
	Yes	51.0	70.7	79.9	85.4	85.9	86.9	86.5	87.7
TablERT (Ma et al., 2022)	No	62.4	68.0	74.8	78.6	81.4	82.0	81.5	80.2
	Yes	66.9	78.2	84.1	87.4	88.7	88.2	88.5	88.5
TablERT-CNN (Ours)	No	80.3	81.1	83.1	85.1	86.6	86.2	86.0	85.9
	Yes	80.5	83.7	85.6	87.0	88.4	88.4	88.3	88.0

Table 3: Micro-average NER F1 scores on the CoNLL04 development set with/without parameter updates within the encoder ($\text{BERT}_{\text{BASE}}$) during fine-tuning. We fed the hidden states at different encoder layers into the prediction model for task-specific predictions.

Method	Parameter Updates	Encoder Layer 0	1	2	4	6	8	10	12
SpERT (Eberts and Ulges, 2020)	No	3.0	3.3	3.7	4.6	7.8	6.0	5.8	0.0
	Yes	16.4	35.4	49.6	64.7	67.2	69.3	70.2	69.1
TablERT (Ma et al., 2022)	No	28.8	37.4	39.3	47.1	53.0	54.0	55.9	51.7
	Yes	36.0	47.9	60.9	66.5	71.3	70.5	71.0	70.7
TablERT-CNN (Ours)	No	53.5	54.8	57.6	64.4	66.2	67.1	64.4	61.5
	Yes	54.0	59.9	62.3	67.8	70.6	70.3	70.1	70.6

Table 4: Micro-average F1 scores of the RE on the CoNLL04 development set with/without parameter updates within the encoder ($\text{BERT}_{\text{BASE}}$) during fine-tuning. We fed the hidden states at different encoder layers into the prediction model for task-specific predictions.

the performance of NER and RE.

In addition, we are interested in how each BERT layer encodes dependencies that are helpful for NER and RE. Previous studies have utilized the outputs of the top BERT layers to produce word representations (Wadden et al., 2019; Eberts and Ulges, 2020; Ma et al., 2022; Wang and Lu, 2020). However, we are curious whether the bottom or middle BERT layers also store useful information for solving the NER and RE. Therefore, we fed hidden states at the $\{0, 1, 2, 4, 6, 8, 10, 12\}$th BERT layer into the prediction model and examined the difference in performance (§ 5.2). Here, the 0th layer denotes the embedding layer of the BERT encoder.

Our analysis includes SpERT (Eberts and Ulges, 2020), TablERT (Ma et al., 2022) and the proposed method. We included TablERT for comparison because it is a counterpart of our system, incorporating no dependencies while performing RE. We included SpERT for comparison because it is a strong baseline utilizing a pretrained BERT encoder. Systems were trained on the CoNLL04 (Roth and Yih, 2004) training set and evaluated on the development set, using $\text{BERT}_{\text{BASE}}$ (Devlin et al., 2019) as the encoder. The experimental results are listed in Tables 3 and 4. The plots corresponding to these results are presented in Appendix B. Finally, we analyze the effect of 2D-CNNs (§ 5.3).

5.1 Effect of Parameter Updates within BERT

As listed in Tables 3 and 4, while freezing the parameters within BERT, we observed a decrease in the performance of both NER and RE for all target systems. SpERT exhibits a drastic decrease in performance while disabling the parameter updates within the encoder. This observation suggests that the system relies heavily on parameter updates of the encoder during task-specific fine-tuning to solve specific tasks.

By contrast, TablERT-CNN exhibited the best performance among the target systems, even with BERT parameters frozen. This result indicates that in a situation in which the parameter updates within the encoder are infeasible (e.g., computational resources are limited), TablERT-CNN can be more promising than TablERT or SpERT in terms of achieving high performance.

Furthermore, while freezing the BERT parameters, utilizing the hidden states of the top layers (i.e., layer 10 and higher) hindered the performance of all target systems. This phenomenon corresponds to the study by Rogers et al. (2020), which concluded that the final layers of BERT are usually the most task-specific. For a pretrained BERT encoder without any parameter updates, the top layers of the model are specified to the pretraining task, i.e., the masked-language modeling (MLM) task. It can therefore be assumed that while using the hidden

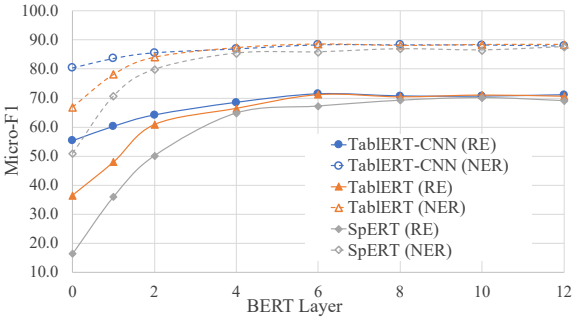


Figure 2: Micro-F1 scores of all target systems while varying the encoder layer whose outputs were fed into the prediction model (CoNLL04 development set).

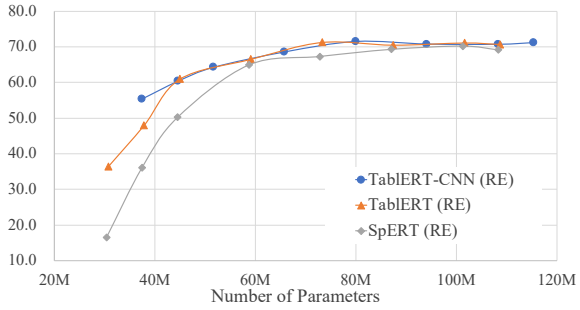


Figure 3: Performance of all target systems while varying the number of trainable parameters, as measured using the RE micro-F1 score (CoNLL04 development set).

states of the top layers of BERT without any task-specific parameter updates, the specificity toward the MLM task adversely affects the performance of the prediction model for both NER and RE.

5.2 Effect of BERT Layer

To visualize the performance change caused by the choice of BERT layer, the hidden states of which were utilized as word embeddings, we plotted the micro-F1 scores of all target systems, as shown in Figure 2.

Incorporating outputs from deeper BERT layers generally improves the prediction of all target systems. The improvement was significant at the bottom layers, but subtle at the top. Specifically, as shown in Figure 2, from layers 0 to 6, we observed a significant boost in the performance of NER and RE for all target systems. The change in performance was more evident with RE than with NER. By contrast, the performance of all target systems remained flat, starting from layer 8. This tendency suggests that, while building a BERT-based NER and RE system, it may be sufficient to employ up to 8 layers for text encoding.

Our findings match those reported by Jawahar et al. (2019), suggesting that BERT encodes a hierarchy of linguistics from bottom to top. Jawahar et al. (2019) found that BERT learns to encode long-distance dependencies, e.g., subject-verb agreements at deeper layers, which possibly explains the significant improvement in the RE performance while using outputs of the deeper BERT layers.

5.3 Effect of 2D-CNNs

As shown in Figure 2, while employing the outputs from the bottom BERT layers (i.e., from layers 0 to 4), TablERT-CNN outperformed the other systems by a relatively large margin. We owe the performance gap to the ability of TablERT-CNN to capture local dependencies. As noted in an existing study, the bottom BERT layers encode the surface information, for example, the phrasal syntax and word order (Jawahar et al., 2019; Rogers et al., 2020). As a result, the outputs at the bottom BERT layers lack contextualized information incorporating long-range dependencies, which are crucial for extracting relations. Therefore, whereas SpERT and TablERT suffer from the absence of word-word interactions, TablERT-CNN overcomes this issue by encoding them in the prediction model. By observing the table representation as a 2D image and each cell as a pixel, our method captures the local dependencies within each convolution kernel using 2D-CNNs. This advantage is apparent when word embeddings are not properly contextualized.

However, the superiority of TablERT-CNN becomes inconspicuous when the depth of the BERT layers increases. This phenomenon indicates that, when the contextualization ability of the encoder improves, the strength of a 2D-CNN to incorporate dependencies diminishes because the encoder has already captured the necessary information.

Notably, although we have shown the superiority of TablERT-CNN when utilizing the bottom BERT layers, it is natural to suspect that the performance gain resulted from the additional parameters introduced by the convolutional layers. Compared with SpERT and TablERT, TalERT-CNN introduces more trainable parameters, thereby increasing the ability of the system to fit the training data. To determine whether the performance gain resulted from the ability of the CNN to capture local dependencies or merely from an increased number of parameters, we replotted Figure 2, as shown in

Figure 3, the result of which shows the relationship between the RE micro-F1 scores and the number of trainable parameters of each target system. From Figure 3, we observed that TablERT-CNN lies on the left-most side among all of the target systems. To paraphrase, when keeping the number of trainable parameters the same, TablERT-CNN performs better than its competitors. This tendency is apparent when the number of trainable parameters is small, which indicates that TablERT-CNN can be a prospective option when the computational resources are limited.

To conclude, TablERT-CNN can be a promising architecture when parameter updates within the encoder are infeasible or when the encoder is not well-contextualized. Under these situations, a 2D-CNN plays an important role in encoding the local dependencies, thus improving the NER and RE predictions.

6 Conclusion

We presented TablERT-CNN, a novel method for jointly extracting entities and relations with 2D-CNNs. The method casts NER and RE as table-labeling problems, representing each table cell as a pixel and each table as a 2D image. By applying 2D-CNNs, the method predicts the label of each table cell to extract entities and relations. Experiments conducted on CoNLL04, ACE05, and ADE demonstrated that TablERT-CNN performed on par with current state-of-art systems when the pretrained encoders were aligned.

To explore why TablERT-CNN did not outperform existing systems by a significant margin, we conducted experiments to compare their performance with and without parameter updates of the BERT encoder during the fine-tuning. We observed that TablERT-CNN performed reasonably well even without updating the encoder parameters, whereas its competitors suffered a decrease in performance. These results indicate that the BERT encoder can capture task-specific dependencies among tokens and labels within its architecture, based on parameter updates during fine-tuning.

In the future, we plan to model the dependencies among table cells using other neural architectures. Prospective directions include 2D-transformers that compute the attention across element pairs in a 2D array, or Routing Transformers that utilize local attentions.

Acknowledgements

This paper is based on results obtained from a project, JPNP18002, commissioned by the New Energy and Industrial Technology Development Organization (NEDO). We appreciate the insightful comments and suggestions of the anonymous reviewers.

References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Markus Eberts and Adrian Ulges. 2020. Span-based joint entity and relation extraction with transformer pre-training. In *24th European Conference on Artificial Intelligence (ECAI)*.

Pankaj Gupta, Hinrich Schütze, and Bernt Andrassy. 2016. Table filling multi-task recurrent neural network for joint entity and relation extraction. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2537–2547, Osaka, Japan. The COLING 2016 Organizing Committee.

Harsha Gurulingappa, Abdul Mateen Rajput, Angus Roberts, Juliane Fluck, Martin Hofmann-Apitius, and Luca Toldo. 2012. Development of a benchmark corpus to support the automatic extraction of drug-related adverse effects from medical case reports. *Journal of Biomedical Informatics*, 45(5):885–892. Text Mining and Natural Language Processing in Pharmacogenomics.

Pere-Lluís Huguet Cabot and Roberto Navigli. 2021. REBEL: Relation extraction by end-to-end language generation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2370–2381, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. What does BERT learn about the structure of language? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3651–3657, Florence, Italy. Association for Computational Linguistics.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations (ICLR)*.

Shen Li, Yanli Zhao, Rohan Varma, Omkar Salpekar, Pieter Noordhuis, Teng Li, Adam Paszke, Jeff Smith, Brian Vaughan, Pritam Damania, and Soumith Chintala. 2020. Pytorch distributed: Experiences on accelerating data parallel training. *Proc. VLDB Endow.*, 13(12):3005–3018.

Xiaoya Li, Fan Yin, Zijun Sun, Xiayu Li, Arianna Yuan, Duo Chai, Mingxin Zhou, and Jiwei Li. 2019. Entity-relation extraction as multi-turn question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1340–1350, Florence, Italy. 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.

Yijin Liu, Fandong Meng, Jinchao Zhang, Jinan Xu, Yufeng Chen, and Jie Zhou. 2019. GCDT: A global context enhanced deep transition architecture for sequence labeling. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2431–2441, Florence, Italy. Association for Computational Linguistics.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*.

Yi Luan, Dave Wadden, Luheng He, Amy Shah, Mari Ostendorf, and Hannaneh Hajishirzi. 2019. A general framework for information extraction using dynamic span graphs. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3036–3046, Minneapolis, Minnesota. Association for Computational Linguistics.

Youmi Ma, Tatsuya Hiraoka, and Naoaki Okazaki. 2022. Named entity recognition and relation extraction using enhanced table filling by contextualized representations. *Journal of Natural Language Processing*, 29(1):187–223.

Makoto Miwa and Mohit Bansal. 2016. End-to-end relation extraction using LSTMs on sequences and tree structures. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1105–1116, Berlin, Germany. Association for Computational Linguistics.

Makoto Miwa and Yutaka Sasaki. 2014. Modeling joint entity and relation extraction with table representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1858–1869, Doha, Qatar. Association for Computational Linguistics.

Giovanni Paolini, Ben Athiwaratkun, Jason Krone, Jie Ma, Alessandro Achille, RISHITA ANUBHAI, Cicero Nogueira dos Santos, Bing Xiang, and Stefano Soatto. 2021. Structured prediction as translation between augmented natural languages. In *International Conference on Learning Representations (ICLR)*.

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL-2009)*, pages 147–155, Boulder, Colorado. Association for Computational Linguistics.

Liliang Ren, Chenkai Sun, Heng Ji, and Julia Hockenmaier. 2021. HySPA: Hybrid span generation for scalable text-to-graph extraction. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 4066–4078, Online. Association for Computational Linguistics.

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. 2020. A primer in BERTology: What we know about how BERT works. *Transactions of the Association for Computational Linguistics*, 8:842–866.

Dan Roth and Wen-tau Yih. 2004. A linear programming formulation for global inference in natural language tasks. In *Proceedings of the Eighth Conference on Computational Natural Language Learning (CoNLL-2004) at HLT-NAACL 2004*, pages 1–8, Boston, Massachusetts, USA. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems (NeurIPS)*, pages 5998–6008.

David Wadden, Ulme Wennberg, Yi Luan, and Hannaneh Hajishirzi. 2019. Entity, relation, and event extraction with contextualized span 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 5784–5789, Hong Kong, China. Association for Computational Linguistics.

Christopher Walker, Stephanie Strassel, Julie Medero, and Kazuaki Maeda. 2006. Ace 2005 multilingual training corpus. *Philadelphia: Linguistic Data Consortium*.

Jue Wang and Wei Lu. 2020. Two are better than one: Joint entity and relation extraction with table-sequence encoders. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1706–1721, Online. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi 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 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.

Zhiheng Yan, Chong Zhang, Jinlan Fu, Qi Zhang, and Zhongyu Wei. 2021. A partition filter network for joint entity and relation extraction. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 185–197, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Meishan Zhang, Yue Zhang, and Guohong Fu. 2017. End-to-end neural relation extraction with global optimization. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1730–1740, Copenhagen, Denmark. Association for Computational Linguistics.

Zexuan Zhong and Danqi Chen. 2021. A frustratingly easy approach for entity and relation extraction. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 50–61, Online. Association for Computational Linguistics.

A Hyper-parameters

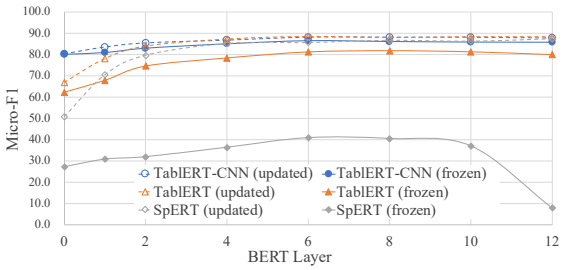
The values of the hyperparameters used during the experiments are listed in Table 5. CNN configurations were determined by conducting grid searches on the development split of each dataset, whereas the training configurations were adopted directly from Ma et al. (2022). We applied a scheduler that linearly increases the learning rate from 0 to the maximum value during the warm-up period and gradually decreases it afterward.

B Effect of Parameter Updates with BERT (cont.)

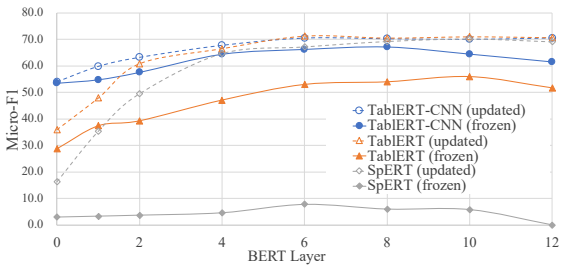
Figures 4(a) and 4(b) correspond to Tables 3 and 4, respectively. As we can see, TablERT-CNN exhibited a relatively high performance, even when the BERT parameters were frozen. In addition, when the BERT parameters were frozen, the performance of all target systems decreased while incorporating the hidden states of the top (10–12) encoder layers.

	CoNLL04	ACE05	ADE
CNN Config.			
kernel size $F_h \times F_w$	3×3	5×5	3×3
# layers L	2	2	3
hidden dim $d^{(l)}$	512	512	512|256
Training Config.			
batch size	8	8	16
Learning rate (BERT)	5e-5	5e-5	5e-5
Learning rate (others)	1e-3	1e-3	1e-3
dropout	0.3	0.3	0.3
warm-up period	0.2	0.2	0.2
# epochs	30	30	30

Table 5: Hyperparameters of our proposed method (TablERT-CNN).



(a) NER micro-F1 scores.



(b) RE micro-F1 scores.

Figure 4: Micro-F1 scores of all target systems while varying the encoder layer whose outputs were fed into the prediction model (CoNLL04 development set). Here, "updated" and "frozen" indicate the status of each parameter within BERT during the fine-tuning process.