ADL Final Project Report
Shared Task Challenge

Po-Heng Chen
Group 10
b07902114@ntu.edu.tw

Wei-Lun Kao
Group 10
b07902027@ntu.edu.tw

Yi-Min Lin
Group 10
b07902016@ntu.edu.tw

Abstract

We choose the shared task challenge as our final project, which is an task that extracts domain specific information from some domain specific documents. In this case, we need to extract information like '調達年度'" from bidding documents of an electric power company. The challenge is that we only received little amount of training/testing data (about 110 documents in total), while the task needs the models to generate detailed and structured output. To tackle such problem, we make use of a mighty pre-trained model BERT and the technique of transfer learning, combined with different network architectures based on BERT, and validate the results by applying the models on the dataset.

1 Introduction

Transfer learning is a technique that plays an important role when domain-specific data is little [4]. Especially in real world application, named entity recognition (NER) tasks are facing with tags that are highly domain-specific (like the dataset for this project). One way to utilize transfer learning to tackle domain-specific NER tasks is to train the network using large scale out-domain NER datasets like CoNLL 2003 [6] and then adapt the network to domains with limited domain-specific data.

A recent research [3] has showed that using BERT [1] as the starting point of transfer learning and combining BERT with other networks for domain-adaption yields impressive results. Inspired by this work, we choose BERT as the basis of transfer learning, and experiment with four approaches. The first two are the combinations of BERT and other networks, which are CNN [5] and Bi-LSTM [2] respectively, and the latter two are the variants of applications of BERT. We finally compare the results on this bidding dataset and discuss the performance of each approach.

2 Approach

2.1 BERT-CNN

2.1.1 Data Preprocess.

Training data. We use BERT-tokenizer to convert ca-data into tokens. Then, we split each tokenized PDF ca-data into k sections, each section has several texts, and the total numbers of tokens in those texts in one section is roughly 384. If total numbers of tokens is too short, model can't catch the meaning of each tokens clearly. After trying many different numbers and referring the reference, we select 384 at the end. Besides, Every two texts in one section are separated with the [SEP] token.

After processing the raw data, we tokenize 20 tags with BERT-tokenizer, and concatenate each tag to every previous step output. For example, if one section is "['品', '目', '分', '類', '番', '号', '２', '## ６','[SEP],'購', '入',...]", we will concatenate each tag with it, as {"['品', '目', '分', '類', '番', '号', '２', '## ６','[SEP],'購', '入',...,'[SEP]','調', '達', '年', '度']", ['[CLS]','品', '目', '分', '類', '番', '号', '２', '## ６','[SEP],'購', '入',...,'[SEP]','都', '道', '府', '県'], ...}. After concatenation, use BERT-tokenizer to convert them into decimal ids.

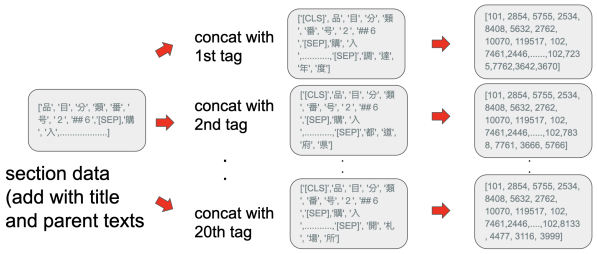

Figure 1. How sections are processed with tags.

Target. The target of the training data is the start/end position of the value corresponding to the tag in that data. If the tag is not relative to that section data, the target will be start position = 0, end position = 0 (span {[CLS], [CLS]}), just as the way we train BERT-QA.

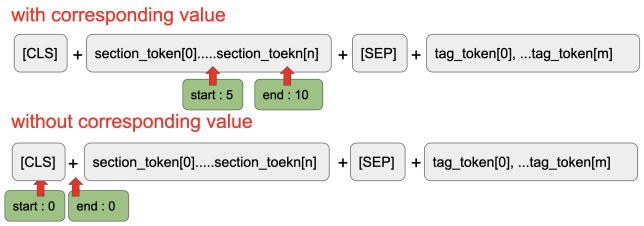

Figure 2. Inputs with targets.

2.1.2 Model.

BERT. Lack of domain specific data, we have to use transfer learning to deal with this task. In addition, owing to the fact that similar or same tokens may have different tag, for example, a date may be tagged with "開札日時", "調達開始

日" or "調達終了日", the model need to know more detailed meaning of each token.

Therefore, multi-layered bidirectional Transformer encoder (BERT),which is pre-trained with large amount of out-domain data, is a great choice we should try. With contextual embedding, BERT can encode all information into high-level hidden vectors, which more precisely represent the meaning of the tokens. After processing all information in section data comprehensively, it can tell the difference between similar values which should be tagged differently.

CNN. Convolution neural network (CNN) is the key of this model, we have tried this approach without CNN layer, and get worsen performance. The reason is that the hidden vectors output by BERT is based on the common text that it was pre-trained. To get better representation vectors toward our domain, we need an additional neural network to learn and extract the detailed information. Deep neural network (DNN) should work but the CNN can do a better job since its more suitable for retaining information using convolution and pooling. By doing this, we can improve our performance substantially.

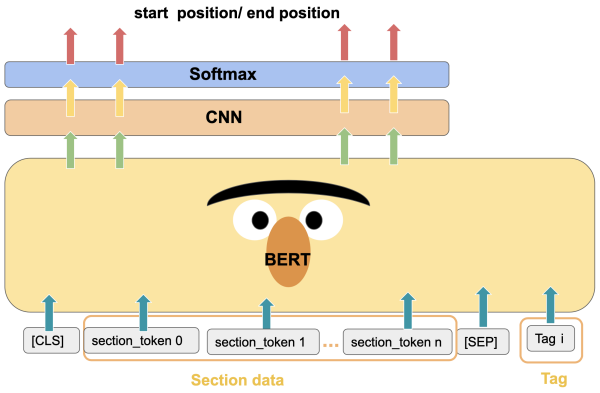

Figure 3. BERT-CNN model.

2.2 BERT-BiLSTM-CRF

2.2.1 Data Preprocess. We use BERT-tokenizer to tokenize data. Let the tokens be $x_1, x_2, ...x_n$, CRF tags corresponding to the tokens be $y_1, y_2, ...y_n$. If $\{x_i, x_{i+1}, x_{i+2}, ...\}$ is a value corresponding to a task_tag$_i$, then we set $\{y_i, y_{i+1}, y_{i+2}, ...\}$ be the crf_tag$_i$. For example, $\{x_i, x_{i+1}, x_{i+2}, ...\}$ = {"平", "成", "29", "年", "9", "月", "22", "日"} is a value of "公告日", then we set $\{y_i, y_{i+1}, y_{i+2}, y_{i+3}, y_{i+4}, y_{i+5}, y_{i+6}, y_{i+7}\}$ = {B-公告日, I-公告日, I-公告日, I-公告日, I-公告日, I-公告日, I-公告日, I-公告日}.

Table 1. Examples of CRF tags

No.	Task tag	CRF tag begin	CRF tag inter
0	None	O	O
1	調達年度	B-調達年度	I-調達年度
2	都道府県	B-都道府県	I-都道府県
3	入札件名	B-入札件名	I-入札件名
⋮	⋮	⋮	⋮

2.2.2 Model.

BERT and Bi-LSTM. We take advantage of the fact that BERT's trained with large amount of data. By using it as the embedding layer, we can get great word vector to represent every token of data. The bidirectional Long Short Term Memory (Bi-LSTM) layer can let us get the embedded token information forward and backward, then combine two direction information and feed them into the CRF layer.

CRF. Given data $(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$ where x_i is the input of CRF and which should be tagged as y_i. Let $x = \{x_1, x_2, ..., x_k\}$, $y = \{y_1, y_2, ..., y_k\}$ Let P be a probability function:

$$P(y|x) = \frac{P(x,y)}{\sum_{y' \in Y} P(x,y)} = \frac{exp(w \cdot \phi(x,y))}{\sum_{y' \in Y} exp(w \cdot \phi(x,y'))}$$

Where w is a vector that can be learned, and ϕ is a function of (x, y), which can be any function that represent relations between tags and values and that between different tags. The ϕ function count the number of $(\hat{x}, \hat{y})$ in x and y, and the number of y_i, y_{i+1}, the consecutive tags pair.

Define the objective function O of w that we want to maximize

$$O(w) = \sum_{n=1}^{N} P(y^n|x^n)$$

The CRF layer can be considered as the relation between tags and values and the relation between tags and tags, so it is often used to solve the NER NLP problem. However, this task's tag are not mutually relative, and there are too many token should be no tag in the data, resulting that CRF is tend to predict all tokens has None tag. Therefore, CRF has very bad performance on this task. (Although predicting all tokens as NONE tag will get 0.87 f1 score, we thought that this score can't really represent the performance of this model. Thus, we don't put Bert-CRF into the table that show f1 score of every model we had tried in Experiment part below)

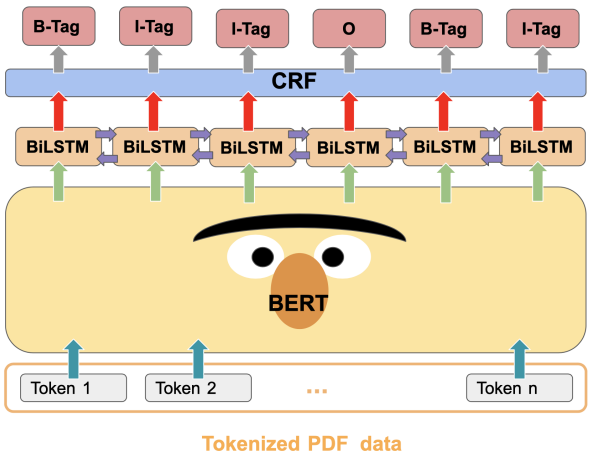

Figure 4. BERT-BiLSTM-CRF model.

2.3 Sentence BERT QA

In the original BERT-CNN, when the same tag exist in the same sample, we can't be sure that whether the training of multiple start/end position in one sample will affect each other. Therefore, a more intuitive strategy would be that, we treat each sentence as a sample, and query start/end on each of them. This way, we can easily obtain whether this sentence contains the corresponding tag.

Preprocess. Basically, it's the same as our first approach - each sample has sentences with each tag concatenated at the tail. That is, let each sentence be $S_i = \{[CLS], s_1^i, ..., s_k^i, [SEP]\}$, a sample be $S = [SOS] + S_1 + ... + S_n + [TAG]$.

The only difference is that, while in the original preprocess strategy, we only give a target per sample, in this new preprocess strategy, we give each sentence in the sample (s_i) a target with the following strategy:

- If the [TAG] exists in S_i with start = 5, end = 10, then we let the target be T_i = (start = 5, end = 10).
- If the [TAG] doesn't exist in S_i, then the target T_i is (start = 0, end = 10).

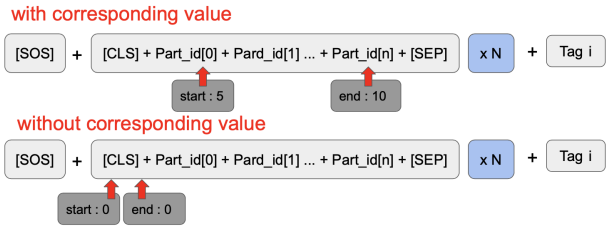

Figure 5. BERT-BiLSTM-CRF model.

Training. When training, we calculate our loss per sample as $L = \sum_{i=1}^{n} CrossEntropy(S_i, T_i)$, like what we did in basic BERT QA.

2.4 BERT + Multilabel Classification

Since all of our work above are based on BERT QA, that is, we query start/end position each sample to find the corresponding tag, we also try treating this problem as a multilabel classification. Note that it can't be a multiclass classification, since a token can be classified into multiple tags at the same time.

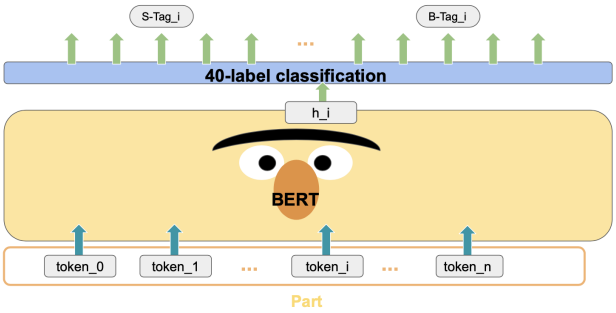

Figure 6. BERT + Multilabel Classification

Each tag has corresponding two class: Start-of-Tag and Body-of-Tag, as figure 6. We have 40 class since there are 20 tags in total.

3 Experiments

Table 2. Comparisons between different models

Model	F1-score
BERT-QA	0.922
BERT-CNN	**0.945**
Sentence-BERT QA	0.913
BERT + Multilabel Classification	0.897

By the above experiment result, we can observe that BERT-CNN outperforms other model, while BERT-QA slightly left behind. Note that in theory, the performance of Sentence-BERT QA should be at least not worse than BERT-QA, by the fact that BERT-QA is the simplified version of Sentence-BERT QA. However, in our experiment, the above result is not observed. We guess the reason is that, in the original BERT QA, if the given tag exists in a sample, then this sample become 'positive' with the tag. However, when preprocessing sample with n sentence using Sentence-BERT QA's, since we treat each sentence as a BERT QA problem, we'll have 1 positive sentence and (n-1) negative sentence, which cause

severe data imbalance problem. Therefore, the performance of Sentence-BERT QA is not as good as expected.

Another interesting fact is that BERT + Multilabel Classification performs worse than other model in our experiment. We think the reason is that, it doesn't fully utilize BERT's strength - output different word embedding based on different context. In the BERT-CNN and other similiar model, we pad the tag with context, so that the BERT and Linear Layer can extract the information needed to specify the given tag together; while in this Multilabel Classification model, we can only rely on linear layers to do all the tag-related information extraction.

4 Conclusion

In this project, we have experimented with four approaches, compared and analyzed the results. BERT is so powerful that transfering the BERT-QA model toward the bidding documents dataset yields results that are good enough. We have also tried stacking BERT with other deep neural networks for more datailed classification, which is shown to outperform the original BERT-QA model. On top of that, we have also experimented with two variants of BERT-QA, and pointed out why the results are not as good as original BERT-QA.

5 Work Distribution

Student name	Part
陳柏衡	Code and PPT
高偉倫	Code and PPT
林義閔	report

References

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv:1810.04805 [cs.CL]

[2] Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF Models for Sequence Tagging. arXiv:1508.01991 [cs.CL]

[3] Minh-Tien Nguyen, Anh Phan, Le Linh, Hong Nguyen, Nguyen Son, Le Dung, Miku Hirano, and Hajime Hotta. 2019. Transfer Learning for Information Extraction with Limited Data.

[4] Juan Diego Rodriguez, Adam Caldwell, and Alexander Liu. 2018. Transfer Learning for Entity Recognition of Novel Classes. In *Proceedings of the 27th International Conference on Computational Linguistics*. Association for Computational Linguistics, Santa Fe, New Mexico, USA, 1974–1985. https://www.aclweb.org/anthology/C18-1168

[5] Hoo-Chang Shin, Holger R. Roth, Mingchen Gao, Le Lu, Ziyue Xu, Isabella Nogues, Jianhua Yao, Daniel Mollura, and Ronald M. Summers. 2016. Deep Convolutional Neural Networks for Computer-Aided Detection: CNN Architectures, Dataset Characteristics and Transfer Learning. *IEEE transactions on medical imaging* 35, 5 (May 2016), 1285–1298. https://doi.org/10.1109/TMI.2016.2528162 26886976[pmid].

[6] Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*. 142–147. https://www.aclweb.org/anthology/W03-0419