ADL Final Project Shared Task (108 Spring)

羅啓帆*
b07902047@ntu.edu.tw
National Taiwan University
Taipei, Taiwan

鄭豫澤
b07902139@ntu.edu.tw
National Taiwan University
Taipei, Taiwan

林庭風
b07902141@ntu.edu.tw
National Taiwan University
Taipei, Taiwan

Applied Deep Learning, June 2020, Taipei, Taiwan

Abstract

Our report presents an approach for information extraction when training data is limited. Our main idea is to leverage the concept of transfer learning, using both pre-trained BERT model and the Delta Reading Comprehension Dataset (DRCD) [1] to deal with limited-data problem. We also used Convolutional Layers instead of classic Linear Layers in our model, and use several training tricks to improve the performance of our model. Our approach is validated by a public data of competitive bids for development projects in Japan. The data is splitted into three parts: train, development, and test (public test & private test), while we only have the answer for train and development. **Our approach can achieve private test f1-score 0.97904[2]**

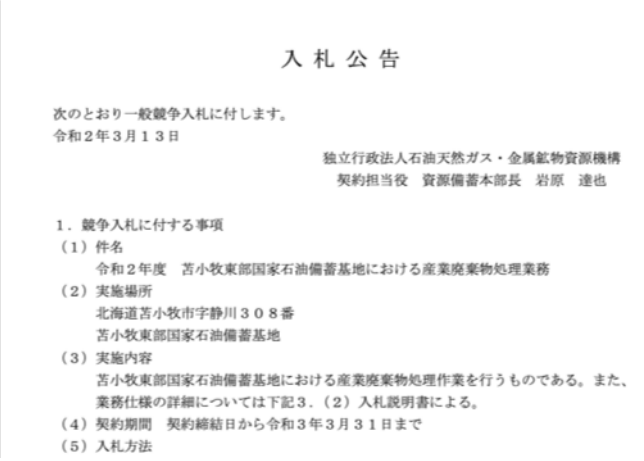
1 Introduction

This section first gives the definition of Information Extraction and Named Entity Recognition, then introduces the dataset used in this report.

Information extraction (IE) is the automated retrieval of specific information related to a selected topic from a body or bodies of text. Information extraction tools make it possible to pull information from text documents, databases, websites or multiple sources [3]. Here, we focus on a subfield called Named Entity Recognition (NER) that seeks out and categorizes specified entities in a body or bodies of texts [4].

The Delta Reading Comprehension Dataset (DRCD) [1] is a Chinese Question Answering (QA) datasets.

Our data used for validating our approach in this report has two format: raw pdf file and processed csv file. The raw pdf file looks like the following:



入 札 公 告

次のとおり一般競争入札に付します。
令和２年３月１３日

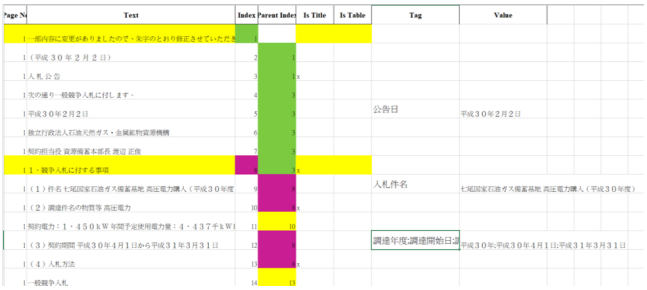
独立行政法人石油天然ガス・金属鉱物資源機構
契約担当役　資源備蓄本部長　岩原　達也

１．競争入札に付する事項
（１）件名
令和２年度　苫小牧東部国家石油備蓄基地における産業廃棄物処理業務
（２）実施場所
北海道苫小牧市字静川３０８番
苫小牧東部国家石油備蓄基地
（３）実施内容
苫小牧東部国家石油備蓄基地における産業廃棄物処理作業を行うものである。また、
業務仕様の詳細については下記３．（２）入札説明書による。
（４）契約期間　契約締結日から令和３年３月３１日まで
（５）入札方法

Fig.1: raw pdf file.

We won't use the raw pdf file in our training or predicting.

The processed csv file looks like the following:



Page No.	Text	Index	Parent Index	Is Title	Is Table	Tag	Value
1	一部内容に変更がありましたので、朱字のとおり修正させていただき	1		Y			
1	（平成３０年２月２日）	2	1				
1	入札公告	3	1				
1	次の通り一般競争入札に付します。	4	1				
1	平成３０年２月２日	5	1			公告日	平成３０年２月２日
1	独立行政法人石油天然ガス・金属鉱物資源機構	6	1				
1	契約担当役 資源備蓄本部長 渡辺 正俊	7	1				
1	１１．競争入札に付する事項	8	1	Y			
1	（１）件名 七尾国家石油ガス備蓄基地 高圧電力購入（平成３０年度	9	8			入札件名	七尾国家石油ガス備蓄基地 高圧電力購入（平成３０年度）
1	（２）調達件名の物質等 高圧電力	10	8				
1	契約電力：１，４５０ｋＷ 年間予定使用電力量：４，４３７千ｋＷｈ	11	8				
1	（３）契約期間 平成３０年４月１日から平成３１年３月３１日	12	8			調達年度;調達開始日;調達終了日	平成３０年;平成３０年４月１日;平成３１年３月３１日
1	（４）入札方法	13	8				
1	一般競争入札	14	13				

Fig.2: csv file converted from pdf file.

where each column represents:

- Page No.：page number in which the text line appears
- Text：text line as appeared in the original document
- Index：index of each text line, starting from 1
- Parent Index：refer to the index of its immediate title as appeared in the original document
- Is Title：whether the current text line is a title or not
- Is Table：whether the current text line is in a table or not
- Tag：name of tag(s) in the current text line
- Value：extracted value from the current text line

We have 20 tag(s) to extract, including:

No.	Tag Name (JP)	Tag Name (EN)	Value Type
T1	調達年度	Year of Procurement	datetime (year only)
T2	都道府	Prefecture	text
T3	入札件名	Bid Subject	text
T4	施設名	Facility Name	text
T5	需要場所(住所)	Address for Demand	text
T6	調達開始日	Start Date of Procurement	datetime
T7	調達終了日	End Date of Procurement	datetime
T8	公告日	Public Announcement Date	datetime
T9	仕書交付期限	Deadline for Delivery of the Specification	datetime
T10	質問票締切日時	Deadline for Questionnaire	datetime
T11	資格申請締切日時	Deadline for Applying Qualification	datetime
T12	入札書締切日時	Deadline for Bidding	datetime
T13	開札日時	Opening Application Date	datetime
T14	質問箇所 所/者	PIC for Inquiry of Questions	text
T15	質問箇所 TEL/FAX	TEL/FAX for Inquiry of Questions	text
T16	資格申請送付先	Address for Submitting Application	text
T17	資格申請送付先 部署/者名	Department/PIC for Submitting Application	text
T18	入札書送付先	Address for Submitting Bid	text
T19	入札書送付先 部署/者名	Department/PIC for Submitting Bid	text
T20	開札場所	Place of Opening Bid	text

2 Approach

Our approach can be summarized by the figure below:

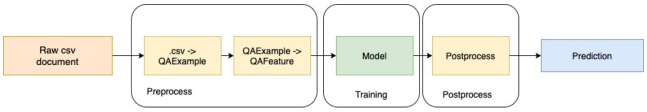


Fig.3: Approach flow.

We will explain each section in details below.

2.1 Preprocess

2.1.1 Definition of class. We used two classes in our preprocess. The *QAExmaple* class store the conversion result from raw csv file and *QAFeature* class is the "trainable" version of *QAExample* (contains corresponding input indexes, attention mask, token type indexes . . .).

```
class QAExample:
    def __init__(self, document_id, index,
    page_id, context_text, question_text,
    answer_text):
        self.document_id = document_id
        self.index = index
        self.page_id = page_id
        self.context_text = context_text
        self.question_text = question_text
        self.answer_text = answer_text
        self.answerable = (answer_text != '')
class QAFeature:
    def __init__(self, example_id, input_ids,
    attention_mask, token_type_ids,
    start_position, end_position, tokens):
        self.example_id = example_id
        self.input_ids = input_ids
        self.attention_mask = attention_mask
        self.token_type_ids = token_type_ids
        self.start_position = start_position
        self.end_position = end_position
        self.tokens = tokens
```

2.1.2 Convert from csv to QAExamples.

- Method 1: We first parse the csv file into sentences. Then, for each tag, we use a sliding window to scan through the file; for each window, we generate an *QAExample*, where the question text is exactly the tag, the context is the concatenation of those sentences separated by BERT's special token *[SEP]*, and the answer text is the first occurrence of the tag in this window. If there are contiguous sentences with same tag, their answers will be concatenated together with BERT's special token *[SEP]*, and other answers in this window will be ignored.
- Method 2: We first parse the csv file into sentences, and then group sentences together by their parent indexes. Then, we find all ancestors of the parent index (parent of parent, parent of parent of parent…). Similarly to Method 1, for each tag and each group, we use a sliding window to scan through it and generate an *QAExample* for each window. The question text is the tag. The context text is the concatenation of the current context, its ancestor's(including parent's) contexts, and contexts in current window. They are seperated by BERT's special token *[SEP]*. The answer text is handled with the same way in Method 1 (the first contiguous answers will be the answer of this *QAExample*). *QAExample*s generated by this method consider the relation between each sentences and its ancestors, which may give more information to the model and perform better (It indeed performs better than Method 1, as mentioned in the third section).
- An Example

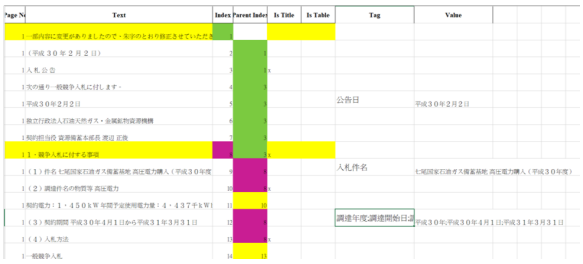


Fig.4: An example of method 2

The group with parent index 8 will be converted to:

- 'document_id': '300351813'
- 'index': [1, 3, 8, 9, 10, 12, 13]
- 'page_id': 1
- 'context_text': '一部容更 、朱字修正 。[SEP]入札公告[SEP]１・競入札付事項[SEP]（１）件名七尾家石油備蓄基地高電力購入（平成３０年度）[SEP]（２）調達件名物質等高電力[SEP]（３）契約期間平成３０年４月１日平成３１年３月３１日[SEP]（４）入札方法',
- 'question_text': '調達年度'
- 'answer_text': '平成３０年'
- 'answerable': True

2.1.3 Convert from QAExamples to QAFeatures.

For each QAExample, do the following process to create a corresponding QAFeature:

- Create a new string "[CLS] + question_text + [SEP] + context_text + [SEP]"
- Find start/end position of answer_text in the new string
- Generate the input_ids, attention_mask, and token_type_ids

2.2 Model

We take advantage of the pretrained BERT model "bert-base-multilingual-cased" from Huggingface [5] by using it as our initialization, combined with the convolution layer to fine tune our model. More precisely, the convolution layer is only stacked on the output of context tokens. The architecture of our model is showed in the following.

- **BERT**: BERT model makes use of the encoder part of Transformer, an attention mechanism that allows to learn contextual relation between words. In contrast to the directional models like RNN,read the context input sequentially, Transformer reads the whole sequence of words at once since it is a bidirectional model. This characteristic make the model learn the context of a word based on all its surrounding, including left and right side of the word.
- **Convolution Layer**: A convolution layer is stacked on the top of BERT to fine tune our BERT model to fit in this task. More concretely, the convolution layer is used to capture local information more carefully. As a pretrained model, BERT produces the hidden vectors from tags and context from our input. To fit in our domain, convolution layer tries to retain the important information between local relation. These information will be fed into the activation function for classification.

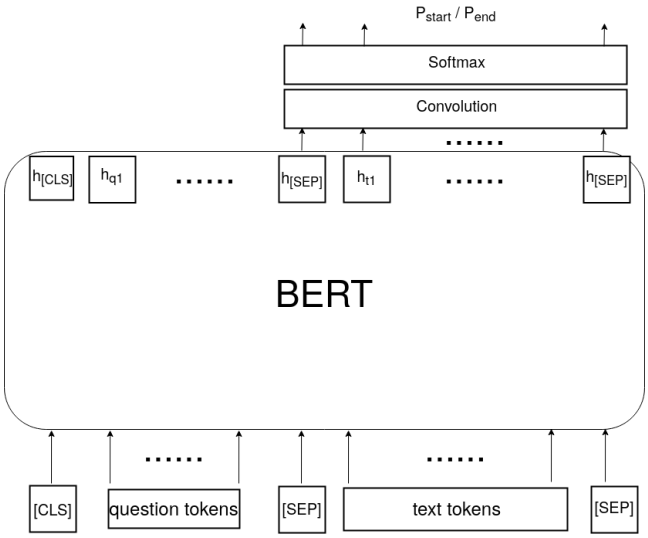


Fig.5: The BERT-CNN model for extracting the value of a tag.

2.3 Training

2.3.1 Main training process.

Our main training process looks like the following:

```
def Train(
    model,
    train_dataloader,
    dev_dataloader,
    epochs
):
    best_f1, best_loss = 0, inf
    best_thres = None

    for epoch in 1, 2, ..., epochs:
        # First, iterate through
    train_dataloader and train the model every
    step

        # Then, iterate through dev_dataloader
    to get:
        #   raw_start_logits,
        #   raw_end_logits,
        #   dev_loss

        # Then, enumerate through different null
     threshold (used in postprocess, 0.1, 0.2,
    ..., 0.7) to calculate the best dev_f1 score

        # Finally, Update best_f1, best_thres,
    best_loss with dev_f1, dev_loss and save
    current model if either of them improved

    return (best_f1, best_thres), best_loss
```

2.3.2 Training tricks.

We used several tricks in training to improve our model performance, including:

- Weight sampling Use weight sampling from dataset instead of enumerate over the dataset to solve the imbalance problem between numbers of answerable and unanswerable *QAFeatures* instances.
- Double transfer learning Fine tune the pretrained "bert-base-multilingual-cased" from Huggingface [5] on the DRCD dataset [1] first before our actual training.

2.4 Postprocess

After training the model, we pass input indexes through model to get logits, then pass the logits through softmax function to get the probability of start/end position for each context token. After that, we find an index pair, *(start_idx, end_idx)*, with the largest probability as the prediction. Moreover, A null threshold is set to verify whether the probability is large enough, and if the probability of our prediction is smaller than the threshold, we will predict "NONE" for this tag.

Furthermore, we perform blending to increase our performance. As different models may have different null threshold, our blending method is divided into two parts:

- Determine whether it's answerable: Each model determine whether it thinks the *QAFeature* is answerable or not and vote its decision. If over less then or equal to half models think it's unanswerable, then predict NONE
- Determine start/end position: Otherwise, we use the average probability of each models' prediction and use the same way above to determine the start/end position.

3 Experiments

We first analyze the performance of different hyperparameters and model structure, then share our hyperparameters used for our best result on private test dataset.

3.1 Model structure: Linear Layers v.s. One/Multi Convolution Layers

In this experiment, we want to know whether convolution layers is better than linear layers or not. Also, we want to know whether or not do the performance of model have positive correlation with the depth of convolution layers.

Stacked Layer	Dev f1 Score
Linear	0.906
1 Convolution	0.933
2 Convolution	0.928

Tab.1: F1 score with different stacked layer.

It seems that convolution layer performs a lot better linear layer. We believe that acing at capturing local relations (in our case, capturing relations between surrounding word embeddings) is why convolution layer performs a lot better than linear layers. Also, deeper convolution layer seems to perform worse than a simple one-layer convolution. This suggests that one convolution layer is enough for capturing the relation and deep convolution layers might easily overfit the training data and perform slightly poor on development dataset.

3.2 Kernel size in Convolutional Layer

This experiment compares different kernel size used in the convolution layer. Using Method-1 preprocess and fix other hyperparameters, we can obtain the following result:

Kernel Size	F1 Score
3	0.919
5	0.922
7	0.933
9	0.929
11	0.928

Tab.2: F1 score with different kernel size.

One can observe that the kernel size with the best result is 7. Our explanation for this phenomenon is that larger kernel size can capture more local information about the context. However, if the kernel size is too large, the model might suffer from overfitting.

Therefore we will use kernel size 7 in the following experiments.

3.3 Data iterations: Simple enumerate v.s. Weight Sampling

Since our data has high proportion of unanswerable examples, simply iterate through all data implies the model have less chance tuning on answerable data, and causes the model to crash after few epochs. By crashing we means that the model's loss increase abruptly and all of its prediction is NONE (unanswerable).

To solve the problem mentioned above, we utilized weight sampling (*WeightRandomSampler* in pytorch). We set the weight so that answerable data has more probability to be sampled than unanswerable data. we can successfully train the model. In our experiment, we discovered model trained with moderate ratio (probability ratio of answerable and unanswerable data) sampler converges faster.

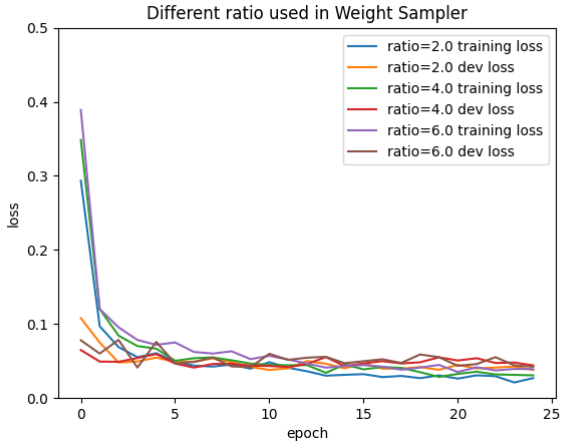


Fig.6: The dev loss decreases faster in ratio=4.0.

3.4 Data preprocessing: Method 1 v.s. Method 2

In this experiment, we want to show that Method-2 preprocess is better than Method-1 preprocess.

Using Method-1 preprocess, we can achieve **0.940** development f1 score. However, using Method-2 preprocess, we can achieve **0.965** development f1 score.

Our explanation for this is that Method-2 preprocess contains not only contexts of neighboring sentences but also contexts of ancestors' sentences. This allows the model to determine which type of contexts it is reading by reading its parent sentences. This implies that Method-2 preprocess probably works good with itemized data. As our data have

a lot of itemization parts, Method-2 preprocess gives us a better performance.

3.5 Single transfer learning v.s. Double transfer learning

In this experiment, we want to know whether transferring from Question-Answering training performs better than transferring directly Language-Masking training. In Single transfer learning, we initialize our model directly with "bert-base-multilingual-cased" from Huggingface [5]. In Double transfer learning, we first initialize our model with "bert-base-multilingual-cased" from Huggingface [5], then train it on DRCD dataset [1] and use the final state of model as our initialization. The results is given in the following graph:

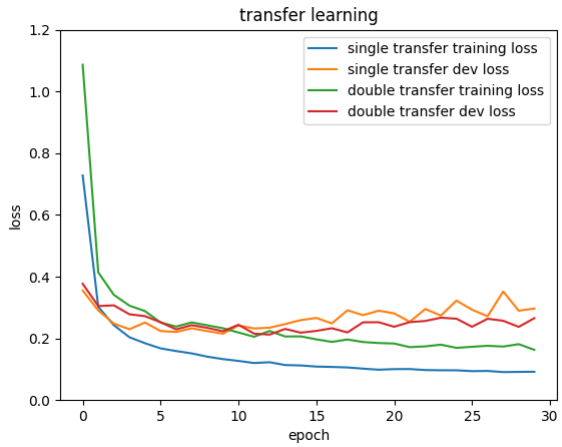


Fig.7: Comparison of single transfer learning model and double transfer learning model.

We can observe that the overfitting is more severe in single transferring than in double transferring. Also, double transferring may achieve a lower loss compared to single transferring. We think that this is because although DRCD dataset is Chinese (which is not Japanese), both data are QA datasets.

3.6 Blending or not

Our blending method have detailed description in the postprocess section.

Using single model and keeping the model with best development f1 score while training, we can achieve only development f1 score **0.969**.

However, if we blend four models with development f1 score **0.959 / 0.958 / 0.960 / 0.963**, we can achieve a better f1 score **0.974** (Note that each model performs worse than the no-blending score **0.969**).

3.7 Best result

Lastly, we share our hyperparameters for our best result on Kaggle competition website [2]. Our best result is blended from four models trained with *kernel_size=7, learning_rate=5e-6, optimizer=Adam, loss=CrossEntropyLoss*, and keep the model with best development loss. The only difference between them is the ratio (probability ratio of answerable and unanswerable data) used in the weighted sampler (2.0, 4.0, 6.0, 8.0 respectively). The development f1 score of the four models are **0.959 / 0.958 / 0.960 / 0.963**, and after blending the development f1 score is **0.974**. The public/private test f1 score is **0.97915 / 0.97904**.

Finally, we plot the tag level accuracy with best hyperparameters mentioned above.

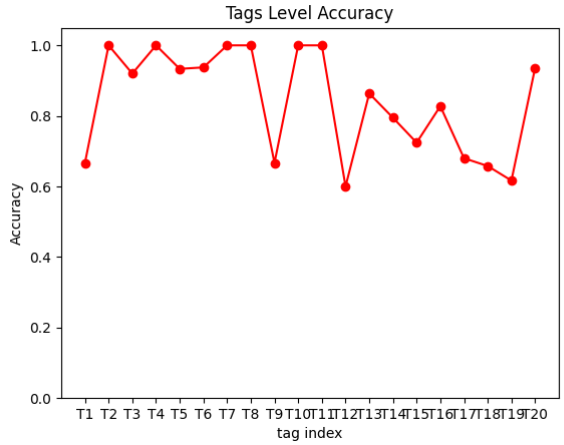


Fig.8: Tag level accuracy with our best model on dev dataset. The exact name of tags can be found at the table in section 1.

4 Work Distribution

- b07902047 羅啓帆：Postprocess, model design & training, report
- b07902139 鄭豫澤：Preprocess, model design & training, report
- b07902141 林庭風：Model design & training, report

5 Conclusion

In this report, we compared several different methods used to solve this limited-data NER task, including different preprocess methods, different model structure (linear/convolution), different ways to iterate through data and different transfer learning source. We concluded that, on this "competitive bids for development projects in Japan" dataset, using Method-2 preprocess, convolution model with double transferring and a training with weight sampler can achieve a better result. Then, we use blending to further improve our performance.

Finally, we achieve public/private test f1 score **0.97915 / 0.97904**, which ranked 1 in the corresponding kaggle competition [2].

References

[1] Delta Reading Comprehension Dataset

[2] Kaggle Competition Website: ADL Final Project Shared Task (108 Spring)
[3] Information extraction (IE)
[4] Named Entity Recognition (NER)
[5] HUGGING FACE: On a mission to solve NLP, one commit at a time.