Team 21 Applied Deep Learning Final Report

Han-Wei Chen
b05705025
b05705025@ntu.edu.tw

Liang-Ying Huang
b05705017
b05705017@ntu.edu.tw

Guan-Min Lien
b05705009
b05705009@ntu.edu.tw

Abstract

In the final shared task project, due to the limited amount of data, we applied transfer learning with Bert and Albert to extract information from Japanese bidding documents. To be more specific, we used 78 documents as training set and 22 documents as validation set to train a deep neural model, then further using our trained model to extract important tags and its corresponding values from plain testing documents. The result showed that our model have capability to extract key information correctly.

ACM Reference Format:
Han-Wei Chen, Liang-Ying Huang, and Guan-Min Lien. 2020. Team 21 Applied Deep Learning Final Report. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introduction

There is large amount of information in un-structured document data. How to efficiently and effectively extract useful information from them (i.e. document information extraction) is a classical problem in the field of natural language processing. In this report, we presented BERT, ALBERT base models to deal with this task in a real world Japanese bidding documents dataset. The training sample of this dataset is relative small, we aim to find specified names and entities within these limited data. Also, to get a better performances, we conducted ensemble technique to gather multiple models' extractions. Here is our codes: https://github.com/DylanHuang126/Japanese-document-IE

The report will contain the following section: Approach, where we explain how we define this task, introduce dataset, and present our proposed model; Performances and Experiments, where we show the performances of our model and demonstrate some experiments to evaluate models goodnesses; Summary and Future work where we summarize the result and discuss possible future works.

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

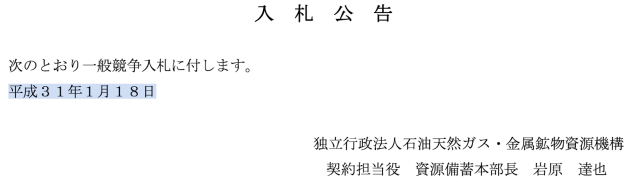
2 Approach

2.1 Task definition

We took the task as a question-answering scenario. In this case, BERT will be appropriate in this task. Bert will be used for fine-tuning, and questions, answers, contexts are needed for BERT. Thus, tags and values from source data will be seen as questions and answers respectively, and texts per paragraph will be seen as contexts for BERT.

2.2 Dataset

There are two format of each data, PDF and Excel file. The original paragraph in PDF may seem like this way:


入　札　公　告

次のとおり一般競争入札に付します。
平成３１年１月１８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

Figure 1. Sample Paragraph in the PDF file

So the date 「平成３１年１月１８日」is one of the answer, and the related question will be the tag as 「公告日」. 「公告日」is the date of announcement, so it can be the question like when was the announcement date.

Let's see how the excel file looks like:

Text	Index	Parent Index	Is Title	Is Table	Tag	Value
入 札 公 告	1		x			
次のとおり一般競争入札に付します。	2	1				
平成３１年１月１８日	3	1			公告日	平成３１年１月１８日
独立行政法人石油天然ガス・金属鉱物資源機構	4	1				
契約担当役 資源備蓄本部長 岩原 達也	5	1				

Figure 2. Same Paragraph in the Excel file

We would like to look for tags per paragraph when we are testing, and it's obvious that there's the Parent Index column in the excel file. So we should find the paragraph to which the tag belongs according to the Parent Index. Then, make tags and values pairs with context like (tag, value, context), and add them into the dataset.
Also, we will record whether there is tag or not in each line.

If there's at least one tag in one line, it will be labeled as 'positive'. Otherwise, it should be 'negative'. For the training data, there are 38807 negative lines and 1980 positive lines. For the develop data, there are 10768 negative lines and 524 positive.

	Negative	Positive
Train Data	38807	1980
Dev Data	10768	524

Figure 3. statistics of positive and negative data

Basically, we fine-tuned BERT with context question. Like above paragraph, we will concatenate the text which belongs to paragraph 1 in the excel file as the context. Similarly, the question should be the labeled tag. When we put them in BERT, It would be look like:

[CLS] 入札公告 [SEP] 次のとおり一般競争入札に付します。 [SEP] 平成３１年１月１８日 [SEP] 独立行政法人石油天然ガス・金属鉱物資源機構 [SEP] 契約担当役 資源備蓄本部長 岩原 達也 [SEP] 公告日

Figure 4. Same Paragraph in the Excel file

2.3 Proposed model

In this task, we only have limited data that can be used for training. Therefore we would want to use the transfer learning approach to deal with this problem. We then try out 2 similar but different approach of the model structure:

- BERT
- ALBERT

But there are still a little difference between general BERT and ALBERT model in implement stage(such as tokenize method and pretrain data). After survey, we decide to use bert-based-multilingual, bert-based-japanese and albert-japanese-v2 to comparing the difference.

BERT is the encoder of transformer. It is designed to find the contextualize representaion of the input sentence vector. Which means it can consider not only the front but bi-directional of sentence contexts. It shows SOTA performance in almost every nlp task in 2018, include the QA task we want to solve here. Therefore we want to apply this task to BERT QA question and conduct fine-tuning to help us extract the possible tag under each parent paragraph.

ALBERT is the brief name of **A Lite Bert**, which use cross-layer parameter sharing and factorized embedding parameterization two method to significantly reduce the parameter of BERT. And it also use the technique of sentence-order prediction to replace BERT's next sentence prediction. Which will have better influence on model's learning about the contextualize representation. We want to take these advantage to see if ALBERT can outperform BERT in this task. And also want to observe if ALBERT will catch different representation and predict differently than the BERT model.

For both models we apply the QA structure which the output of the models will be the start and end index of the paragraph, we then use the post-processing steps to clean up and do the prediction.

In preprocessing steps, we will concat all the sentence under the same parent indexes and seperate them with "[SEP]", then we calculate the start end indexes as the final ground truth. If the tag is not exist in the corresponding paragraph. We will assign start and end =(0,0) point to "[CLS]".

In the post-processing step, we will check wether the argmax of start and end prediction score are lies under (0,0), if is not then extract the range as the final answer. But if it is lies under (0,0) then we predict the tag has no answer in this paragraph. This strategy seems pretty strict and will cause a lot of no answer situation. But under this task, the none answerable data occupy 80% of the data, the unbalance situation is suitable for this method.

2.4 Proposed improvement approach

After getting decent results on individual models. we then want to further improve the model for different methods, we then develop 2 approach to get better performance in the task:

- 2 steps training approach
- Model ensemble approach

In 2 steps training approach, we want to fine-tune the model to a better starting point for the actual training process. Because the pretrain dataset of the models is quite different from the current task. We also think that if we want to let the model learning about the representation in the first stage and learn about where to extract the answer in second stage. Therefore we generate a special dataset for this process. The new dataset contains only the positive data which means only the data with answers(About 1600). After several epoch of fine tuning, the model will be encourage to extract every possible tag of the testing data. After that we added negative samples (Tag without answers) into the dataset to begin the step 2 training process. The outcome distribution after step 2 training will be close to the actual tag-answer distribution.

In ensemble approach, we observe that the real reason the f1 score can't be further improved are mainly due to our unanswerable prediction are too much, not because of our extraction make the wrong prediction. And we also find out that each models have different strength on predicting different tags. Therefore we tries to aggregate the results of each prediction and tries to merge into a better output. In order to make it perform better. We set up several constraint

Table 1. Performances

Model name	F1 score on develop set
Baseline	0.8749
ALBERT-Japanese non-pretrain	0.9413
ALBERT-Japanese pretrain	0.9403
BERT-Multilingual non-pretrain	0.9153
BERT-Multilingual pretrain	0.9342
BERT-Japanese non-pretrain	**0.9499**
BERT-Japanese pretrain	0.9480

while merge to decide the priority when encounter awkward situation:

- Guess one Get one Policy - Concatenate all possible predicted tag, which means that if there are any model predict a new tag that no one else predicted, then we will add it to the final results.
- Longest First - Choosing the longest sentence if encounter different model predict the same tag based on the observation our model tend to have too short extraction range.
- Best Model First - Using validation f1 as a priority when encounter same tag with same length.

With these constraints and rules, we can aggregate the results with the best advantage and will also decrease the amount of unanswerable data. In the following parts we can see that this strategy actually work very well and improved our f1 scores for about 1.5%.

3 Performances and Experiments

3.1 Performances

We set the baseline as when the predictions of model are all majority answer (i.e. 'NONE'), the F1 score is 0.8749. In Table 1, it is not difficult to observe our models all outperformed baseline by above 0.04. The best model-Bert-Japanese non-pretrain increase its F1 scores by 8.5% compared to baseline. The result showed our proposed models are able to extract correct tags and their corresponding values but there is still room for improvement.

3.2 Effect of two steps training

The effect of two steps training could be indicated from Table 1. Model with non-pretrain setting meant conducting only one step training (i.e. without training on only positive data) while pretrain meant two step training. In Table 1, multilingual model had about 0.02 improvement with and without pretrain, which referred that two steps training is useful. However, we were not able to see same improvement in Albert or Bert-Japanese. The main reason should be when we use multilingual model as base model, we need to let the model focus on positive data as well as let it transfer more to Japanese documents at the same time. In addition, in Figure 5 to 7, the pretrain models can acquire high F1 score in relative early steps of training, which again showed our two steps training process are worth it.

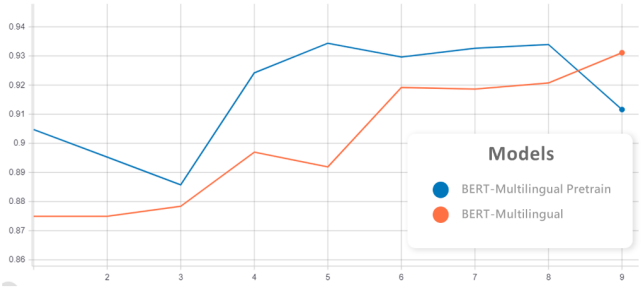


Figure 5. F1 scores of pretrained and non-pretrained bert multilingual model per epoch

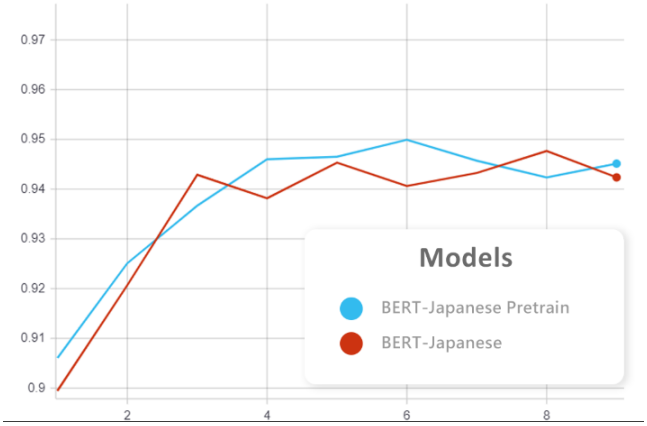


Figure 6. F1 scores of pretrained and non-pretrained bert Japanese model per epoch

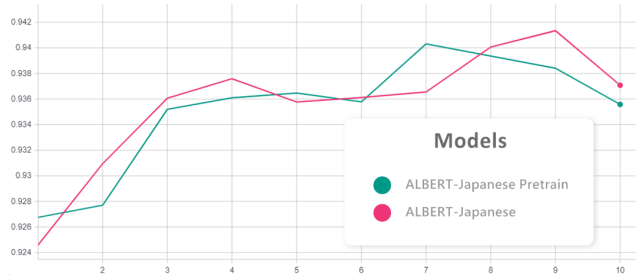


Figure 7. F1 scores of pretrained and non-pretrained albert model per epoch

3.3 Accuracy of each tag

Figure 8 and Figure 9 are the bar charts of accuracy of each tag where Figure 8 contain Tag 1 to 10 and Figure 9 contain Tag 11 to 20. The blue, orange, green, red-colored bars are

the result of Bert-Japanese, Bert-Multilingual, Albert and Ensemble, respectively. The reference table of tag numbers and tag entities is also shown.

Except for the ensemble model, the Bert-Japanese model had overall better accuracy in most of the twenty kind of tags. The main reason behind this is that all the documents in the dataset are written in Japanese so the pretrained embedding and tokenization method in Bert-Japanese are more suitable for this task. However, Bert-Multilingual model outperform the other two models in Tag 2, 6, 9 and 20. On the other hand, Albert model only surpass the other models in Tag 3. We tried to figure out if there is some relation between the well-performing model and tag entities such as multilingual model was good at extract datetime or place, but we failed. However, we still assumed each model had its strength in extracting different tags.

To investigate the result of our ensemble approach, one can see that ensemble model have highest accuracy among all models in every tags. That is to say, it not only preserved the performance of the best model but also took advantages of other models. The most obvious improvements can be observed from Tag 8 and 16. These result proved our policies in ensemble approach are indeed effective.

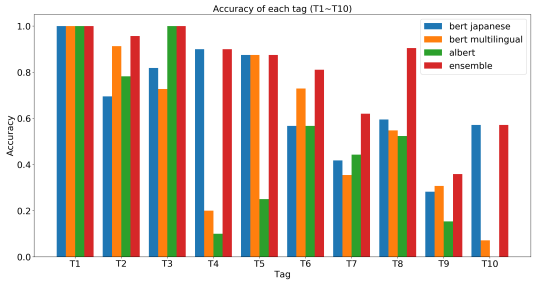


Figure 8. Accuracy of T1 to 10

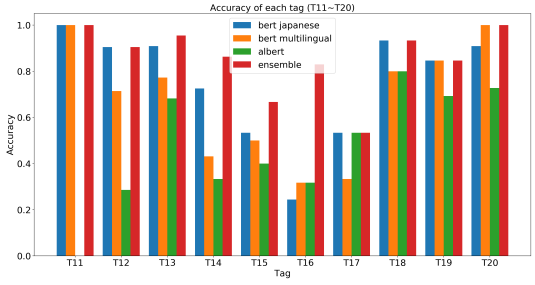


Figure 9. Accuracy of T11 to 20

Table 2. reference table of tag numbers and entities

T1: Public Announcement Date
T2: Bid Subject
T3: Year of Procurement
T4: End Date of Procurement
T5: Start Date of Procurement
T6: Address for Submitting Bid
T7: PIC for Inquiry of Questions
T8: Department/PIC for Submitting Bid
T9: TEL/FAX for Inquiry of Questions
T10: Deadline for Delivery of the Specification
T11: Deadline for Questionnaire
T12: Deadline for Applying Qualification
T13: Opening Application Date
T14: Place of Opening Bid
T15: Address for Submitting Application
T16: Department/PIC for Submitting Application
T17: Deadline for Bidding
T18: Address for Demand
T19: Prefecture
T20: Facility Name

4 Summray and Future work

4.1 Summary

From the above experiments, we can compare the results and make a brief summary for the topic:

- Comparing several model and structure, we find out that **BERT-based-Japanese** is most suitable for this task. But this result doesn't have significant improvement from the other. And the variable in the tokenize method make it hard to analysis why it has the best performance. So we still need to design a more detailed experiments to find the reasons.
- Two step training method is more effective in multilingual model. Our pretrained method is not so effective as we assume in japanese-based pretrained mdels. But in a diverse pretrained model like multilingual. It can effectively fine-tuning an better representation as we proposed.
- Each model will learn and predict differently on each tag. From Figure 8 and 9 we can observe that each model has their own strength on different topic. Besides from the randomness in training steps, we assume the reasons for this phenomenon has to do with the structure of the models and the pretrained representation of each token. Like the difference in prediction on albert-japanese and bert-based-multilingual are most significant. Because they not only have different model structure but also have different representation on pretrained datasets. But the difference between

albert-japanese and bert-based-japanese is not so obvious since they use the same pretrained datasets.
- Ensemble method will have significant improvement on our approach. Not only just from the above reasons. Our rule-based approach of concatenation also effective on the final results.

4.2 Future work

- Try out different dataset structure or context splitting rules and collect more similar documents for training.
- Specific tag prediction on different model based on their advantages.
- Stack more complex model structure such as CNN, LSTM on bert base model.

5 Work distribution

- **B05705009 Guan-Min Lien**: data resource, data preprocessing, report writing (Task Definition, Dataset).
- **B05705017 Liang-Ying Huang**: data preprocessing, model implementation (bert japanese), report writing (Abstract, Introduction and Performances and Experiments, overleaf template arrangement), code organization and release.
- **B05705025 Han-Wei Chen**: data preprocessing, model implementation (bert multilingual, albert japanese), report writing(proposed models, Proposed improvement approach and summary).

6 References

1. Transfer Learning for Information Extraction with Limited Data
2. . Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. (2018)
3. Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, Radu Soricut: ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. (2019)