ADL Project

Cinnamon - Information Extraction

Group: 19
R08942087 吳彬睿 NTU GICE
R08942035 許軒瑋 NTU GICE

1. Abstract

This work is carried out on the dataset provided by Cinnamon Company. The challenge of shared tasks is mainly focused on information extraction, which is similar to NLP NER(Named Entity Recognition) tasks. The purpose of the task is to extract the important informations from the official documents.

2. Introduction

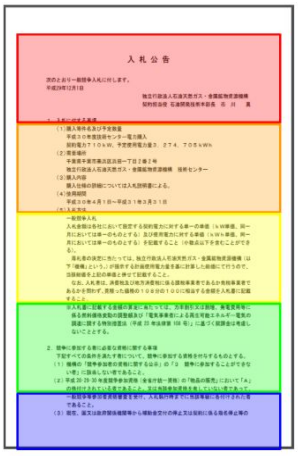
The Named Entity Recognition aims to locate and classify named entity in the text into predefined categories, such as personnel name, locations, organizations, etc. Generally, four types of algorithms would be used in NER tasks as follows: rule-base, unsupervised, feature-based and deep learning methods. Among unsupervised classification algorithms, the well-known classic algorithm is clustering, which recognizes named entities based on statistics and text similarity.Today, researchers usually choose deep learning methods to complete NER tasks. Compared with traditional machine learning HMM, CRF and other feature-based methods, deep learning methods can achieve better performance. In our work, we use the BERT pre-trained model as the model backbone. Through the application of transfer learning, although there is not much training data, we still can get a good performance.

3. Approach

3-1. Pre-processing

We have try three different type of preprocessing, Pre-process v0~v3 , and in the experiments the v0 is the poorest, v1 and v2 are much more better than v0, and v2 is a

little bit performance better than v1. For v0 and v1, the model acts like human reading a paragraph, but for v2 it doesn't act like human we give it pairs to let it learn pattern of pairs.

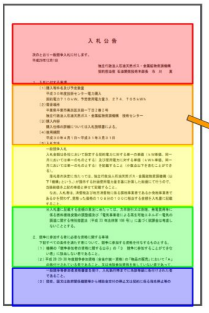


(Pre-process v0)

(Pre-process v1)

(Pre-process v2)

[Pre-process V0]



(Pre-process v0)

1. sample (concate 10 texts)

   {"text_0", "text_1", "text_2",..."text_10"} , {"text_11", ...

2. model input

   [CLS] <text_0> [SEP] <text_1> ... [SEP] <text_N>

3. performance

   Poor !

For V0 we apply the simplest way to preprocess the document, we read the xlsx file and fifo text with 10 rows concatenate together, and each sub paragraph then become one training sample to our dl model.

[Pre-process V1]

(Pre-process v1)

1. sample (group by parent index)

   {"text" , "text" , "text" , ... } , {"text" , "text" , "text" , ...

2. model input

   [CLS] <text_0> [SEP] <text_1> ... [SEP] <text_N>

3. performance

   better !

For V1 we group the text by it's parent index, we think that text have same parent index should have information share with each other so every sample can get furthermore informations, and it indeed perform better than V0.

[Pre-process V2]

1. **sample (text & parent text)**

   {"text", "p_text", "Index", "p_Index", "doc"} , {"text

2. **model input**

   [CLS] <text...> [SEP] <p_text...> [SEP]

3. **performance**

   better !

(Pre-process v2)

For V2 we don't let model read paragraphs. Alternatively, we pairs datas text with parent index's text. And for this kind of pre-processing, our model performed the best, finally we choose this as our preprocessin approach.

3-1-1. data cleaning

```
入札件名82] train loss:0.499 acc:0.82 f1:0.01
[23, 17, 24, 3356, 28581, 266, 2921, 853, 25, 1387, 36, 2988, 1594, 11144, 10728, 8727, 17705, 397, 701, 6012, 3601, 28822, 23,
19910, 21680, 28985, 24, 502, 107, 5170, 28742, 5326, 3638, 15800, 3042, 28658, 28813, 6012, 3601, 1880, 23, 3152, 19479, 3908,
28548, 1108, 5, 953, 4286, 1097] (1)件名 ：令和２年度「ガスクロマトグラフ燃焼同位体比質量分析計（DeltaV）および高分解能誘導結合プラズマI
CP質量分析装置（ELEMENT XR）」の年間保守契約
[2921, 853, 25, 1387, 36, 2988, 1594, 11144, 10728, 8727, 17705, 397, 701, 6012, 3601, 28822, 23, 19910, 21680, 28985, 24, 502,
107, 5170, 28742, 5326, 3638, 15800, 3042, 28658, 28813, 6012, 3601, 1880, 23, 3152, 19479, 29296, 28548, 1108, 5, 953, 4286, 1
097] 令和2年度「ガスクロマトグラフ燃焼同位体比質量分析計(DeltaV)および高分解能誘導結合プラズマICP質量分析装置(ELEMENTXR)」の年間保守契約
300366257
```

Like the example above some values can't work well by finding the substrings in text, so we need to apply maximum length sequence (MLS) to find the most possible place of index, which is probably not exact matching, but it is the best way to label this kind of unclean datas.

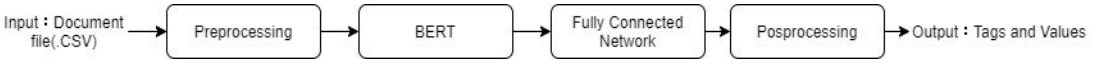
3.2 Model Architecture

3.2.1 Bert

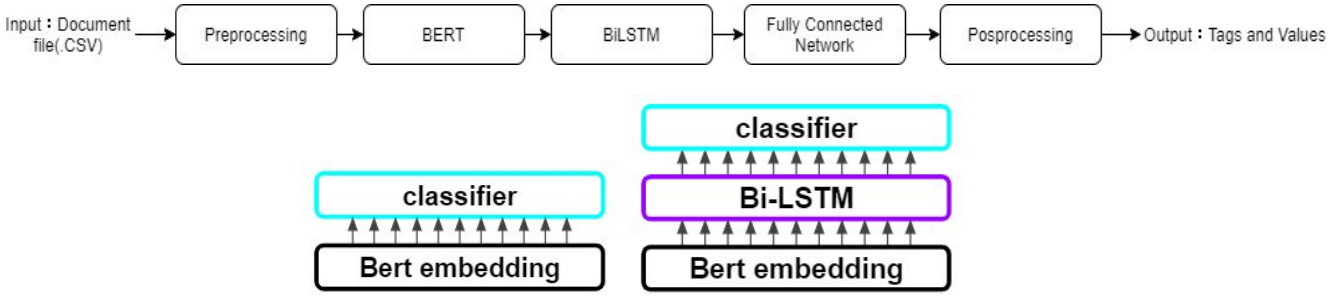
pretrained : '**cl-tohoku**/bert-base-japanese-whole-word-masking'
tokenizer : **BertJapaneseTokenizer**, BertTokenizer

There are plentiful pretrained model can choose, we choose cl-tohoku because it was used by most people, and for tokenizer there are two choices, for BertJapaneseTokenizer it tokenized the sentences into words, and for BertTokenizer it tokenized sentences into chars, and for our experiences BertJapaneseTokenizer is better, so we choose it as our tokenizer.

3.2.2 Baseline Model



3.2.3 BiLSTM Model



Our model is quite easy because the cinnamon information extraction dataset is only 80 docs, it may be overfitting if we apply too complicated models. And for this kind of dataset, we should pay more attention on data preprocessing and postprocessing it will make more improve for performance.

3-3. Post-processing

We find that the char is sometimes full char, so we will make checks if of our prediction texts matches the size of char in text.

4. Experiments & ANALYSIS

4.1 Training

- Batch size : 32
- Learning rate : 2e-5 (with decline)
- Critirion: BCE Loss (pos_weight=[40])
- Optimizer : AdamW

4.2 Ablation test

- Pre-processing 1:

	Epoch	F1 (ours)	dev Score	F1 EM (kaggle)
naive baseline	60	0.72	0.92147	0.93612
Bi-LSTM	90	0.74	0.92168	0.93322

- Pre-processing 2:

	Epoch	F1 (ours)	dev Score	F1 EM (kaggle)

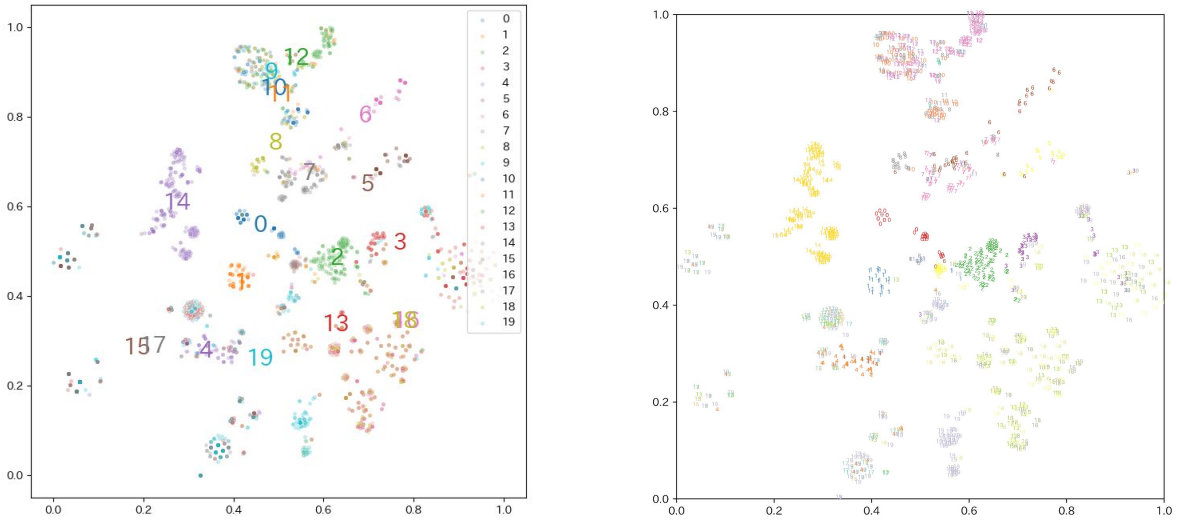
naive baseline	60	0.77	0.94640	0.95169
Bi-LSTM	90	0.81	0.95234	**0.95518**

* f1(ours) : The metric defined by ourselves, which is different from kaggle's em.
* f1 em (kaggle) : The score of kaggle submission.

Our experiment

4-3. t-SNE

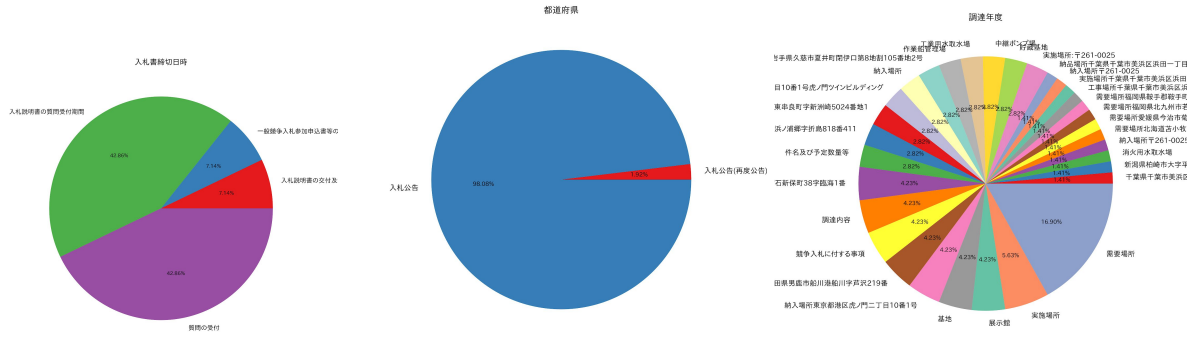
調達年度	都道府県	入札件名	施設名	需要場所(住所)	調達開始日	調達終了日	公告日	仕様書交付期限	質問票締切日時	資格申請締切日時	入札書締切日時	開札日時	質問箇所所属/担当者	質問箇所TEL/FAX	資格申請送付先	資格申請送付先部署/担当者名	入札書送付先	入札書送付先部署/担当者名	開札場所
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19



The above figure shows the visual distribution of each sentence embedded in the Cinnamon Dataset (including the training/dev dataset) corresponding to the ground

turch tag-value pair. After t-SNE processing, the distribution is simplified to a two-dimensional data distribution. (For example, it is to perform t-SNE analysis after sentence embedding of the values in the ground truth "tag:公告日　value:平成29年9月22日".) The sentence embedding method we adopt is to convert the value into word embedding through the bert pre-training model (Japanese). After embedding all the tokens in this sentence, the average value of the word embedding of the tokens are as the representative of this sentence embedding. As shown in the figure above, it can be found that the distribution of sentence embedded values of some tags is more concentrated than the distribution of sentence embedded values of other tags, such as type 14(tag:質問箇所TEL/FAX). However, for some tags, the sentence embedding distribution of their corresponding values is very similar, such as type 9, type 10, type 11, type12(tag: 質問票締切日時、資格申請締切日時、入札書締切日時、開札日時). After careful observation, we found that this is because the target tags they are looking for correspond to the same text - date and time. If you want to distinguish each other, you need to use keywords before and after the date paragraph to help classification. This thing can be discussed later.

4-4. Statistics



The statistical analysis in this part is to collect the parent texts corresponding to the value of the same tag. We want to know the relationship between parent text and the tag-value pairs. After statistics, we found that some tags almost only appear under a specific parent text. Like, the tag - *都道府県* only appears in the paragraph where *入札公告* is the parent index text. It can be speculated that the occurrence of this

phenomenon is related to the file format. Some tags will only appear in certain paragraphs of the document. By the way, if it is a date and time tag (such as *調達年度*), we guess that the document format has no restrictions, so this tag may appear in any paragraph of the file. This leads to various possible parent texts for this kind of tag.

4-5. BERT model attention hidden layers

Intra-Similarity(cosine similarity):

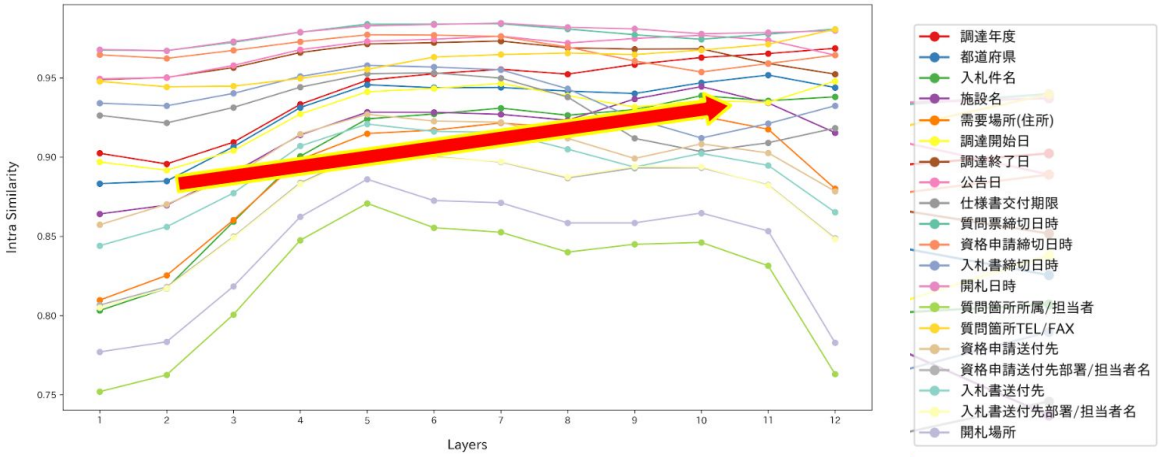
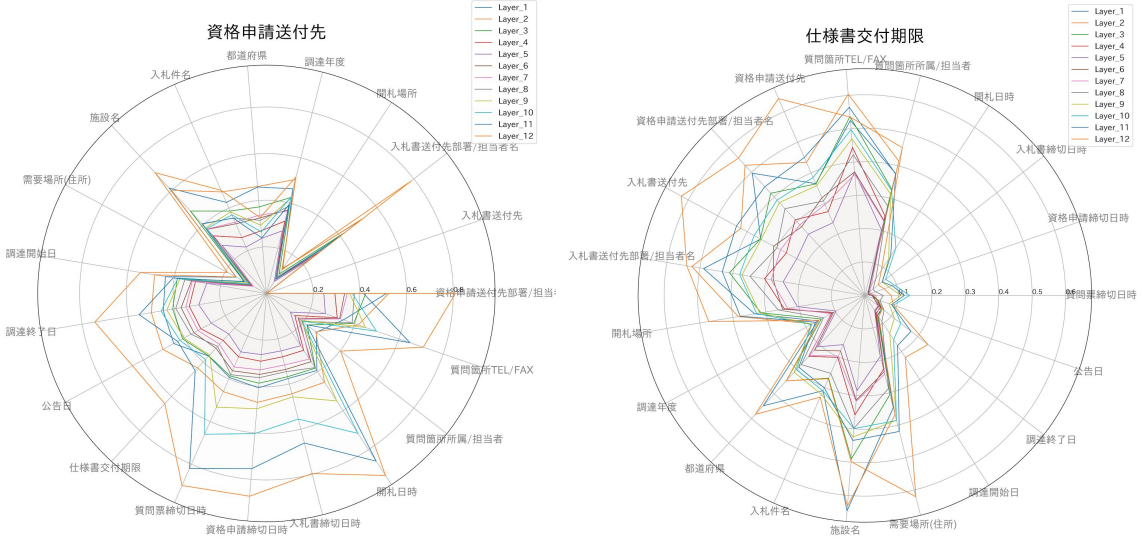


figure 10

Inter-Similarity(-log(cosine similarity)):



This analysis is to analyze the BERT model. We know that the BERT model is a deep neural network and can be divided into 12 layers. The embedding performance of these 12 layers should be as deep as possible. Since we have 20 kinds of tags, we divided the ground truth value into 20 groups according to their respective tags. We want to observe the performance of inter-similarity between different groups and intra-similarity inside the same groups on the 12 different layers of the bert model. For similarity calculation we use the cosine similarity. Sentence embedding selection is the same as previous in section 5-1.

Usually the intra-similarity of the cluster is high and the inter-similarity is low, which means that the classification effect is better. It is reasonable to say that as the number of layers increases, cluster classification should be better. However, as shown in Figure 5-1, it can be found that although for the inter-similarity between different layers, for most tags, the similarity value also increases as the model layer increases. Nevertheless, for the intra-similarity corresponding to some tags, the intra-similarity value decreases with the increase of the BERT model of layers. We suspect that this phenomenon may be similar to the situation when we do t-SNE analysis previous. Due to the values of certain tags, their corresponding values are very similar sentences, such as date and time. Therefore, if we want to classify the tags correctly, we should find some keywords through the context to make a correct judgment. In addition, when considering the clustering performance of clustering, not only the performance of intra-similarity but also the performance of inter-similarity must be considered. This means that the model is indeed learning in the right direction. For the above problem, if the decoder is properly selected and attached to the fine-tuned bert model, we think this should be an effective solution.

5. CONCLUSION

In whis work, we use bert pre-trained model to extract the sentence feature and use bi-LSTM to decode which label the sentence token should be. By analyzing the given training dataset and the model we use, we know there is still room for improvement

in our work. Through pre-processing and post-processing, the final mean f-score in the test dataset can reach 0.95. For us, if we want to improve the performance of our work, we think that perhaps a more complex model as Fuzzy-LSTM-CRF, may be used to complete this work.

6. Work Distribution

- Dataset : 吳彬睿
- Analysis : 許軒偉
- Training : 吳彬睿, 許軒偉
- 吳彬睿50%, 許軒偉50%

7. Reference

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805, 2018.
(python package：[Transformers] https://huggingface.co/transformers/index.html)

[2] Xiang Dai, Sarvnaz Karimi, Ben Hachey, and Cecile Paris. 2019. Using similarity measures to select pretraining data for NER. In Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL).

[3] Jingbo Shang, Liyuan Liu, Xiaotao Gu, Xiang Ren, Teng Ren, and Jiawei Han. 2018. Learning named entity tagger using domain-specific dictionary. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2054–2064, Brussels, Belgium. Association for Computational Linguistics.

[4] Unsupervised NER using BERT (website link：
https://towardsdatascience.com/unsupervised-ner-using-bert-2d7af5f90b8a)

[5] MAKING DYNAMIC DECISIONS AND THE BI-LSTM CRF

(Link：https://pytorch.org/tutorials/beginner/nlp/advanced_tutorial.html)