NLP_Final_project

B06902012 龔柏年
B06902032 楊則軒
B06705059 魏任擇
F06945029 陳彥斌

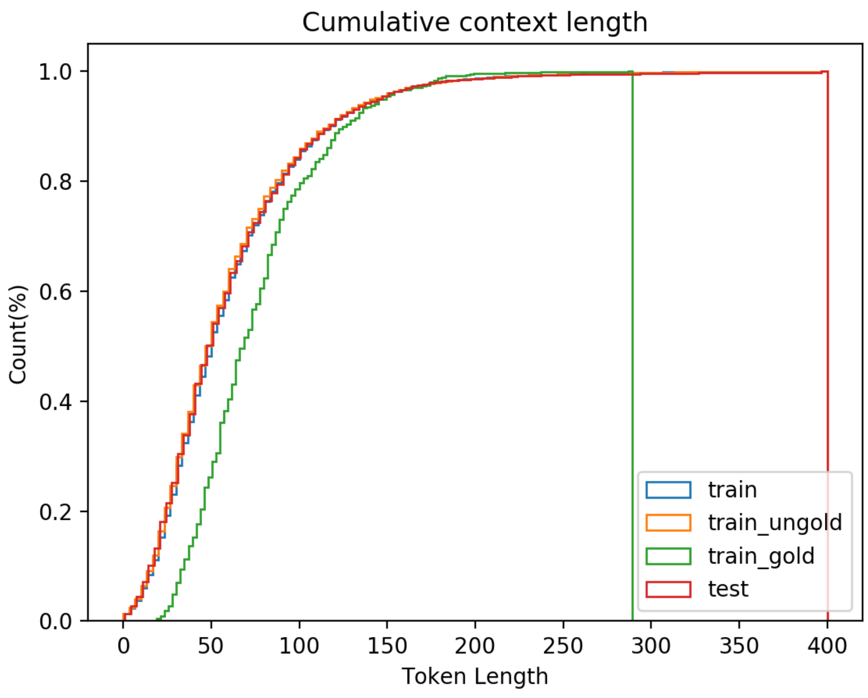
Task 1

How to run

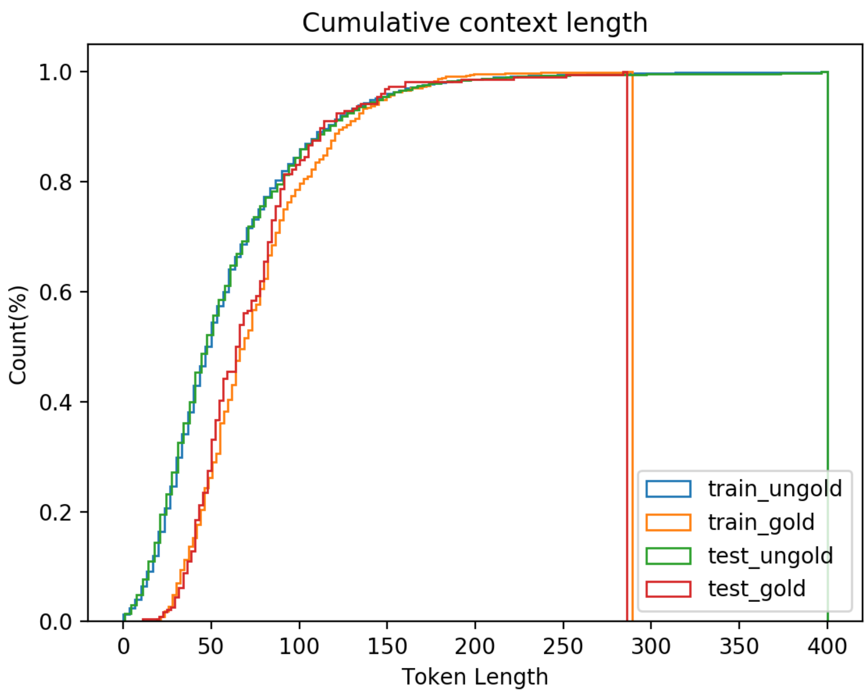
```
bash task1.sh test
python s2s_v203_lstm.py test

result:
./task1_method1.csv                 (ELECTRA)-best
./task1_method2.csv                 (ELMo)-poor
```

Data analysis

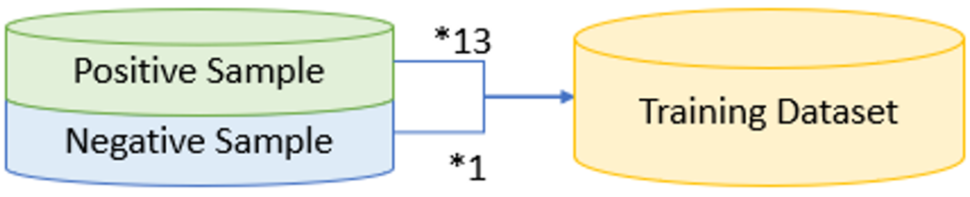
- Data num
  - Train: 10837
  - Test : 2710
- Data context length(Token)
  - Train and Test
    - 

  - Train and Predict(gold = 0 and gold = 1)

- 

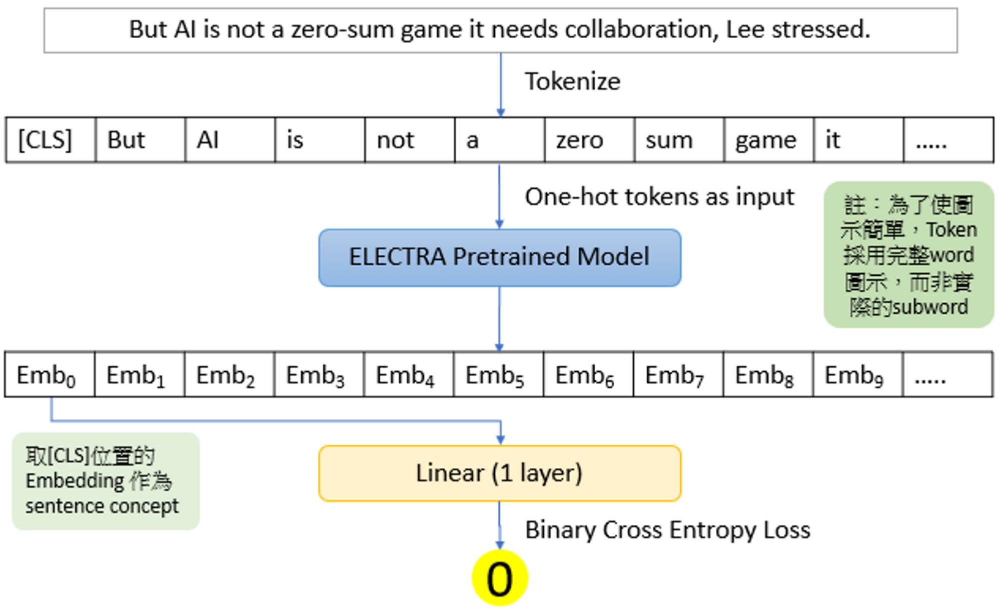
- Analysis
  - In the First pic, it is seen that the cummulative context length of Train set and Test set are nearly same, therfore we can assume that the Train set and Test set are very similar.
  - For the data with gold=0, they have average longer context length than those with gold=1, show in the figure above.
  - In the second figure, we compare the context length(gold=0 and gold=1) in training set and our prediction in testing set. It can be seen that the context with gold=1 predict by our model has shorter context length than the training set.

Method

- Data Preprocessing
  - Because we find that the number of datas labeled with gold=0 and gold=1 are extremely not balanced, we tried to do some data preprocessing to make data more balance such that the model will have a good performance.
  - 

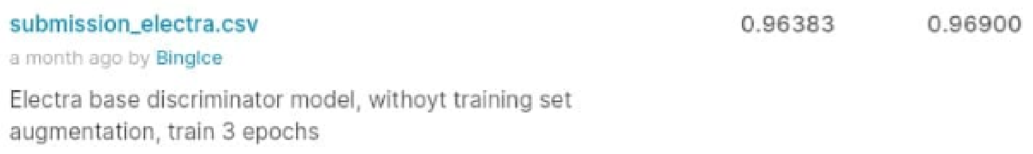
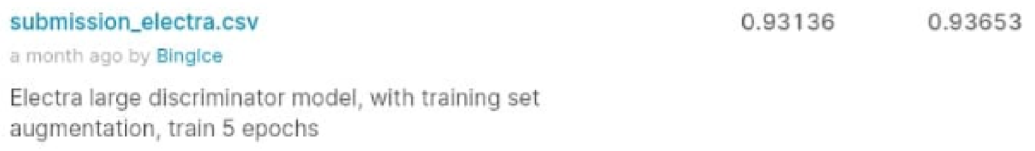
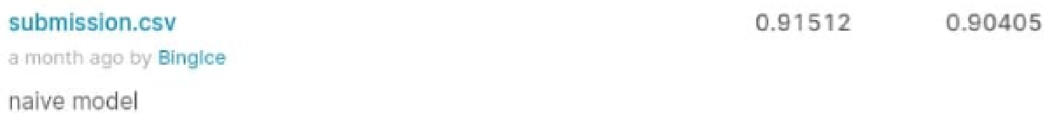
    - we solved this problem by repeating the small numbers of data by 13 times.
  - We first input text to Electra tokenizer. Then we cut it into subword tokens and put a token [CLS] on the top of sentence. At the same time, we will cut it to 512 tokens if it is composed of more than 512 tokens and turn it into one-hot encoding.
- **Model 1**
  - ELECTRA base-pretrain Model discriminator
  - We input the sentence composed of tokens into language model, then it will output a embedding vector.

- We know that the first dimension of this embedding vector represents the meaning of whole sentence. So we input this vector into an one layer of Linear model, which will project $\mathbb{R}^{768} \rightarrow \mathbb{R}^{2}$. Then we can use Binary crossentropy loss.


But AI is not a zero-sum game it needs collaboration, Lee stressed.
Tokenize
[CLS] But AI is not a zero sum game it .....
One-hot tokens as input
註：為了使圖示簡單，Token採用完整word圖示，而非實際的subword
ELECTRA Pretrained Model
Emb0 Emb1 Emb2 Emb3 Emb4 Emb5 Emb6 Emb7 Emb8 Emb9 .....
取[CLS]位置的Embedding 作為sentence concept
Linear (1 layer)
Binary Cross Entropy Loss
0

- **Model 2**
  - ELMo (character base Naïve model)
  - We used wikipedia corpus pre-trained, but pre-trained need a huge resources to have a nice performance. So this method didn't have a nice performance.
  - The structure is same with Model 1 mentioned above.

Results

- **Model 1**
  - ELECTRA base-pretrain Model discriminator
  - 
    submission_electra.csv 0.96383 0.96900
    a month ago by BingIce
    Electra base discriminator model, withoyt training set augmentation, train 3 epochs
  - 
    submission_electra.csv 0.93136 0.93653
    a month ago by BingIce
    Electra large discriminator model, with training set augmentation, train 5 epochs
  - why
- **Model 2**
  - ELMo (Naïve model)
  - 
    submission.csv 0.91512 0.90405
    a month ago by BingIce
    naive model
  - We find that ELMO model has the worse performance (only 0.90405), this maybe as a resulf of lacking of pre-trained resources.

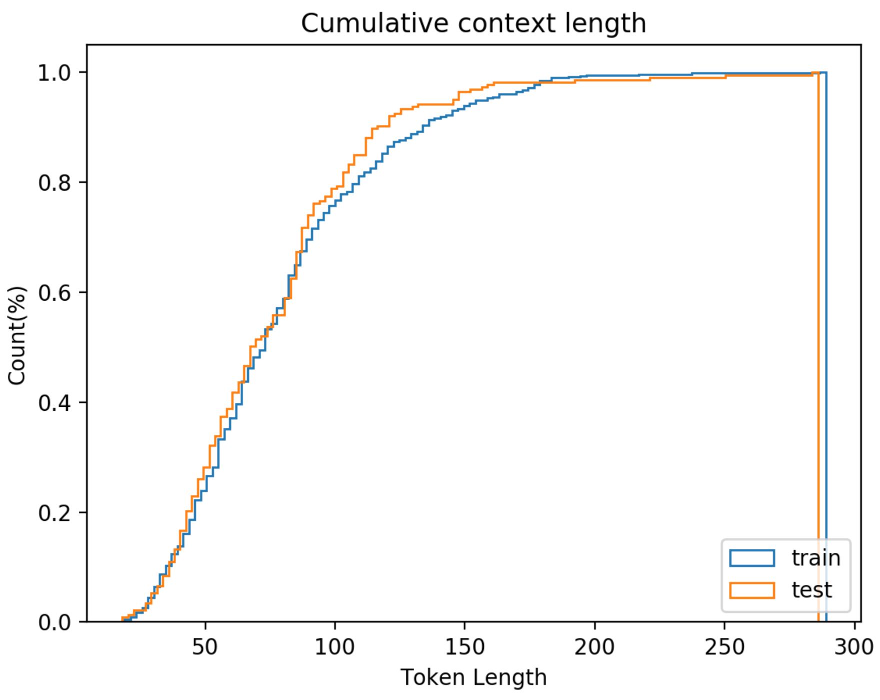
Task 2

How to run

```
bash task2_electra.sh test
bash task2_bert.sh test

results csv:
./data/task2_method1.csv             (ELECTRA)-Best
./data/task2_method2.csv             (BERT)-baseline
```

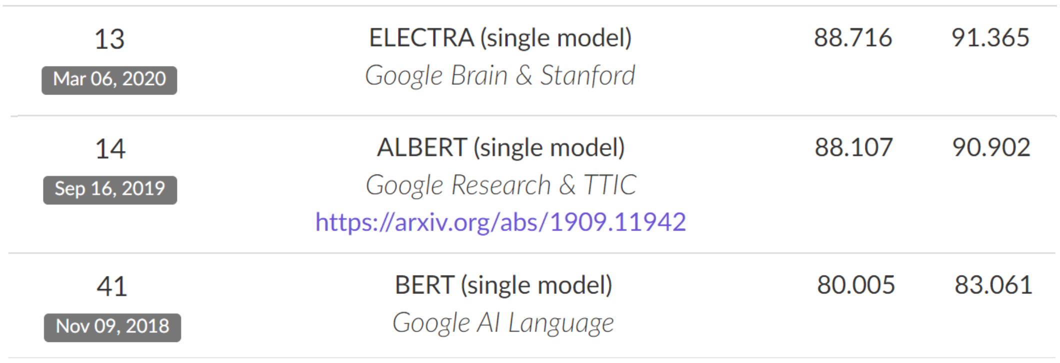
Data analysis

- Data num
    - Train: 901
    - Test : 227
- Data context length(Token)
    - Train and Test
        - Avg token num:
            - Train: 79.82
            - Test: 76.49
        - 

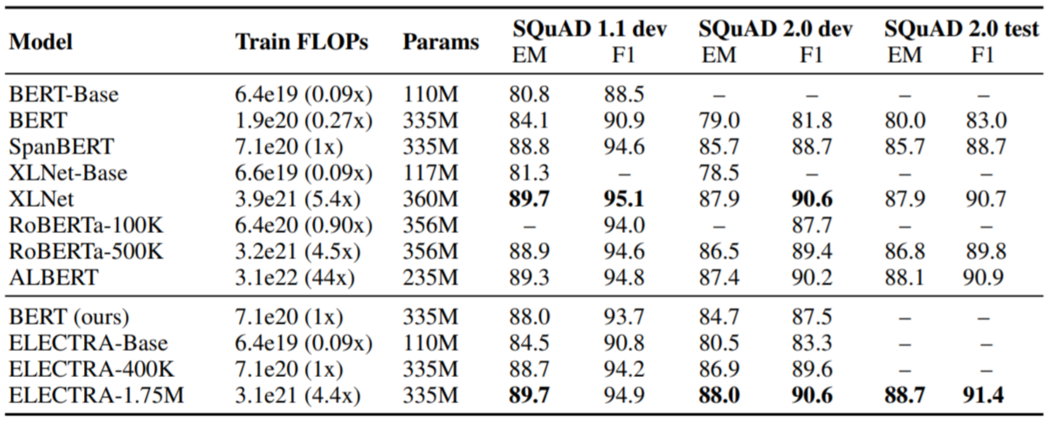
    - Cause and Effect(Train)
        - Avg token num:
            - Cause: 32.43
            - Effect: 33.87
        -
- Analysis

Method

- Expect the performance

- 

13 Mar 06, 2020	ELECTRA (single model) *Google Brain & Stanford*	88.716	91.365
14 Sep 16, 2019	ALBERT (single model) *Google Research & TTIC* https://arxiv.org/abs/1909.11942	88.107	90.902
41 Nov 09, 2018	BERT (single model) *Google AI Language*	80.005	83.061

- 

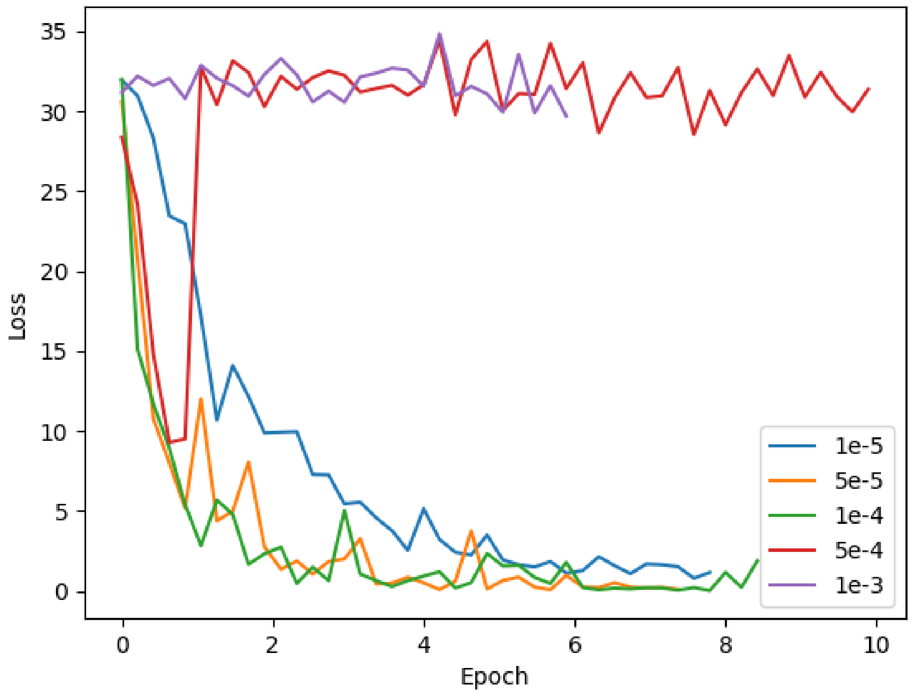
Model	Train FLOPs	Params	SQuAD 1.1 dev		SQuAD 2.0 dev		SQuAD 2.0 test	
			EM	F1	EM	F1	EM	F1
BERT-Base	6.4e19 (0.09x)	110M	80.8	88.5	–	–	–	–
BERT	1.9e20 (0.27x)	335M	84.1	90.9	79.0	81.8	80.0	83.0
SpanBERT	7.1e20 (1x)	335M	88.8	94.6	85.7	88.7	85.7	88.7
XLNet-Base	6.6e19 (0.09x)	117M	81.3	–	78.5	–	–	–
XLNet	3.9e21 (5.4x)	360M	**89.7**	**95.1**	87.9	**90.6**	87.9	90.7
RoBERTa-100K	6.4e20 (0.90x)	356M	–	94.0	–	87.7	–	–
RoBERTa-500K	3.2e21 (4.5x)	356M	88.9	94.6	86.5	89.4	86.8	89.8
ALBERT	3.1e22 (44x)	235M	89.3	94.8	87.4	90.2	88.1	90.9
BERT (ours)	7.1e20 (1x)	335M	88.0	93.7	84.7	87.5	–	–
ELECTRA-Base	6.4e19 (0.09x)	110M	84.5	90.8	80.5	83.3	–	–
ELECTRA-400K	7.1e20 (1x)	335M	88.7	94.2	86.9	89.6	–	–
ELECTRA-1.75M	3.1e21 (4.4x)	335M	**89.7**	94.9	**88.0**	**90.6**	**88.7**	**91.4**

- In the figures above, we find that BERT-Base has 110M parameters and ELECTRA-Base also has 110M parameter. It's suggested that the ability of computing will be similar.
- In the figures above, we also find that the score of SQUAD1.1 of ELECTRA-Base is higher than BERT-Base. So we expect the performance of ELECTRA will be better than the performance of BERT.

- Data preprocessing
  - We find that the same document maybe have more than one cause parts and effect parts. As a result, we collect the same document but different parts of cause and effect into one data. We suppose that only two sets of cause and effect in the same document.
  - Besides, in order to check the correctivity, we divide the train dataset to train part and validation part, where validation part is about 8 percent of original train dataset.
  - According to original cause and effect part, we turn it into the position of token respectively.

Method 1

- ELECTRA base-pretrain Model discriminator
- Trained by different learning rates

- ○ 

    - We can find that if the learning rate is bigger than 5 · 10^{-4} then it can't reach relative minimum loss. In contrast, if learning rate is less than 5 · 10^{-4} then it can reach relative minimum loss to get better performance.

- We input the sentence composed of tokens, which are produced by data preprocessing, into ELECTRA model. Then we know that the output is a embedding vector,

  $\mathbb{R}^{n \times 768}$, where n is the length of token.

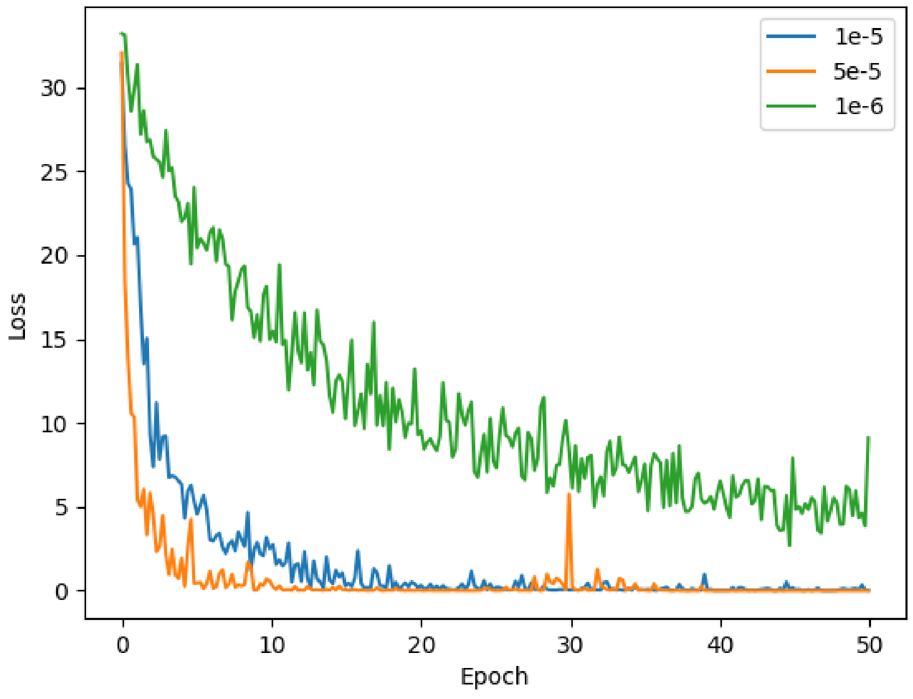
- After that, we will pass this embedding vector into a one layer of linear model,

  $\mathbb{R}^{n \times 768} \rightarrow \mathbb{R}^{n \times 8}$, where 8 dimensions represent the first related cause-start, cause-end, effect-start, effect-end,and the second related cause-start, cause-end, effect-start, effect-end.

- The last, we use softmax to get the most probably position of each dimension and then train the model by comparing with gold standard label.

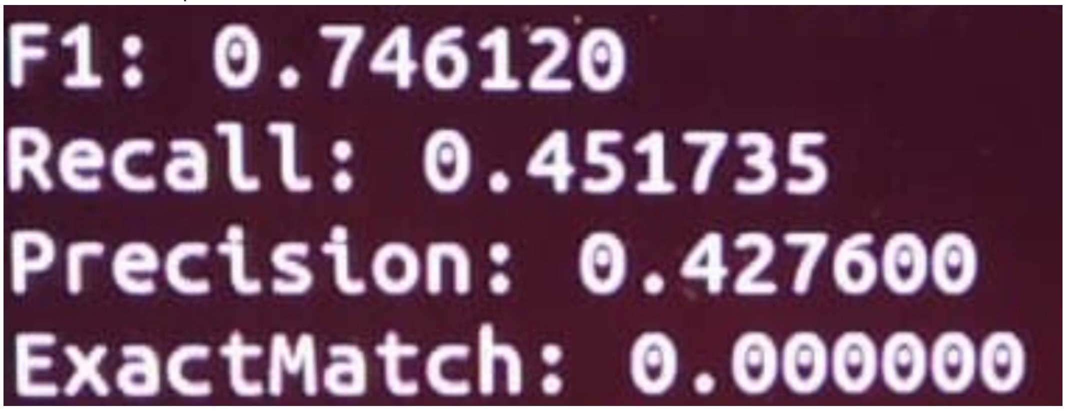
Method 2

- BERT
- Trained by different learning rates

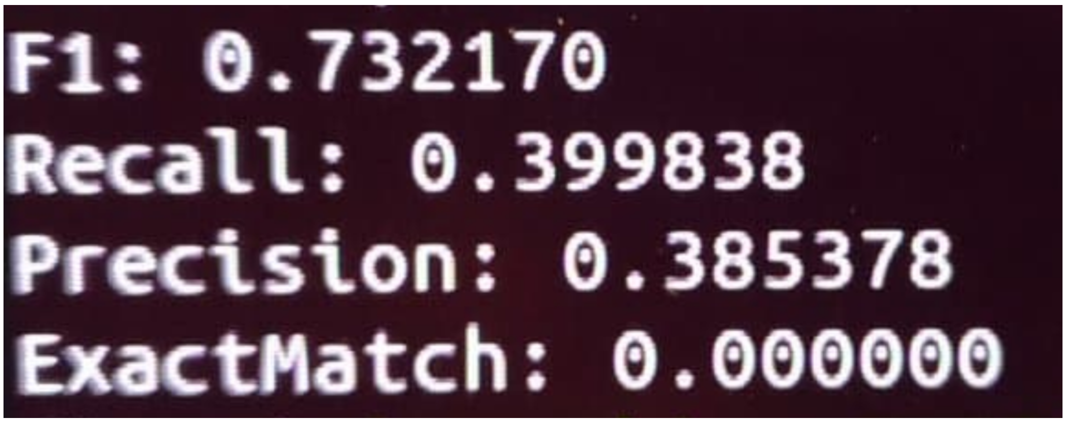
- 

-
- The training method is same with the method 1, just using different pretrained model.

Results

- Method 1
  - ELECTRA base-pretrain Model discriminator
  - 

```
F1: 0.746120
Recall: 0.451735
Precision: 0.427600
ExactMatch: 0.000000
```

- Method 2
  - BERT
  - 

```
F1: 0.732170
Recall: 0.399838
Precision: 0.385378
ExactMatch: 0.000000
```

  - We find that trained based on BERT model has the worse performance.

- We find that maybe there are some wrong print order in task2_evaluate.py because F1 scored should lie between recall score and precision score.
- The performance of using ELECTRA is better than the perforamce of using BERT. We think that the reason is that ELECTRA is a newly model taking different trainning method. Therefore the performance of ELECTRA is better than BERT-base.

Learned from Project

魏任擇：感謝這次的Project隊友，在剛修這門課時完全不懂ML這塊領域，老師講述的都大多是關於理論上的做法，而透過這次Project的學習，讓我了解到如何將所學的應用在實作上，逐漸對NLP這個領域有了些了解和興趣。

楊則軒:這學期的課學到了很多NLP領域不同的做法，由於語言是有高度多樣性的，所以NLP在不同task會有很多不同的處理方法，即使是同一個task也可能會有許多不同的做法，在這次的project中就發現當資料量差距很大時，不能用普通分類的方法來訓練模型。

陳彥斌：感謝組員一起合作，讓我們更熟悉pretrained model的使用，並從task1中得知想要從頭訓練ELMo(任何NN的language model)，需要相當多的運算資源/時間，有好的pretrained model能讓我們不用重新製造輪子，能直接站在巨人的肩膀完成很厲害的自然語言處理問題。

龔柏年：從學期初上課到現在，學了各種NLP task的做法，但是都比較像是學了方法，不知道確切怎麼用。Final project的實作讓我們對NLP 模型（深度學習的）有更深的了解，也學會了使用一些當前比較紅的transformers 模型，覺得對上課教的這些知識感覺有了更進一步的理解。

Team cooperate

陳彥斌：主要程式撰寫、模型訓練、增進模型表現
楊則軒：實驗不同參數及結果分析(task1)
龔柏年：資料分析、處理
魏任擇：實驗不同參數及結果分析(task2)

Reference

https://openreview.net/pdf?id=r1xMH1BtvB
https://github.com/huggingface/transformers
HuggingFace's Transformers: State-of-the-art Natural Language Processing