

Team_BC

2021 국립국어원

인공지능 언어능력 평가

김강민 김현우 이동빈

신문종 서석민 임도훈

INDEX

- 1 | 인과 관계 추론 : COPA
- 2 | 문장 문법성 판단 : COLA
- 3 | 동형이의어 구별 : WiC
- 4 | 판정의문문 : BoolQ
- 5 | 모델 구동 과정

이 인과 관계 추론

인과 관계 추론 : 입력 생성 방식

```
def load_data(dataset_dir):
    dataset = pd.read_csv(dataset_dir, delimiter='\t', names=['ID', 'sentence', 'question', '1', '2', 'answer'], header=0)
    dataset["label"] = dataset["answer"].astype(int) - 1

    new_sentence1_1 = []
    new_sentence1_2 = []
    new_sentence2_1 = []
    new_sentence2_2 = []
    for i in range(len(dataset)):
        s = dataset.iloc[i]['sentence']
        q = dataset.iloc[i]['question']
        s1 = dataset.iloc[i]['1']
        s2 = dataset.iloc[i]['2']
        lb = dataset.iloc[i]['label']

        if q == "결과":
            new_sentence1_1.append("[결과]" + s)
            # new_sentence1_1.append(s)
            new_sentence1_2.append(s1)
            new_sentence2_1.append("[결과]" + s)
            # new_sentence2_1.append(s)
            new_sentence2_2.append(s2)

        else:
            new_sentence1_1.append("[원인]" + s1)
            # new_sentence1_1.append(s1)
            new_sentence1_2.append(s)
            new_sentence2_1.append("[원인]" + s2)
            # new_sentence2_1.append(s2)
            new_sentence2_2.append(s)

    dataset["new_sentence1_1"] = new_sentence1_1
    dataset["new_sentence1_2"] = new_sentence1_2
    dataset["new_sentence2_1"] = new_sentence2_1
    dataset["new_sentence2_2"] = new_sentence2_2

    return dataset
```

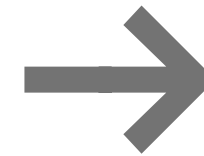
[주의] question이 원인, 결과에 따라 문장 순서가 뒤집어짐

Sentence : 전쟁이 시작되었다.

Question : 결과

1 : 병사들이 집으로 돌아왔다.

2 : 병사들이 전투에 파견되었다.



S1 : [결과] 전쟁이 시작되었다. 병사들이 집으로 돌아왔다.

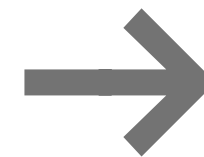
S2 : [결과] 전쟁이 시작되었다. 병사들이 전투에 파견되었다.

Sentence : 차가 찌그러졌다.

Question : 원인

1 : 운전자가 차에 시동을 걸었다.

2 : 운전자가 전신주에 차를 박았다.



S1 : [원인] 운전자가 차에 시동을 걸었다. 차가 찌그러졌다.

S2 : [원인] 운전자가 전신주에 차를 박았다. 차가 찌그러졌다.

인과 관계 추론 : 모델 구성

```

class Roberta(RobertaModel):
    def __init__(self, config, args):
        super(Roberta, self).__init__(config)
        self.roberta = RobertaModel.from_pretrained("klue/roberta-large", config=config) # Load pretrained Electra

        self.num_labels = config.num_labels

        self.pooling = PoolingHead(input_dim=config.hidden_size,
                                    inner_dim=config.hidden_size,
                                    pooler_dropout=0.1)
        self.qa_classifier = nn.Linear(config.hidden_size, self.num_labels - 1)

    def forward(self, input_ids=None, attention_mask=None, token_type_ids=None, input_ids2=None, attention_mask2=None, token_type_ids2=None, labels=None):
        outputs = self.roberta(
            input_ids, attention_mask=attention_mask
        ) # sequence_output, pooled_output, (hidden_states), (attentions)
        outputs2 = self.roberta(
            input_ids2, attention_mask=attention_mask2
        )
        sequence_output = outputs[0]
        sequence_output2 = outputs2[0]
        pooled_output = outputs[0][:, 0, :] # [CLS]
        pooled_output2 = outputs2[0][:, 0, :]

        sentence_representation = torch.cat([pooled_output, pooled_output2], dim=1)

        pooled_output = self.pooling(pooled_output)
        pooled_output2 = self.pooling(pooled_output2)

        logits1 = self.qa_classifier(pooled_output)
        logits2 = self.qa_classifier(pooled_output2)

        logits = torch.cat([logits1, logits2], dim=1)

        outputs = (logits,) + outputs[2:] # add hidden states and attention if they are here

    return outputs # logits, (hidden_states), (attentions)

```

```

> outs = model(**item)
  loss = criterion(outs[0], item['labels'])

  preds = torch.argmax(outs[0], dim=-1)

```

각 backbone model을 2번 통과하여 두개의 output중 확률값이 높은 index를 선택

Bert모델 등에서 학습했던, 문장 순서 예측을 [CLS]토큰에서 한 것과 유사한 방식을 사용

인과 관계 추론 : Tuning

```
class Roberta(RobertaModel):
    def __init__(self, config, args):
        super(Roberta, self).__init__(config)
        self.roberta = RobertaModel.from_pretrained("klue/roberta-large", config=config) # Load pretrained Electra

        self.num_labels = config.num_labels

        self.pooling = PoolingHead(input_dim=config.hidden_size,
                                   inner_dim=config.hidden_size,
                                   pooler_dropout=0.1)
        self.qa_classifier = nn.Linear(config.hidden_size, self.num_labels - 1)

    def forward(self, input_ids=None, attention_mask=None, token_type_ids=None, input_ids2=None, attention_mask2=None, token_type_ids2=None, labels=None):
        outputs = self.roberta(
            input_ids, attention_mask=attention_mask
        ) # sequence_output, pooled_output, (hidden_states), (attentions)
        outputs2 = self.roberta(
            input_ids2, attention_mask=attention_mask2
        )
        sequence_output = outputs[0]
        sequence_output2 = outputs2[0]
        pooled_output = outputs[0][:, 0, :] # [CLS]
        pooled_output2 = outputs2[0][:, 0, :]

        sentence_representation = torch.cat([pooled_output, pooled_output2], dim=1)

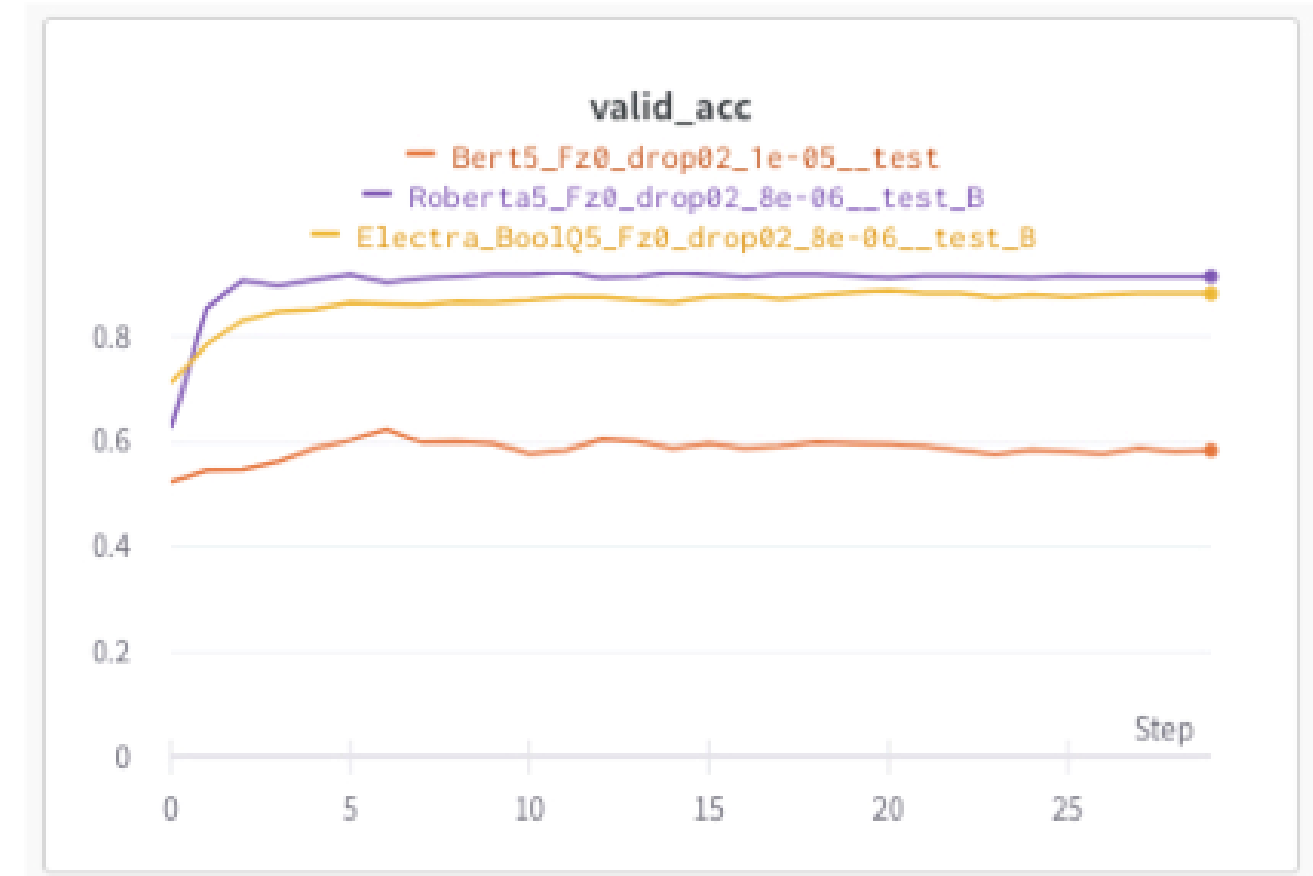
        pooled_output = self.pooling(pooled_output)
        pooled_output2 = self.pooling(pooled_output2)

        logits1 = self.qa_classifier(pooled_output)
        logits2 = self.qa_classifier(pooled_output2)

        logits = torch.cat([logits1, logits2], dim=1)

        outputs = (logits,) + outputs[2:] # add hidden states and attention if they are here

        return outputs # logits, (hidden_states), (attentions)
```



Bert, Roberta, Electra model에 대하여 Hyperparameter 튜닝 진행.

- Roberta 모델이 val_acc 92.6으로 최고 성능
- KLUE/Roberta-large 채택

인과 관계 추론 : 추가 번역 Data 사용

```
from selenium import webdriver
from selenium.webdriver.chrome.options import Options
from selenium.webdriver.common.by import By
```

```
from selenium import webdriver
import time
```

```
chrome_options = webdriver.ChromeOptions()
chrome_options.add_argument('--headless')
chrome_options.add_argument('--no-sandbox')
chrome_options.add_argument("--single-process")
chrome_options.add_argument("--disable-dev-shm-usage")
path="/usr/bin/chromedriver"
driver = webdriver.Chrome(path, chrome_options=chrome_options)
driver.get("https://papago.naver.com/")
search_box = driver.find_element_by_css_selector('textarea')
search_box.send_keys('ok there you go.')
time.sleep(2)
search_box.clear()
answer = driver.find_element_by_css_selector('div#txtTarget')
print(driver.title)
driver.quit()
```

외부 데이터

```
<item id="1" asks-for="cause" most-plausible-alternative="1">
  <p>My body cast a shadow over the grass.</p>
  <a1>The sun was rising.</a1>
  <a2>The grass was cut.</a2>
</item>
```

번역 데이터

Sentence : 나의 몸은 그 풀에 그림자를 드리웠다.
 Question : 원인
 1 : 해가 뜨고 있었다.
 2 : 풀이 깎였다
 Answer : 1

Copa 영어 dataset을 번역하여 train에 포함시켜 학습

- valid accuracy : 92.6 -> 93.2

- 출처 : <https://people.ict.usc.edu/~gordon/copa.html>

[최종 제출] Roberta base + Roberta_adddata soft ensemble

02 문장 문법성 판단

문장 문법성 판단 : EDA

문법적으로 문제가 있는 문장들에는 추가적으로 속성값 [“*”, “?”]이 존재

- 오답인 이유를 분류하여 제공한 것으로 판단하였으나 명확한 분류 기준을 파악하기 어려움

오답의 종류를 크게 세가지로 분류 가능

- 부적절한 도치로 인한 맥락상의 오류 : “사람은 죽는다. 누구나”
- 부적절한 조사, 접미사, 종결어미 사용 : “철수를 영희를 좋아한다”
- 부정/긍정 의미의 부사 및 접속사가 부정/긍정인 문맥을 역접하는 경우 : “철수는 좀처럼 공부하였다.”

국어 지식을 이용하여 여러 문법적인 오류를 잡도록 새로운 special token을 이용해 학습하는 모델을 만들기는 현실적으로 어려움.

주어진 학습Set와 비슷한 데이터로 학습한 모델을 사용하는 것이 중요하다고 판단

- 문장 내 형태소들의 순서를 변경하거나 형태소 자체를 변경한다면 의미 있는 오답 문장을 추가적으로 생성해 학습시킬 수 있을 것

문장 문법성 판단 : 모델 선정 및 결과

Bert (monologg/ko-bert)

- Vocab size : 8002
- Valid acc : 0.6458 (LB matthew's Corr : 22.61)
- Vocab size에 의해 <unk> Token이 많이 발생하는 문제 발생

RoBerta (KLUE/Roberta-large)

- Vocab size : 32000
- Valid acc : 0.74 (LB matthew's Corr : 48.86)
- KoBERT에 비해 큰 Vocab size와 단어예측 사전학습 방식에 의해 성능 향상

Electra (monologg/koelectra, Tunib/electra)

- Vocab size : 30000
- Valid acc : 0.76 (LB matthew's Corr : 57~58)
- 단어예측에 원본여부 판단이 추가되어진 사전학습 방법이 문법성 판단에 적합하여 성능향상

문장 문법성 판단 : 성능 향상 시도

Data augmentation

- 생성모델 bart와 electra를 생성자와 판별자로 이용한 semi-supervised learning (성능 하락)

추가적인 데이터 학습

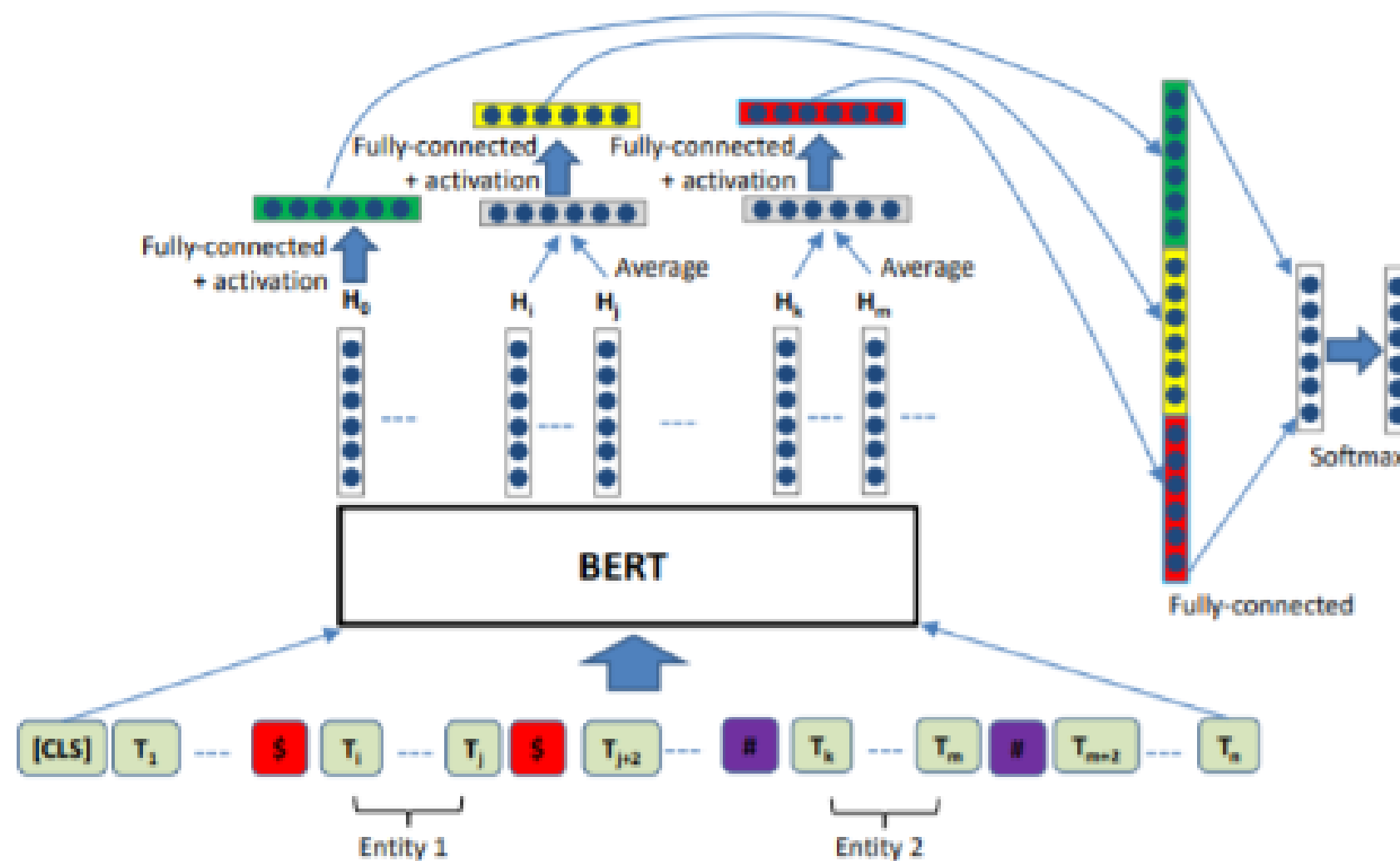
- 모두의 말뭉치(신문 말뭉치 2020)에서 문장길이 20 ~ 30사이 문장을 선별
(출처 : <https://corpus.korean.go.kr/main.do#none>)
- 형태소 분석기로 부사, 조사 치환하여 문법 오류 문장 생성
- Matt's corr기준 단일모델 58.79(tunib-electra) 기록

최종 제출

- Model 1 : Tunib/electra를 사용해 CLS Token만을 이용한 분류
- Model 2 : Tunib/electra를 사용해 모든 Token의 평균을 이용한 분류 + 55만개의 신문말뭉치 추가
- Model 3 : monologg/koelectra를 사용해 9Fold 에서 각 최고 성능 모델의 예측확률을 평균내어 분류
- 위 세가지 모델의 예측확률을 SoftVoting Ensemble 후 최종 제출

03 동형이의어 구별

동형이의어 구별 : 구조



R-BERT

1. Sentence with Special token
 - Insertion of the special tokens
2. using pretrained model
 - Feature extraction by pretrained model
3. Concat and Classification
 - Concat $[CLS]$, Entity1, Entity2
 - Last layers and classification

동형이의어 구별 : 구조

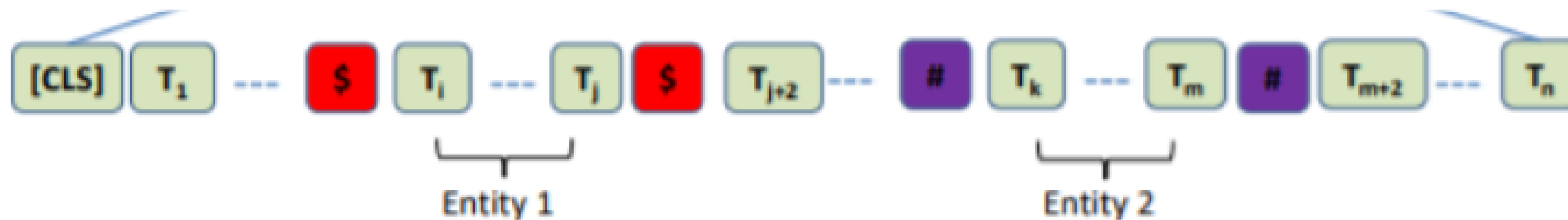
1. Sentence with Special token

- Entity의 위치 강조해줄 수 있으며, Pretrained language model이 해당 위치 정보를 포착하게 도와줌
- 문장 앞에 [CLS] Token을 추가, Entity사이에도 special token을 넣어줍니다.

ex. sentence 1 : 백제는 의자왕의 방탕과 실정으로 패망했다고 보는 견해가 있다.

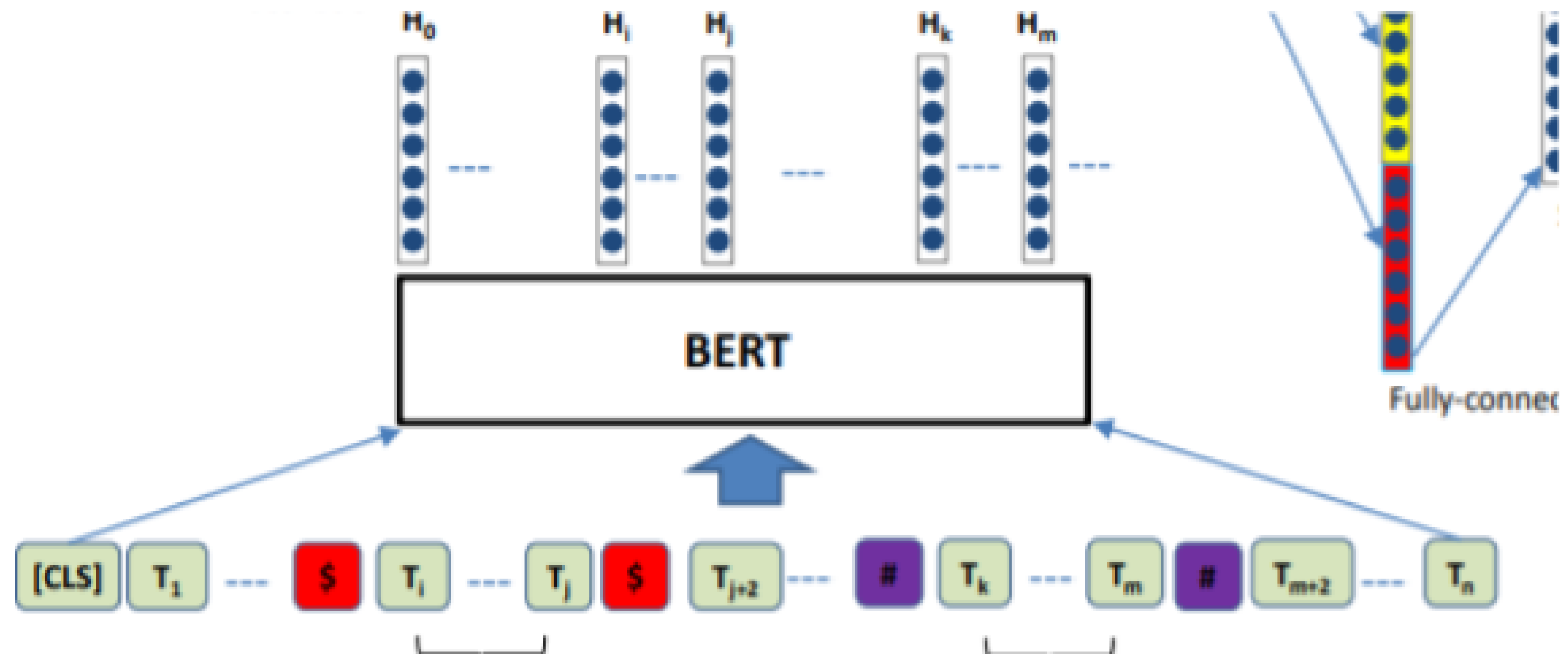
sentence 2 : 지금 회사는 자금난으로 공장을 운영할 수 없는 실정이다.

Input : <s> 백제는 의자왕의 방탕과 <e1> 실정 </e1> 으로 패망했다고 보는 견해가 있다.</s> <s>지금 회사는 자금난으로 공장을 운영할 수 없는 <e2> 실정 </e2> 이다.</s>



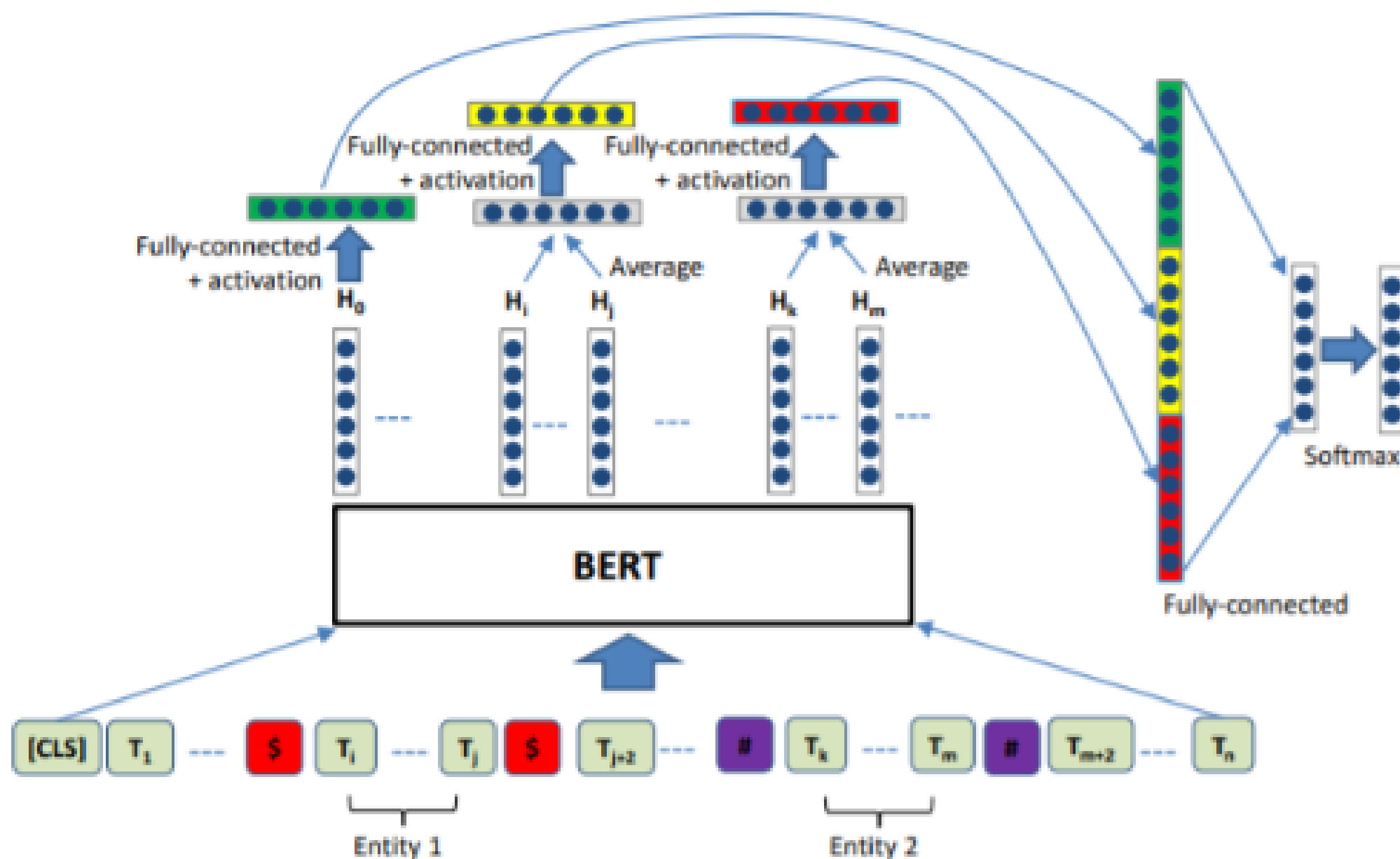
동형이의어 구별 : 구조

2. Pretrained Language model(PLM, 사전학습모델)
- Input : Token으로 분리되어 있는 문장
 - Output : 문장에 대한 사전학습 기반 Representation



동형이의어 구별 : 구조

3. Concat and Classification



$$H'_1 = W_1 \left[\tanh \left(\frac{1}{j-i+1} \sum_{t=i}^j H_t \right) \right] + b_1$$

$$H'_2 = W_2 \left[\tanh \left(\frac{1}{m-k+1} \sum_{t=k}^m H_t \right) \right] + b_2$$

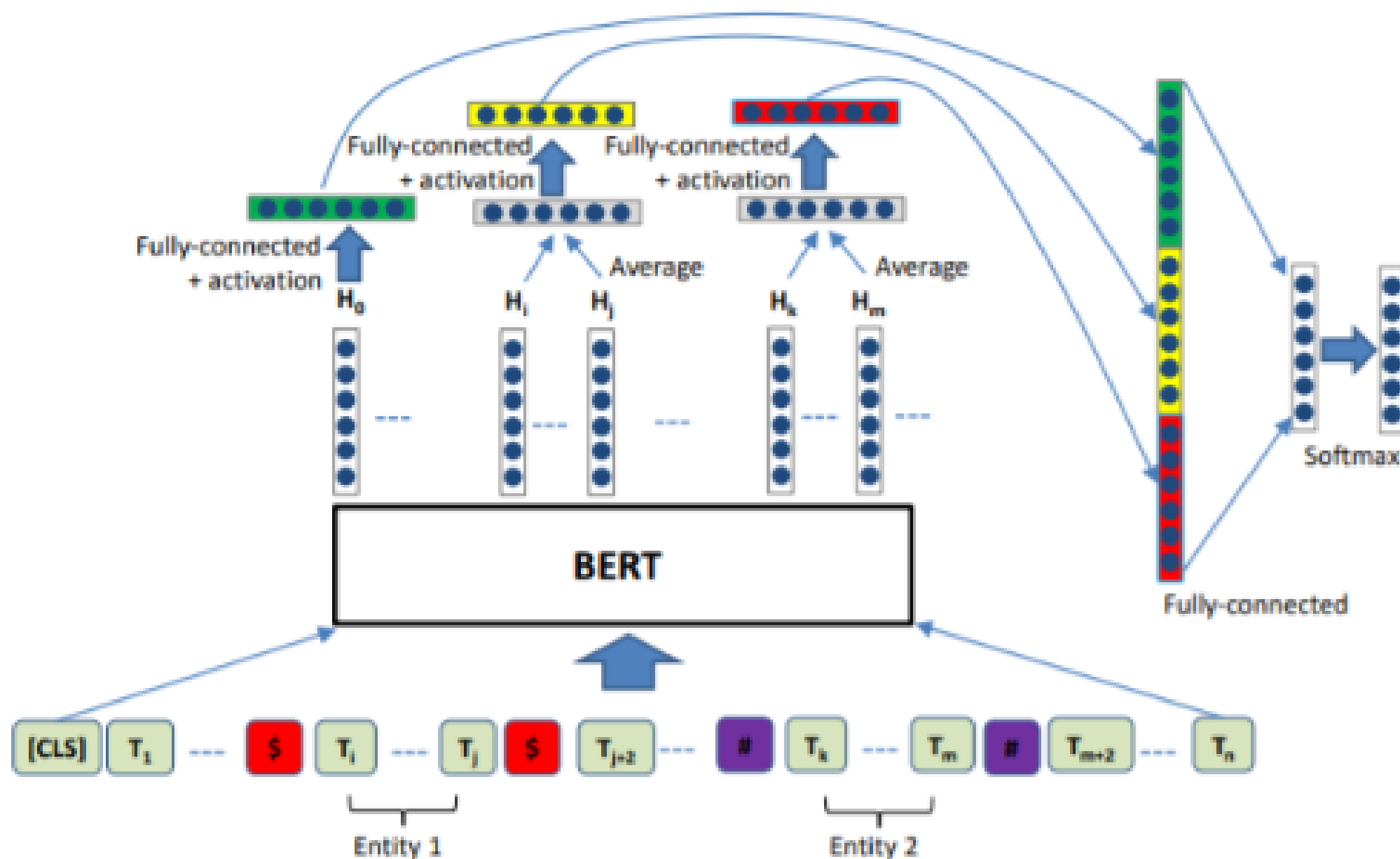
Entity token들의 hidden vector을
average후 Linear layer 통과

$$H'_0 = W_0 (\tanh(H_0)) + b_0$$

[CLS] token의 hidden vector를
Linear layer 통과

동형이의어 구별 : 구조

3. Concat and Classification



$$h'' = W_3 \left[\text{concat} (H'_0, H'_1, H'_2) \right] + b_3$$

$$p = \text{softmax}(h'')$$

모든 Vectors들을 Concat하여
Linear layer 통과 후 Softmax값을 취해
동형이의어 구별

동형이의어 구별 : 시도 및 최종 제출

Pretrained Model 선정 (Validation accuracy 기준)

- Tunib/koelectra : 89.97
- monologg/koelectra : 90.99
- KLUE/roberta-large : 91.25

최종 제출

- KLUE/roberta-large를 5-Fold로 나누어 학습한 후 SoftVoting Ensemble (Leader Board accuracy 기준)
- roberta-large : 88.52
- roberta-large(wih k-fold(k=5) & soft voting) : 91.65

04 판정의문문

판정의문문 : Dataset 구성

예시

- Text : 고위공직자범죄수사처는 고위 공직자 및 대통령 친인척의 범죄행위를 상시적으로 수사·기소할 수 있는 독립된 국가기관 이다.
- Question: 고위공직자범죄수사처는 대통령 친인척을 수사할 수 있나?
- [CLS] + Text + [SEP] + Question + [SEP]

Tokenize Option

- Max length : 350
- Padding : True
- Truncation : True

판정의문문 : 시도(Modeling)

다양한 사전학습 모델 사용

- Electra : monologg/koelectra, Tunib/koelectra
- Roberta : KLUE/Roberta 채택
- GPT2 : SKT/Kogpt2
- Bart : KoBart

[CLS] weight sharing

- 일반적으로 Transformer Layer를 통과한 output에서 [CLS] Token을 이용해 분류를 진행
- 판정의문문의 경우 다수의 문장으로 Text가 구성
- 각 문장의 시작에 [CLS] Token을 사용하여 [CLS] Token의 Weight를 공유하도록 구성하여 실험 진행
- 성능 향상

판정의문문 : 시도(전처리)

```
def pre_process(st):
    st = re.sub('\(.|\)|\s-\s.*', '', st) # () 괄호도형 사이 문장 제거
    st = re.sub('\[.|\]|\s-\s.*', '', st) # [] 괄호도형 사이 문장 제거
    st = st.lower() # 영어들은 모두 소문자로 변환

    # 같은 의미를 가지는 특수문자들을 통일
    st = re.sub('[~]', '\'', st)
    st = re.sub('[']', '\'', st)
    st = re.sub('[>>> > >]', '>', st)
    st = re.sub('[<<< < <]', '<', st)
    st = re.sub('[---]', '-', st)
    st = re.sub('[...]', '.', st)
    st = st.replace('/', '/')
    st = st.replace('℃', '도')
    st = st.replace('+', '에서')
    st = st.replace('!', '')
    st = st.replace('.', ',')
    st = st.replace('m', 'km')
    st = st.replace('~', '~')
    st = st.replace('mm', 'mm')
    st = st.replace('x', '곱하기')
    st = st.replace('=', '는')
    st = st.replace('8', '')
    st = st.replace('ml', 'ml')
    st = st.replace('e', 'l')
    st = st.replace('°C', '도')
    st = st.replace('°', '도')
    st = st.replace('°C', '도')
    st = st.replace('°', '도')
    st = st.replace('+', '+')
    st = st.replace('*', '')
    st = st.replace(':', ':')

    return st
```

1. (), [] 등의 괄호 내의 문장들의 경우 아랍어, 힌두어 등의 다국어가 다수 포함
- 괄호 내의 내용을 통해 질문의 답을 내야하는 경우가 없었기에 괄호포함 사이 문장 제거

- ## 2. 영어의 경우 모두 소문자로 변환

- ### 3. 같은 의미를 가지는 특수문자들을 통합

- 여|시|

```
st = st.replace('°C', '도')
st = st.replace('°C', '도')
st = st.replace('°', '도')
st = st.replace('°C', '도')
```

결과

- 8 Batch의 경우 성능 향상
- 16 Batch의 경우 성능 감소

판정의문문 : 최종제출

- 16 Batch로 5-Fold 진행
- 5번째 fold가 성능이 가장 낮아 앙상블에서 제외
- Model 1 : 1~4 Fold
- Model 2 : 기존 Train/Dev set 학습 모델
- Model 3 : 기존 Train/Dev set + [CLS] weight sharing 학습 모델

최종적으로 위의 Model 1,2,3을 Soft Voting Ensemble을 통해 예측하여 제출

05 모델 구동 과정

모델 구동 과정

NIKL-GLUE

pretrained model & data

need extract inside each task folder

boolq : <https://drive.google.com/file/d/1k-6W3bTqFVlSBBtsJ42kPcBdBCO7bQkD/view?usp=sharing>

copa : <https://drive.google.com/file/d/1PxwwOiYxKb7PUBVByX0LYwnXm9lY6lfh/view?usp=sharing>

cola : <https://drive.google.com/file/d/1DJDch9YquGUrf9GQBwD64X0ecrSAdUbS/view?usp=sharing>

wic : <https://drive.google.com/file/d/1DJDch9YquGUrf9GSEJR4X0ecrSAdUbS/view?usp=sharing>

Docker inference

1. make docker image

```
docker build -t inference-docker -f Dockerfile .
```

2. run docker container

```
docker run -v "$PWD:/mnt/work" -it inference-docker
```





3. do inference

```
sh all_inference.sh
```

1. Github Readme 확인

2. 각 Task별 Pretrain 다운로드

3. 각 폴더에 다운로드한 압축파일 해제

	BoolQ	init	2 days ago
	Cola	modifying data path	14 minutes ago
	Copa	init	2 days ago
	WIC	docker file	2 days ago

4. Docker Inference 순서대로 진행