NLP 파트 - ElectraSpacer - # 19

ElectraSpacer

1. Data Architecture

(1) Dtata Format

- ElectraSpacer 모델을 거치기 위한 데이터 포맷은 csv 파일 형식
- 파일을 DataFrame으로 읽고, DataFrame의 feature는 'wrong_sentence'와 'correct_sentence'를 입력 값으로 받음
- correct_sentence는 올바른 띄어쓰기가 된 문장이고 wrong_sentence는 띄어쓰기 공백을 모두 제거한 문장

```
# example
wrong_sentence = 나는걸어가고있는중입니다.나는밥을먹고있는중입니다.
correct_sentence = 나는 걸어가고 있는 중입니다. 나는 밥을 먹고 있는 중입니다.
```

(2) Dataset

```
,correct_sentence,wrong_sentence
0,나는 철수에게 공을 던져다 주었다.,나는철수에게공을던져다주었다.
1,먹은 것을 다 소화시켜야 한다.,먹은것을다소화시켜야한다.
2,그가 노래를 부르고는 내가 피아노를 쳤다.,그가노래를부르고는내가피아노를쳤다.
3,철수가 영수의 손을 잡아서 눈물을 글썽거렸다.,철수가영수의손을잡아서눈물을글썽거렸다.
4,그의 업적은 길이 빛난다.,그의업적은길이빛난다.
5,별이 반짝반짝한다.,별이반짝반짝한다.
```

- train.csv : 15,876개
- dev.csv : 1,060개
- test.csv : 1,060개

2. Model Architecture

(1) Preprocessing

1) Tokenizer

- 보통 Tokenizing은 WordPiece(단어) 단위로 토큰화
- ElectraSpacer는 아래와 같이 KoCharElectraTokenizer를 사용하여 character(음절) 단위로 토큰화
- 음절 단위로 Tokenizing 할 경우 띄어쓰기의 위치를 더 정확히 판단

```
from tokenization_kocharelectra import KoCharElectraTokenizer
tokenizer = KoCharElectraTokenizer.from_pretrained("monologg/kocharelectra-base-discriminator")
tokenizer.tokenize("나는 걸어가고 있는 중입니다.")
```

```
['나', '는', ' ', '걸', '어', '가', '고', ' ', '있', '는', ' ', '중', '입', '니', '다', '.']
```

2) [CLS]&[SEP] Token 추가

```
tokens : [CLS] 나 는 걸 어 가 고 있 는 중 입 니 다 . [SEP] 나 는 밥 을 먹 고 있 는 중 입 니 다 . [SEP]
```

3) Token to Index

```
input_ids: 2 40 8 5 374 38 14 13 5 36 8 5 75 142 57 7 10 3 40 8 5 733 11 5 445 13 5 36 8 5 75 142 57 7 10 3 0 0 0 0
token_type_ids: 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0
attention_mask: 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0
```

(2) Model : ELECTRA

```
[Last layer hidden state]
Size: torch.Size([1, 40, 768])
Tensor: tensor([[[ 0.1453, -0.0629,  0.2065, ...,  0.5304, -0.4602,  0.6803],
                 [ 0.8824, -0.3448, -0.3342, ...,  0.4652, -0.2378,  0.2560],
                 [ 0.3114, -0.3019, -0.1159, ...,  0.4712, -0.6678,  0.3425],
                 ...,
                 [-0.0830, -0.2008,  0.2107, ..., -0.2890, -0.0297,  0.5241],
                 [ 0.0587, -0.2498,  0.4193, ..., -0.2537,  0.1526,  0.5394],
                 [ 0.1337, -0.2736,  0.6251, ..., -0.1580,  0.2323,  0.5248]]])
```

- Token to Index 한 결과 값들을 ELECTRA 모델을 거치기 전에 Embedding layer를 거침
- outputs.last_hidden_state 찍어보면 위와 같은 결과가 나옴
- torch.Size([1, 40, 768]) -> [1문장, 문장의 길이, 차원]
- 문장 한 개에 속한 각각의 토큰(최대 길이 10)을 768 차원의 벡터로 변환했다는 의미

(3) BiLSTM-CRF

- LSTM 모델 input 값으로 last_hidden_state을 받아 새로운 Embedding Vector를 출력
- 이 과정에서 Overfitting 방지를 위해 dropout을 해줌
- 최종적으로 문장의 인덱스가 0이면 'B', 나머지 인덱스 값은 'I'로 태그

```
sentence : 나는 걸어가고 있는 중입니다. 나는 밥을 먹고 있는 중입니다.
BI tag : B I B I I I B I B I I I I B I B I B I B I B I I I I
```

3. Train Architecture

(1) Train Model

1) Model Config

```
tokenizer_name: "monologg/kocharelectra-small-discriminator"
model_name: 'monologg/kocharelectra-small-discriminator'
train_data_path: '../data/train.csv'
val_data_path: '../data/dev.csv'
test_data_path: '../data/test.csv'
model_path: './models'

# training arguments
output_dir: './results'
max_len: 256
epochs: 5
steps: 100
batch_size: 32
```

2) Optimizer

```
# Optimizer : Adam
# Learning Rate : 0.00001
optimizer = torch.optim.Adam(params=model.parameters(), lr=1e-05)
```

(2) Metrics

1) Calculate

$$\textbf{Accuracy} = \frac{\#\ of\ correct\ BI\ taggings}{\#\ of\ chracters}$$

$$\textbf{Boolean Accuracy} = (prediction == ground\ truth)$$

$$\textbf{F1 Score} = \frac{2*(precision*recall)}{(precision+recall)}, Precision\ and\ recall\ are\ calculated\ by\ tag\ 'B'$$

$$\textbf{Word Error Rate}(\textbf{WER}) = \frac{\#\ of\ Substitutions+\#\ of\ Deletions+\#\ of\ Insertions}{\#\ of\ Words\ in\ the\ ground\ truth}$$

2) Example

Ground Truth
안녕 나는 범블비야 반가워
B I B I B I I I B I I -> Prediction
안녕 나는 범 블비야반 가워
B I B I B B I I I B I -> $Accuracy = \frac{8}{11}$
$Boolean\ Accuracy = 0$
$F1\ Score = \frac{2}{3}$
$Word\ Error\ Rate = \frac{3}{4}$

4. Output

(1) Inference


```
{
        "0":  [
                "나는철수에게공을던져다주었다.",
                "나는  철수에게  공을  던져다  주었다.",
                "  나는  철수에게  공을  던져다  주었다."
        ],
        "1":  [
                "먹은것을다소화시켜야한다.",
                "먹은  것을  다  소화시켜야  한다.",
                "  먹은  것을  다소화시켜야  한다."
        ],
        "2":  [
                "그가노래를부르고는내가피아노를쳤다.",
                "그가  노래를  부르고는  내가  피아노를  쳤다.",
                "  그가  노래를  부르고는  내가  피아노를  쳤다."
        ],
        "3":  [
                "철수가영수의손을잡아서눈물을글썽거렸다.",
                "철수가  영수의  손을  잡아서  눈물을  글썽거렸다.",
                "  철수가  영수의  손을  잡아서  눈물을  글썽거렸다."
        ],
        "4":  [
                "그의업적은길이빛난다.",
                "그의  업적은  길이  빛난다.",
                "  그의  업적은  길이  빛난다."
        ],
```


...


```
        "1057":  [
                "그는나를바보여긴다.",
                "그는  나를  바보  여긴다.",
                "  그는  나를  바보여긴다."
        ],
        "1058":  [
                "수호는모든일에전혀무감각하다.",
                "수호는  모든  일에  전혀  무감각하다.",
                "  수호는  모든  일에  전혀  무감각하다."
        ],
        "1059":  [
                "나는할아버지가제일무서우시다.",
                "나는  할아버지가  제일  무서우시다.",
                "  나는  할아버지가  제일  무서우시다."
        ]
}
```


- inference.py 실행하면 results 경로에 prediction.json 파일 생성

- 추론 결과는 위에서 wrong_sentence, correct_sentence, predict_sentence 순으로 출력

(2) Predict

1) Method #1

```
,Unnamed: 0,correct_sentence,wrong_sentence,predict_sentence
0,0,나는 철수에게 공을 던져다 주었다.,나는철수에게공을던져다주었다., 나는 철수에게 공을 던져다 주었다.
1,1,먹은 것을 다 소화시켜야 한다.,먹은것을다소화시켜야한다., 먹은 것을 다소화시켜야 한다.
2,2,그가 노래를 부르고는 내가 피아노를 쳤다.,그가노래를부르고는내가피아노를쳤다., 그가 노래를 부르고는 내가 피아노를 쳤다.
3,3,철수가 영수의 손을 잡아서 눈물을 글썽거렸다.,철수가영수의손을잡아서눈물을글썽거렸다., 철수가 영수의 손을 잡아서 눈물을 글썽거렸다.
4,4,그의 업적은 길이 빛난다.,그의업적은길이빛난다., 그의 업적은 길이 빛난다.
5,5,별이 반짝반짝한다.,별이반짝반짝한다., 별이 반짝 반짝한다.
6,6,영수와 철수가 같이하였다.,영수와철수가같이하였다., 영수와 철수가 같이 하였다.
7,7,화초가 시들었다. 꽃밭에 물을 줄까?,화초가시들었다.꽃밭에물을줄까?, 화초가 시들었다. 꽃밭에 물을 줄까?
8,8,철호가 성실하다고 나에 의해 보인다.,철호가성실하다고나에의해보인다., 철호가 성실하다고 나에 의해 보인다.
9,9,노래는 순이가 부르고는 춤은 추지 않는다.,노래는순이가부르고는춤은추지않는다., 노래는 순이가 부르고는 춤은 추지 않는다.
10,10,철수가 제 방에 가서 잔다.,철수가제방에가서잔다., 철수가 제방에 가서 잔다.
11,11,성은 나서 날뛰는 사람은 물과 불을 모른다.,성은나서날뛰는사람은물과불을모른다., 성은 나서 날뛰는 사람은 물과 불을 모른다.
```

- predict.py를 실행하면 results 경로에 prediction.csv 파일 생성
- wrong_sentence를 input 값으로 받아서 곧바로 띄어쓰기를 예측해줌

2) Method #2

```
electraspacer = ElectraSpacer()
```

```
{
    "0": [
        "안녕하세요저는연구소직원입니다.",
        "안녕하세요저는연구소직원입니다.",
        " 안녕하세요 저는 연구소 직원입니다."
    ]
}
```

- predict.py를 실행하면 results 경로에 prediction.json 파일 생성
- 클래스를 직접 불러와 문장을 입력하여 곧 바로 띄어쓰기 예측

파일

clipboard-202207111644-adfl9.png	14.2 KB	2022-07-11	신재영
clipboard-202207121009-dk4sy.png	45.6 KB	2022-07-12	신재영
clipboard-202207121016-ofe40.png	6.78 KB	2022-07-12	신재영
clipboard-202207121016-rjsux.png	6.92 KB	2022-07-12	신재영
clipboard-202207121017-xfe5m.png	10.3 KB	2022-07-12	신재영
clipboard-202207121117-vxzsm.png	33.1 KB	2022-07-12	신재영
clipboard-202207121123-rbj8k.png	4.67 KB	2022-07-12	신재영