Final project with RoBERTa

중어중문학과 박사과정 이다연

사용모델 : **KLUE RoBERTa base**

1. Transformers 설치

```
!pip install transformers
```

2. 모델과 토크나이저 설정

```
model = AutoModelForSequenceClassification.from_pretrained("klue/roberta-base")
tokenizer = AutoTokenizer.from_pretrained("klue/roberta-base")
```

3. Cuda 설정, learning rate 설정 등

```
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
random.seed(24)
np.random.seed(24)
torch.manual_seed(24)
model.to(device)
learning_rate = 1e-5
optimizer = AdamW(model.parameters(), lr=learning_rate, eps=1e-5)
```

* Lr과 AdamW의 epsilon을 하이퍼패러미터로 사용

4. 데이터프레임 포맷 변경

	ID	sentence	question	candi_first	candi_second	answer
0	1	이퀄라이저로 저음 음역대 소리 크기를 키웠다.	결과	베이스 소리가 잘 들리게 되었다.	베이스 소리가 들리지 않게 되었다.	1
1	2	음료에 초콜렛 시럽을 넣었다.	결과	음료수가 더 달아졌다.	음료수가 차가워졌다.	1
2	3	남자는 휴대폰을 호수에 빠뜨렸다.	결과	휴대폰이 업그레이드 되었다.	휴대폰이 고장났다.	2
3	4	옆 집 사람이 이사를 나갔다.	원인	옆 집 사람은 계약이 완료되었다.	옆 집 사람은 계약을 연장했다.	1
4	5	문을 밀었다.	결과	문이 잠겼다.	문이 열렸다.	2

→ 전제 문장(sentence)을 question 및 후보 문장들과 연결

→ 이때 question의 결과 / 원인을 "그래서"/"왜냐면"으로 바꿔서 후보 문장들과 합친다

→ 후보 문장들을 [SEP]으로 연결

→ Binary classification이 되도록 answer을 0, 1 label로 변경

5. 데이터 처리 결과

	ID	sentence	question	candi_first	candi_second	answer
0	1	이퀄라이저로 저음 음역대 소리 크기를 키웠다.	그래서	이퀄라이저로 저음 음역대 소리 크기를 키웠다. 그래서 베이스 소리가 잘 들리게 되었다.	이퀄라이저로 저음 음역대 소리 크기를 키웠다. 그래서 베이스 소리가 들리지 않게 되었다.	0
1	2	음료에 초콜렛 시럽을 넣었다.	그래서	음료에 초콜렛 시럽을 넣었다. 그래서 음료수가 더 달아졌다.	음료에 초콜렛 시럽을 넣었다. 그래서 음료수가 차가워졌다.	0
2	3	남자는 휴대폰을 호수에 빠뜨렸다.	그래서	남자는 휴대폰을 호수에 빠뜨렸다. 그래서 휴대폰이 업그레이드 되었다.	남자는 휴대폰을 호수에 빠뜨렸다. 그래서 휴대폰이 고장났다.	1
3	4	옆 집 사람이 이사를 나갔다.	왜냐면	옆 집 사람이 이사를 나갔다. 왜냐면 옆 집 사람은 계약이 완료되었다.	옆 집 사람이 이사를 나갔다. 왜냐면 옆 집 사람은 계약을 연장했다.	0

6. Encoding 함수 설정 → 두 개의 후보 문장을 pair로 만들기

```
def encode_data(tokenizer, candi_seconds, candi_firsts, max_length):

    input_ids = []
    attention_masks = []

    for candi_second, candi_first in zip(candi_seconds, candi_firsts):
        encoded_data = tokenizer.encode_plus(candi_second, candi_first,
                                             max_length=max_length,
                                             pad_to_max_length=True,
                                             truncation_strategy="longest_first")
        encoded_pair = encoded_data["input_ids"]
        attention_mask = encoded_data["attention_mask"]

        input_ids.append(encoded_pair)
        attention_masks.append(attention_mask)

    return np.array(input_ids), np.array(attention_masks)
```

7. Train set과 Valid set(Dev set)의 값을 컬럼별로 저장

```
candi_firsts_train = train_data_df.candi_first.values
candi_seconds_train = train_data_df.candi_second.values
answers_train = train_data_df.answer.values.astype(int)

candi_firsts_dev = dev_data_df.candi_first.values
candi_seconds_dev = dev_data_df.candi_second.values
answers_dev = dev_data_df.answer.values.astype(int)
```

8. 그 값을 Encoding 함수 적용 → answer과 합쳐 feature에 저장

```
max_seq_length = 512
input_ids_train, attention_masks_train = encode_data(tokenizer, candi_seconds_train, candi_firsts_train, max_seq_length)
input_ids_dev, attention_masks_dev = encode_data(tokenizer, candi_seconds_dev, candi_firsts_dev, max_seq_length)

train_features = (input_ids_train, attention_masks_train, answers_train)
dev_features = (input_ids_dev, attention_masks_dev, answers_dev)
```

9. Feature의 정보들을 텐서로 변환하고 Data loader 만들기

```
batch_size = 8

train_features_tensors = [torch.tensor(feature, dtype=torch.long) for feature in train_features]
dev_features_tensors = [torch.tensor(feature, dtype=torch.long) for feature in dev_features]

train_dataset = TensorDataset(*train_features_tensors)
dev_dataset = TensorDataset(*dev_features_tensors)

train_sampler = RandomSampler(train_dataset)
dev_sampler = SequentialSampler(dev_dataset)

train_dataloader = DataLoader(train_dataset, sampler=train_sampler, batch_size=batch_size)
dev_dataloader = DataLoader(dev_dataset, sampler=dev_sampler, batch_size=batch_size)
```

```
for _ in tqdm(range(epochs), desc="Epoch"):

  # Training
    epoch_train_loss = 0 # Cumulative loss
    model.train()
    model.zero_grad()

    for step, batch in enumerate(train_dataloader):

        input_ids = batch[0].to(device)
        attention_masks = batch[1].to(device)
        labels = batch[2].to(device)

        outputs = model(input_ids, token_type_ids=None, attention_mask=attention_masks, labels=labels)

        loss = outputs[0]
        loss = loss / grad_acc_steps
        epoch_train_loss += loss.item()

        loss.backward()

        if (step+1) % grad_acc_steps == 0: # Gradient accumulation is over
            torch.nn.utils.clip_grad_norm_(model.parameters(), 1.0) # Clipping gradients
            optimizer.step()
            model.zero_grad()

    epoch_train_loss = epoch_train_loss / len(train_dataloader)
    train_loss_values.append(epoch_train_loss)
```

```
epochs = 7
grad_acc_steps = 1
train_loss_values = []
dev_acc_values = []
```

10. Training, Evaluation

→ 데이터로더에 담긴 파이토치 텐서 batch들을 모델에 넣는다

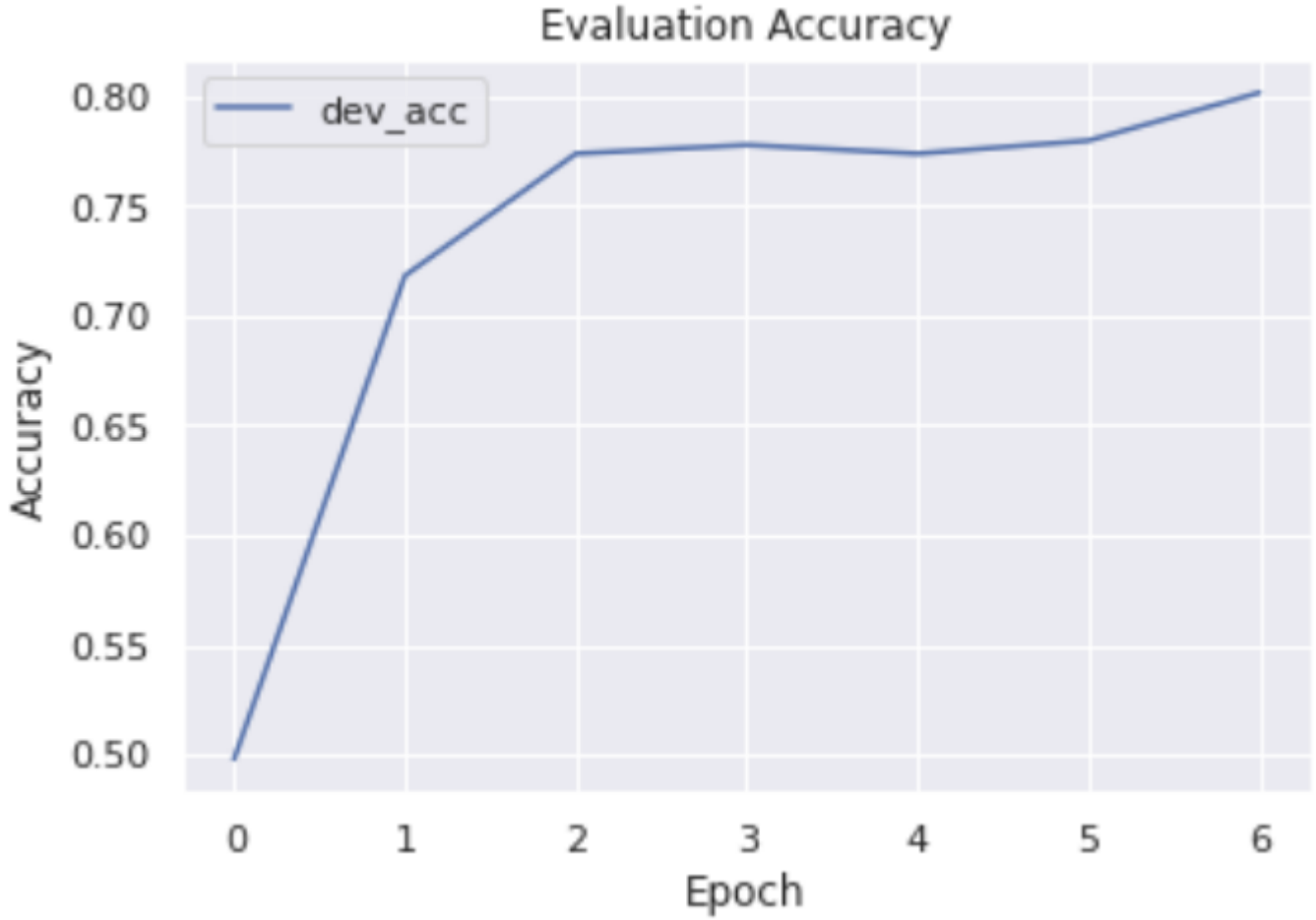
인과관계 추론(COPA)

```
# Evaluation
epoch_dev_accuracy = 0 # Cumulative accuracy
model.eval()

for batch in dev_dataloader:

    input_ids = batch[0].to(device)
    attention_masks = batch[1].to(device)
    labels = batch[2]

    with torch.no_grad():
        outputs = model(input_ids, token_type_ids=None, attention_mask=attention_masks)

    logits = outputs[0]
    logits = logits.detach().cpu().numpy()

    predictions = np.argmax(logits, axis=1).flatten()
    labels = labels.numpy().flatten()

    epoch_dev_accuracy += np.sum(predictions == labels) / len(labels)

epoch_dev_accuracy = epoch_dev_accuracy / len(dev_dataloader)
dev_acc_values.append(epoch_dev_accuracy)
```

→ valid dataset에 대해 평가

→ Accuracy 도출

11. 훈련 결과:



AdamW--eps: 1e-5

Max_len: 512

Batch size: 8

Epoch: 7

Learning rate: 0.00001

→ Accuracy: 0.8095

판정 의문문(BoolQ)

1. Binary classification(Yes or No Question)

	ID	passage	question	answer
0	1	로마 시대의 오리엔트의 범위는 제국 내에 동부 지방은 물론 제국 외부에 있는 다른 ...	오리엔트는 인도와 중국, 일본을 이루는 광범위한 지역을 지칭하는 단어로 쓰인다.	1
1	2	비글을 키우려면 비글이 뛰어놀수 있는 넓은 놀이공간 등을 확보하고 있는 단독주택이 ...	비글은 넓고 뚤린 공간에서 키워야 한다.	1
2	3	타이완 요리의 특징은 토속 요리(일본 통치 전)에서 기름을 많이 사용하는 다른 지역...	타이완 요리는 다른 지역의 중국 요리처럼 기름을 많이 사용하는 것이다.	0
3	4	연하곤란은 음식물이 구강에서 식도로 넘어가는 과정에 문제가 생겨 음식을 원활히 혹은...	연하곤란이 생기면 식도가 막히나요?	0
4	5	인문과학 또는 인문학(人文學, 영어: humanities)은 인간과 인간의 근원문제...	인문과학은 경험적인 접근을 주로 사용하는가?	0

2. 다른 과정은 인과관계 추론과 같은 방식으로 진행

3. 텍스트 클리닝

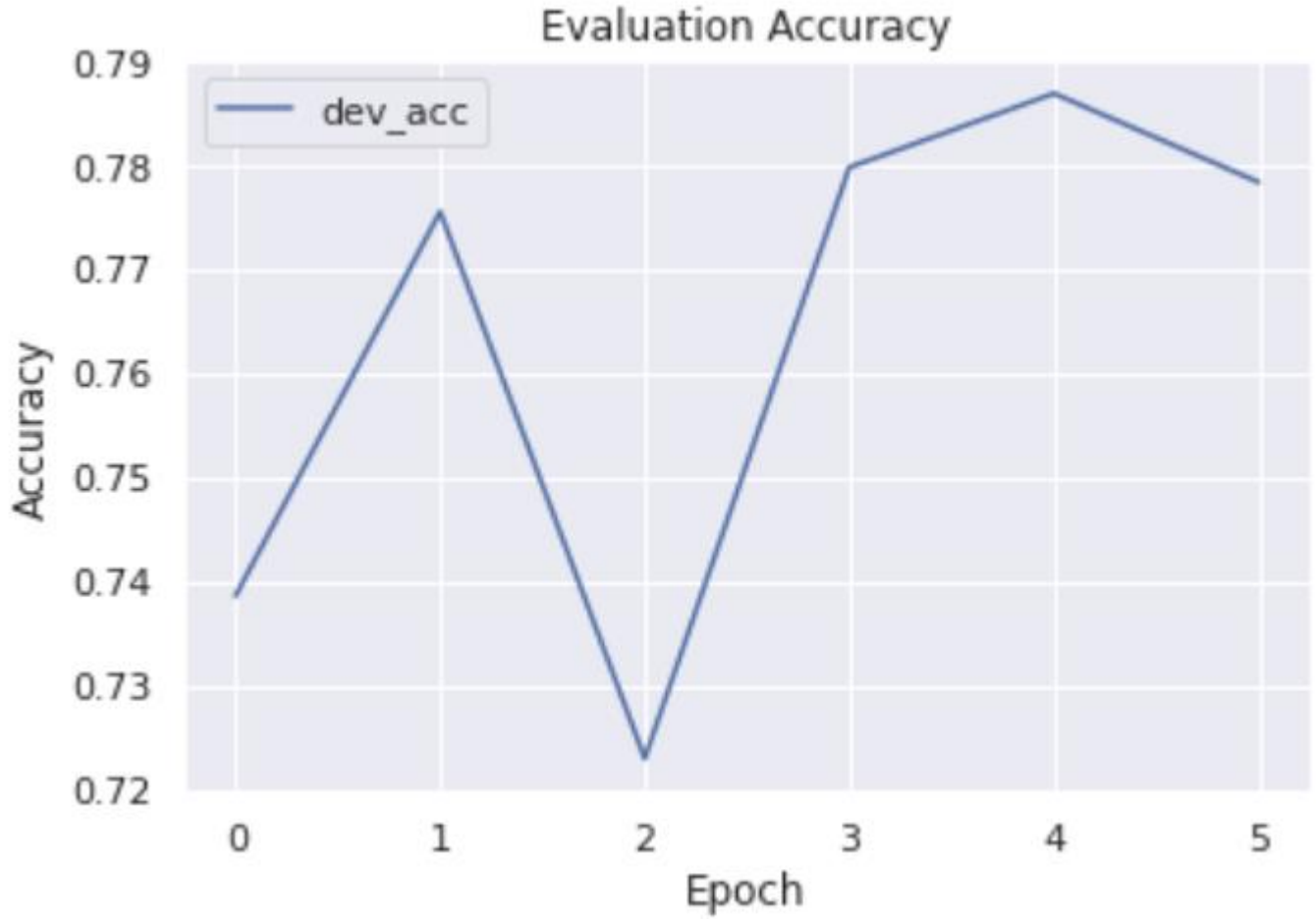
```python
import re

def cleaning(document):
    document = re.sub(r'\((.*?)\)', '', document)
    # 괄호 ()안에 있는 영어, 한자, 각종 외국어나 설명 제거

    return document
```

판정 의문문(BoolQ)

4. 클리닝 결과

	ID	passage	question	answer
0	1	로마 시대의 오리엔트의 범위는 제국 내에 동부 지방은 물론 제국 외부에 있는 다른 ...	오리엔트는 인도와 중국, 일본을 이루는 광범위한 지역을 지칭하는 단어로 쓰인다.	1
1	2	비글을 키우려면 비글이 뛰어놀수 있는 넓은 놀이 공간 등을 확보하고 있는 단독주택이 ...	비글은 넓고 뚤린 공간에서 키워야 한다.	1
2	3	타이완 요리의 특징은 토속 요리에서 기름을 많이 사용하는 다른 지역의 중국 요리와 ...	타이완 요리는 다른 지역의 중국 요리처럼 기름을 많이 사용하는 것이다.	0
3	4	연하곤란은 음식물이 구강에서 식도로 넘어가는 과정에 문제가 생겨 음식을 원활히 혹은...	연하곤란이 생기면 식도가 막히나요?	0
4	5	인문과학 또는 인문학은 인간과 인간의 근원문제, 인간과 인간의 문화에 관심을 갖거나...	인문과학은 경험적인 접근을 주로 사용하는가?	0

5. 훈련 결과



AdamW--eps: 1e-5

Max_len: 512

Batch size: 8

Epoch: 6

Learning rate: 0.00001

→ Accuracy: 0.7784

1. Train 데이터를 훈련시키는 과정까지는 앞과 동일

```
predictions , true_labels = [], []

for batch in dev_dataloader:

    batch = tuple(t.to(device) for t in batch)

    b_input_ids, b_input_mask, b_labels = batch

    with torch.no_grad():
        outputs = model(b_input_ids, token_type_ids=None,
                        attention_mask=b_input_mask)

    logits = outputs[0]

    logits = logits.detach().cpu().numpy()
    label_ids = b_labels.to('cpu').numpy()

    predictions.append(logits)
    true_labels.append(label_ids)
```

예측 값과 실제 값을 가지고 MCC를 구해야 함

→ Logits 값을 예측 값 리스트에 저장, 실제 값을 실제 라벨 리스트에 저장

2. 사이킷런에서 matthews_corrcoef를 임포트한 뒤에 batch마다 MCC값 계산

```
from sklearn.metrics import matthews_corrcoef

matthews_set = []

print('Calculating Matthews Corr. Coef. for each batch...')

for i in range(len(true_labels)):

    pred_labels_i = np.argmax(predictions[i], axis=1).flatten()

    matthews = matthews_corrcoef(true_labels[i], pred_labels_i)
    matthews_set.append(matthews)
```

3. 예측 값과 실제 값을 MCC로 계산한 결과 도출

```
flat_predictions = [item for sublist in predictions for item in sublist]
flat_predictions = np.argmax(flat_predictions, axis=1).flatten()

flat_true_labels = [item for sublist in true_labels for item in sublist]

mcc = matthews_corrcoef(flat_true_labels, flat_predictions)

print('MCC: %.4f' % mcc)
```

AdamW--eps: 1e-10

Max_len: 32

Batch size: 32

Epoch: 4

Learning rate: 0.00001

→ MCC: 0.4318

동형이의어(WiC)

1. 데이터 확인

	ID	Target	SENTENCE1	SENTENCE2	ANSWER	start_s1	end_s1	start_s2	end_s2
0	1	단정	그의 죽음은 타살로 단정이 되었다.	단정이 된 교실은 정돈되어 있다.	False	11	13	0	2
1	2	단수	현대 생활에서 단전과 단수의 고통은 겪어 보지 않으면 짐작도 못 한다.	사업자를 단수로 할지 복수로 할지를 놓고 관계자들 사이에 입씨름이 벌어졌다.	False	12	14	5	7
2	3	화성	화성은 밤과 낮, 하루의 길이와 계절의 변화가 지구와 매우 비슷하다.	화성은 서양 음악을 이루는 중요한 요소이다.	False	0	2	0	2
3	4	자전	달의 자전 주기는 달이 지구의 둘레를 공전하는 주기와 같다.	태양계의 모든 행성은 자전을 한다.	True	3	5	12	14

Classification 형식으로 변경 → ANSWER: False와 True를 0과 1로 변경

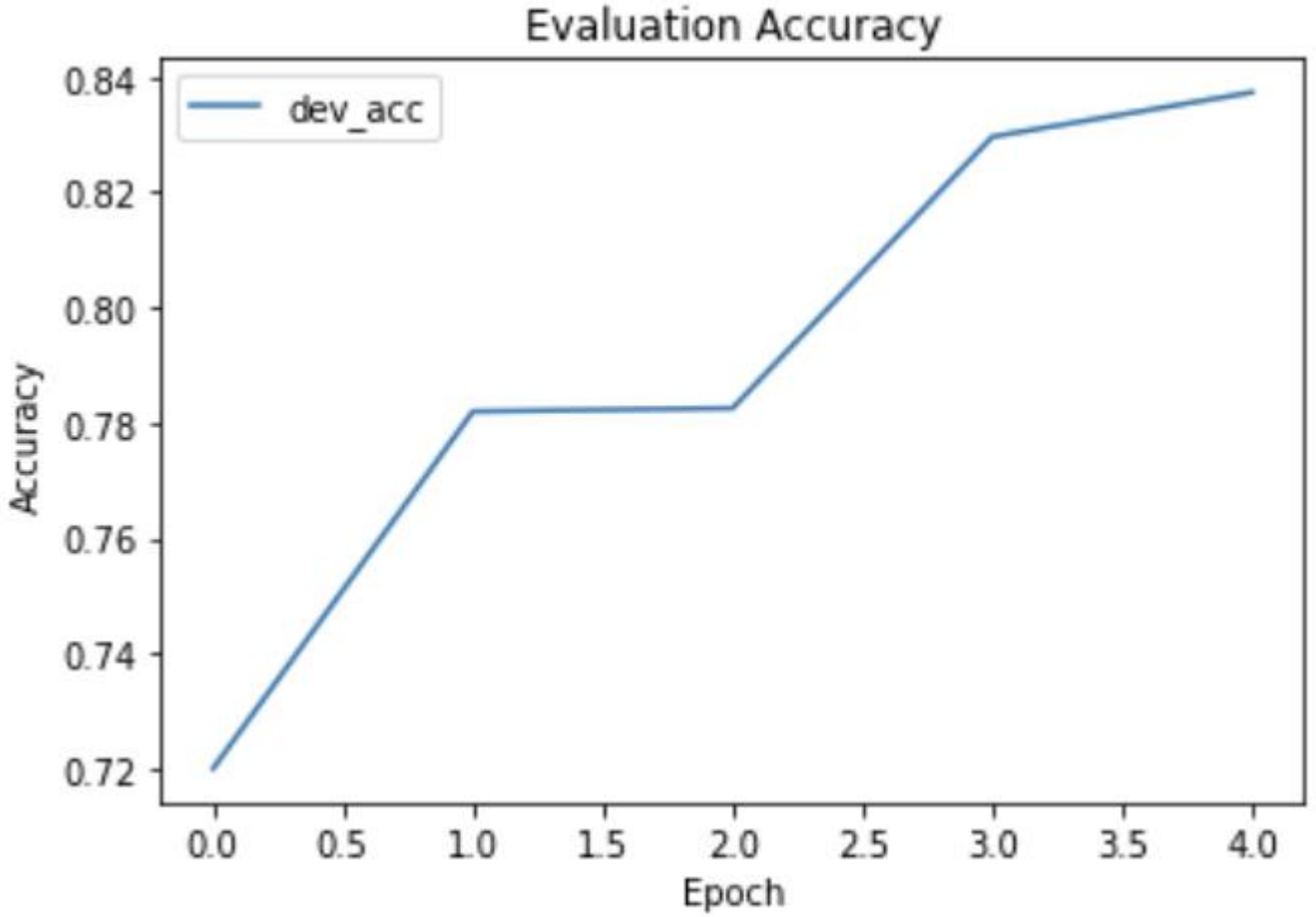
2. Encoding에 각 문장의 start position과 end position을 포함시켜서 input id와 attention mask에 정보 저장

```
def encode_data(tokenizer, sentence_firsts, sentence_seconds, max_length,
                start_positions1, start_positions2, end_positions1, end_positions2):
```

```
input_ids_train, attention_masks_train = encode_data(tokenizer, sentence_firsts_train, sentence_seconds_train,
                                                     max_seq_length,
                                                     start_positions1, start_positions2,
                                                     end_positions1, end_positions2)
input_ids_dev, attention_masks_dev = encode_data(tokenizer, sentence_firsts_dev, sentence_seconds_dev,
                                                 max_seq_length,
                                                 start_positions1, start_positions2,
                                                 end_positions1, end_positions2)
```

→ 다른 과정은 앞과 동일

3. 결과



AdamW--eps: 1e-8

Max_len: 32

Batch size: 8

Epoch: 5

Learning rate: 0.00001

→ Accuracy: 0.8378

감사합니다

* 참조 사이트

Deep Learning has (almost) all the answers: Yes/No Question Answering with Transformers | by Micheli Vincent | Illuin | Medium