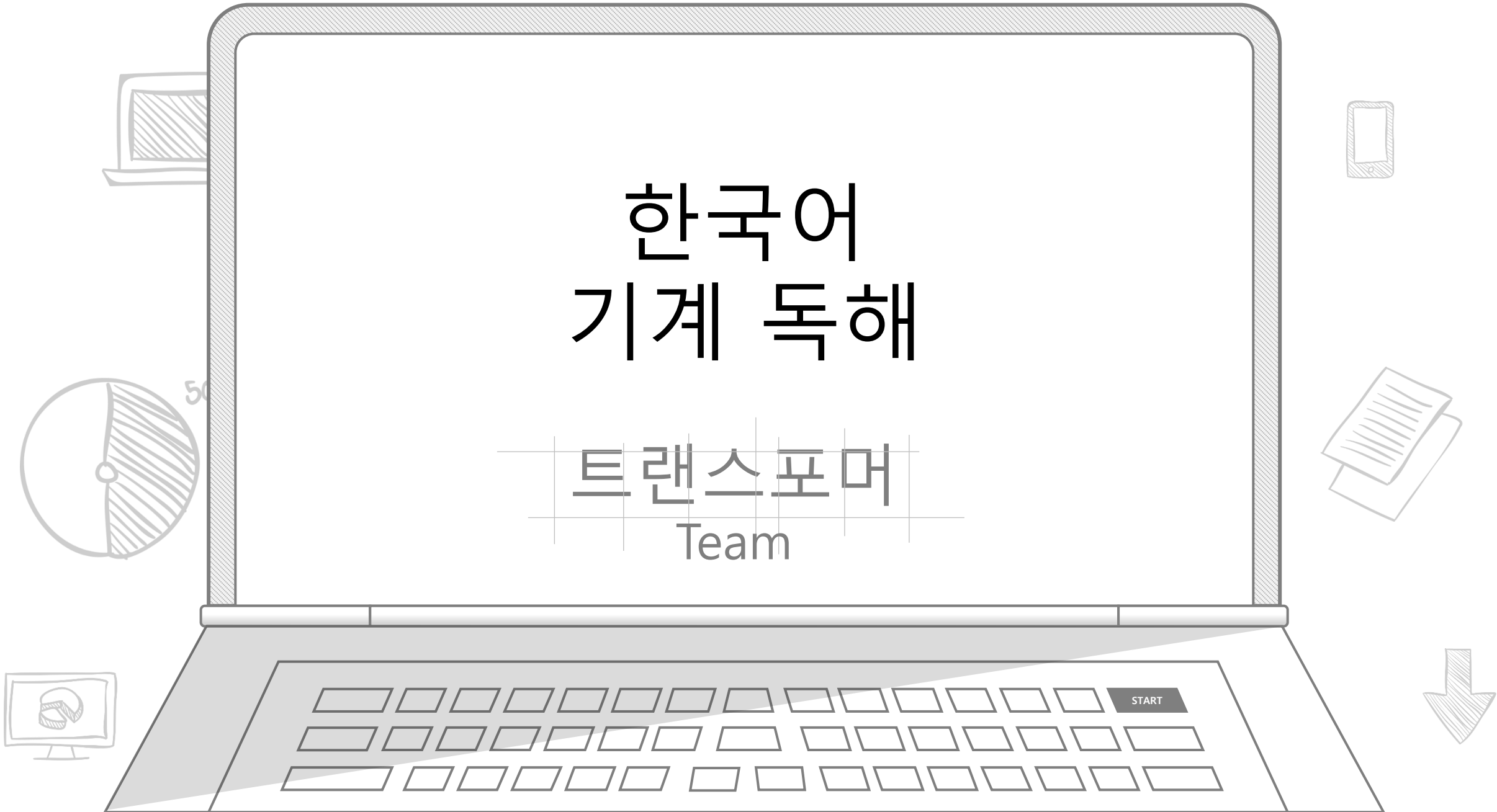
한국어
기계 독해



트랜스포머

Team

CONTENTS

Ⅰ. 팀 소개

Team Introduction

박정호	이동근	이상규	호연경
팀장	팀원	팀원	팀원

※ 공동 작업 : 데이터 모델 성능 향상, 코드 구현

Ⅱ. 프로젝트 개요

주제

- 한국어 기계독해 Question Answering

프로젝트 목표

- 모델을 도출하여 평가 척도인 편집 거리(Levinstein Distance) 최소화
- Learning Rate, Batch Size, Epoch 등의 하이파 퍼라미터와 후선처리를 통해 학습 훈련

활용 라이브러리 및 프레임워크

PyTorch colab

- **활용 데이터**:
  - AI Hub의 기계독해 데이터 셋 사용
  - 다양한 주제의 기계독해 데이터 셋으로 약 243,425 건의 데이터로 구성
  - Train Data: 기계독해 데이터 셋 (243,425 개의 데이터 중 랜덤으로 추출한 219,083 개)

    Evaluation Data: train.json + 기계독해 데이터 셋 (12,037 + Train_data에서 추출하지 않은 24,342 개)

Ⅱ. 프로젝트 개요

01. 데이터 소개

▶ 사용 데이터 개요
▶ AI Hub의 도서자료 기계독해 데이터셋 사용

데이터 출처	데이터 분류 및 개수	데이터 특징
도서자료 기계독해	Train: 900,000 (실제 사용 데이터: 225,000) Valid: 50,000 (실제 사용 데이터: 47,314)	SQuAD와 같은 지문-질문-답변으로 이루어진 데이터셋 (데이터셋 70만 건과 정답이 없는 데이터셋 30만 건으로 구성됨)
기계독해	243,425	질문과 답으로 이루어져 있는 뉴스 본문 기반 학습 데이터셋 25만 건
train.json	17,663 (실제 사용 데이터: 12,037)	Unique Value를 포함한 학습을 위한 데이터셋

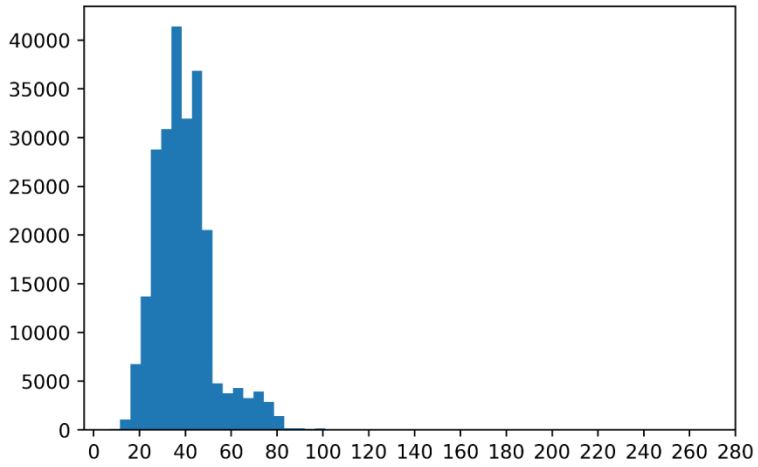
Ⅱ. 프로젝트 개요

01. 데이터 소개

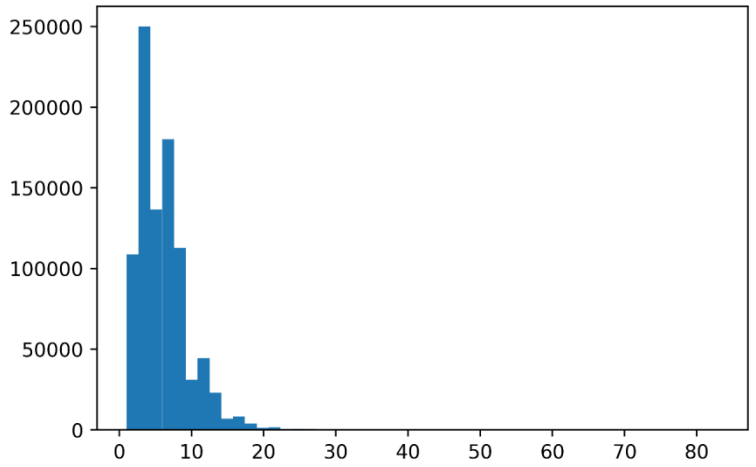
▶ 사용 데이터 시각화

기계독해 데이터의 글자 길이 분포

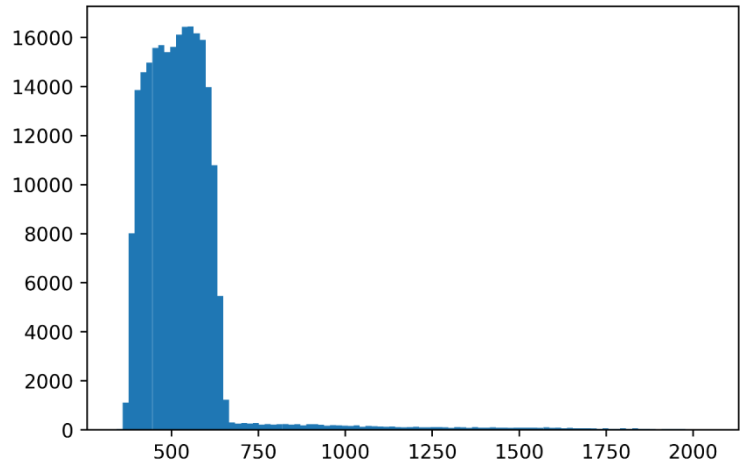
▪ Context



▪ Answer



▪ Question



- 히스토그램을 통해, Answer 데이터의 글자수가 20자를 넘어가면 그 수가 급격히 줄어드는 것을 확인 가능
- Answer 데이터에서 0 ~ 10 길이의 글자수에 대부분의 데이터가 분포하는 것을 확인 가능

Ⅲ. 데이터 분석 및 결과

01. Tokenization 및 전처리

```
def qa_preprocess(df, batch_size=32, method='train'):

    if method == 'train':
        batch_input = tokenizer(df['contexts'].tolist(), df['questions'].tolist(), truncation=True, padding=True)

        start_ids = df['start_ids'].tolist()
        end_ids = df['end_ids'].tolist()

        start_positions = [batch_input.char_to_token(i, start_ids[i]) for i in range(len(start_ids))]
        end_positions = [batch_input.char_to_token(i, end_ids[i]-1) for i in range(len(end_ids))]
        deleting_list = [i for i, v in enumerate(end_positions) if v == None]

        batch_input.update({'start_positions': start_positions, 'end_positions': end_positions})

        batch_input = {key : [v for ids, v in enumerate(value) if ids not in deleting_list] for key, value in batch_input.items()}
        batch_input = {key : torch.tensor(np.array(value,dtype=float),astype(int)) for key, value in batch_input.items()}

        input_ids = batch_input['input_ids']
        segments = batch_input['token_type_ids']
        masks = batch_input['attention_mask']
        start_ids = batch_input['start_positions']
        end_ids = batch_input['end_positions']

        dataset = TensorDataset(input_ids, masks, segments, start_ids, end_ids)
        dataset_sampler = RandomSampler(dataset)
        dataset = DataLoader(dataset, sampler=dataset_sampler, batch_size=16)

    elif method == 'valid':
        batch_input = tokenizer(df['contexts'].tolist(), df['questions'].tolist(), truncation=True, padding=True)

        start_ids = df['start_ids'].tolist()
        end_ids = df['end_ids'].tolist()
start_positions = [batch_input.char_to_token(i, start_ids[i]) for i in range(len(start_ids))]
end_positions = [batch_input.char_to_token(i, end_ids[i]-1) for i in range(len(end_ids))]
deleting_list = [i for i, v in enumerate(end_positions) if v == None]
        batch_input.update({'start_positions': start_positions, 'end_positions': end_positions})

        batch_input = {key : [v for ids, v in enumerate(value) if ids not in deleting_list] for key, value in batch_input.items()}
        dataset = {key : torch.tensor(np.array(value,dtype=float),astype(int)) for key, value in batch_input.items()}

    elif method == 'test':
        batch_input = tokenizer(df['contexts'].tolist(), df['questions'].tolist(), truncation=True, padding=True)
        dataset = {key : torch.tensor(value) for key, value in batch_input.items()}

    return dataset
```

※Tokenization

BertTokenizerFast() → 빠른 속도, 성능에 영향을 줌

tokenizer = BertTokenizerFast.from_pretrained('klue/bert-base')

※전처리

만약 start_position 또는 end_position 둘 중 하나라도 input_ids에 포함되지 않는다면 position index 값이 None으로 표시 됨

↓

제거해야 할 리스트를 생성한 후 리스트 기반으로 데이터를 지움
- end_position에서 None인 경우 deleting_list에 넣기
- deleting_list 해당 값 지우기

<결과 예시>

```
Input_ids :  [2, 1446, 22555, 11477, 2116, 3932, 2210, 6530, 2713
token_type_ids :  [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
attention_mask :  [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1
start_positions :  241
end_positions :  244
```

```
input_ids :  [2, 5689, 7780, 5430, 2084, 2170, 5120, 5430, 2194
token_type_ids :  [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
attention_mask :  [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
start_positions :  None
end_positions :  None
```

Ⅲ. 데이터 분석 및 결과

02. 학습 모델 비교

▶ Bert 모델 사용
▶ Hugging Face에 등록된 사전학습 모델에 대한 비교

beomi/kcbert-bert

Model Configuration	▪ Max Length = 300 ▪ Learning Rate = 2e-5 ▪ Training Step = 1M ▪ Vocab_size = 30000
특징	▪ 기본코퍼스 - 70GB - 350M sents - 11.3B 형태소 ▪ 특허문서 - 75GB - 270M sents - 15B 형태소

kykim/bert-kor-base

Model Configuration	▪ Max Length = 512 ▪ Learning Rate = 1e-12 ▪ Training Step = 1.9M ▪ Vocab_size = 42000
특징	• 총 70GB 텍스트 • 위키/나무위키 • 주요 커머스 리뷰 1억개 + 블로그형 웹사이트 2000만개 • 모두의 말뭉치

klue/bert-base

Model Configuration	▪ Max Length = 512 ▪ Learning Rate = 1e-4 ▪ Training Step = 1M ▪ Vocab_size = 32000
특징	▪ 연합 뉴스 헤드라인, 위키피디아, 위키뉴스, 위키트리 ▪ Policy News (POLICY) - 정부 부처, 국가 기관 및 정부 기관에서 배포하는 다양한 기사의 데이터 세트 ▪ PARAKQC(ParAKQC) - 스마트홈과 상호작용할 때 발생할 수 있는 다양한 주제 ▪ 에어비앤비 리뷰, NSMC), ▪ Acrofan News (ACROFAN), - 새로운 제품이나 기업의 이벤트를 소개하는 경우가 많음 ▪ 한국 경제 신문

Ⅲ. 데이터 분석 및 결과

02. 학습 모델 비교

▶ 추가적인 모델 사용
▶ 모델 성능 비교

Monologg/koeletra-base-v3-discriminato

Model Configuration	▪ Max Length = 512 ▪ Learning Rate = 2e-4 ▪ Training Step = 1.5M ▪ Vocab_size = 32000
특징	▪ 34GB의 한국어 text ▪ PTD 방식으로 MLM방식과 비교하여 더 효과적임

klue/roberta-base

Model Configuration	▪ Max Length = 512 ▪ Learning Rate = 1e-4 ▪ Training Step = 1M ▪ Vocab_size = 32000
특징	▪ klue/bert-base와 동일함

적용한 학습 모델 Best 결과

model	Klue/bert -base	kykim/bert -kor-base	Beomi/kcbert -base	monologg/koelectra -base-v3-discriminator	klue/roberta-base
Batch Size	32	32	32	16	16
Learning Rate	1e-5	1e-5	1e-5	2e-5	5e-5
Epoch	4	4	4	4	3
Edit distance	2.653	2.77	4.567	2.884	5.847

Klue vs Kor 버트 성능 비교

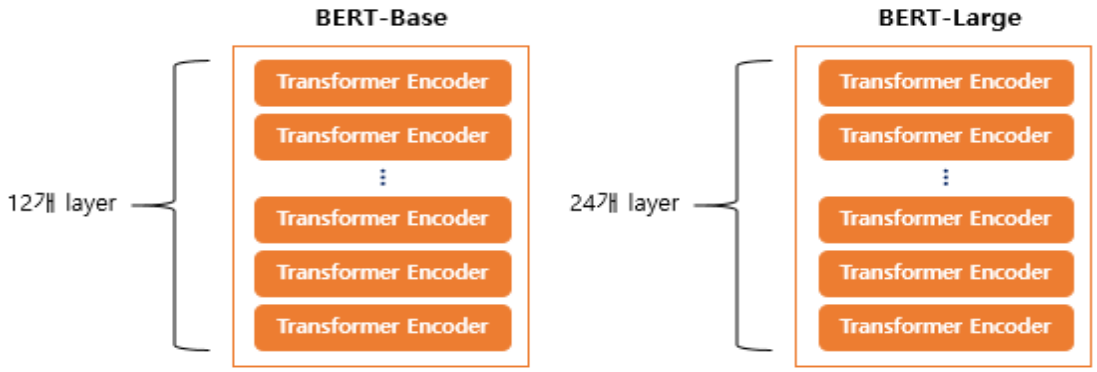
model	Klue/bert -base	Klue/bert -base	kykim/bert -kor-base	kykim/bert -kor-base
Batch Size	16	32	16	32
Learning Rate	1e-5	1e-5	1e-5	1e-5
Epoch	4	4	4	4
Edit distance	2.695	2.653	2.84	2.77

Ⅲ. 데이터 분석 및 결과

03. 최종 모델

▶ BERT 모델
- 토큰의 왼쪽과 오른쪽 모두에서 텍스트를 볼 수 있기 때문에 단방향이 아닌 양방향이라는 장점이 있음

▶ Klue/bert-base 사전 학습 모델
- 한국어 NLP 발전을 위해 도입된 한국어 기본 모델
- 한국어 모델의 자연어 이해 능력을 평가하기 위한 일련의 데이터 세트
- Huggingface에 등록되어 있어 손쉽게 접근이 가능함



➢ Bert의 기본 구조는 트랜스포머의 인코더를 쌓아 올린 구조
➢ Base 버전은 12개, Large 버전은 24

→ " Bert-Base 선택"

```
model = BertForQuestionAnswering.from_pretrained('klue/bert-base')
model = model.to(device)

epochs = 4

optimizer = AdamW(model.parameters() , lr=1e-5, eps=1e-8)

total_steps = len(train_set) * epochs

scheduler = get_linear_schedule_with_warmup(optimizer,
                                            num_warmup_steps = 0,
                                            num_training_steps = total_steps)
```

Scheduler 적용
- 학습을 통해 효율적인 러닝레이트 조정

최종 선택 이유
- 다양한 업무와 말뭉치 커버
- 제한 없이 누구나 접근 가능
- 위와 같은 이유로 향후 다른 태스크에서 높은 성능을 보일 것을 기대 됨

최종 모델 및 하이퍼파라미터
- Klue Bert-base 모델
- Batch_size =32, learning_rate = 1e-5, epoch = 4
- 후처리: ##토큰제거, 답의 길이가 20 이상 제거

iv. 데이터 분석 및 결과

아쉬운 점

- 25만개의 데이터를 사용해 학습하는 데 오랜 시간이 걸려 모델 훈련 관리에 불편한 점이 있었다
  따라서 효율적인 모델 훈련 관리와 데이터 관리가 필요해 보인다

- 다양한 사전 학습 모델을 이용하여 학습을 하려고 했으나 특정사전 학습 모델은 unbuntu 서버를 사용해야 하는 등 로컬 자원의 문제가 있어서 다양한 사전학습 모델로 학습을 시도하지 못함

- 지난 분류 프로젝트와는 다르게 성능 향상을 위해 결과물에서 추가적인 처리를 거칠 필요가 있었다
  따라서 점수 향상을 위해 결과물에 효과적인 후처리 방법 사용에 필요성을 느낌


Q & A