[KUBIG - DL]

뉴스토픽 분류

"15기 김지호 신윤 염윤석 우명진 최민경"

목차

① 데이터 소개 및 프로젝트 소개

	index	title	topic_idx
1	29708	웹툰 나이트에 참가한 작가들	3
2	1189	북한인권법 4일 시행...북한인권재단 다음주 출범	6
3	40319	與 검찰개혁 강공 드라이브...한국당 검찰 겁박 홍위병 정치	6
4	11860	이라크총리 민생고 시위 100여명 사망 진상조사 지시	4
5	15621	그래픽 기업경기실사지수 낙폭 메르스 이후 최대	1
6	30193	민주 당권주자 제주서 유세대결...1강2중 관측 속 경쟁 본격화	6
7	6567	카메라 4개 달린 스마트폰 지오니 S10	0
8	34239	내년 첫 5G폰 평균가 80만원 육박...2023년 60만원대 ↓	0
9	40441	IBS 천진우 나노의학연구단장팀	0
10	21726	고발장 접수하는 김진태 의원실 관계자	6
11	8858	아시안피스컵 남북 男배구 열전 펼쳐...북한팀 32 승종합	5

데이터 설명

YNAT(주제 분류를 위한 연합 뉴스 헤드라인) 데이터
데이터 출처 : KLUE 데이터셋
Train : 45654개, test : 9131개

TITLE | 뉴스 헤드라인
TOPIC_IDX | 뉴스 주제 인덱스

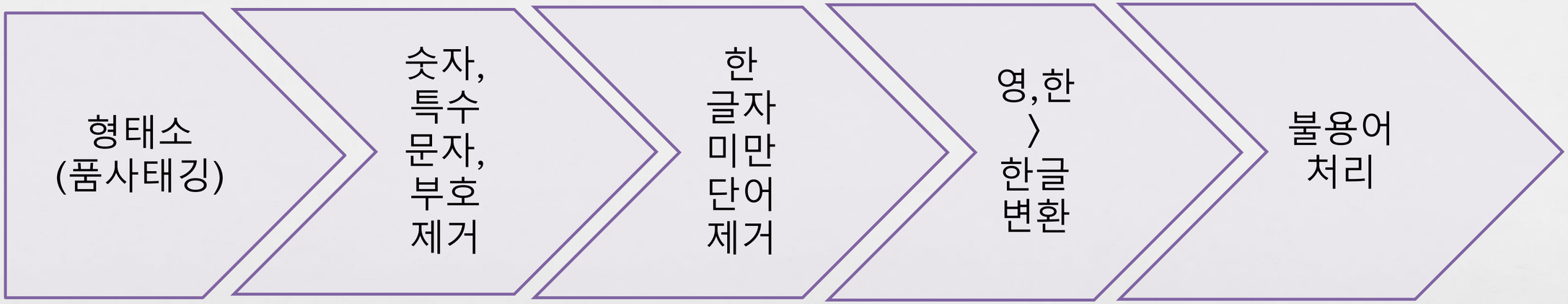
IT과학 0, 경제 1, 사회 2, 생활문화 3,
세계 4, 스포츠 5, 정치 6

프로젝트 설명

한국어 뉴스 헤드라인을 이용하여 뉴스의 주제를 분류하는 알고리즘 개발

② 전처리

전처리 과정



② 전처리

형태소(품사태깅)

```
okt = Okt() #형태소 분석기
```

```
def clean1(text):
    clean = []

    for word in okt.pos(text, stem = True): #어간 추출
      if word[1] not in ["Josa", "Eomi", "Punctuation"]:
        clean.append(word[0])

    return " ".join(clean)
```

- 형태소 분석기 사용
- 언어적 속성의 구조를 파악
- 어간 추출

기타전처리

```
def clean2(text):
  sent_clean = re.sub(r"[^a-zA-Z가-힣]"," ",text) #substraction 숫자, 특수문자 제거
  clean =[]
  for word in sent_clean.split(" "):
    if len(word)>1: #한글자 미만 제거
      clean.append(word)

  return " ".join(clean)
```

- 숫자, 특수문자 제거
- 한 글자 미만 단어 제거

② 전처리

형태소(품사태깅)

```
okt = Okt() #형태소 분석기
```

```
def clean1(text):
    clean = []

    for word in okt.pos(text, stem = True): #어간 추출
     if word[1] not in ["Josa", "Eomi", "Punctuation"]:
        clean.append(word[0])

    return " ".join(clean)
```

- 형태소 분석기 사용
- 언어적 속성의 구조를 파악
- 어간 추출

② 전처리

```
def clean2(text):
  sent_clean = re.sub(r"[^a-zA-Z가-힣]"," ",text) #substraction 숫자, 특수문자 제거
  clean =[]
  for word in sent_clean.split(" "):
    if len(word)>1: #한글자 미만 제거
      clean.append(word)

  return " ".join(clean)
```

```
punct = "/-'?!.,#$%₩'()*+-/:;<=>@[₩₩]^_`{|}~" + '"" “” ’ ' + '∞ θ ÷ α •à− β Ø³ π ‘₹’
punct_mapping = {" ‘": "'", "₹": "e", "´": "'", "° ": "", "€": "e", "™": "tm", "√

def clean_punc(text, punct, mapping):
    for p in mapping:
        text = text.replace(p, mapping[p])

    for p in punct:
        text = text.replace(p, f' {p} ')

    specials = {'₩u200b': ' ', '…': ' ... ', '₩ufeff': '', 'करना': '', 'है': ''}
    for s in specials:
        text = text.replace(s, specials[s])

    return text.strip()
```

- 숫자, 특수문자 제거
- 한 글자 미만 단어 제거

② 전처리

한자〉한글

```
def clean_text(texts):
    corpus = []
    for i in range(0, len(texts)):
        texts[i] = texts[i].replace("外人","외국인")
        texts[i] = texts[i].replace("日","일본")
        texts[i] = texts[i].replace("美","미국")
        texts[i] = texts[i].replace("北","북한")
        texts[i] = texts[i].replace("英","영국")
        texts[i] = texts[i].replace("中","중국")
        texts[i] = texts[i].replace("與","여당")
        texts[i] = texts[i].replace("靑","청와대")
        texts[i] = texts[i].replace("野","야당")
        texts[i] = texts[i].replace("伊","이탈리아")
```

영어〉한글

```
review = re.sub(r'UFG20','한미 합동 군사', str(review))
review = re.sub(r'F35','전투기', str(review))
review = re.sub(r'WP','워싱턴포스트', str(review))
review = re.sub(r'TK','대구와 경북', str(review))
review = re.sub(r'ACL','아시아축구연맹 챔피언스리그', str(review))
review = re.sub(r'IT','정보기술', str(review))
review = re.sub(r'AI','인공지능', str(review))
review = re.sub(r'TF','태스크포스', str(review))
review = re.sub(r'ML','메이저리그', str(review))
review = re.sub(r'FC','축구 클럽', str(review))
review = re.sub(r'SI','스포츠 일러스트레이티드', str(review))
review = re.sub(r'㈜','', str(review))
```

③ Modeling

사용모델1 : KLUE-RoBERTa

RoBERTa는 Robustly optimizrd BERT approac라는 뜻으로 기본적인 구조는 전부 BERT를 따라가면서 기존 BERT 모델에 비해 RoBERTa에 추가되는 부분은 다음과 같다

(1) dynamic masking : 매 epoch마다 새로운 마스킹을 시키는 태스크
(2) NSP(Next Sentence Prediction) 제거: 두 문장이 이어졌는지 판단하는 pretraining 과정
(3) 더 긴 시퀀스로 학습
(4) 더 많은 데이터 사용하여 더 큰 배치로 학습

KLUE-RoBERTa는 RoBERTa기반 한국어 자연어 처리 모델

③ Modeling

사용모델1 : KLUE-RoBERTa

Model	Embedding Size	Hidden Size	# Layers	# Heads
KLUE-RoBERTa-small	768	768	6	12
KLUE-RoBERTa-base	768	768	12	12
KLUE-RoBERTa-large	1024	1024	24	16

③ Modeling

Modeling 공통부분

토크나이징 함수

```
def tokenized(tokenizer, total_dataset):
  tokenized = total_dataset.map(lambda x :tokenizer(text = x["clear_title"], add_special_tokens = True,
                  max_length = MAX_LEN, padding = "max_length",
                  truncation = True) , batched = True)

  tokenized = tokenized.remove_columns(["index", "clear_title"])

  tokenized["train"] = tokenized["train"].rename_column("topic_idx", "labels")
  if "valid" in tokenized.keys():
    tokenized["valid"] = tokenized["valid"].rename_column("topic_idx", "labels")

  #torch tensor로 바꾸기
  tokenized.set_format("torch")

  if "valid" in tokenized.keys():
    return tokenized["train"], tokenized["valid"], tokenized["test"]
  else:
    return tokenized["train"], tokenized["test"]
```

하이퍼파라미터 설정

```
epochs = 10
MAX_LEN = 46
batch_size = 32
num_cores = 2

np.random.seed(42)

device = torch.device("cuda:0")
print(torch.cuda.is_available())


lr = 1e-5
log_interval = 200
```

입력 데이터로 만드는 함수

```
class TrainDataset(Dataset):
    def __init__(self, df):
        self.df_data = df
    def __getitem__(self, index):
        # get the sentence from the dataframe
        sentence = self.df_data.loc[index, 'clear_title']
        encoded_dict = tokenizer(
          text = sentence,
          add_special_tokens = True,
          max_length = MAX_LEN,
          pad_to_max_length = True,
          truncation=True,           # Pad & truncate all sentences.
          return_tensors="pt")

        padded_token_list = encoded_dict['input_ids'][0]
        token_type_id = encoded_dict['token_type_ids'][0]
        att_mask = encoded_dict['attention_mask'][0]
        target = torch.tensor(self.df_data.loc[index, "topic_idx"])
        sample = (padded_token_list, token_type_id , att_mask, target)
        return sample
    def __len__(self):
        return len(self.df_data)
```

③ Modeling

사용모델1 : KLUE-RoBERTa-small, base, large

입력 데이터로 만들기

```
model_name = 'klue/roberta-small'
```

```
tokenizer = AutoTokenizer.from_pretrained('klue/roberta-small')

train_data1 = TrainDataset(train)

test_data1 = TestDataset(test)

train_dataloader1 = torch.utils.data.DataLoader(train_data1,
                                    batch_size=BATCH_SIZE,
                                    shuffle=True,
                                   num_workers=NUM_CORES)
test_dataloader1 = torch.utils.data.DataLoader(test_data1,
                                    batch_size=BATCH_SIZE,
                                    shuffle=False,
                                   num_workers=NUM_CORES)
```

모델 학습 및 예측

```
model = AutoModelForSequenceClassification.from_pretrained('klue/roberta-small',num_labels =7)
model.to(device)

# for scheduling
warmup_ratio = 0.1
t_total = len(train_dataloader1) * num_epochs
warmup_step = int(t_total * warmup_ratio)

optimizer = AdaBelief(model.parameters(), lr = lr,
                      eps=1e-16, betas=(0.9,0.999), weight_decouple = True, rectify = True)



scheduler = get_cosine_schedule_with_warmup(optimizer, num_warmup_steps =warmup_step,
                                            num_training_steps = t_total)

train_acc_mean ,trained_model= Training(model, train_dataloader1, optimizer, scheduler, num_epochs, device)
print("Average : ", train_acc_mean)

#save model
model_path = "/content/drive/Shareddrives/2022-1 KUBIG 딥러닝 콘테스트/Code/4주차_최종/민경"
trained_model.save_pretrained(model_path + "/{}".format(model_name))
```

```
preds = []
model.eval()
torch.set_grad_enabled(False)
for batch_id, (input_id,token_type_id,attention_mask) in enumerate(tqdm_notebook(test_dataloader1)):
    input_id = input_id.long().to(device)
    token_type_id = token_type_id.long().to(device)
    attention_mask = attention_mask.long().to(device)
    outputs = model(input_ids=input_id, token_type_ids=token_type_id, attention_mask=attention_mask)
    out = outputs[0]
    for inp in out:
      preds.append(inp.detach().cpu().numpy())
Preds = np.array(preds)
```

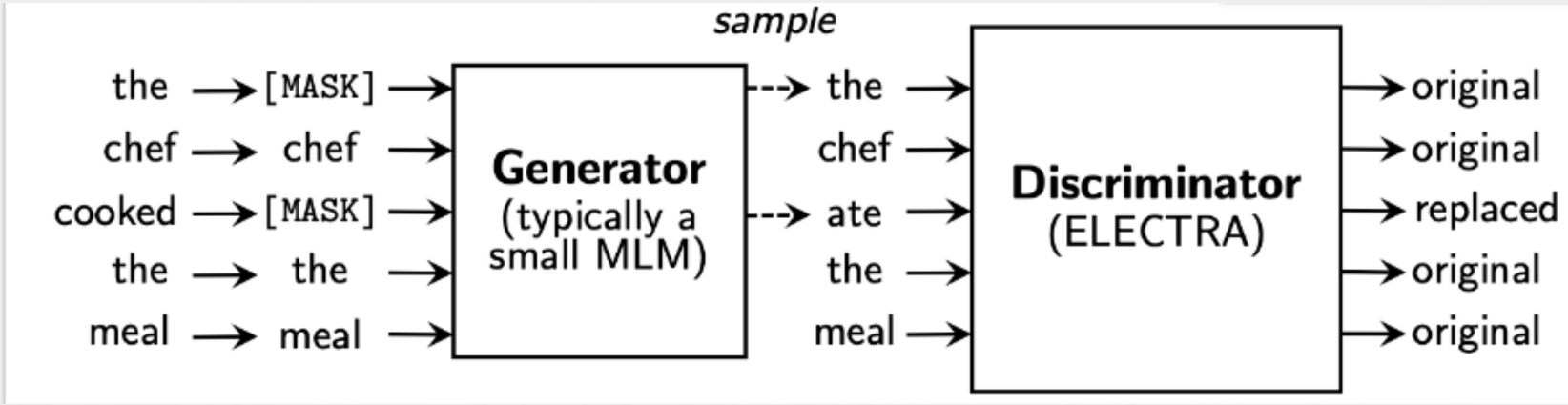
〉 이와 같은 방법으로 KLUE-RoBERTa-base, large도 학습

〉〉 Optimizer는 AdaBelief 사용
(사용한 모든 모델에 대해 AdamW, AdaBelief 둘 다 적용시켰을 때 AdaBelief가 더 좋은 성능을 보임)

③ Modeling

사용모델2 : Koelectra - base

- ELECTRA는 Replaced Token Detection, 즉 generator에서 나온 token을 보고 discriminator에서 real인지 fake인지를 판별하는 방법으로 학습을 한다.
- 즉 MLM에서 masking을 했던 것을 예측을 해서 나온 token이 맞는지를 여부를 판단해서 학습한다.
- input token에 대해 학습할 수 있다는 장점이 있어 BERT와 비교하였을 때, 더 좋은 성능을 보인다.



③ Modeling

사용모델2 : Koelectra - base

입력 데이터로 만들기

```
model_name = "monologg/koelectra-base-v3-discriminator"

tokenizer =ElectraTokenizer.from_pretrained(model_name)

Train_dataset4, Test_dataset4 = tokenized(tokenizer, total_dataset)

#data loader

train_dataloader4 = torch.utils.data.DataLoader(Train_dataset4,
                                      batch_size=BATCH_SIZE,
                                      shuffle=True,
                                    num_workers=NUM_CORES)

test_dataloader4 = torch.utils.data.DataLoader(Test_dataset4,
                                      batch_size=BATCH_SIZE,
                                      shuffle=False,
                                    num_workers=NUM_CORES)
```

〉Optimizer는 AdaBelief 사용

모델 학습 및 예측

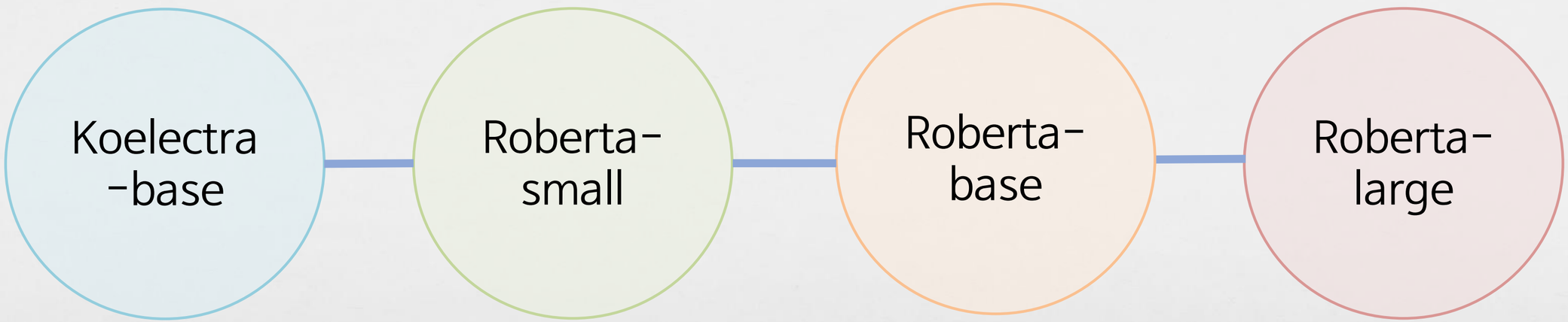
```
model = AutoModelForSequenceClassification.from_pretrained(model_name,num_labels =7)
model.to(device)

# for scheduling
warmup_ratio = 0.1
t_total = len(train_dataloader4) * num_epochs
warmup_step = int(t_total * warmup_ratio)

optimizer = AdaBelief(model.parameters(), lr = lr,
                      eps=1e-16, betas=(0.9,0.999), weight_decouple = True, rectify = True)



scheduler = get_cosine_schedule_with_warmup(optimizer, num_warmup_steps =warmup_step,
                                            num_training_steps = t_total)

train_acc_mean ,trained_model= Training(model, train_dataloader4, optimizer, scheduler, num_epochs, device)
print("Average : ", train_acc_mean)

#save model
model_path = "/content/drive/Shareddrives/2022-1 KUBIG 딥러닝 콘테스트/Code/4주차_최종/염"
trained_model.save_pretrained(model_path + "/{}".format(model_name))
```

```
preds = []
model.eval()
torch.set_grad_enabled(False)
for batch_id, (input_id,token_type_id,attention_mask) in enumerate(tqdm_notebook(test_dataloader4)):
    input_id = input_id.long().to(device)
    token_type_id = token_type_id.long().to(device)
    attention_mask = attention_mask.long().to(device)
    outputs = model(input_ids=input_id, token_type_ids=token_type_id, attention_mask=attention_mask)
    out = outputs[0]
    for inp in out:
      preds.append(inp.detach().cpu().numpy())
Preds = np.array(preds)
```

③ Modeling

Voting

: 서로 다른 종류의 알고리즘들을 결합



③ Modeling

- 네 개 모델 load & pred

1. roberta-small

```
[ ]  model_path = "/content/drive/Shareddrives/2022-1 KUBIG 딥러닝 콘테스트/Code/4주차_최종/민경"
     model_name = 'klue/roberta-small'
     save_path = model_path + "/"+ model_name
     model = AutoModelForSequenceClassification.from_pretrained(save_path, num_labels=7).to(
```

2. Roberta-base

```
[ ]  model_path = "/content/drive/Shareddrives/2022-1 KUBIG 딥러닝 콘테스트/Code/4주차_최종/윤"
     model_name = 'klue/roberta-base'
     save_path = model_path + "/"+ model_name
     model = AutoModelForSequenceClassification.from_pretrained(save_path, num_labels=7).to(device)
```

3. Roberta-large

```
[ ]  model_path = "/content/drive/Shareddrives/2022-1 KUBIG 딥러닝 콘테스트/Code/4주차_최종/지호"
     model_name = 'klue/roberta-large'
     save_path = model_path + "/"+ model_name
     model = AutoModelForSequenceClassification.from_pretrained(save_path, num_labels=7).to(device)
```

4. Koelectra-base

```
[ ]  model_path = "/content/drive/Shareddrives/2022-1 KUBIG 딥러닝 콘테스트/Code/4주차_최종/염"
     model_name = 'monologg/koelectra-base-v3-discriminator'
     save_path = model_path + "/"+ model_name
     model = AutoModelForSequenceClassification.from_pretrained(save_path, num_labels=7).to(device)
```

③ Modeling

Voting

```
[ ]  Pred_values = Roberta_small*0.5 + Roberta_base * 0.1 + Roberta_large * 0.3 + Koelectra_base * 0.1
```

```
[ ]  results = np.argmax(Pred_values, axis=1)
     submission['topic_idx']= results
```

```
[ ]  submission
```

	index	topic_idx
0	45654	3
1	45655	3
2	45656	2
3	45657	0
4	45658	3
...	...	...
9126	54780	3

- 각 모델 반영 비율 다르게 해서 예측
- 데이콘 제출 후 가장 성능 좋은 모델 채택

④ 결과 및 느낀점

결과

RoBERTa-large

Public 점수: 0.8545454545
Private 점수: 0.8151554972

Ensemble(voting)
-koelectra(0.1) large(0.1)
small(0.5) base(0.3)

Public점수 : 0.8403066813
Private 점수: 0.8136224266

느낀점

- 앙상블이 더 결과가 좋을 것이라 예측했지만 RoBERTa-large 단독모델이 더 성능이 좋은 것을 보고 여러가지 시도가 필요함을 알 수 있었다.

- NLP에서 전처리 과정보다 모델 선택이 더 중요하게 작용된다는 점을 알게 되었다.