NLP - [분류]

감성분석을 통한 긍정적인 말하기의 실천

18기 - 심서현

16기 - 이수찬

TABLE OF CONTENTS

01: 프로젝트 소개

- NLP 분반 수업 과정에서 학습한 내용의 응용
- 감정분석 모델 활용 방법 탐구
- SNS에 게시되는 과격한 표현
  대상과 상황에 맞지 않는 표현으로 인한 오해 및 갈등 방지

01: 프로젝트 소개

- 주어진 텍스트가 긍정, 혹은 부정적으로
  받아들여질지를 미리 판별할 수 있는
  감성 분석 모델을 구축

➢ 온라인 대화 상에서 갈등이 발생하는
현상을 완화하는 수단으로써 활용 가능한 기능을 구현

02: 데이터 전처리 - 1차 모델 구현

	Unnamed: 0	연령	성별	상황키워드	신체질환	감정_대분류	감정_소분류	사람문장1	시스템문장1	사람문장2	시스템문장2	사람문장3	시스템문장3
0	1	청년	여성	진로, 취업, 직장	해당없음	분노	노여워하는	일은 왜 해도 해도 끝이 없을까? 화가 난다.	많이 힘드시겠어요. 주위에 의논할 상대가 있나요?	그냥 내가 해결하는 게 나아. 남들한테 부담주고 싶지도 않고.	혼자 해결하기로 했군요. 혼자서 해결하기 힘들면 주위에 의논할 사람을 찾아보세요.	NaN	NaN
1	2	청년	여성	진로, 취업, 직장	해당없음	분노	노여워하는	이번 달에 또 급여가 깎였어! 물가는 오르는데 월급만 자꾸 깎이니까 너무 화가 나.	급여가 줄어 속상하시겠어요. 월급이 줄어든 것을 어떻게 보완하실 건가요?	최대한 지출을 억제해야겠어. 월급이 줄어들었으니 고정지출을 줄일 수밖에 없을 것 같아.	월급이 줄어든 만큼 소비를 줄일 계획이군요.	NaN	NaN

```
#감정 대분류를 기준으로 pos, neg 판단
train = train[['감정_대분류','사람문장1']]
test = test[['감정_대분류','사람문장1']]
```

```
train = train.rename(columns={"감정_대분류": "label", "사람문장1": "document"})
test = test.rename(columns={"감정_대분류": "label", "사람문장1": "document"})
```

- 감정 대화 텍스트 데이터 사용

➔ 감정 대분류, 사람문장 1을 사용

[감정, 문장] 형식의 Train, Test 데이터셋 구현

➔ 감정 대분류 내 감정 레이블을 사용해 긍정/부정 감정 판별

02: 데이터 전처리 - 1차 모델 구현

```
#기쁨은 1 이외의 label은 다 0으로 취급 (0: negative, 1: positive)
train['label'] = train['label'].replace(['불안','분노','상처','슬픔','당황'], 0)
train['label'] = train['label'].replace(['기쁨'], 1)
test['label'] = test['label'].replace(['불안','분노','상처','슬픔','당황'], 0)
test['label'] = test['label'].replace(['기쁨'], 1)
```

```
[ ] train.label.value_counts()
    #부정 데이터가 약 6배 정도 많은 불균형 존재.
```

```
불안    9320
분노    9160
상처    9143
슬픔    9125
당황    8756
기쁨    6126
Name: label, dtype: int64
```

- 기쁨은 1,
  이외의 label은 0으로 분류
  (0 : negative, 1 : positive)
- 부정 데이터 label 다수
  ➢ **부정 판독 경향성**

02: 데이터 전처리 - 2차 모델 구현

```
[ ]  # 네이버 영화 리뷰 데이터 다운로드

     !git clone https://github.com/e9t/nsmc.git
```

```
[ ]  train_new = rate_train[rate_train['label'] == 1]
     movie_train = train_new[['label', 'document']]


     movie_train.head()
```

	label	document
1	1	흠...포스터보고 초딩영화줄....오버연기조차 가볍지 않구나
4	1	사이몬페그의 익살스런 연기가 돋보였던 영화!스파이더맨에서 늙어보이기만 했던 커스틴 ...
8	1	액션이 없는데도 재미 있는 몇안되는 영화
9	1	왜케 평점이 낮은건데? 꽤 볼만한데.. 헐리우드식 화려함에만 너무 길들여져 있나?
10	1	걍인피니트가짱이다.진짜짱이다♥

- 데이터 불균형 문제 해결

➔ 네이버 영화 리뷰 데이터 중 긍정 리뷰 데이터 (label = 1) 사용

02: 데이터 전처리 - 최종 데이터셋

```
[ ]  train = pd.concat([movie_train[['label', 'document']],  train[[ 'label', 'document']]], ignore_index=True)
```

```
train.head()
```

	label	document
0	0	일은 왜 해도 해도 끝이 없을까? 화가 난다.
1	0	이번 달에 또 급여가 깎였어! 물가는 오르는데 월급만 자꾸 깎이니까 너무 화가 나.
2	0	회사에 신입이 들어왔는데 말투가 거슬려. 그런 애를 매일 봐야 한다고 생각하니까 스...
3	0	직장에서 막내라는 이유로 나에게만 온갖 심부름을 시켜. 일도 많은 데 정말 분하고 ...
4	0	얼마 전 입사한 신입사원이 나를 무시하는 것 같아서 너무 화가 나.

- 기존 train 데이터셋에 네이버 영화 리뷰 데이터셋 추가
- ['label', 'document'] 형식의 최종 데이터셋을 구현

03: 감성분류 모델 구현

- 문장의 시작,끝 구분

```
[ ]  #시작을 [CLS], 끝을 [SEP] 로 표시.
     document_bert = ["[CLS] " + str(s) + " [SEP]" for s in train.document]
     document_bert[:5]
```

```
['[CLS] 흠...포스터보고 초딩영화줄....오버연기조차 가볍지 않구나 [SEP]',
 '[CLS] 사이몬페그의 익살스런 연기가 돋보였던 영화!스파이더맨에서 늙어보이기만 했던 커스틴 던스트가 너무나도 이뻐보였다 [SEP]',
 '[CLS] 액션이 없는데도 재미 있는 몇안되는 영화 [SEP]',
 '[CLS] 왜케 평점이 낮은건데? 꽤 볼만한데.. 헐리우드식 화려함에만 너무 길들여져 있나? [SEP]',
 '[CLS] 걍인피니트가짱이다.진짜짱이다♥ [SEP]']
```

- 전이 학습

```
[ ]  #Pretrained 된 multilingual bert 모델을 불러와서 전이 학습.
     tokenizer = BertTokenizer.from_pretrained('bert-base-multilingual-cased', do_lower_case=False)
     tokenized_texts = [tokenizer.tokenize(s) for s in document_bert]
     print(tokenized_texts[0])
```

03: 감성분류 모델 구현

- 패딩 추가 및 구별 처리

```
[ ]  #패딩 추가
     MAX_LEN = 128
     input_ids = [tokenizer.convert_tokens_to_ids(x) for x in tokenized_texts]
     input_ids = pad_sequences(input_ids, maxlen=MAX_LEN, dtype='long', truncating='post', padding='post')
     input_ids[0]
```

```
     #패딩 있는 곳과 없는 곳을 나누어 지정

     attention_masks = []

     for seq in input_ids:
         seq_mask = [float(i>0) for i in seq]
         attention_masks.append(seq_mask)

     print(attention_masks[0])
```

03: 감성분류 모델 구현

- BERT 모델

```
[ ]  # Pretrain Bert 모델 로딩하여 GPU로 올림.

     model = BertForSequenceClassification.from_pretrained("bert-base-multilingual-cased", num_labels=2)
     model.cuda()
```

- Optimizer, Scheduler 설정

```
[ ]  #Optimizer 설정
     optimizer = AdamW(model.parameters(),
                       lr = 2e-5,
                       eps = 1e-8
                     )

     epochs = 4

     total_steps = len(train_dataloader) * epochs

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

03: 감성분류 모델 구현

```
[ ]  seed_val = 42
     random.seed(seed_val)
     np.random.seed(seed_val)
     torch.manual_seed(seed_val)
     torch.cuda.manual_seed_all(seed_val)

     model.zero_grad()
```

- 데이터 학습 및 정확도 계산

➢ 정확도 98% 달성

```
======== Epoch 4 / 4 ========
Training...
  Batch   500  of  3,557.    Elapsed: 0:05:09.
  Batch 1,000  of  3,557.    Elapsed: 0:10:17.
  Batch 1,500  of  3,557.    Elapsed: 0:15:26.
  Batch 2,000  of  3,557.    Elapsed: 0:20:34.
  Batch 2,500  of  3,557.    Elapsed: 0:25:43.
  Batch 3,000  of  3,557.    Elapsed: 0:30:51.
  Batch 3,500  of  3,557.    Elapsed: 0:35:59.

  Average training loss: 0.03
  Training epcoh took: 0:36:34

Running Validation...
  Accuracy: 0.98
  Validation took: 0:01:19

Training complete!
```

03: 감성분류 모델 구현 - 감정 도출

```
#예측할 문장을 직접 입력하게 해주는 함수
def pred_text(input_text):
  text = [input_text]
  text_bert = ["[CLS] " + t + " [SEP]" for t in text]
  tokenized_text = [tokenizer.tokenize(s) for s in text_bert]

  input_text = [tokenizer.convert_tokens_to_ids(x) for x in tokenized_text]
  input_text = pad_sequences(input_text, maxlen=MAX_LEN, dtype='long', truncating='post', padding='post')

  attention_mask_text= []
  for seq in input_text:
    seq_mask = [float(i>0) for i in seq]
    attention_mask_text.append(seq_mask)



  input_text=torch.tensor(input_text)
  attention_mask_text =torch.tensor(attention_mask_text)
  input_text = input_text.to(device)
  attention_mask_text =attention_mask_text.to(device)
  with torch.no_grad():
    outputs = model(input_text,
    token_type_ids=None,
    attention_mask=attention_mask_text)
  logit = outputs[0].detach().cpu().numpy()
  pred = np.argmax(logit, axis=1)
  return pred
```

- 입력받은 문장 긍정/부정 판독

```
x=''
while x != 'stop':
  x = input()
  print(pred_text(x))
```

```
오늘 기분이 좋아
[1]
나 그 사람 마음에 안들어
[0]
```

04: 프로젝트 결과

- 2차 모델 구현으로 정확도 향상 성공 .
  94% -> 98%
- 부정 표현 판독 편중 개선 성공.
- 긍정 표현을 판독하는 정확도가 높아짐.

➢ 감정 분석 데이터셋 학습을 통한 긍정/부정 감정 판독 기능을 구현

➢ 감정 판독에 있어 높은 정확도를 보이는 모델을 구현하는데 성공

```
Running Validation...
  Accuracy: 0.98
  Validation took: 0:01:19

======== Epoch 4 / 4 ========
Training...
  Batch   500  of  3,557.    Elapsed: 0:05:09.
  Batch 1,000  of  3,557.    Elapsed: 0:10:17.
  Batch 1,500  of  3,557.    Elapsed: 0:15:26.
  Batch 2,000  of  3,557.    Elapsed: 0:20:34.
  Batch 2,500  of  3,557.    Elapsed: 0:25:43.
  Batch 3,000  of  3,557.    Elapsed: 0:30:51.
  Batch 3,500  of  3,557.    Elapsed: 0:35:59.

  Average training loss: 0.03
  Training epcoh took: 0:36:34

Running Validation...
  Accuracy: 0.98
  Validation took: 0:01:19

Training complete!
```

04: 프로젝트 결과

- 데이터 불균형으로 인해 부정으로 판정하는 데이터의 비율이 높음.
- 데이터셋에 없는 표현은 잘 파악하지 못하는 경향.
- 짧은 텍스트의 경우 감정을 잘 파악하지 못함.

- 텍스트의 길이가 늘어나면 감정 파악 정확도가 높아짐.

```
x=''
while x != 'stop':
  x = input()
  print(pred_text(x))
```

```
별로야
[1]
그 사람 별로야. 어떻게 그렇게 행동하지?
[0]
```

04: 프로젝트 결과

- 글을 게시하기 전, 자신의 글에 어떠한 감정이 지배적인지,
  긍정적, 부정적으로 받아들여질지를 판별하는 기능 구현
- 부정 표현의 민감도가 높은 경향
  - ➢ 혐오 표현, 오해를 야기할 수 있는 표현 감지 및 방지에 유용

➔ **추후 다양한 대화 스타일이나 신조어를 학습**
긍정, 부정 판독 외 세부적인 감정을 분석할 수 있는 모델로 발전 가능

THANK YOU!