Information Search Project

B813005 고민재

주제 결정

에브리타임 홍.맛.게 Sentiment Analysis

사람들이 쓴 글을 분석해서 긍정후기와 부정후기로 분류하는 딥러닝을 해보자

대학생만을 위한 커뮤니티

에브리타임

에브리타임 <야 넌 홍대생인데 홍대맛집도 모르냐-를 방지하기 위한 홍대맛집게시판> 크롤링

Keras RNN으로 네이버 영화 리뷰 감성 분석

Hugging Face의 PyTorch BERT로 모델 만들어 전이 학습

진행 상황

데이터 수집

HuggingFace BERT를 사용한 네이버 영화리뷰 감정분석 https://github.com/deepseasw/bert-naver-movie-review

BERT 모델

Utilize

RNN/ LSTM 모델

딥러닝을 이용한 자연어 처리 입문 https://wikidocs.net/44249

최종 완성

진행 상황

BERT.ipynb ☆

파일 수정 보기 삽입 런타임 도구 도움말 모든 변경사항이 저장됨

+ 코드 + 텍스트

모델 생성

```
[ ] if torch.cuda.is_available():
      device = torch.device("cuda")
      print("We will use the GPU:", torch.cuda.get_device_name(0))
```

```
We will use the GPU: Tesla T4
```

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

```
Downloading: 100% 625/625 [00:16<00:00, 38.5B/s]

Downloading: 100% 714M/714M [00:12<00:00, 59.5MB/s]

Some weights of the model checkpoint at bert-base-multilingual-cased were not used when initializing BertF
- This IS expected if you are initializing BertForSequenceClassification from the checkpoint of a model tr
- This IS NOT expected if you are initializing BertForSequenceClassification from the checkpoint of a mode
Some weights of BertForSequenceClassification were not initialized from the model checkpoint at bert-base-
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and infere
BertForSequenceClassification(
  (bert): BertModel(
    (embeddings): BertEmbeddings(
      (word_embeddings): Embedding(119547, 768, padding_idx=0)
      (position_embeddings): Embedding(512, 768)
      (token_type_embeddings): Embedding(2, 768)
      (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
```

RNN.ipynb ☆

파일 수정 보기 삽입 런타임 도구 도움말 모든 변경사항이 저장됨

+ 코드 + 텍스트

```
[23] X_test = []
     for sentence in test_data['document']:
       temp_X = []
       temp_X = okt.morphs(sentence, stem=True) #토큰화
       temp_X = [word for word in temp_X if not word in stopwords] #불용어 제거
       X_test.append(temp_X)
```

```
[24] tokenizer = Tokenizer()
     tokenizer.fit_on_texts(X_train)
```

```
[25] print(tokenizer.word_index)
```

```
{'영화': 1, '보다': 2, '을': 3, '없다': 4, '이다': 5, '있다': 6, '좋다': 7, '너무': 8, '다': 9, '정말
```

```
[26] threshold = 3
     total_cnt = len(tokenizer.word_index) #단어의 수
     rare_cnt = 0 #등장 빈도수가 threshold보다 작은 단어의 개수를 카운트
     total_freq = 0 #훈련 데이터의 전체 단어 빈도수 총합
     rare_freq = 0 #등장 빈도수가 threshold보다 작은 단어의 등장 빈도수 총합

     #단어와 빈도수의 쌍을 key와 value로 받는다
     for key, value in tokenizer.word_counts.items():
       total_freq = total_freq + value

       #단어의 등장 빈도수가 threshold보다 작으면
       if(value < threshold):
         rare_cnt = rare_cnt + 1
         rare_freq = rare_freq + value

     print("단어 집합의 크기 : ", total_cnt)
     print("등장 빈도가 %s번 이하인 희귀 단어의 수 : %s" %(threshold-1, rare_cnt))
     print("단어 집합에서 희귀 단어의 비율 : ", (rare_cnt/total_cnt)*100)
     print("전체 등장 빈도에서 희귀 단어 등장 비도의 비율 : ", (rare_freq/total_freq)*100)
```

```
단어 집합의 크기 :  43752
등장 빈도가 2번 이하인 희귀 단어의 수 : 24337
단어 집합에서 희귀 단어의 비율 :  55.62488571950996
전체 등장 빈도에서 희귀 단어 등장 비도의 비율 :  1.8715872104872904
```

```
#전체 단어 개수 중 빈도수 2 이하인 단어 개수는 제거
#0번 패딩 토큰과 1번 OOV 토큰을 고려하여 +2
vocab_size = total_cnt - rare_cnt + 2
print("단어 집합의 크기 : ", vocab_size)
```

```
단어 집합의 크기 :  19417
```

```
tokenizer = Tokenizer(vocab_size, oov_token = 'OOV')
tokenizer.fit_on_texts(X_train)
X_train = tokenizer.texts_to_sequences(X_train)
X_test = tokenizer.texts_to_sequences(X_test)
```

```
[29] print(X_train[:3])
```

```
[[51, 455, 17, 261, 660], [934, 458, 42, 603, 2, 215, 1450, 25, 962, 676, 20], [387, 2445, 1, 2316, 5672, 3, 223, 10]]
```

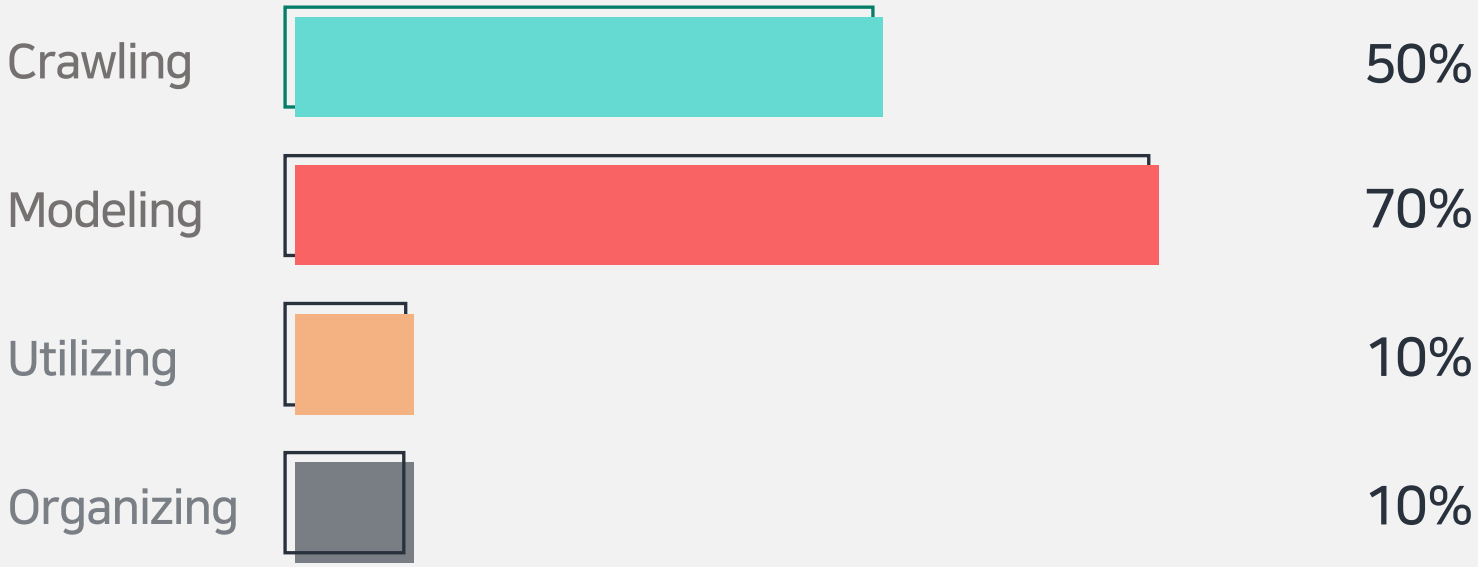
```
[30] y_train = np.array(train_data['label'])
     y_test = np.array(test_data['label'])
```

```
[31] drop_train = [index for index, sentence in enumerate(X_train) if len(sentence) < 1]
```

```
[32] #빈 샘플들을 제거
     X_train = np.delete(X_train, drop_train, axis=0)
     y_train = np.delete(y_train, drop_train, axis=0)
     print(len(X_train))
     print(len(y_train))
```

```
145380
145380
```

진행 상황



이후 계획

```
def sentiment_predict(new_sentence):
  new_sentence = okt.morphs(new_sentence, stem=True) # 토큰화
  new_sentence = [word for word in new_sentence if not word in stopwords] # 불용어 제거
  encoded = tokenizer.texts_to_sequences([new_sentence]) # 정수 인코딩
  pad_new = pad_sequences(encoded, maxlen = max_len) # 패딩
  score = float(loaded_model.predict(pad_new)) # 예측
  if(score > 0.5):
    print("{:.2f}% 확률로 긍정 리뷰입니다.₩n".format(score * 100))
  else:
    print("{:.2f}% 확률로 부정 리뷰입니다.₩n".format((1 - score) * 100))
```

```
sentiment_predict('이 영화 개꿀잼 ㅋㅋㅋ')
sentiment_predict('감독 뭐하는 놈이냐?')
sentiment_predict('이 영화 핵노잼 ㅠㅠ')
sentiment_predict('이딴게 영화냐 ㅉㅉ')
sentiment_predict('와 개쩐다 정말 세계관 최강자들의 영화다')
```

```
92.34% 확률로 긍정 리뷰입니다.

98.30% 확률로 부정 리뷰입니다.

98.74% 확률로 부정 리뷰입니다.

99.85% 확률로 부정 리뷰입니다.

90.54% 확률로 긍정 리뷰입니다.
```

```
sample_file = abstract_yonsei_reviews[100]
df_abstract = pd.read_csv(sample_file, encoding="utf-8")
df_abstract.sample(5)
```

	No	제목	학과	과정	년도	href
49	37	SCSU	심리학과	학부	2009	/partner/expReport.asp?id=3989&page=5&bgbn=R
11	75	세인트 클라우드 주립대학교	경제학과	학부	2013	/partner/expReport.asp?id=6900&page=2&bgbn=R
10	76	세인트 클라우드에서의 한 학기	영어영문학과	학부	2013	/partner/expReport.asp?id=6922&page=2&bgbn=R
16	70	세인트클라우드에서의 소중한 시간	문화인류학과	학부	2012	/partner/expReport.asp?id=6077&page=2&bgbn=R
55	31	08 SCSU 가을학기	법학과	학부	2008	/partner/expReport.asp?id=3743&page=6&bgbn=R

```
one_review_title = df_abstract["제목"][16]
print(one_review_title)
sentiment_predict(one_review_title)
```

```
세인트클라우드에서의 소중한 시간
76.41% 확률로 긍정 리뷰입니다.
```

활용자료

https://github.com/deepseasw/bert-naver-movie-review
https://github.com/snoop2head/yonsei-exchange-program
https://wikidocs.net/44249
https://blog.yourssu.com/post/everytime-bot-1/
https://itsallgoodman.tistory.com/9

Thank You