네이버 쇼핑 리뷰 데이터 감성 분석(분류)

박은정

INDEX

01

데이터 소개 및 가설 설정

사용한 데이터: 네이버 쇼핑 리뷰 데이터

네이버 쇼핑의 제품별 후기와 별점이 함께 수집된 데이터

(의류, 잡화, 미용, 가전, 식품 등 다양한 분야 상품 포함)

	ratings	reviews
0	5	배공빠르고 굿
1	2	택배가 엉망이네용 저희집 밑에층에 말도없이 놔두고가고
2	5	아주좋아요 바지 정말 좋아서2개 더 구매했어요 이가격에 대박입니다. 바느질이 조금 ...
3	2	선물용으로 빨리 받아서 전달했어야 하는 상품이었는데 머그컵만 와서 당황했습니다. 전...
4	5	민트색상 예뻐요. 옆 손잡이는 거는 용도로도 사용되네요 ㅎㅎ

출처: 크롤링을 통해 한 브랜드의 리뷰 데이터를 수집하려고 했으나 시간 상의 문제로 Github에 공개된 데이터 사용

https://github.com/bab2min/corpus/blob/master/sentiment/README.md

사용한 데이터: 네이버 쇼핑 리뷰 데이터

데이터 정보

- 데이터 수: 200,000 개
- 데이터 수집 기간: 2020.06 ~ 2020.07 데이터
- 데이터 분포: 긍정(99,963), 부정(100,037)
- 긍/부정으로 분류하기 애매한 3점에 해당하는 텍스트들은 제외하고 긍정(4~5점) & 부정(1~2점) 비율이 1:1이 되도록 샘플링

주제 및 데이터 선정 이유

회사는 소비자의 의견을 파악 위해
회사 제품이나 서비스에 대한 사용자
리뷰 데이터를 분석

이때 텍스트 감성 분석 or NLP 기반
키워드 추출을 사용해 마케팅에 활용

따라서 텍스트와 같은 비정형 데이터
처리하는 스킬을 기르기 위해
감성 분석 모델이라는 주제를 선정

01 데이터 소개 및 가설 설정 2. 가설 설정

가설

초기 가설

BERT 모델을 활용해 사용자 리뷰
핵심 키워드 추출 & 감성 분석 실시

키워드 추출을 통해 추출된 핵심어와
감성분석 모델의 분석 결과 사용해
리뷰가 어떤 부분에서 긍정적인지
부정적인지 판별할 수 있다

수정한 가설

GRU 모델과 BERT 모델을 활용해
감성 분석 실시

BERT 모델의 정확도가 GRU 모델의
정확도보다 높을 것이다

02

모델 소개 및 모델 학습

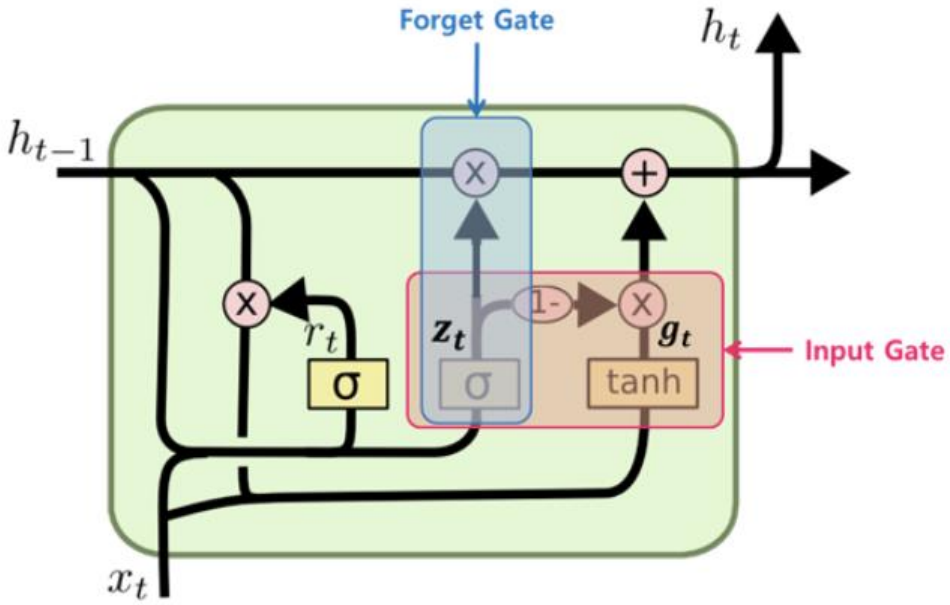
02 모델 소개 및 모델 학습 1. 모델 소개

1. GRU 모델

2. BERT 모델

02 모델 소개 및 모델 학습 1. 모델 소개

GRU 모델

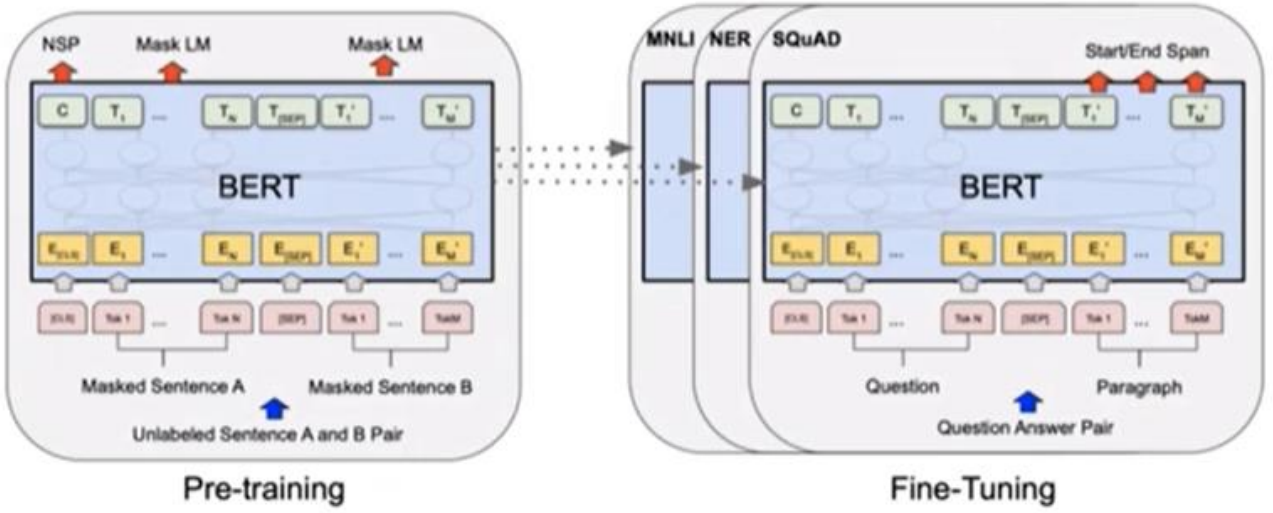


RNN(순환 신경망) 기반의 LSTM 모델의 간소한 버전

자연어와 같이 연속형 데이터 처리에 특화되어 있음

02 모델 소개 및 모델 학습 1. 모델 소개

BERT 모델



Transformer의 Encoder 사용해 문맥을 양방향으로 읽어내는 모델

사전학습: 대량의 데이터를 사용해 미리 학습하는 과정(레이블링 되지 않은 말뭉치 학습: 비지도 학습)

Fine-tuning: 사전학습이 끝난 모델에 우리가 하고자 하는 태스크에 특화된 데이터 학습시켜 모델의 성능 최적화

02 모델 소개 및 모델 학습 2. 모델 학습

공통

데이터 전처리

ratings	reviews	label
5	배공빠르고 굿	1
2	택배가 엉망이네용 저희집 밑에층에 말도없이 놔두고가고	0
5	아주좋아요 바지 정말 좋아서2개 더 구매했어요 이가격에 대박입니다. 바느질이 조금 ...	1
2	선물용으로 빨리 받아서 전달했어야 하는 상품이었는데 머그컵만 와서 당황했습니다. 전...	0
5	민트색상 예뻐요. 옆 손잡이는 거는 용도로도 사용되네요 ㅎㅎ	1

네이버 쇼핑 리뷰 데이터 업로드 후
Ratings 4,5 -> 1, Ratings 1,2->0 라벨 생성

```
[8]  #중복값 제거
     df.drop_duplicates(subset=['reviews'], inplace=True)
     print(len(df))

     199908
```

```
[9]  #결측치(null값) 확인
     df.isnull().sum()

     ratings    0
     reviews    0
     label      0
     dtype: int64
```

```
[10] #훈련/테스트 데이터 분리
     #테스트 데이터 -> 25%
     train, test = train_test_split(df, test_size = 0.25, random_state = 42)
     train.shape, test.shape

     ((149931, 3), (49977, 3))
```

중복 값/결측치 처리 및 train/test 데이터 분리
분리한 이후 train 데이터: 149931, test 데이터: 49977

02 모델 소개 및 모델 학습 2. 모델 학습

GRU 모델

1. 데이터 정제

```
#한글, 공백 제외하고 제거
train['reviews']=train['reviews'].str.replace("[^ㄱ-ㅎㅏ-ㅣ가-힣 ]","")
#한글, 공백 제외하고 제거
test['reviews']=test['reviews'].str.replace("[^ㄱ-ㅎㅏ-ㅣ가-힣 ]","")
```

reviews	label	tokenized
사이즈를 센치씩 늘린건데도 작아요 그리고 색상은 완전 달라요 칙칙한핑크네요ㅠㅠ 많이...	0	[사이즈, 센치, 씩, 늘린, 건데, 작, 아요, 그리고, 색상, 완전, 달라요, ...
ㅂ불만족 빗이 아픔 멍이피부에 빗질못해주겟네요	0	[ㅂ, 불, 만족, 빗이, 아픔, 멍, 피부, 빗, 질, 못해, 주, 겟, 네요]
이 제품쓰고 삼일만에 변기물이 잘 안내려갔어요 혹시나해서 다시 빼보니 물이 다시 잘...	0	[제품, 쓰, 삼, 일, 만, 변기, 물, 잘, 안, 내려갔, 어요, 혹시나, 해서...
적당하고 만족합니다	1	[적당, 만족, 합니다]
편하자고 이용하는 밀키튼데 손 은근 많이 가서 저는 패쓰요	0	[편하, 자고, 이용, 밀키, 튼, 데, 손, 은근, 많이, 서, 저, 패, 쓰, 요]

리뷰 한글, 공백 제외하고 제거 -> Mecab 이용해 토큰화 & 불용어 제거

02 모델 소개 및 모델 학습 2. 모델 학습

GRU 모델

1. 데이터 정제

```
[ ] #단어마다 index 값 붙음
    #정수는 데이터에서 등장 빈도수가 높은 순서대로 부여됨 -> 높은 정수가 부여된 단어들은 등장 빈도수가 낮
    tokenizer = Tokenizer()
    tokenizer.fit_on_texts(X_train)
```

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

```
{'네요': 1, '좋': 2, '어요': 3, '는데': 4, '아요': 5, '잘': 6, '있': 7, '구매': 8, '안': 9, '배송': 10
```

```
#빈도수 2 이하인 단어 제거
#0번 패딩 토큰, 1번 OOV 토큰 고려해 +2
vocab_size = total_count - rare_count + 2
print(vocab_size)
```

```
21787
```

```
[ ] #max_len=50으로 맞추기
    max_len=50
    X_train = pad_sequences(X_train, maxlen=max_len)
    X_test = pad_sequences(X_test, maxlen=max_len)
```

정수 인코딩 수행 -> 빈도 수가 낮은 단어 제거 -> 패딩

02 모델 소개 및 모델 학습 2. 모델 학습

GRU 모델

2. 학습 진행

```
model = Sequential()
model.add(Embedding(vocab_size, 100))  #임베딩 벡터 차원: 100
model.add(GRU(128))
model.add(Dense(1, activation='sigmoid')) #0,1 이진 분류
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, None, 100)         2178700
_________________________________________________________________
gru (GRU)                    (None, 128)               88320
_________________________________________________________________
dense (Dense)                (None, 1)                 129
=================================================================
Total params: 2,267,149
Trainable params: 2,267,149
Non-trainable params: 0
```

```
model.compile(optimizer='rmsprop', loss='binary_crossentropy', metrics=['acc'])
model.fit(X_train, y_train, epochs=10, callbacks=[es, mc], batch_size=100, validation_split=0.2)
```

Earlystopping 적용해 검증 데이터 손실 4회 증가하면 학습 조기 종료
에포크 10번 수행
훈련 데이터 중 20% 검증데이터로 사용

02 모델 소개 및 모델 학습 2. 모델 학습

GRU 모델

3. 학습 결과

```
#검증 데이터의 정확도가 가장 높았을 때 저장된 모델 로드
loaded = load_model('best_model.h5')
loaded.evaluate(X_test, y_test)
```

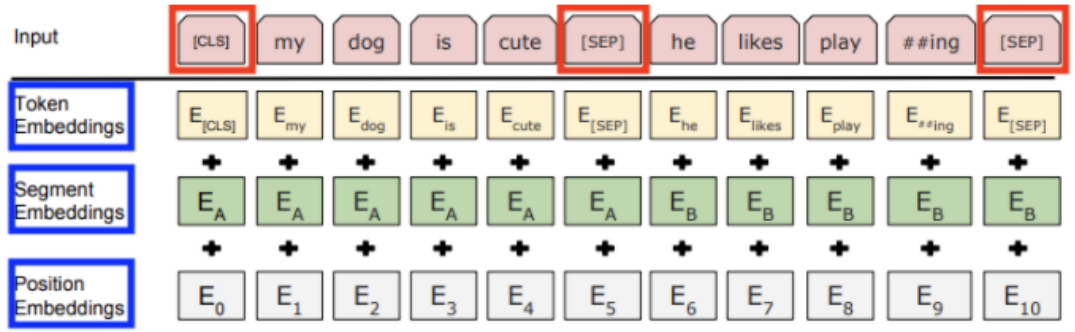
```
1562/1562 [==============================] - 10s 6ms/step - loss: 0.2066 - acc: 0.9264
[0.20664027333259583, 0.9263661503791809]
```

검증 데이터의 정확도가 가장 높았을 때 저장된 모델 로드 후 test 데이터 acc 확인 -> 0.9264

02 모델 소개 및 모델 학습 2. 모델 학습

BERT 모델

1. 데이터 정제




```
#BERT INPUT 형식에 맞게 변환
sentences = ["[CLS] " + str(sentence) + " [SEP]" for sentence in sentences]
sentences[:10]
```

```
['[CLS] 사이즈를 3센치씩 늘린건데도 작아요 그리고 색상은 완전 달라요 칙칙한펑크네요ㅠㅠ 많이 아쉽지만 암막효과는 좋아요 [SEP]',
 '[CLS] ㅂ불만족.. 빗이 아픔 .. 멍이피부에 빗질못해주겟네요 [SEP]',
 '[CLS] 이 제품쓰고 삼일만에 변기물이 잘 안내려갔어요. 혹시나해서 다시 빼보니 물이 다시 잘 내려가네요.., 이 많은걸 다 어쩌나요, ..반품하고싶다.., [SEP]',
 '[CLS] 적당하고 만족합니다 [SEP]',
 '[CLS] 편하자고 이용하는 밀키트데 손 은근 많이 가서 저는 패쓰요 [SEP]',
 '[CLS] 아기 어릴때부터 항상 써오던거에요~ [SEP]',
 '[CLS] 너무좋가겨 방금우리집데도둑기다 [SEP]',
 '[CLS] 몸에 좋은 카카오닙스 강력 추천 배송도 빠르고 지프펙으로 250g 2팩 왔네요. 보관도 쉽고 먹기도 편합니다.올해는 건강하게 출발할거 같네요.ㅎㅎㅎ [SEP]',
 '[CLS] 바르먄 건조해요. 아직 잘 모르겠다는 ㅡㅡ [SEP]',
 '[CLS] 빠른배송~슬림핏이라서 너무 타잇하면어쩌나했는데 딱이쁩니다~ [SEP]']
```

리뷰 문장 추출한 후 BERT의 입력 형식에 맞게 변환
문장의 시작을 알리는 [CLS] 토큰과 문장 종결을 의미하는 [SEP] 토큰 추가

02 모델 소개 및 모델 학습 2. 모델 학습

1. 데이터 정제

```
#BERT 토크나이저로 토큰으로 분리
tokenizer = BertTokenizer.from_pretrained('bert-base-multilingual-cased', do_lower_case=False)
tokenized = [tokenizer.tokenize(s) for s in sentences]

print(sentences[0], tokenized[0])
```

[CLS] 사이즈를 3센치씩 늘린건데도 작아요 그리고 색상은 완전 달라요 칙칙한펑크네요ㅠㅠ 많이 아쉽지만 암막효과는 좋아요 [SEP]

['[CLS]', '사', '##이', '##즈', '##를', '3', '##센', '##치', '##씩', '늘', '##린', '##건', '##데', '##도', '작', '##아', '##요', '그리고', '색', '##상', '##은', '완', '##전', '달', '##라', '##요',

'[UNK]', '많이', '아', '##쉽', '##지만', '암', '##막', '##효', '##과', '##는', '좋', '##아', '##요', '[SEP]']

BERT의 토크나이저로 각 문장을 토큰으로 분리

02 모델 소개 및 모델 학습 2. 모델 학습

1. 데이터 정제

```
MAX_LEN = 128 #최대 시퀀스 길이
input = [tokenizer.convert_tokens_to_ids(x) for x in tokenized] #토큰 -> 숫자 인덱스로 변환
input = pad_sequences(input, maxlen=MAX_LEN, dtype="long", truncating="post", padding="post") #문장 MAX_LEN 길이에 맞추고, 모자란 부분은 패딩 0

input[0]
```

```
array([   101,   9405,  10739,  24891,  11513,    124, 119044,  18622,
       119108,   9044,  27654,  71439,  28911,  12092,   9652,  16985,
        48549,  23289,   9416,  14871,  10892,   9591,  16617,   9061,
        17342,  48549,    100,  47058,   9519, 119072,  28578,   9526,
       118907, 119449,  11882,  11018,   9685,  16985,  48549,    102,
            0,      0,      0,      0,      0,      0,      0,      0,
            0,      0,      0,      0,      0,      0,      0,      0,
            0,      0,      0,      0,      0,      0,      0,      0,
            0,      0,      0,      0,      0,      0,      0,      0,
            0,      0,      0,      0,      0,      0,      0,      0,
            0,      0,      0,      0,      0,      0,      0,      0,
            0,      0,      0,      0,      0,      0,      0,      0,
            0,      0,      0,      0,      0,      0,      0,      0,
            0,      0,      0,      0,      0,      0,      0,      0])
```

각 토큰을 인덱스로 변환 -> 패딩을 통해 모든 데이터의 길이 통일

02 모델 소개 및 모델 학습 2. 모델 학습

1. 데이터 정제

```
#BERT의 경우 제로 패딩으로 입력된 토큰 -> 항상 마스킹 처리, 이 토큰에 대해서는 페널티 부과해 어텐션 점수를 받지 못하도록 구현
attentionmask = [] #어텐션 마스크

for sequence in input:
    seq_mask = [float(i > 0) for i in sequence] #어텐션 마스크: 패딩이 아니면 1, 패딩이면 0
    attentionmask.append(seq_mask)

print(attentionmask[0])
```

```
[1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
```

이후 패딩에 해당하는 부분은 0, 그 외에는 1로 표시해 패딩에 해당하는 부분은 모델 학습 시 어텐션을 수행하지 않도록 함
입력값, 어텐션 마스크, 라벨을 하나로 묶어 입력 데이터를 구성한 후 모델 학습

02 모델 소개 및 모델 학습 2. 모델 학습

1. 데이터 정제

```
#train/val 분리
train_input, val_input, train_labels, val_labels = train_test_split(input, labels, random_state=42, test_size=0.1)

#어텐션 마스크 -> train/val 분리
train_masks, val_masks, _, _ = train_test_split(attentionmask,input,random_state=42, test_size=0.1)

#Pytorch 텐서(코어 데이터 구조)로 변환
train_input = torch.tensor(train_input)
train_labels = torch.tensor(train_labels)
train_masks = torch.tensor(train_masks)

val_input = torch.tensor(val_input)
val_labels = torch.tensor(val_labels)
val_masks = torch.tensor(val_masks)
```

```
#배치 사이즈
batch_size = 32

#Pytorch DataLoader로 input, masks, labels 묶어 데이터 설정
#train
train_data = TensorDataset(train_input, train_masks, train_labels)
train_sampler = RandomSampler(train_data)
train_dataloader = DataLoader(train_data, sampler=train_sampler, batch_size=batch_size) #학습 시 batch size만큼 데이터 가져옴
#val
val_data = TensorDataset(val_input, val_masks, val_labels)
val_sampler = SequentialSampler(val_data)
val_dataloader = DataLoader(val_data, sampler=val_sampler, batch_size=batch_size)
```

train/val 분리 한 후 입력값, 어텐션 마스크, 라벨을 하나로 묶어 입력 데이터를 구성한 후 모델 학습

02 모델 소개 및 모델 학습 2. 모델 학습

2. 학습 진행

```
#분류 위한 BERT 모델 생성
model = BertForSequenceClassification.from_pretrained("bert-base-multilingual-cased", num_labels=2)
model.cuda()
```

```
  (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)
    )
    (encoder): BertEncoder(
```

```
#옵티마이저: transformers에서 제공하는 옵티마이저 AdamW를 사용
optimizer = AdamW(model.parameters(),lr = 2e-5, eps = 1e-8) # 0으로 나누는 것을 방지하기 위한 epsilon 값

#에폭
epochs = 4

#총 훈련 스텝 : 배치반복 횟수(이터레이션) * 에폭
total_steps = len(train_dataloader) * epochs

#처음에 학습률을 조금씩 변화시키는 스케줄러 생성
scheduler = get_linear_schedule_with_warmup(optimizer, num_warmup_steps = 0, num_training_steps = total_steps)
```

BERT 모델 생성
에포크 4번 수행

02 모델 소개 및 모델 학습 2. 모델 학습

3. 학습 결과

```
======== Epoch 4 / 4 ========
Training...
  Batch   500  of  4,217.    Elapsed: 0:05:56.
  Batch 1,000  of  4,217.    Elapsed: 0:11:51.
  Batch 1,500  of  4,217.    Elapsed: 0:17:47.
  Batch 2,000  of  4,217.    Elapsed: 0:23:42.
  Batch 2,500  of  4,217.    Elapsed: 0:29:38.
  Batch 3,000  of  4,217.    Elapsed: 0:35:34.
  Batch 3,500  of  4,217.    Elapsed: 0:41:29.
  Batch 4,000  of  4,217.    Elapsed: 0:47:25.

  Average training loss: 0.13
  Training epcoh took: 0:49:59

Running Validation...
  Accuracy: 0.92
  Validation took: 0:01:54
```

```
  Batch   100  of  1,562.    Elapsed: 0:00:24.
  Batch   200  of  1,562.    Elapsed: 0:00:48.
  Batch   300  of  1,562.    Elapsed: 0:01:13.
  Batch   400  of  1,562.    Elapsed: 0:01:37.
  Batch   500  of  1,562.    Elapsed: 0:02:01.
  Batch   600  of  1,562.    Elapsed: 0:02:26.
  Batch   700  of  1,562.    Elapsed: 0:02:50.
  Batch   800  of  1,562.    Elapsed: 0:03:14.
  Batch   900  of  1,562.    Elapsed: 0:03:39.
  Batch 1,000  of  1,562.    Elapsed: 0:04:03.
  Batch 1,100  of  1,562.    Elapsed: 0:04:27.
  Batch 1,200  of  1,562.    Elapsed: 0:04:51.
  Batch 1,300  of  1,562.    Elapsed: 0:05:16.
  Batch 1,400  of  1,562.    Elapsed: 0:05:40.
  Batch 1,500  of  1,562.    Elapsed: 0:06:04.

Accuracy: 0.92
Test took: 0:06:19
```

검증 데이터의 정확도: 0.92
테스트 데이터의 정확도: 0.92

03

모델 비교

03 모델비교

GRU 모델

리뷰 예측

```
def predict(sentence):
  sentence = re.sub(r'[^ㄱ-ㅎㅏ-ㅣ가-힣 ]', '', sentence)
  sentence = mecab.morphs(sentence) #토큰화
  sentence = [word for word in sentence if not word in stopwords] #불용어 처리
  encoded = tokenizer.texts_to_sequences([sentence]) #정수 인코딩
  pad = pad_sequences(encoded, maxlen=max_len) #패딩
  score = float(loaded.predict(pad)) #예측
  if score > 0.5:
    print("{:.2f}% 확률로 긍정 리뷰".format(score * 100))
  else:
    print("{:.2f}% 확률로 부정 리뷰".format((1-score) * 100))
```

```
predict('예쁜 줄 알고 샀는데 생각보다는 그렇게 예쁘지는 않다')
predict('흠 매우 애매함.. 뭐가 4점주긴 아깝고 2점주긴 괜찮고')
predict('역시 리뷰를 보고 사니 실패하지 않네') #왜 부정 리뷰로 나올까,,
predict('생각보단 괜찮음')
```

```
86.47% 확률로 부정 리뷰
86.84% 확률로 부정 리뷰
56.56% 확률로 부정 리뷰
56.72% 확률로 긍정 리뷰
```

세 번째 문장
'역시 리뷰를 보고 사니 실패하지 않네'
-> 부정 리뷰로 예측

03 모델비교

BERT 모델

리뷰 예측

```
logits = test_sentences(['예쁜 줄 알고 샀는데 생각보다는 그렇게 예쁘지는 않다'])

print(logits)
print(np.argmax(logits))
```

```
[[ 2.3514857 -2.4168046]]
0
```

```
logits = test_sentences(['역시 리뷰를 보고 사니 실패하지 않네'])

print(logits)
print(np.argmax(logits)) #GRU 모델처럼 여기서도 부정리뷰로 나옴
```

```
[[ 2.4488897 -2.4136095]]
0
```

```
logits = test_sentences(['흠 매우 애매함,, 뭔가 4점주긴 아깝고, 2점주긴 괜찮고'])

print(logits)
print(np.argmax(logits))
```

```
[[ 2.9188883 -2.9486833]]
0
```

```
logits = test_sentences(['생각보단 괜찮음'])

print(logits)
print(np.argmax(logits))
```

```
[[-1.054081   1.1439103]]
1
```

세 번째 문장
'역시 리뷰를 보고 사니 실패하지 않네'
-> GRU모델과 동일하게 부정 리뷰로 예측