WeTIE 머신러닝 톺아보기

NLP를 이용한 뉴스기사 카테고리 분류 및 감정 분석

A조 김서연 정다솜 박민서 이준서

Table of Contents

1. 분석 주제 및 목적

01

1. 분석 주제 및 목적

분석 주제

영어 뉴스 기사에 대한 카테고리 분류와 감정 분석을 위한 딥러닝 모델 설계

분석 목적

- 매일 수많은 뉴스기사가 발행됨
- 개인은 자신에게 맞는 뉴스를 찾는 것을 어려워함
- 카테고리 분류, 긍정/부정 분류를 통해 개인화된 뉴스 추천 시스템을 만들고자 함

2. 방법론

02

2. 방법론

실험 개요

1) 뉴스 기사 카테고리 분류

2) 뉴스 기사 감정 분석

1) 카테고리 분류

1) 데이터 전처리 후 토큰화

2) GloVe 모델 이용하여 단어 임베딩

3) LSTM 딥러닝 모델 훈련 및 성능 측정

2) 감정 분석

1) 데이터 전처리 후 토큰화

2) BERT 모델 이용한 감정 분석 모델 훈련 및 성능 측정

3. 카테고리 분류 모델

03

3. 카테고리 분류 모델

데이터

category	title	body
ARTS & CULTURE	Modeling Agencies Enabled Sexual Predators For Years, Former Agent Says	In October 2017, Carolyn Kramer received a disturbing phone call. The former modelin harassed or assaulted by photographers — likely, she said, because they were terrified than done to walk out of a shoot, especially if you're young, maybe English isn't your fi
ARTS & CULTURE	Actor Jeff Hiller Talks "Bright Colors And Bold Patterns" and More (AUDIO)	This week I talked with actor Jeff Hiller about the hit Off Broadway play Bright Colors A
ARTS & CULTURE	New Yorker Cover Puts Trump 'In The Hole' After 'Racist' Comment	The New Yorker is taking on President Donald Trump after he asked why the U.S. woul

데이터 출처: Kaggle의 News Article Category Dataset

데이터 크기: 6877행

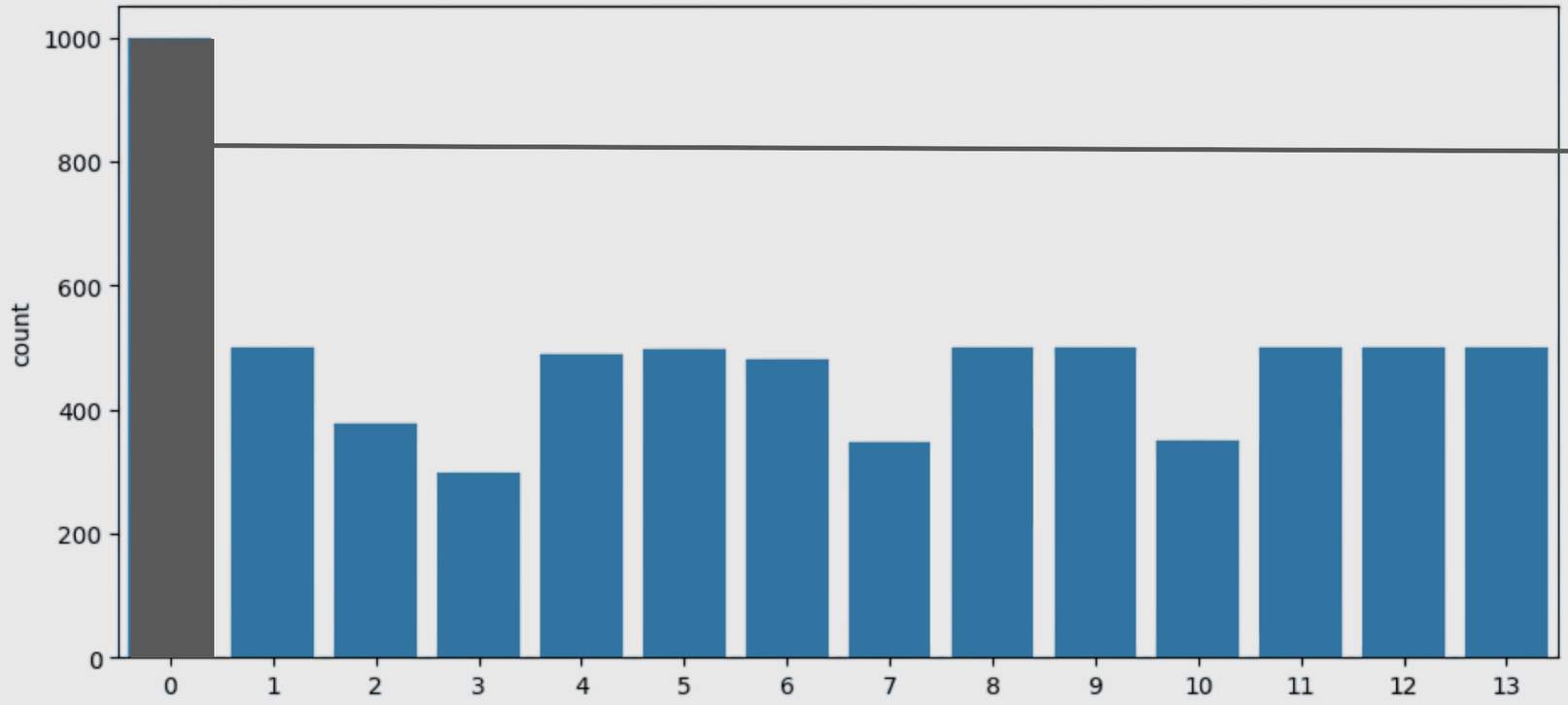
언어: English

컬럼: 'category', 'title', 'body'

카테고리 수: 14개(ARTS & CULTURE, BUSINESS, COMETY etc.)

3. 카테고리 분류 모델

데이터 분포



불균형 데이터

ARTS & CULTURE 의
데이터 개수가 월등히 많음

3. 카테고리 분류 모델

데이터 전처리

기본 전처리

1) 결측치, 중복 제거

2) 'title' 컬럼과 'body' 컬럼을 합쳐서 'combined' 컬럼으로 만듦

3) 텍스트 정제: 대문자→소문자 / 숫자와 특수기호 제거

이후 전처리

1) nltk를 이용한 불용어 제거

2) X_data 생성: 토큰화, 정수 인코딩 및 시퀀스 패딩

3) y_data 생성: 라벨 인코딩

4) GloVe 모델을 이용한 Embedding Matrix 만들기

3. 카테고리 분류 모델

1) NLTK를 이용한 불용어 제거

불용어(Stopword)

문장에 자주 등장하지만 의미 분석에는 필요 없는 단어

ex) the, and, is, at, a 등

제거함으로써 분석 정확도 향상+처리 속도 개선 필요

NLTK 라이브러리

영어 불용어 리스트 제공

불용어 제거 결과

The news is about the new technology in AI

news new technology AI

2) X_data 생성: 토큰화, 정수 인코딩 및 시퀀스 패딩

토큰화(Tokenization)

문장 토큰화: 텍스트를 문장 단위로 나누는 작업

단어 토큰화: 문장을 단어 단위로 나누는 작업

Tokenizer() from tensorflow.keras.preprocessing.text

토큰화 결과

news new technology AI

↓

[news, new, technology, AI]

정수 인코딩(Integer Encoding)

각 단어를 고유한 정수로 처리하여 숫자 시퀀스로 된 데이터로 만듦

단어의 빈도수가 높은 순서대로 1, 2, 3, 4... 정수가 매핑됨

Tokenizer().fit_on_text() + text_to_sequence()

정수 인코딩 결과

[news, new_technology, AI]

↓

[1, 2, 3, 4]

3. 카테고리 분류 모델

2) X_data 생성: 토큰화, 정수 인코딩 및 시퀀스 패딩

시퀀스 패딩(Sequence Padding)

문장의 길이를 전부 동일하게 맞춤

maxlen을 넘는 문장: 자름

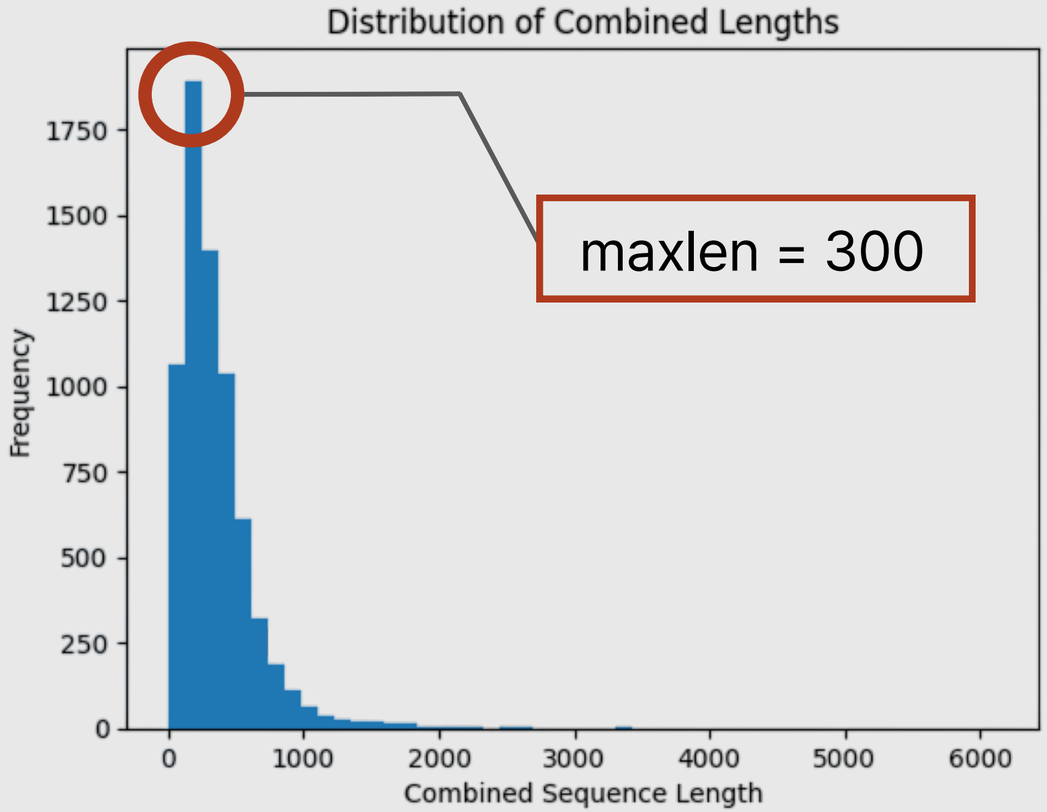
maxlen을 넘지 않는 문장: 0으로 채움(Padding)

pad_sequences()

from tensorflow.keras.preprocessing.sequences

시퀀스 패딩 결과(max_len=6)

[1, 2, 3, 4] ➡ [0, 0, 1, 2, 3, 4]



3. 카테고리 분류 모델

3) y_data 생성: 라벨 인코딩

라벨 인코딩

모델이 이해할 수 있도록 Label(category)을 정수로 변환

LabelEncoder() from sklearn.preprocessing

라벨 인코딩 매핑 목록

정수	Category
0	ARTS & CULTURE
1	BUSINESS
2	COMEDY
3	CRIME
4	EDUCATION
...	...
13	WOMEN

4) GloVe 모델 이용한 Embedding Matrix 만들기

워드 임베딩(Word Embedding)

단어를 고차원 공간에서의 숫자 벡터로 바꾸는 방법

단어 간의 의미적 유사성을 벡터 거리로 표현할 수 있게 해줌

GloVe 모델

단어 간 동시 등장 통계를 기반으로 학습된 사전 학습 워드 임베딩 모델

유사한 의미의 단어는 비슷한 벡터로 표현됨

100차원의 벡터로 표현

워드 임베딩 결과

happy

[-0.0904, 0.1964, 0.2947, ..., 0.2020]

3. 카테고리 분류 모델

모델링

LSTM

긴 시퀀스 데이터 패턴을 효과적으로 학습할 수 있는 구조

4개의 Layer를 바탕으로 매 시점마다 중요한 정보는 기억하고,
중요하지 않은 정보는 지움

LSTM의 핵심 구조

1) Cell State: 중요한 정보를 전달하는 핵심 구조

2) 입력 게이트: 새로운 정보 수용 여부 결정

3) 망각 게이트: 어떤 정보를 잊을지 결정

4) 출력 게이트: 다음 단계로 보낼 정보를 결정

3. 카테고리 분류 모델

모델링

1) train, validation, test로 나누기

train: validation: test = 6 : 2 : 2

레이블의 분포는 동일하게 유지(stratify 옵션 사용)

2) y_data 를 One-Hot Encoding

단어	One-Hot Vector
apple	[1, 0, 0]
banana	[0, 1, 0]
peach	[0, 0, 1]

3. 카테고리 분류 모델

모델링

모델 구성

```
model_cate = Sequential()
model_cate.add(Embedding(vocab_size, embedding_dim, weights=[embedding_matrix], input_length=max_length, trainable=Fals
e))
model_cate.add(Bidirectional(LSTM(hidden_units, dropout=0.3, recurrent_dropout=0.3)))
model_cate.add(Dense(embedding_dim, activation='relu'))
model_cate.add(Dense(num_classes, activation='softmax'))
```

1) Sequential() 모델 구성: 레이어를 순차적으로 쌓는 방식

2) Embedding 레이어: GloVe를 이용한 단어 임베딩

3) Bidirectional LSTM 레이어: 양방향 LSTM을 이용해 문맥 추출

4) 'ReLu' Dense 레이어: 중간 은닉층으로 정보 변환

5) 'SoftMax' Dense 레이어: 다중 클래스 분류를 위한 출력층

3. 카테고리 분류 모델

모델링

콜백 함수 정의

```
es = EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=4)
mc = ModelCheckpoint('/content/drive/MyDrive/Colab Notebooks/best_category_model.h5', moni
tor='val_acc', mode='max', verbose=1, save_best_only=True)
```

1) EarlyStopping: val_loss 기준으로 학습 조기 종료

2) ModelCheckpoint: val_acc가 최대가 될 때의 모델 저장

3. 카테고리 분류 모델

모델링

모델 컴파일 및 학습

```
model_cate.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['acc'])
history_cate = model_cate.fit(X_train, y_train, batch_size=128, epochs=30, callbacks=[es,
mc], validation_data=(X_val, y_val))
```

1) compile(): 손실 함수(loss function), 옵티마이저(optimizer), 성능 평가지표 설정

2) fit(): 모델 학습 및 검증 데이터 적용

4. 감정 분석 모델

04

4. 감정 분석 모델

데이터

id	headline	dominant_emotion
7d3fe468	Cops in One Village Have Been Convicted of 70 Crimes. Here's What They Had to Say About It.	anger
86693d59	DIY penis enlargements are a 'nationwide problem' in Papua New Guinea	negative_surprise
0fb40e90	Dam breaking: New Epstein accuser comes forward	anger
fa7750d6	David Beckham gets six-month driving ban for using phone at wheel	negative_surprise

데이터 출처: GoodNewsEveryone 뉴스감성 데이터(University of Stuttgart)

데이터 크기: 5000개

언어: English

컬럼: 'id', 'headline', 'dominant emotion' 등 15개 컬럼

Emotion 종류: anger, negative_surprise, sadness, disgust 등

4. 감정 분석 모델

데이터 전처리

1) 텍스트 정제(대소문자 통일, 특수문자 제거)

2) 토큰화

NLTK를 이용하여 토큰화 시도

⬇

NLTK의 리소스 파일 중 일부 다운로드 실패
(LookupError 발생)

⬇

NLTK 대신 spaCy 사용

	NLTK	spaCy
주요 특징	다양한 NLP 기능과 코퍼스 제공, 세세한 조작 가능	빠르고 효율적인 처리, 산업 현장에 적합
속도	느린 편	빠름(Cython 기반)
토크나이저	rule-based (punkt, RegexpTokenizer 등)	자체 엔진 사용 (정확도 높음)
한국어 지원	매우 제한적	거의 없음(영어 중심)

4. 감정 분석 모델

데이터 전처리

3) 불용어 제거

4) 어간추출/표제어 추출

5) 텍스트 데이터 정수 인코딩

전처리 결과

Cops in One Village Have Been Convicted

↓

['cop', 'one', 'village', 'convicted']

↓

[1, 2, 3, 4]

4. 감정 분석 모델

BERT

Google이 제안한 사전학습된 언어 표현 모델

- 문맥을 양방향으로 이해
  - 기존의 RNN, LSTM 등은 문장을 한 방향으로만 처리했지만, BERT는 좌우 문맥을 동시에 반영해 문장의 의미를 더 정확하게 파악 가능
- 주요 특징

특징	설명
텍스트	문장의 앞뒤 문맥을 모두 사용함(예: '나는 [MASK]을 좋아해요'에서 [MASK]의 앞뒤 단어를 다 반영)
Transformer 기반	Self-Attention 메커니즘을 사용하여 문장 내 단어들 간 관계를 파악
Pre-trained & Fine-tuning	미리 방대한 데이터로 학습된 모델을 특정 작업(감정분석, QA)에 맞게 미세 조정 가능

4. 감정 분석 모델

BERT

■ 벡터화 과정에서 BERT 모델이 하는 일

단계	역할
1. Tokenization	입력 문장을 WordPiece 단위로 분해하여 토큰화([CLS]나는, 좋아, ##해요)
2. Embedding	각 토큰을 고정된 차원의 벡터로 변환 (ex.768차원)
3. Contextual Encoding	Transformer를 통해 각 단어의 의미를 문맥 기반으로 조정한 벡터로 인코딩
4. Output	각 단어의 벡터 또는 [CLS]

4. 감정 분석 모델

데이터 전처리 및 토크나이저 준비 Code

```
# 5. 데이터 로드 및 전처리
def load_and_preprocess(test_path, train_path):
    test_df = pd.read_csv(test_path)
    train_df = pd.read_csv(train_path)
    # tokenized_headline이 문자열 리스트라면 실제 리스트로 변환 후 공백으로 합침
    train_df['text'] = train_df['tokenized_headline'].apply(lambda x: ' '.join(eval(x)) if isinstance(x, str) else '')
    test_df['text'] = test_df['tokenized_headline'].apply(lambda x: ' '.join(eval(x)) if isinstance(x, str) else '')
    # 레이블 매핑
    train_df['label'] = train_df['emotion_binary'].map({'positive':1, 'negative':0})
    test_df['label'] = test_df['emotion_binary'].map({'positive':1, 'negative':0})
    return train_df, test_df

train_data, test_data = load_and_preprocess(TEST_PATH, TRAIN_PATH)

# 6. 토크나이저 준비 (영어 BERT 사용, bert-base-uncased 권장)
MODEL_NAME = 'bert-base-uncased'
tokenizer_sent = BertTokenizer.from_pretrained(MODEL_NAME)
MAX_LEN = 64
```

4. 감정 분석 모델

벡터화 함수 Code

```python
# 7. 벡터화 함수
def bert_vectorize(texts, labels, tokenizer_sent, max_len=64):
    input_ids, attention_masks, token_type_ids = [], [], []
    for text in texts:
        encoded = tokenizer_sent.encode_plus(
            text,
            add_special_tokens=True,
            max_length=max_len,
            padding='max_length',
            truncation=True,
            return_attention_mask=True,
            return_token_type_ids=True
        )
        input_ids.append(encoded['input_ids'])
        attention_masks.append(encoded['attention_mask'])
        token_type_ids.append(encoded['token_type_ids'])
    return {
        'input_ids': np.array(input_ids),
        'attention_mask': np.array(attention_masks),
        'token_type_ids': np.array(token_type_ids)
    }, np.array(labels)

train_features, train_labels = bert_vectorize(train_data['text'], train_data['label'], tokenizer_sent, MAX_LEN)
test_features, test_labels = bert_vectorize(test_data['text'], test_data['label'], tokenizer_sent, MAX_LEN)
```

4. 감정 분석 모델

감정분류

다양한 감정 레이블을 긍정/부정으로 구분

원본 데이터 파일의 감정 관련된 컬럼들

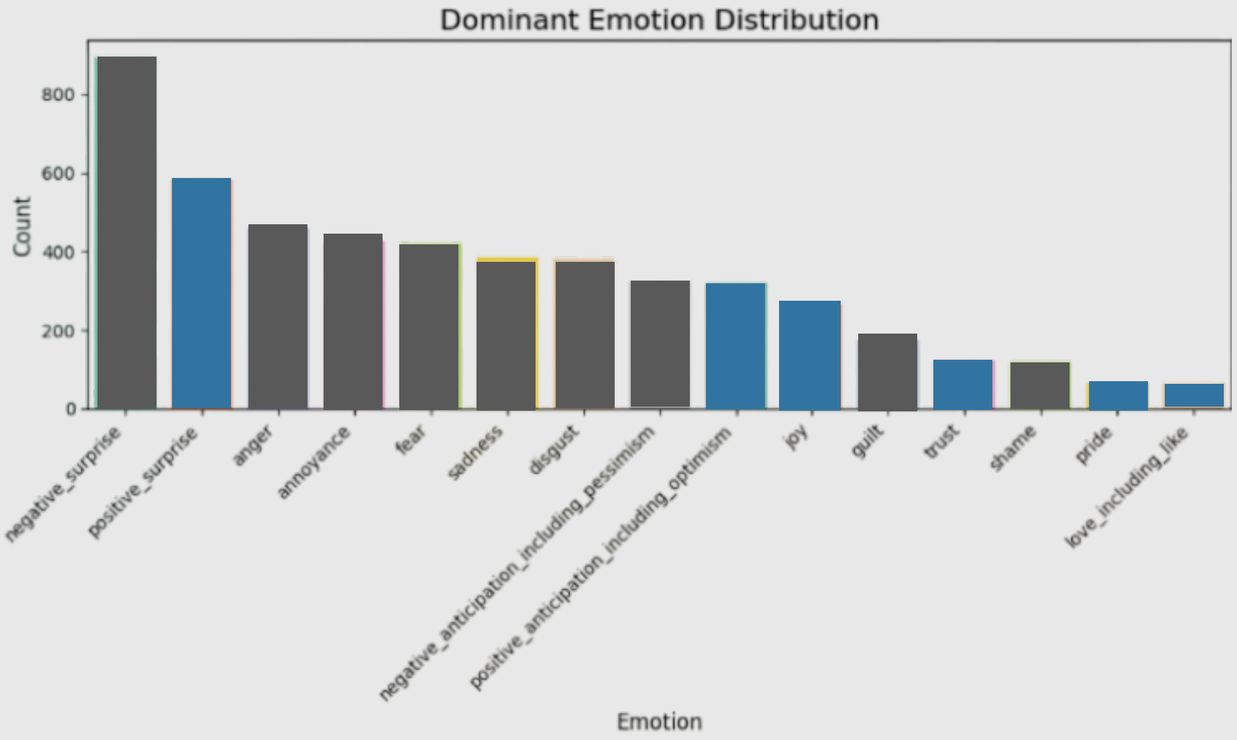
컬럼명	의미
dominant_emotion	뉴스 기사에서 **가장 뚜렷하게 드러난 감정(주감정)** (anger, disgust, sadness, negative_surprise 등)
intensity	감정의 **강도(intensity level)** (low, medium, high 중 하나)
other_emotions	해당 뉴스 기사에서 **주 감정 외에 추가로 표현된 감정**
reader_emotions	뉴스 기사를 읽은 독자가 **느꼈다고 응답한 감정**

dominant_emotion 컬럼으로 감정 분류 진행

4. 감정 분석 모델

감정분류

dominant_emotion 컬럼 데이터 분포



긍정 감정

positive_surprise,
positive_anticipation_including_optimism,
joy, trust, pride, love_including_like

부정 감정

negative_surprise, anger, annoyance, fear,
sadness, disgust, guilt, shame,
negative_anticipation_including_pessimism,

4. 감정 분석 모델

감정분류

감정분류 결과

- 긍정 감정 개수: 1417개
- 부정 감정 개수: 3583개

약 2.5배

클래스 불균형 문제 해결 방법

방법	설명
언더샘플링(Under-sampling)	많은 쪽(예: 부정) 데이터를 일부만 사용해서 균형 맞춤
오버샘플링(Over-sampling)	적은 쪽(예: 긍정) 데이터를 복제해서 수를 늘림
SMOTE	적은 클래스의 데이터를 기반으로 유사한 가짜 샘플 생성

긍정:부정 = 1:1

4. 감정 분석 모델

모델 수립 및 학습

최종 데이터셋

- 3개의 컬럼으로 구성
- 뉴스기사제목 원문, 벡터화된 뉴스기사 제목, 긍정/부정 감정 분류

1) train : test = 8 : 2 로 나누기

2) BERT + 이진분류기를 사용하여 모델을 수립한 후 train data로 학습

4. 감정 분석 모델

BERT 모델 구성 Code

```python
# 8. BERT 모델 구성
def build_bert_model(model_name, num_labels=2):
    bert = TFBertModel.from_pretrained(model_name)
    input_ids = tf.keras.Input(shape=(MAX_LEN,), dtype=tf.int32, name='input_ids')
    attention_mask = tf.keras.Input(shape=(MAX_LEN,), dtype=tf.int32, name='attention_mask')
    token_type_ids = tf.keras.Input(shape=(MAX_LEN,), dtype=tf.int32, name='token_type_ids')
    bert_outputs = bert({
        'input_ids': input_ids,
        'attention_mask': attention_mask,
        'token_type_ids': token_type_ids
    })
    pooled_output = bert_outputs.pooler_output
    x = tf.keras.layers.Dropout(0.3)(pooled_output)
    x = tf.keras.layers.Dense(64, activation='relu')(x)
    x = tf.keras.layers.Dropout(0.2)(x)
    outputs = tf.keras.layers.Dense(num_labels, activation='softmax')(x)
    model_sent = tf.keras.Model(
        inputs={'input_ids': input_ids,
                'attention_mask': attention_mask,
                'token_type_ids': token_type_ids},
        outputs=outputs
    )
    optimizer = Adam(learning_rate=3e-5)
    model_sent.compile(
        optimizer=optimizer,
        loss='sparse_categorical_crossentropy',
        metrics=['accuracy']
    )
    return model_sent
```

4. 감정 분석 모델

모델 학습 Code

```
# 9. 콜백 및 학습
checkpoint = ModelCheckpoint(
    '/content/drive/MyDrive/Colab Notebooks/best_bert_model.h5',
    monitor='val_accuracy',
    save_best_only=True,
    verbose=1
)
early_stopping = EarlyStopping(
    monitor='val_loss',
    patience=3,
    restore_best_weights=True
)

history_sent = model_sent.fit(
    {
        'input_ids': train_features['input_ids'],
        'attention_mask': train_features['attention_mask'],
        'token_type_ids': train_features['token_type_ids']
    },
    train_labels,
    epochs=10,
    batch_size=32,
    validation_split=0.2,
    callbacks=[checkpoint, early_stopping]
)
```

4. 감정 분석 모델

테스트 Code

```
# 11. 테스트 및 리포트
results = model_sent.evaluate(
    {
        'input_ids': test_features['input_ids'],
        'attention_mask': test_features['attention_mask'],
        'token_type_ids': test_features['token_type_ids']
    },
    test_labels
)
print(f"테스트 정확도: {results[1]:.4f}")

y_pred = model_sent.predict(
    {
        'input_ids': test_features['input_ids'],
        'attention_mask': test_features['attention_mask'],
        'token_type_ids': test_features['token_type_ids']
    }
).argmax(axis=1)
print("\n분류 리포트:")
print(classification_report(test_labels, y_pred, target_names=['negative', 'positive']))
```

정확도 87%으로 우수한 편

5. 결과 및 결과 분석

05

5. 결과 및 결과 분석

뉴스 기사 자동 분석 시스템

뉴스 기사 제목과 본문 입력

↓

데이터 전처리

↓

감정 분석 / 카테고리 분류 모델 이용하여 예측

↓

결과 출력

실제 출력 결과

```
뉴스 제목을 입력하세요: Dortmund confident of signing Jobe Bellingham
뉴스 본문을 입력하세요: Borussia Dortmund are confident of completing the
1/1 [==============================] - 0s 41ms/step
Text: Football and other premium TV being pirated at 'industrial scale' A
Prediction: negative (Confidence: 0.9954)
--------------------------------------------------
1/1 [==============================] - 0s 150ms/step

카테고리 예측 (상위 3개):
- SPORTS: 95.56%
- ENTERTAINMENT: 0.98%
- WOMEN: 0.81%
```

6. 한계점 및 개선방안

06

6. 한계점 및 개선방안

한계점 및 개선방안

문제점	개선방안
카테고리와 감정 레이블을 동시에 가지고 있는 뉴스 기사 데이터가 없었음	카테고리와 감정 레이블을 동시에 가진 데이터를 이용하여 훈련을 진행했다면 성능이 더 개선될 것
카테고리 별 데이터의 개수가 동일하지 않아 데이터의 불균형 존재	불균형을 해소한 오버샘플링, 언더샘플링 등을 사용
딥러닝 모델은 그 결과를 낸 이유를 잘 설명하지 못함	SHAP와 같은 XAI를 이용하여 설명력을 높일 수 있음

활용 방안

- 하루에 발행되는 뉴스 기사를 분류하여 사용자 맞춤 뉴스 추천 시스템을 만들 때 활용할 수 있음
- 뉴스기사의 입장을 분류하는 시스템을 만드는 기반이 될 수 있음

발표를 들어주셔서 감사합니다.

Q & A

A조 김서연 정다솜 박민서 이준서