

텍스트 마이닝과 데이터 마이닝

Part 05. 토픽 모델링과 워드 클라우드

정 정 민

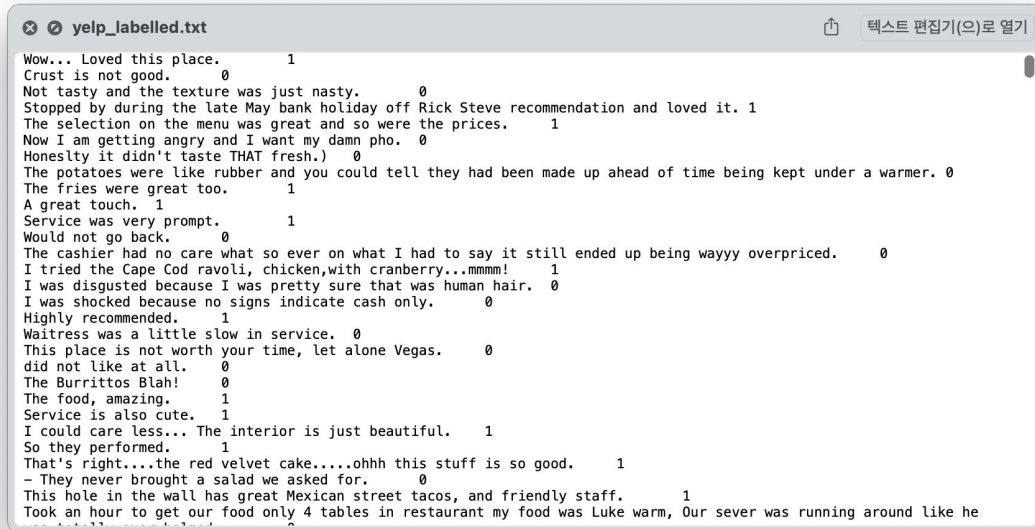
Chapter 13. 토픽 모델링 실습

1. 전처리
2. LDA 실행
3. 결과 분석

전처리

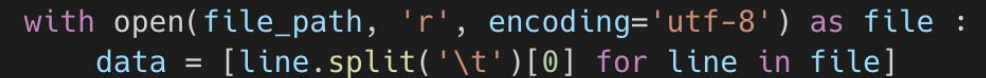
사용 데이터

- 이번 실습에서 사용할 데이터는 Part4의 Yelp 데이터와 동일
 - Kaggle 다운로드 : [링크](#)
- 단, 감성 분석이 목표가 아니므로 감정 상태를 나타내는 0과 1은 사용하지 않음
 - 본문과 tab('t')으로 구분되어 있음



A screenshot of a text editor window titled 'yelp_labelled.txt'. The window displays a list of Yelp reviews, each followed by a sentiment label (0 or 1). The reviews are separated by tab characters. The labels represent sentiment: 0 for negative and 1 for positive. The text is as follows:

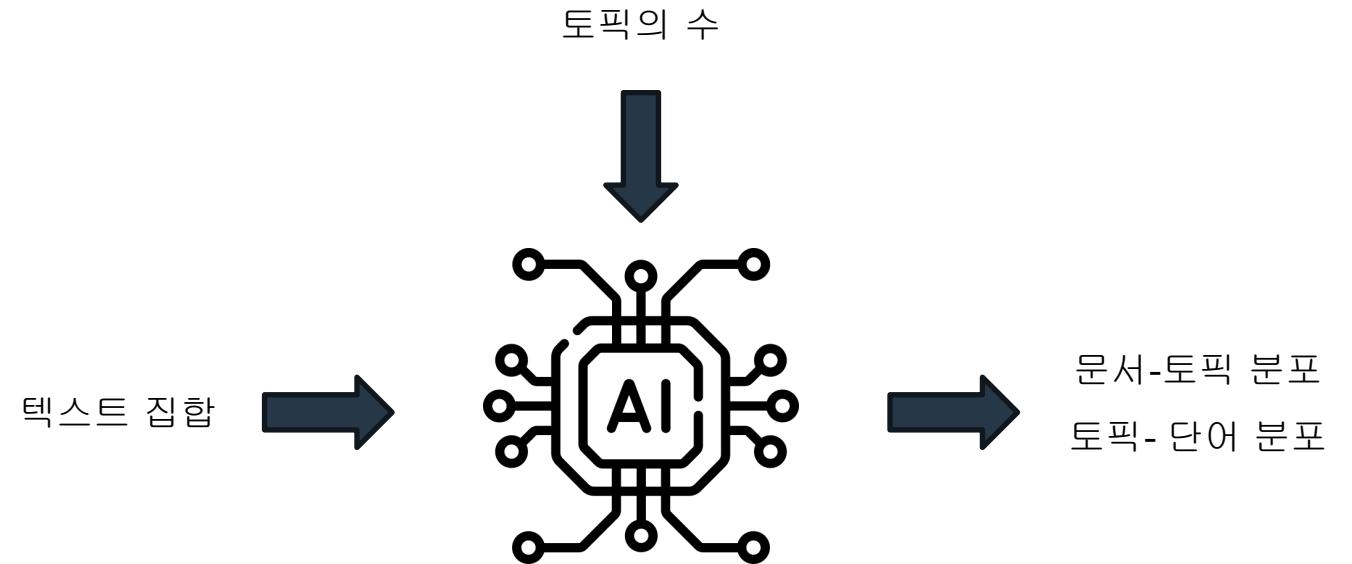
```
Wow... Loved this place. 1
Crust is not good. 0
Not tasty and the texture was just nasty. 0
Stopped by during the late May bank holiday off Rick Steve recommendation and loved it. 1
The selection on the menu was great and so were the prices. 1
Now I am getting angry and I want my damn pho. 0
Honestlty it didn't taste THAT fresh.) 0
The potatoes were like rubber and you could tell they had been made up ahead of time being kept under a warmer. 0
The fries were great too. 1
A great touch. 1
Service was very prompt. 1
Would not go back. 0
The cashier had no care what so ever on what I had to say it still ended up being wayyy overpriced. 0
I tried the Cape Cod ravioli, chicken,with cranberry...mmmm! 1
I was disgusted because I was pretty sure that was human hair. 0
I was shocked because no signs indicate cash only. 0
Highly recommended. 1
Waitress was a little slow in service. 0
This place is not worth your time, let alone Vegas. 0
did not like at all. 0
The Burrittos Blah! 0
The food, amazing. 1
Service is also cute. 1
I could care less... The interior is just beautiful. 1
So they performed. 1
That's right....the red velvet cake....ohhh this stuff is so good. 1
- They never brought a salad we asked for. 0
This hole in the wall has great Mexican street tacos, and friendly staff. 1
Took an hour to get our food only 4 tables in restaurant my food was Luke warm, Our sever was running around like he
```



```
with open(file_path, 'r', encoding='utf-8') as file :
    data = [line.split('\t')[0] for line in file]
```

문제 정의

- 풀어야 하는 문제
 - Yelp 데이터셋에 잠재하는 주제를 찾아내기
- 입력과 출력
 - 입력 :
 - 전체 텍스트 문서 집합
 - 토픽의 수
 - 출력 :
 - 문서 별 토픽 분포
 - 토픽 별 단어 분포



전처리 함수

- LDA의 결과 분석에 단어 해석이 사용되므로
- 단어의 레벨로 Token을 설정!
- 기본적인 텍스트 데이터 전처리 진행
 - Tokenize
 - 띄어쓰기 단위로
 - Stop Words 제거
 - Stemming
 - 많이 사용하는 PorterStemmer 사용
 - 기타
 - 소문자화 (정규화), 비 단어적 요소 제거

```
def preprocessing(text) :  
    text = text.lower()  
    text = re.sub(r'\W', ' ', text)  
    text = text.split()  
    text = [t for t in text if t not in stop_words]  
    text = [stemmer.stem(word) for word in text]  
    return text
```

```
for d in data[:3] :  
    print(f'원래 문장 : {d}')  
    print(f'전처리 후 문장 : {preprocessing(d)}')
```

원래 문장 : Wow... Loved this place.
전처리 후 문장 : ['wow', 'love', 'place']

원래 문장 : Crust is not good.
전처리 후 문장 : ['crust', 'good']

원래 문장 : Not tasty and the texture was just nasty.
전처리 후 문장 : ['tasti', 'textur', 'nasti']

LDA 실행

문서-단어 행렬 (Document-Term Matrix, DTM)

- 전처리된 데이터에서 **각 단어의 빈도를 나타내는 행렬**을 생성
- 행의 방향으로 문서를 나타내며, 열의 방향으로 단어를 나타냄
- 따라서 문서-단어 행렬 (Document-Term Matrix, DTM)이라고 함
- 먼저 어떤 열 번호(index)에 어떤 단어(term)가 들어갈지 정해야 함
 - Gensim 패키지의 Dictionary Class 이용
- 만들어진 단어(term)-번호(index) 객체를 활용 DTM 생성
 - (단어 ID, 빈도수)의 형태를 가짐

```
from gensim import corpora

# Gensim의 Dictionary 객체를 생성
dictionary = corpora.Dictionary(preproc_data)
# {'love': 0, 'place': 1, 'wow': 2, 'crust': 3, ...}

# 문서-단어 행렬을 생성
corpus = [dictionary.doc2bow(text) for text in preproc_data]
# [
#   [(0, 1), (1, 1), (2, 1)], : 1번째 문장
#   [(3, 1), (4, 1)], : 2번째 문장
#   ...
# ]
```

LDA 모델 생성 및 학습

- Gensim에서 제공하는 LDA 모델을 사용
- 아래의 입력을 제공해 학습 진행
 - 전체 텍스트 집합 (corpus)
 - 선정할 토픽의 수
 - 단어와 인덱스 데이터 (dictionary)
 - 학습 횟수

```
from gensim.models import LdaModel

topicK = 3
num_trains = 10

lda_model = LdaModel(corpus,
                      num_topics=topicK,
                      id2word=dictionary,
                      passes=num_trains)
```

LDA 모델 추론

- 학습된 모델을 활용해 아래의 정보를 확인할 수 있음
- **문서 별 토픽 분포**
 - LDA_model[문서]의 형태로 호출하면, 해당 문서의 토픽 분포를 얻음
 - 이 결과는 (토픽 번호, 토픽에 속할 확률)의 형태로 반환
- **토픽 별 단어 분포**
 - LDA_model.show_topic(토픽_번호)의 형태로 호출
 - List[(단어, 단어의 확률]으로 반환
 - 상위 단어는 topn의 변수로 설정 가능 (기본 : 10)

```
# 문서 별 토픽 분포 확인
for document in corpus[:3]:
    print(lda_model[document])

# [(0, 0.08787388), (1, 0.10013846), (2, 0.81198764)]
# [(0, 0.75954294), (1, 0.12738658), (2, 0.113070466)]
# [(0, 0.11008714), (1, 0.8055659), (2, 0.08434695)]

# 토픽 별 단어 분포 확인
for k in range(topicK):
    print(lda_model.show_topic(k, topn=3))

# [('servic', 0.017086321), ('good', 0.014856899),
#  ('disappoint', 0.010963259)]
# [('food', 0.042987805), ('back', 0.02138683),
#  ('good', 0.019900084)]
# [('place', 0.021426972), ('great', 0.017665138),
#  ('restaur', 0.007841083)]
```

결과 분석

토픽 내용 설정

- LDA의 결과로 생성된 토픽의 의미는 사용자가 선정해야 함
 - 토픽을 구성하는 단어를 보고 토픽의 의미를 선정
- 이 과정은 데이터 친숙도, 분야의 전문성이 큰 힘을 발휘
- **0번째 토픽**
 - 주요 단어 : 'service', 'good', 'disappoint', 'like', 'best', 'great', 'also', 'really', 'place', 'staff'
 - 식당 서비스 품질과 고객 경험
- **1번째 토픽**
 - 주요 단어 : 'food', 'back', 'good', 'service', 'place', 'go', 'time', 'wait', 'would', 'ever'
 - 음식 품질과 재방문에 관련된 의사 표현
- **2번째 토픽**
 - 주요 단어 : 'place', 'great', 'restaurant', 'like', 'salad', 'star', 'delicious', 'time', 'get', 'pretty '
 - 식당 전반의 분위기와 음식의 퀄리티에 대한 내용

원 문장에 대한 토픽 분석 비교 확인

- 분석한 토픽의 의미를 실제 문장 결과로 비교
- “Service was very prompt”
 - 모델의 추론 결과 0번 토픽의 비율이 가장 큼
 - 식당의 서비스와 고객 경험을 이야기하는 0번 토픽과 제일 가까움
 - 실제 문장 의미와 분석 결과가 일맥상통함

```
print('원 문장 : ', data[target_idx])
print('전처리 문장 : ', preproc_data[target_idx])
print('문장 내 토픽 분포 : ')
for topic_idx, prob in lda_model.corpus_data[target_idx]:
    print(f' - {topic_idx}번 토픽 : {prob*100:.2f}%')

# 원 문장 : Service was very prompt.
# 전처리 문장 : ['servic', 'prompt']
# 문장 내 토픽 분포 :
#   - 0번 토픽 : 76.57%
#   - 1번 토픽 : 12.26%
#   - 2번 토픽 : 11.16%
```

E.O.D