리뷰데이터 감성분석

제로베이스 9기 머신러닝 2조
강사라 김인희
김현승 황산하

CONTENTS

01

프로젝트 선정 배경

프로젝트 선정 배경

핵심 가치

비즈니스 관점

리뷰 데이터는 기업의 별도 노력 없이
객관적인 고객의 니즈를 파악할 수 있고,
기업의 서비스 및 상품의
지속적 개선에 사용 가능

프로젝트 팀

텍스트 마이닝, 자연어 처리가
다양한 서비스 산업에서 활용되고 있어
범용성, 확장가능성 우수

기술적 관점

일괄적 기준을 제공함으로써
상품간 비교가능성을 제고,
다양한 쇼핑몰에 포진되어 있는 상품을
하나의 플랫폼에서 확인할 수 있는
장점을 더욱 활용

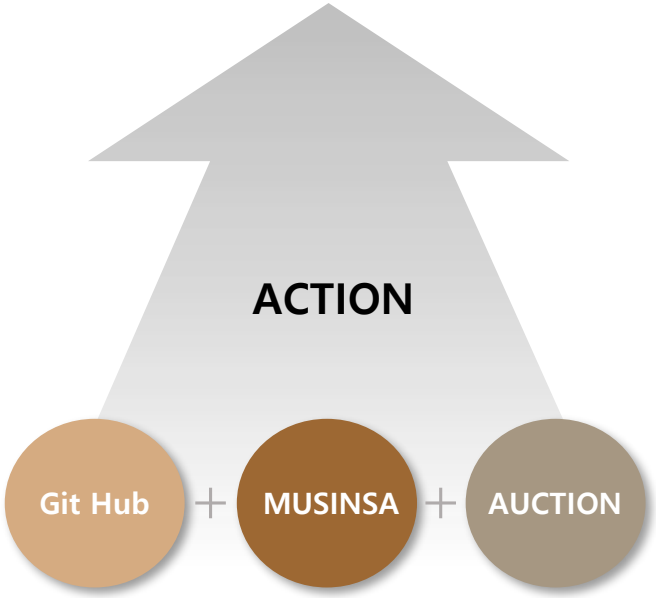
향후 전망

비즈니스	• 리뷰 수가 많은 상위 고객 리뷰 분석으로 시장 세분화에 대한 의사 결정 지표로 사용 가능 • 긍정적 및 부정적 핵심 키워드 그룹을 통해 소셜네트워크 분석 등의 기초 데이터로 활용
기술	고객 쇼핑의 편의성을 향상시켜 매출 증대에 대한 효과를 기대할 수 있음
프로젝트 팀	다양한 분석으로 확장할 수 있는 분석의 토대를 마련하는 것을 지향

데이터 수집

VISION

감성분석 모델링을 통해 별 점이 없는 리뷰도 판별



GitHub

https://github.com/bab2min/corpus/tree/master/sentiment

네이버 쇼핑에서 제품별 별점과 후기를 수집한 데이터 1, 2, 4, 5점으로 구성

남성의류 데이터

스타일 후기(착장 사진 필수), 상품 후기(제품 사진 필수), 일반 후기(20자 이상 필수) 중에서 일반 후기 선택 - 20자 이상이 필수인 콘텐츠 질 우위

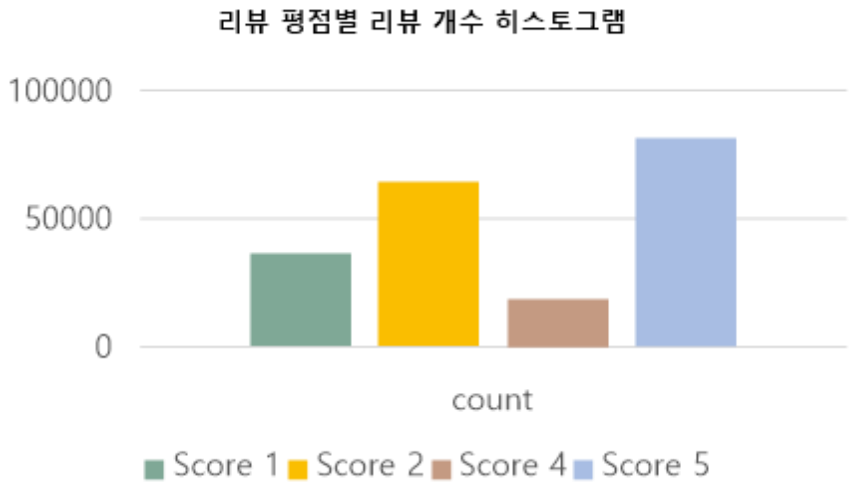
여성의류 데이터

한 사람이 여러 제품을 구매하는 경우 일부 중복되는 리뷰도 발생하나 각각의 상품에 대해 리뷰를 별도로 다는 경우도 있어 중복을 제거하지 않기로 판단

02

수집, EDA, 전처리

[긍정, 부정에 대하여
고른 분포를 띄고 있음]



	1	2	3	4	5
count	36048	63989	0	18786	81177
rate (%)	50.01			49.9	

평점별 데이터 분포

- 1점 18.0%, 2점 32.0%, 3점 전무, 4점 9.4%, 5점 40.6%로 구성되어 있음.
- 별점이 3점인 리뷰 데이터는 데이터를 공개한 Github에서 제공되지 않음

긍정, 부정 데이터 분포

- 1점, 2점을 부정, 4점, 5점을 긍정으로 분류
- 긍정과 부정이 1:1로 고른 분포를 띄고 있음

평점별 리뷰 TOP 10 키워드

배송, 구매, 사용, 생각 등은 모든 평점에 주요 키워드로 등장, 불용어 사전에 추가

1점

배송, 그냥, 별로, 제품, 사용, 구매, 주문, 반품, 생각, 냄새

2점

배송, 그냥, 생각, 별로, 제품, 사용, 구매, 가격, 부분, 주문

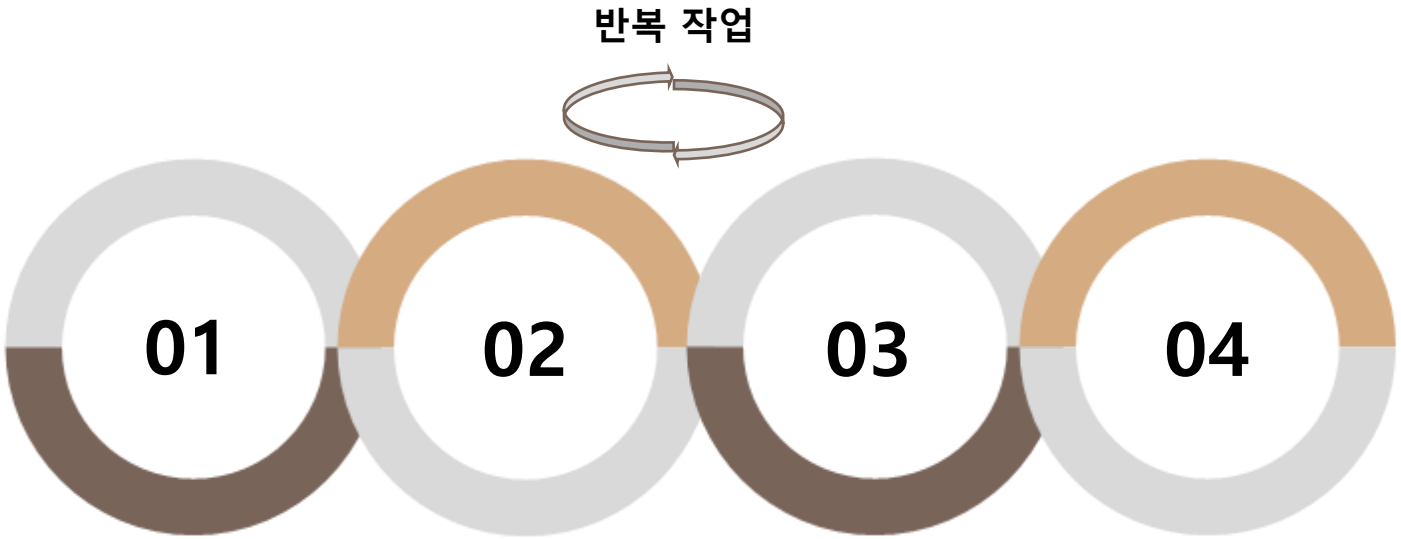
4점

배송, 사용, 구매, 가격, 생각, 제품, 조금, 주문, 사이즈, 포장

5점

배송, 구매, 사용, 가격, 제품, 주문, 아주, 생각, 포장, 사이즈

전처리 절차 및 EDA와 상호작용 메커니즘



리뷰 데이터 수집

워드클라우드 생성

빈출단어 추출

사전 작성

유의어 사전 생성
불용어 사전 갱신

데이터 셋팅

NetMiner

- GUI 기반의 네트워크 분석도구
- 시간과 노력을 단축시킬 수 있어 단기 프로젝트의 빠른 진행에 효과적임

+

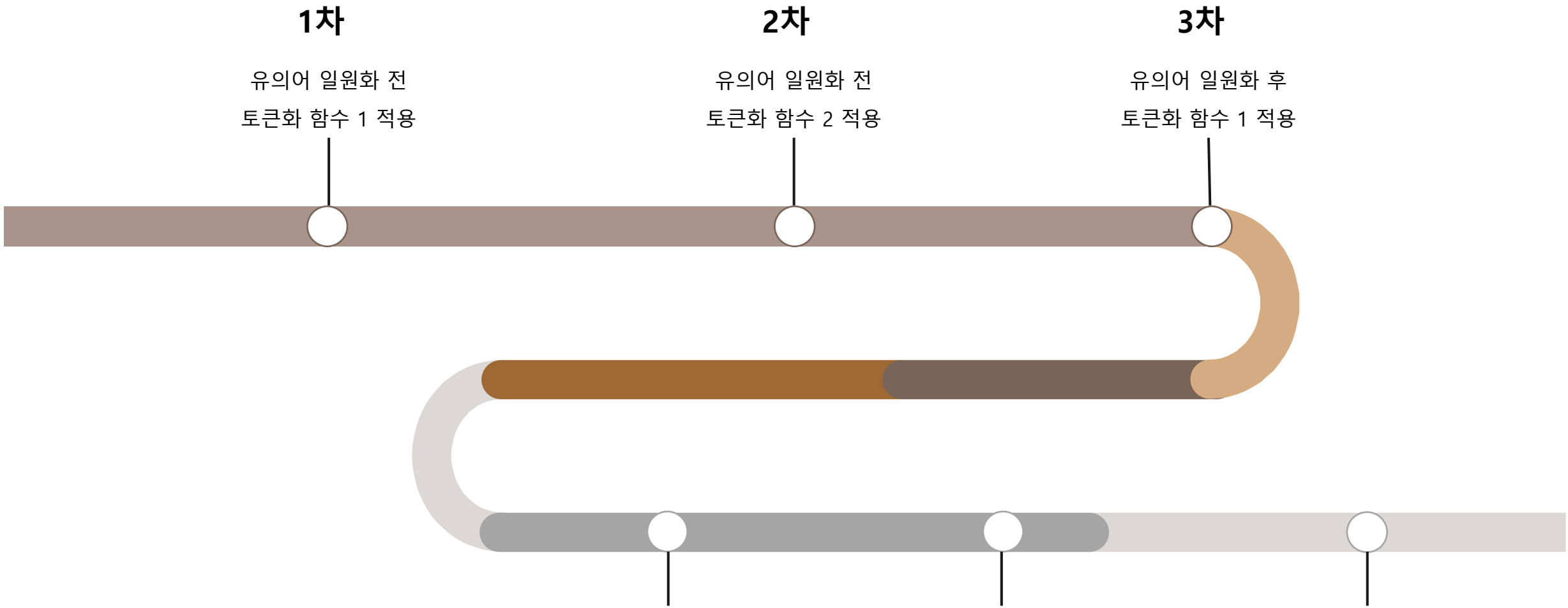
코드생성

- 특수문자 삭제
- 제외어 삭제 = 불용어 사전 + 해당 데이터
- 유사어 통일 = 유사한 의미의 토큰 일원화
- 지정어 추출 = 복합명사 그대로 추출

03

모델링 및 성능평가

모델링 및 성능평가


1차
유의어 일원화 전
토큰화 함수 1 적용
2차
유의어 일원화 전
토큰화 함수 2 적용
3차
유의어 일원화 후
토큰화 함수 1 적용
4차
유의어 일원화 후
토큰화 함수 2 적용
5차
유의어, 불용어 사전 보완
토큰화 함수 1, 2 적용
최종 제출안
유의어, 불용어 사전 보완
분석, EDA

모델링 및 성능평가

토큰화 함수 1

- 명사, 형용사, 동사만 추출
- 2글자 이상인 경우만 추출
- 불용어 제거

```
def text_preprocessed(text):
    hangul = re.compile('[^ ㄱ-ㅣ가-힣]')
    result = hangul.sub('', text)
    okt = Okt()
    Okt_morphs = okt.pos(result)
    words = []

    for word, pos in Okt_morphs:
        if pos == 'Adjective' or pos == 'Verb' or pos == 'Noun':
            if len(word) > 1 and word not in stopwords:
                words.append(word)

    words_str = ' '.join(words)
    return words_str
```

토큰화 함수 2

- 모든 형태소 활용
- 1글자도 활용
- 불용어 제거

```
okt2 = Okt()
def tokenizer_2(text):
    tokens_ko = okt2.morphs(text)

    result = []
    for word in tokens_ko:
        if word not in stopwords:
            result.append(word)
    return ' '.join(result)
```

모델링 및 성능평가

DTM(Document-Term Matrix)

01 BoW 기반 문서 정보를 행렬화

단어 출현의 빈도 수 기준으로 접근합니다.

보편적으로 비슷한 문서에는 비슷한 단어가 등장한다는 개념에 착안합니다

02 한계점

대용량 문서의 경우 리소스가 낭비될 수 있다.

불용어 성격의 단어(예를 들면 영어의 경우 'the')는 빈도수가 높다고 해도 유사도를 대변해 주지 않는다.

TF-IDF(Term Frequency-Inverse Document Frequency)

01 단어의 중요도에 따라 가중치 부여

단어가 등장한 횟수와 특정 단어가 등장한 문서의 수의 비율에 로그함수를 적용한 값을 기준으로 행렬화

02 한계점

맥락적 유사도를 반영하지는 못하는 지표

DTM

성능 최저

1차

accuracy: 0.79
Precision : 0.768
Recall : 0.828
F1 : 0.796

성능 우수

2차

accuracy: 0.92
Precision : 0.923
Recall : 0.918
F1 : 0.921

성능 개선 (1차 대비)

3차

accuracy: 0.90
Precision : 0.905
Recall : 0.892
F1 : 0.898

성능 유사 (2차 대비)

4차

accuracy: 0.91
Precision : 0.914
Recall : 0.910
F1 : 0.912

TFIDF

성능 최저

성능 우수

성능 개선 (1차 대비)

성능 유사 (2차 대비)

1차	2차	3차	4차
accuracy: 0.79 Precision : 0.777 Recall : 0.806 F1 : 0.791	accuracy: 0.92 Precision : 0.929 Recall : 0.916 F1 : 0.923	accuracy: 0.90 Precision : 0.907 Recall : 0.894 F1 : 0.900	accuracy: 0.90 Precision : 0.883 Recall : 0.931 F1 : 0.906

모델링 및 성능평가 - 5차 최종 선정 모델 선정 기준

토큰화

다양한 모델에서 전반적으로
성능이 우수하게 나타났던
토큰화함수2(모든 형태소 활용) 적용

불용어 배제

배송, 구매 등 평점과 무관하게
빈출되는 형태소는 배제

하이퍼파라미터 튜닝

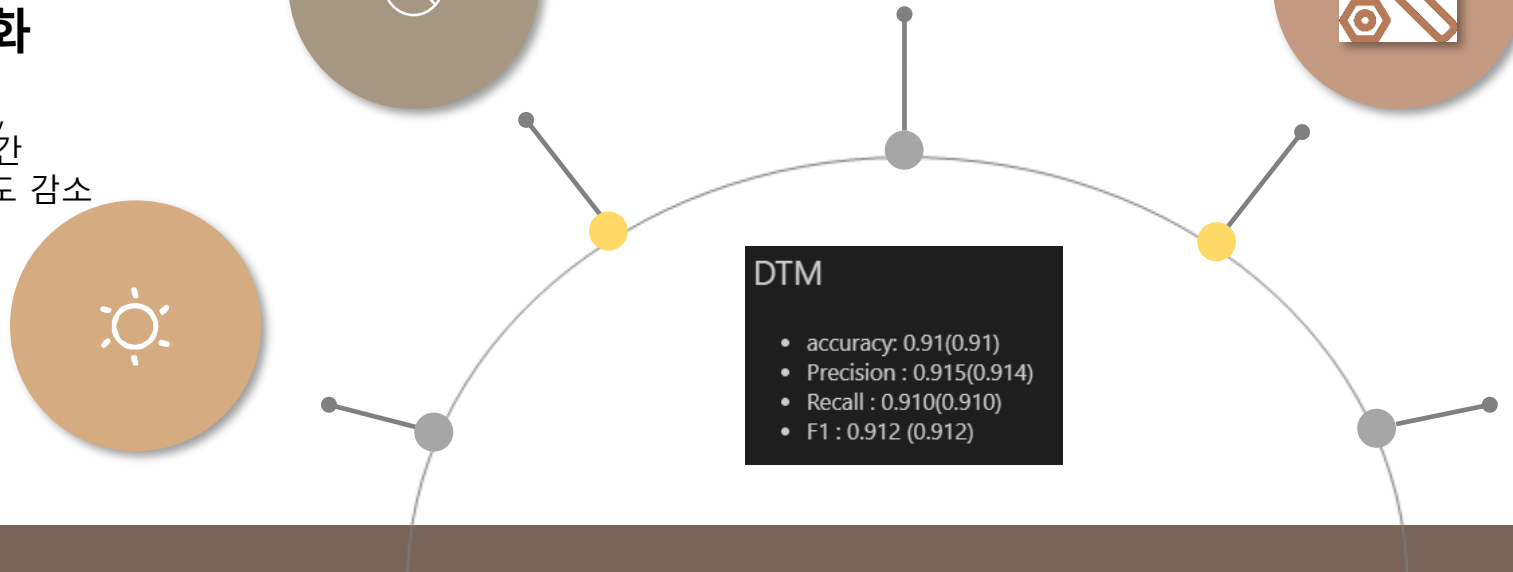
logistic regression의
solver를 liblinear로 지정하고
규제변수 C에 대하여
GridSearchCV

유의어 일원화

주요 긍정 키워드,
주요 부정 키워드 간
중복되는 단어 출현 빈도 감소

성능평가

Accuracy(정확도),
F1 score(정밀도 및 재현율)
우수



04

분석

토큰화함수 1보다 토큰화함수2 성능이 더 좋은 이유

접근 방법

- 1차 모델은 예측이 틀렸지만, 2차 모델의 예측은 맞은 경우에 대하여 데이터프레임 추출
- 추출된 데이터프레임에 대하여 토큰화함수1로 토큰화된 리뷰와 토큰화함수 2로 토큰화된 리뷰에 대하여 각각 워드클라우드를 형성

유의미한 한 글자 또는 부사 존재

- 잘, 굿, ㅠ, ^^, 싹과 같이 유의미하고 빈출되는 한 글자, 이모티콘 및 부사가 존재
- ngram_range = (1,2)로 설정되어 있기 때문에 잘 왔어요와 같은 주요 긍정키워드 top 10안에 드는 단어가 토큰화 함수 1에서는 누락

주요 긍정 키워드

	DTM_tk1	TFIDF_tk1	DTM_tk2	TFIDF_tk2
1	맛있어요	좋아요	맛있어요	좋아요
2	튼튼합니다	맛있어요	만족해요	맛있어요
3	만족해요	만족해요	튼튼합니다	만족해요
4	잘쓰고있어요	예뻐요	최고	예뻐요
5	좋아요	최고	좋아요	최고
6	최고	빠르고	잘쓰고있어요	빠르고
7	튼튼하네요	튼튼하고	빠르고	^^
8	존맛	만족	잘 왔어요	만족
9	예쁘네요	편	예뻐요	ㅎㅎ
10	좋아하시네요	걱정 했는데	괜찮네요	!

주요 부정 키워드

	DTM_tk1	TFIDF_tk1	DTM_tk2	TFIDF_tk2
1	최악	실망	최악	실망
2	실망	반품	실망	별로
3	비추	별로	비추	반품
4	다시는	최악	다시는	최악
5	약	비추	좋아요 좋아요	비추
6	좋아요 좋아요	다시는	그래요	그렇다
7	맛있어요 맛있어요	그렇다	맛있어요 맛있어요	다시는
8	맛없어요	환불	별로	——
9	심하네요	약	약	불편
10	그래요	엉망	심하네요	환불

05

프로젝트를 마치며

프로젝트를 마치며

유의어 일원화 작업 핵심

• 만족스럽습니다, 만족합니다, 만족스럽네요, 만족스러워요 등 유의어를 일원화한 후 다양한 모델들의 전반적인 성능이 상향 평준화되는 현상을 통해 유의어 일원화 작업이 가지는 핵심적인 역할을 체감하였습니다.

토큰화 함수는 윤활제

• 전반적으로 토큰화함수 1(두 글자 이상의 명사, 형용사, 동사만 사용)보다 토큰화함수 2(모든 형태소 활용)를 적용했을 때 성능이 개선되는 것을 관찰할 수 있었습니다.

모델링 및 분석

• 유의어 일원화 작업 및 토큰화함수 적용에 따른 성능 차별화는 두각을 드러내었으나, DTM과 TFIDF모델 간의 성능차이는 크지 않았습니다. 적당한 모델을 차용하는 것도 중요하나, 전처리에 대한 중요성을 더욱 체감할 수 있었습니다.


THANK YOU