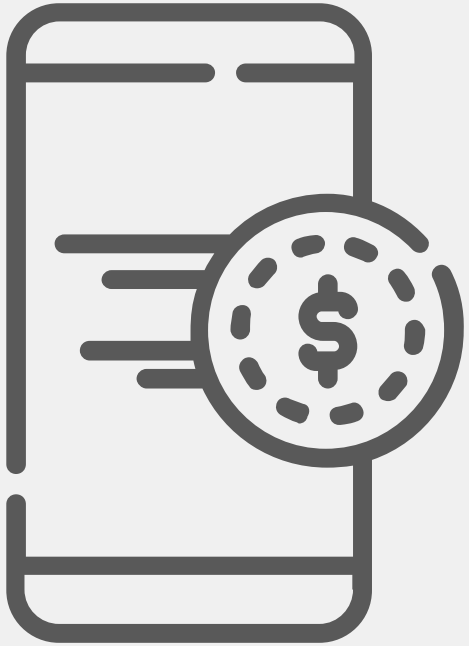
MOBILE BANKING APP REVIEWS ANALYSING

Final Report, team 4



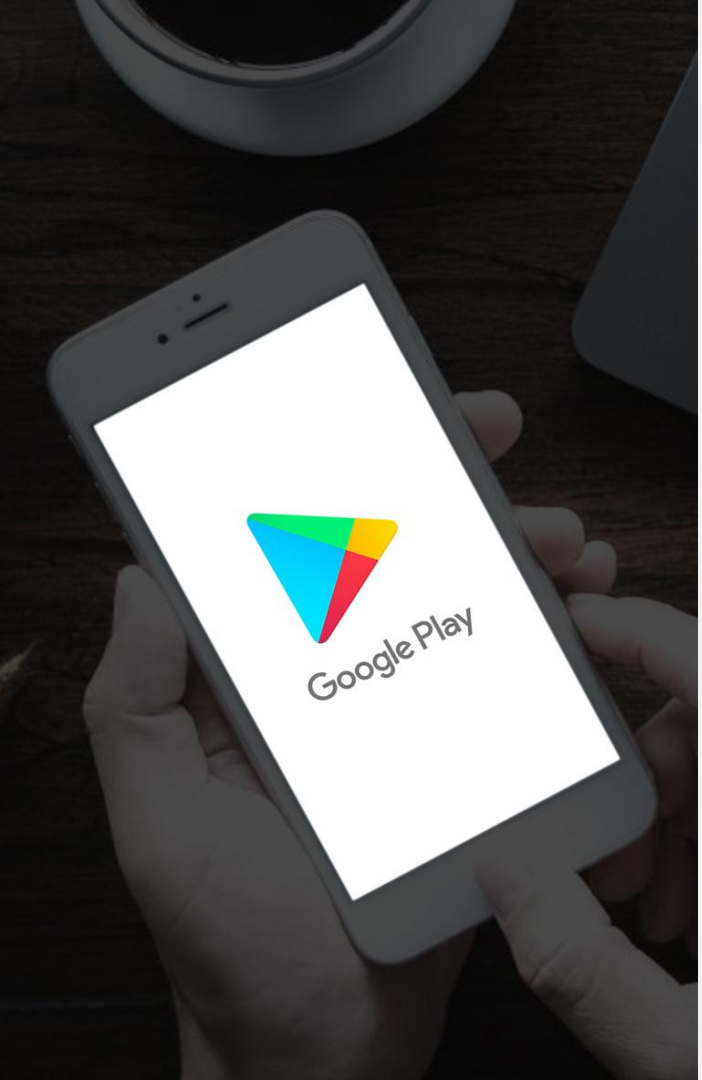


TABLE OF CONTENTS

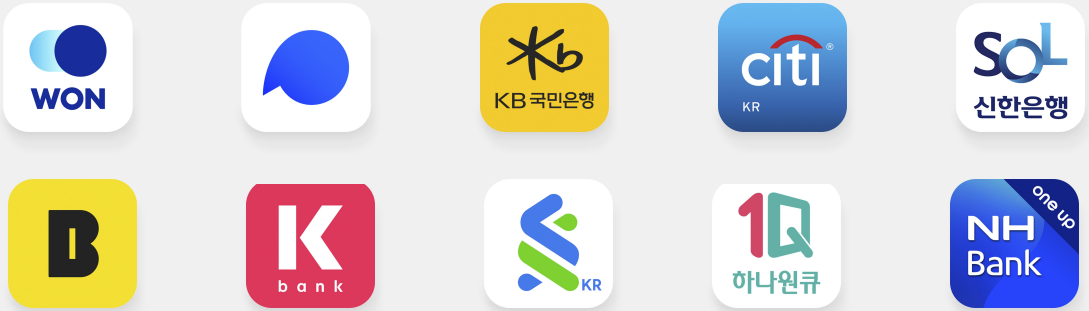
01

OBJECTIVES

프로젝트 소개

1. 프로젝트 소개

Google Play



주제	모바일 뱅킹 어플 리뷰 분석 및 감성사전 구축
내용	금융소비자연맹에서 선정한 은행 순위에서 지방은행을 제외한 상위*10개 은행의 모바일뱅킹 앱에 대해 구글 플레이 스토어에서 리뷰를 분석함
목표	감성 사전 구축 및 은행별 어플 비교 및 개선 방향성 도출

*10개 은행 : 우리, 토스, KB, 한국씨티, 신한, 카카오, 케이, 수협, 하나, 농협

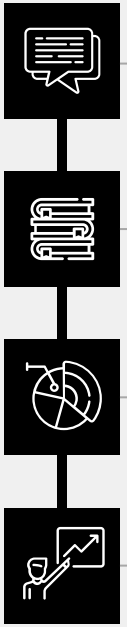
02

PROCESS

프로젝트 개발 경과

2. 프로젝트 개발 경과

- 과정



1. 프로젝트 주제 선정

브레인 스토밍을 통해 주제 선정

2. 데이터 수집, 데이터 전처리

플레이 스토어에서 리뷰 수집, 2021년도의 리뷰만으로 데이터 축소,
맞춤법 교정 및 무의미한 리뷰와 글자수 10개 미만의 리뷰 제거

3. 사용자 사전, 감성 사전 구축

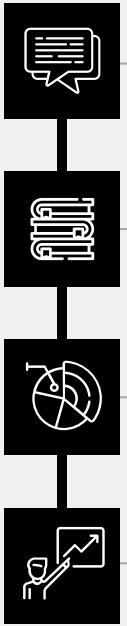
9개 품사의 단어와 빈도 수를 추출하여 사용자 사전을 만들고,
팀원 간 합의 방식으로 감성 사전 구축

4. 긍부정 점수 계산 알고리즘 구상 및 구현

직면한 문제들을 최대한 해결하고, 은행 리뷰에 알맞는 최적의 알고리즘을
구상하여 파이썬을 통해 코드로 구현함

2. 프로젝트 개발 경과

- 과정



5. 긍부정 점수 계산 및 신뢰도 계산

구현한 알고리즘을 통해 긍부정 점수 계산(긍정은 양수값, 부정은 음수값)을 시행하고, 실제 리뷰의 별점과 비교하여 신뢰도를 계산함.

6. 점수 계산 결과 분석

앞서 계산한 점수들을 토대로 은행별 긍부정 비율과
긍정, 부정 리뷰 각각에서 많이 등장하는 단어들을 워드클라우드로 시각화함.

7. 개선 요구 리뷰만 추출 후, 개선 필요 카테고리 분류

전체 리뷰 중 개선을 요구하는 리뷰만을 추출하여,
개선이 필요한 카테고리를 크게 오류, 속도, 편리성으로 분류함.

8. 은행별 개선 필요 카테고리 분석 및 시각화

은행별 개선이 필요한 부분을 조금 더 자세하게 분석해보고,
각 카테고리별로 방사형 차트로 시각화함.

2. 프로젝트 개발 경과

- 주제 선정

아이디어 브레인스토밍 주요 아이디어 | 4조 (이은, 문결, 여윤아)

도서 리뷰 분석

자기계발서, 금융 서적 등 분야를 정하고, 베스트셀러들을 모아 리뷰를 통시적으로 분석하여 유행의 흐름을 살핀다.

장점

- 이용자 성향에 따라, 다른 콘텐츠 리뷰에 비해 비교적 정돈된 텍스트
- 유용성 (시장 예측 가능성)

단점

- 리뷰의 양이 상대적으로 적음
- 주요한 리뷰 사이트를 특정하기 어려움
- 이북의 경우 본문을 통한 추천 시스템 이미 존재함

예시

- 감성적 vs 이성적
- 위로함 vs 다그침
- 긍정적(추천할 만함) vs 부정적(추천하지 않음)
- 통시적으로 각 사회적 배경마다 주요한 키워드/분야 분석

어플 리뷰

은행 어플, 오픈뱅킹 어플, 다이어리 어플, 독서 기록 어플 등 분야를 정해 해당 어플에서 자주 지적되는 문제점 등을 분석

장점

- 구체적인 문제 사항을 지적하는 텍스트 많음
- 유용성 (개선 가능성)
- 긍정-부정이 두드러짐

단점

- 텍스트의 위치가 어플(앱스토어 등) 내부인 경우 크롤링 어려움
- 선정되는 분야에 따라 리뷰를 남기는 이용자 성향 달라짐

예시

- 긍정적(추천할 만함) vs 부정적(추천하지 않음)
- 이용자마다 가중치를 두는 요소별로 분류(디자인, 직관성, 안정성 등)
- 리뷰에서 특히 자주 제안되는 개선 사항 분석

2. 프로젝트 개발 경과

- 주제 선정

기타 아이디어

유튜브 댓글

장점
크롤링 가능, 가치평가를 포함하는 경우 많음,
활용 가능성 다양하고 큼

단점
잘못된 맞춤법, 신조어 등이 많이 사용됨,
영상의 내용 자체에 관한 단어가 많음

음식 리뷰

장점
선호/비선호, 맛있음/맛없음, 위치, 가격, 대기시간, 서비스 등 비교적 명확한 기준

단점
어플의 경우 크롤링 어려움,
흔한 주제,
일반적으로 함께 있는 별점 시스템 자체가 숫자화 역할

숙박 리뷰

장점
비교적 명확한 기준,
부킹닷컴이나 에어비엔비 등 세분화된 리뷰를 제공

단점
흔한 주제,
일반적으로 함께 있는 별점 시스템 자체가 숫자화 역할

문학사 연구

*시대별 대표적 문학 텍스트를 읽어내 각 시대 혹은 사조별 분위기, 어휘 등을 구분

장점
비교적 컴퓨터가 읽기 쉬운 정돈된 텍스트,
가치평가 없이 간단한 계산

단점
활용 가능성 한정적,
본문을 충분히 확보할 방안 필요

증권 레포트

장점
경제적 유용성, 정돈된 텍스트

단점
대상이 되는 기업 등의 선별 기준이 불명확함,
많은 전문용어,
이미 긍정성의 정도 등을 수치화해서 제공하는 경우 많음

넷플릭스

장점
평가 시스템 부재,
유용성

단점
흩어진 리뷰를 모을 방법,
크롤링 어려움,
왓챠에는 존재하는 시스템

전시/공연 리뷰

장점
배우별, 극장별 등 비교적 명확한 기준,
유용성

단점
충성도 높은 팬층의 무조건적 옹호 존재,
내용과 가치평가 구분 어려움

강의평

장점
명확한 사용 대상, 유용성

단점
어플의 경우 크롤링 어려움,
사람을 대상으로 하는 데서 오는 윤리적 문제

게임 리뷰

장점
선호/비선호 구분 쉬움,
유용성

단점
이미 긍정과 부정을 나눠 리뷰 정보를 제공하는 사이트 존재함,
'너도 당해봐라'식 리뷰문화

웹툰 댓글

장점
활용 가능성, 리뷰의 양,
기존 별점 시스템은 사실상 변별력을 잃음

단점
무의미한 댓글 많음,
정돈되지 않은 텍스트,
가치평가보단 내용 자체에 관한 말이 많을 것

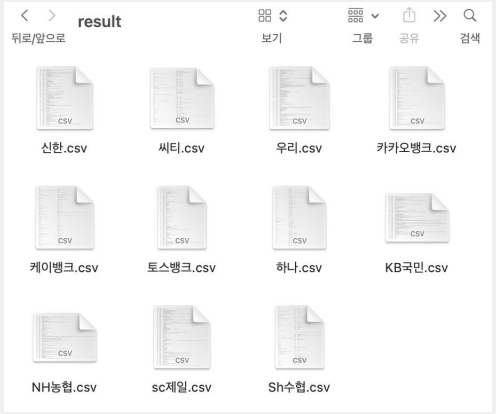
2. 프로젝트 개발 경과

- 데이터 수집 및 전처리

Google Play

google-play-scraper 1.0.2

```
pip install google-play-scraper
```



데이터 수집

(1)

2021년 01월 01일 부터

2021년 09월 30일 까지

해당 기간의 리뷰만 활용

각 은행별 기간을 맞추기 위함과
최신 날짜의 데이터로 축소

(2)

정규 표현식으로

특수문자 및 이모지 제거

(3)

hanspell 패키지

띄어쓰기와 맞춤법 교정

텍스트 전처리

- city_spell_checked.csv
- hana_spell_checked.csv
- k_spell_checked.csv
- kakao_spell_checked.csv
- kb_spell_checked.csv
- nh_spell_checked.csv
- sc_spell_checked.csv
- sh_spell_checked.csv
- shinhan_spell_checked.csv
- toss_spell_checked.csv
- woori_spell_checked.csv

CSV 파일로 추출

2. 프로젝트 개발 경과

- 사전 구축

단어	태그	빈도
있	VV	20369
좋	VA	18626
사용	NNG	17482
이	VCP	16061
하	VV	15405
편리	NNG	11900
…	…	…

리뷰에서 점수매기는 품사의 단어 빈도수

30개 이상 등장하는 단어에 대해 탐색 후 사전에 등록

사용자정의사전

사용자가 원하는 형태로 인식되게 등록

도메인 용어

Ex) 간편결제, 어플

인식 오류 용어

Ex) 명사로 인식되는 '안'

인터넷 용어

Ex) 업뎃, 쩔어, ㄹㅇ

동음이의어

Ex) 은행브랜드 '하나'
숫자 '하나'

감성 사전

단어 점수 등록

Plus Score

명사, 수사, 감탄사, 어근
더하기 점수 등록
곱하기 점수 = 1(중성)

Multiple Score

동사, 부사, 보조용언
곱하기 점수 등록
더하기 점수 = 0(중성)

All Score

형용사
더하기, 곱하기 점수 모두 등록

2. 프로젝트 개발 경과

- 사전 구축

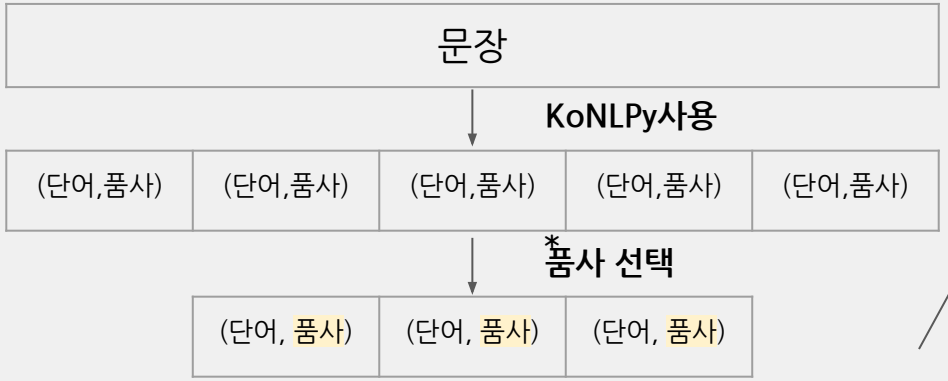
사용자 정의 사전

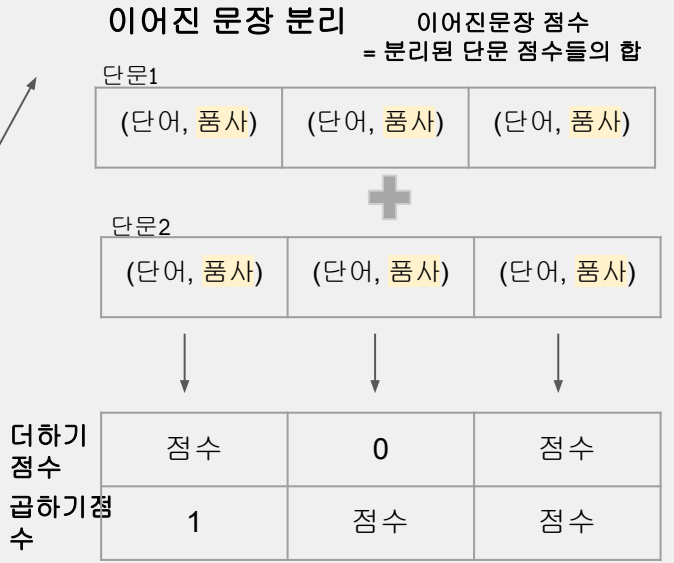
	A	B	C	D
1	WORD	TAG	reinforcer	score
2	1점 주기도	VA	1	-5
3	1점도 아낍	VA	1	-5
4	5점	NNG	1	5
5	5점도 부족	VA	1	5
6	가깝	VA	1	1
7	가능	NNG	1	1
8	가뜩이나	MAG	1.5	0
9	가볍	VA	1.5	2
10	가장	MAG	2	0
11	간결	XR	1	3
12	간단	XR	1	3
13	간단히	MAG	1.5	0
14	간략	XR	1	1
15	간소	XR	1	1
16	간신히	MAG	-0.5	0
17	간절	XR	1	-2
18	간편	NNG	1	3
19	간편	XR	1	3
20	간편히	MAG	1.5	0
21	갈수록	MAG	1.2	0
22	갈아타	VV	-1	0

감성 사전

2. 프로젝트 개발 경과

- 텍스트 점수화





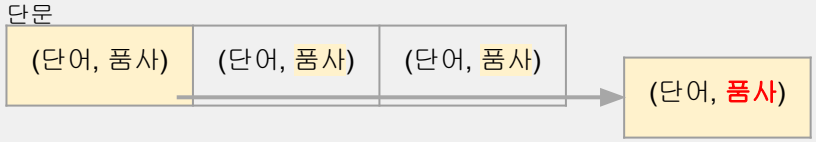
감성 사전에서 단어와 품사가 일치하는 단어 찾아 점수 가져옴

-단어가 없음 : 더히기 0, 곱하기 1점 부여
-단어 일치, 품사 다름 : 일치하는 단어 점수 가져옴

*품사 : NNG(일반명사), NNP(고유명사),VA(형용사), NR(수사), IC(감탄사), 어근(XR),
VV(동사), MAG(부사), VX(보조용언), JKS(주격조사), ETM(관형형 전성 어미), ETN(명사형 전성 어미), EC(연결 어미)

2. 프로젝트 개발 경과

- 단문 텍스트 점수화



단문의 모든 (단어, 품사) 에 대해
왼쪽 과정을 거치면 **계산식 세워짐**

ex) (a+b*c+d*e)*f
= 점수 계산 결과

품사가 *VA이다
- VA +*ETM + NNP/NNG 형태인 경우
  - VA의 곱하기 점수 * NNP 점수
- *NNP/NNG + VA 형태인 경우
  - NNP/NNG점수 * VA의 곱하기 점수
- VA + -게(*EC) + ? 형태인 경우
  - VA의 곱하기점수 * ?
- 위의 3개 경우에 모두 해당되지 않는 형태인 경우
  - VA의 더하기 점수가 더해짐

품사가 VA가 아닌 **덧셈품사이다
- **곱셈품사 * 덧셈품사 형태인 경우
  - 곱셈품사 점수가 1(중성)이 아닌 k값이고
    덧셈품사 점수가 0(중성)인 경우
    - k*1로 계산 (덧셈점수값 0을 1로 변경)

품사가 *VV이다
- 맨 끝인 경우 (?+?+...+VV)
  - 처음부터 VV전까지 계산된 결과에 VV 점수곱해짐
    (계산되어진 점수) * VV
- 맨 끝이 아닌 경우.
  - 뒤에 나오는 단어 점수에 VV 점수 곱하기
    (= VV * ?)

품사가 VA와 VV가 아닌 곱셈품사이다
- 뒤에 나오는 단어 점수에 곱셈품사 점수 곱하기
  (= 곱셈품사점수 * ?)

* VA(형용사), ETM(관형어전성어미), NNP(고유명사), NNG(일반명사), EC(연결어미), VV(동사)

** 덧셈품사 : 더하기점수만 가지는 품사 / 곱셈품사 : 곱하기점수만 가지는 품사

2. 프로젝트 개발 경과

- 리뷰 점수 기반 어플 분석

실제 평점과
계산한 결과 비교
→ 계산 신뢰도 파악

리뷰	별점(1~5)	계산점수
시간과 번거로움 일시에 해결 아주 유용합니다	5	7
타은행보다 앱이 보기 편해요~	5	5
앱이 기본적으로 너무 무겁고 일반 고객이 쓰지 않는 기능이 너무 많아 불편합니다	1	-4
…	…	…

1. 은행간 비교
   은행별 어플 리뷰 평균계산점수 산출
2. 은행별 긍/부정 정도 분석
   은행별 긍부정 리뷰 워드 클라우드
3. 은행별 부정 키워드별 점수
   워드 클라우드 기반 평균 리뷰 산출

03

PROBLEMATICS

직면한 문제들

사전 구축 과정에서의 문제

- 부사인 '안'이 명사로 분류되는 등 일부 단어의 분류상의 문제(코모란 자체의 문제)
- 중립 단어의 점수 부여 문제. 덧셈 점수만 활용될 때는 0이, 강화될 때는 1이 중성적임
- '별 하나도 아깝' 등 하나의 어구가 관용적으로 사용되어 계산되는 점수와 다른 경우가 있음
- '모임 통장'등 개별 은행이 제공하는 특수한 기능들의 존재
- '업데이트' 등 문맥에 따라 긍정적일 수도, 부정적일 수도 있는 단어들의 점수 부여 문제

점수 계산 과정에서의 문제

- 이어진 문장과 안은 문장의 구분, 안은문장의 처리 (특히 강화자를 한정시키는 문제)
- 하나의 오류, 특히 부정 점수의 강화자가 문장의 점수 전체를 좌우하는 위험 부담을 분산시킬 필요
- "다른 좋은 은행 쓰면 되지 왜 이거 씀?" 처럼 직접적으로 부정어가 등장하지 않는 경우
- '아니다' 라는 서술어가 앞의 문장을 어떻게 부정하는지 (0을 곱하는가, -1을 곱하는가?)
- 관형사는 수식하는 명사에만, 서술어는 문장 전체에 곱해지는 등 강화 대상 한정시킬 필요
- 보조사의 역할을 어떻게 표현할 것인지

사전 구축 과정에서의 문제 해결 방안

‘안’ 등 품사가 잘못 나오는 문제	사전에 다른 품사로 등록한다. 잘못 분류된 단어가 해당하는 문장을 찾는 코드를 통해, 가장 많이 쓰이는 맥락을 확인하고 그에 맞춘다. 부사 혹은 명사가 될 수 있는 단어가 문장의 끝에 나올 때 부사로 인식한다.
단어가 깨져서 끊어져 나오는 문제 (끊어져서 나온 단어 때문에 일부 단어 개수가 지나치게 크게 나옴 ex. 번호, 계좌)	이 분야에서 사용되는 합성어들을 user dictionary에 추가해 준다.
맞춤법 교정기가 교정해주지 못하는 오타, 줄임말 등	‘업뎃,’ ‘업댓,’ 등 자주 등장하는 경우 user dictionary에 추가해 준다. 이외의 경우에는 맥락 없이 보아도 무엇의 오타 혹은 줄임말인지 명백한 경우에만 user dic.에 추가해 준다.

일부 명사가 형용사처럼 기능하는 경우	명사 앞에 명사가 나오면 앞의 명사를 형용사처럼 계산하도록 한다. 그 외, 이 분야에서 사용되는 고유한 합성어는 user dic.에 추가해 준다. Ex. 간편 결제 ‘-하-’ 같은 동사파생접미사, 형용사파생접미사가 붙으면 서술어가 되어 reinforcer 값을 가질 수 있도록 한다. Ex. 한눈에 보기 불편하네요: ‘한눈에 보기’라는 긍정적인 명사절을 부정적으로 강화하는 ‘불편(명사) / -하-(접미사)’
양가적인 단어의 경우	별점에 따라 양가적인 값을 부여한다. (문제 해결 방안으로 구상했으나 최종 구현은 되지 않음, 중성값 부여함)
중성이지만, 강화될 때 점수 부여 필요가 있는 경우	더해지기만 할 때는 0, 강화 대상이 될 때는 1을 값으로 갖도록 한다.

점수 계산 과정에서의 문제 해결 방안

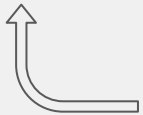
문장 분리 문제

연결어미와 종결어미를 기준으로 문장을 분리한다.
이를 통해 이어진문장을 구분하고,
강화자 하나가 문장 전체의 긍부정을 결정하는 위험을 분산시킨다.

전성어미가 있을 경우 가장 가까운 jks를 찾아 안은문장을 구분한다.
(시도했으나 처리가 오래 걸리고 오류 위험이 커 최종 구현되지 않음)

관형사, 부사, 동사 등 강화자의 강화 대상 한정할 필요

자주 사용되는 문장들을 찾아보고, 유의미한 빈도인 경우
문법을 코드로 구현한다.



어떤 경우에 어떤 것을 강화 대상으로 선택하는지
자세한 방안에 관해서는 이전
프로젝트 경과 부분에서 더 자세히 설명

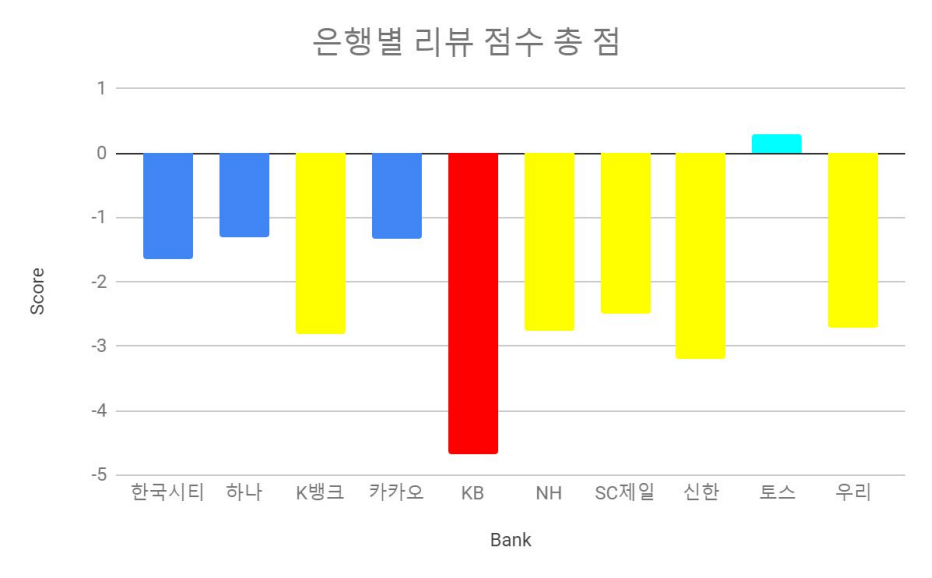
04

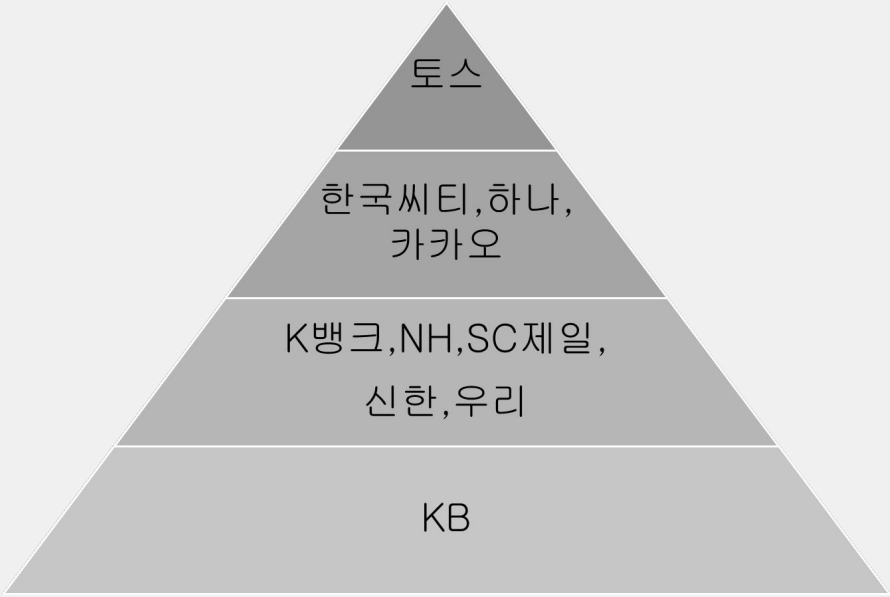
RESULT SUMMARY

결과물 요약

(구체적인 내용은 프로젝트 최종 발표 자료에 포함)

긍부정 계산 최종 결과



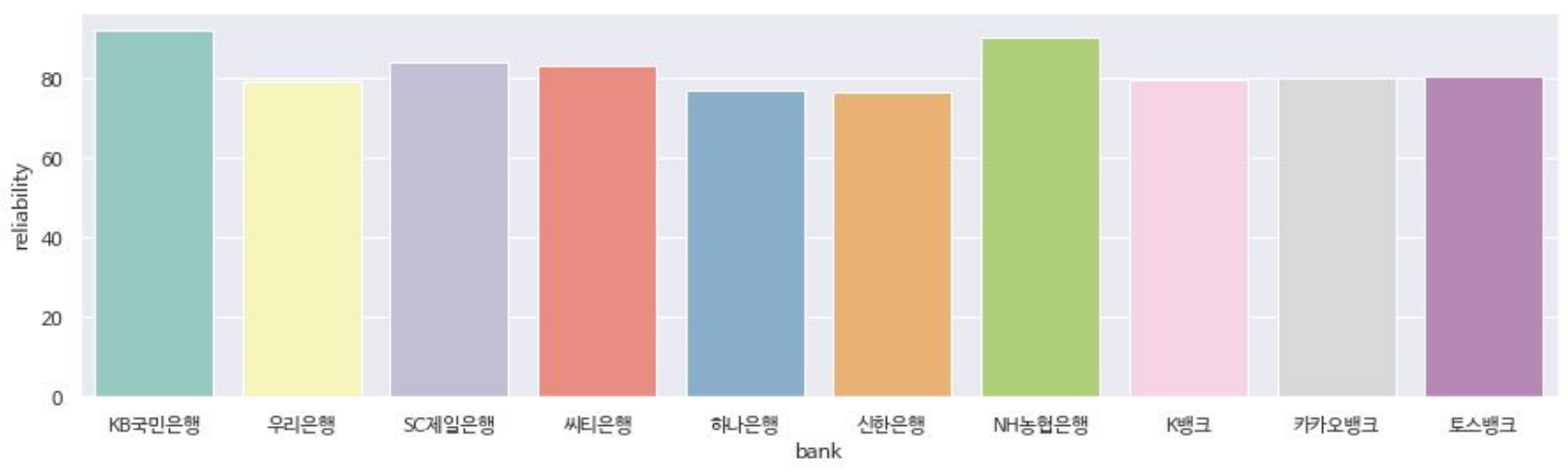


2021.01~2021.09 기간(리뷰 이벤트 기간 제외)
은행별 리뷰 긍부정 계산 점수의 평균

토스>한국씨티,하나,카카오,우리>K뱅크,NH, SC제일, 신한>KB

<고객 불만 지수가 낮은 TOP 4 은행>
토스, 한국씨티, 하나, 카카오

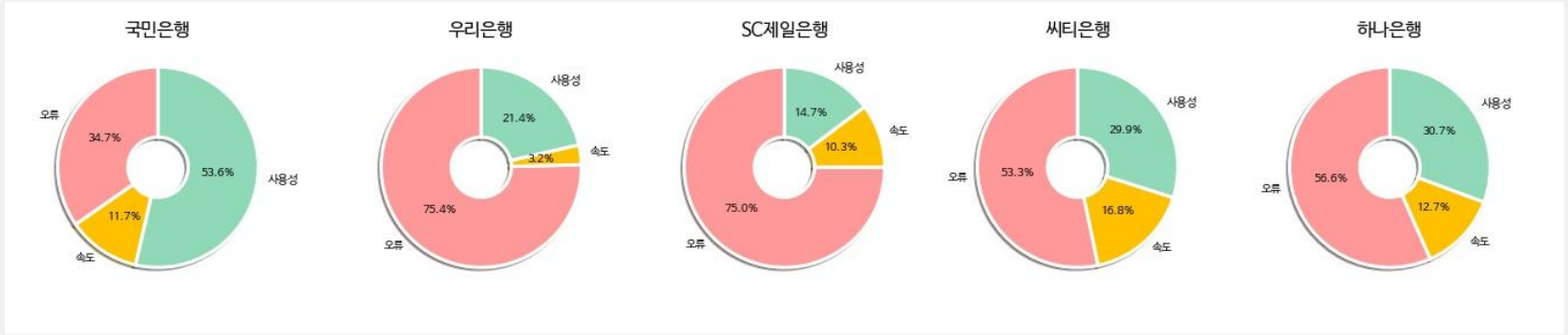
긍부정 계산 신뢰도



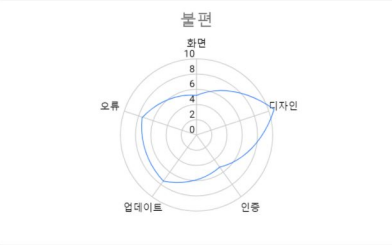
각 10개 은행의 신뢰도 평균: 82.1 %
최고 신뢰도: 91.8% (KB국민은행)
최저 신뢰도: 76.5% (신한은행)

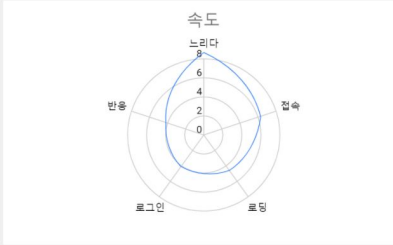
긍정리뷰: 별점 4, 5점　　부정리뷰: 별점 1, 2점
긍정리뷰인데 부정(점수값 음수)으로 계산 &
부정리뷰인데 긍정(점수값 양수)으로 계산된 경우

은행별 개선 필요 카테고리 시각화



은행별 개선 필요 카테고리 시각화





〈한국씨티〉

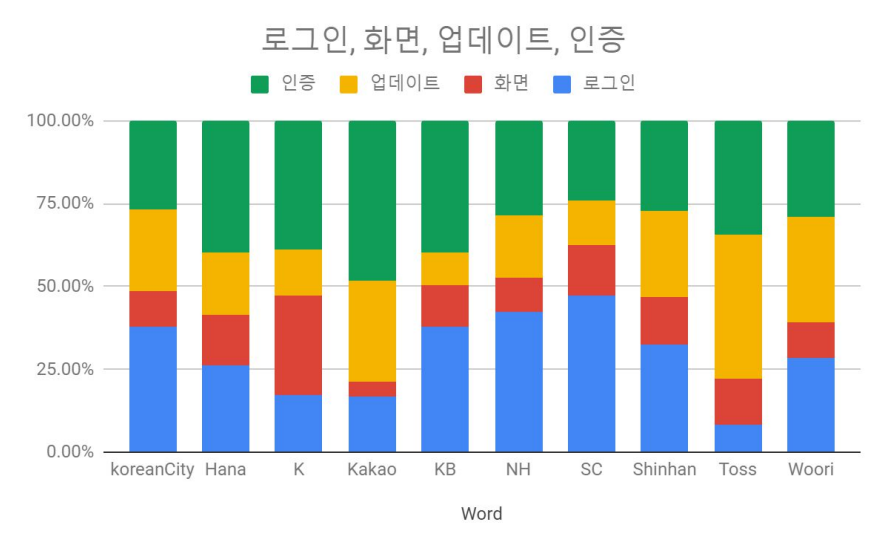
- 사용자들이 불편을 느끼는 대표적인 요소는 **디자인, 업데이트, 오류**에 대한 내용이다.
- 오류는 여러 방면에서 소비자들의 부정 지수 높게 나타난다. 그 중 **로그인과 인증**에 대한 부정 지수가 높음
  → 복잡한 로그인, 로그인 에러, 인증 에러, 인증 절차 문제
- **속도가 느리다**는 의견에 대한 부정 지수가 매우 높음
  → 앱이 전반적으로 느림, 로딩시간이 김

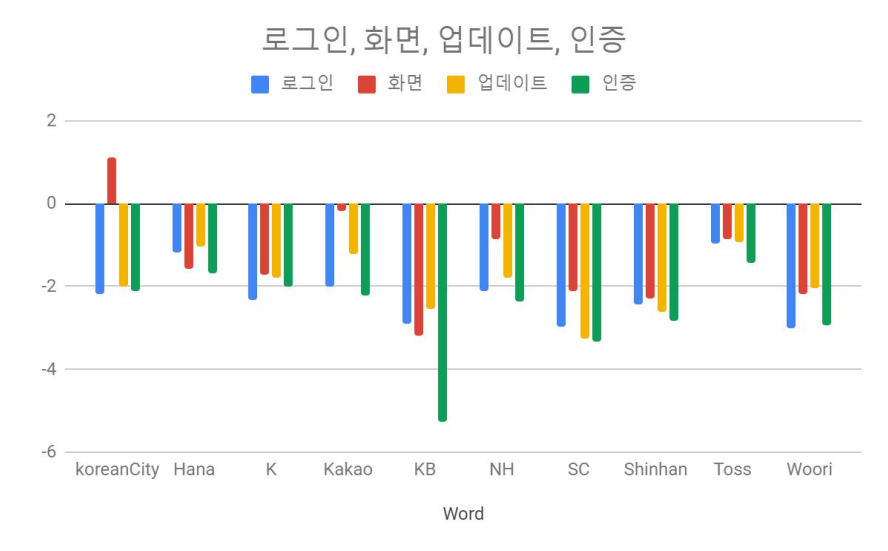
27

좌측 예시와 같이
10개 은행에 대해 각각
불편, 오류, 속도 카테고리에서
가장 두드러지는 불만사항을 도출,
개선 방향성 구체화

4대 불만요소에 대한 은행별 통계

- **10개 은행의 부정 리뷰에서 많이 등장하는 4개의 대표 단어 '로그인', '업데이트', '화면', '인증' 에 대한 은행 별 부정 리뷰에서 비율 및 점수 비교**





한국씨티 : 화면 Good / 로그인,업데이트,인증 Bad
하나은행 : 로그인,업데이트 Good / 화면, 인증 Bad
K뱅크 : 모든 요소 개선 요망
카카오뱅크 : 화면, 업데이트 Good / 로그인,인증Bad
KB : 로그인,업데이트,화면 Bad / 인증 Very Bad

NH : 화면, 업데이트 Good / 로그인, 인증 Bad
SC제일 : 모든 요소 개선 요망
신한은행 : 모든 요소 개선 요망
토스 : 모두 Good
우리 은행 : 모든 요소 개선 요망

05

CODE

자세한 코드 설명

5. 데이터 수집 및 전처리 코드

```python
for review in reviews:

    userName = review['userName']
    score = review['score']
    date = review['at']
    thumbsUpCount = review['thumbsUpCount']
    comment = review['content']

    df = df.append({
        'userName': userName,
        'score': score,
        'date': date,
        'thumbsUpCount': thumbsUpCount,
        'comment': comment
    }, ignore_index=True)
```

```python
# 텍스트 전처리
def cleanText(text):
    specialChar = '[-=+,#/\?:^$.@*\"※~&%·!』\\‘|\(\)\[\]\<\>`\'…》
    ❤♥♡ㅏㅑㅓㅕㅗㅛㅜㅠㅡㅣㄱㄴㄷㄹㅁㅂㅅㅇㅈㅊㅋㅌㅍㅎㄲㄸㅆㅉㅃ]'
    text = ''.join(c for c in text if c not in specialChar)
    only_BMP_pattern = re.compile("["
            u"\U00010000-\U0010FFFF"  #BMP characters 이외 (이모지 제거)
                                "]+", flags=re.UNICODE)
    return only_BMP_pattern.sub(r'', text)

# 맞춤법 검사
def spell_check(comments):
    for k in tqdm(range(len(comments))):
        comments[k] = cleanText(comments[k])
        spelled_sent = spell_checker.check(comments[k])
        checked_sent = spelled_sent.checked
        comments[k] = checked_sent
```

1) PYPI에서 google-play-scraper 패키지를 사용
2) '유저 네임','리뷰 점수','날짜','좋아요 수,"리뷰 내용'을 크롤링
3) data frame형태로 수집하여 csv형태로 추출

1) hanspell 패키지와 정규표현식을 사용
2) '정규표현식으로 특수문자, 이모티콘 및 한글 자음, 모음 제거
3) hanspell 패키지로 맞춤법과 띄어쓰기 교정

5. 품사별 단어 빈도 수 추출 및 csv 병합

```python
# 형태소 분석
def morph_analysis(comments):
  morph_token = []
  for comment in comments:
    token = komoran.pos(comment)
    token = remove_word_by_tag(token, tag_list)
    morph_token.append(token)
  return morph_token


# 품사별 단어 빈도수 계산
def word_count(part, tokens): # part는 원하는 품사
  part_list = defaultdict(int)
  for token in tokens:
    for k in range(len(token)):
      if token[k][1] == part: # token의 품사가 지정한 품사이면
        part_list[token[k][0]] += 1
  return part_list
```

1) Komoran을 이용하여 형태소 분석 실시
2) 점수 계산에 해당되는 품사에 해당되는 단어들만 분리하여, 단어의 빈도수 계산

```python
dir_path = '/content/drive/Shareddrives/언어의미와정보
데이터캡스톤디자인/score_dic'
#files = glob.glob(dir_path + "*.csv")
all_score_dic= pd.DataFrame()

forder = glob.glob("/content/drive/Shareddrives/언어의미와정보
데이터캡스톤디자인/score_dic/*.csv")
total = pd.DataFrame()

for file_name in forder:
  temp = pd.read_csv(file_name, encoding='utf-8')
  total = pd.concat([total,temp])

total.to_csv('all_score_dic.csv')

total = pd.read_csv("/content/all_score_dic.csv")
total.head()

total = total.groupby(['WORD','TAG','reinforcer','score']).sum()
total.tail()

total.to_csv("total.csv")
```

1) 은행별로 단어 점수 계산 csv 파일들에 대해서 중복된 단어들을 제거하고(groupby 이용)
2) 하나의 csv 파일로 병합
(점수 매기기를 쉽게 하기 위해 전체 은행의 단어 빈도 수를 합친 후, 중복 단어는 제거하여 1개의 통합 파일로 만듬)

5. 긍부정 점수 계산 코드

```python
### Seperator_part(한문장)
##한 문장에 대해 형태소 분석하고 (단어, 품사) 형태로 결과를 리턴해주는 함수

def Seperator_part(oneL):
    slash_line = komoran.pos(oneL)
```

```python
# 고려하지 않는 태그는 버리는 함수
def remove_word_by_tag(x):
    temp = list()
    tag_list = ['NNG','NNP','VCP','VCN','VV', 'VA', 'MAG','NR','IC',
    'XR','VX', 'ETM', 'ETN', 'EC', 'JKS'] #고려하는 태그 목록
    for k in range(len(x)):
        if x[k][1] in tag_list:
            temp.append(x[k])
        else:
            continue
    return temp

"""- bring_dic2 : 감성사전 가져오는 함수"""

# 점수 사전 가져와서 dataframe으로 저장하는 함수 (우리가 직접 만든 사전 활용하는 ver)
def bring_dic2():
    df = pd.read_csv("word_dic.csv")
    return df
```

```python
# 단어 점수 가져오는 함수(기본 0,1값 가지는 ver)
def bring_word_score2(w, t, dic, score_type=True):
    temp = dic.loc[(dic['WORD'] == w) & (dic['TAG'] == t)]

    if temp.empty != True :
        if score_type:
            score = float(temp['score'].unique())
            return dic, score

        else:
            rein = float(temp['reinforcer'].unique())
            return dic, rein
```

주요 함수 소개
1) Komoran 형태소 분석 함수
2) 점수 계산에 필요한 품사의 단어만을 걸러내는 함수
3) 점수 사전을 가져와 dataframe으로 저장하는 함수
4) 단어의 점수를 가져오는 함수

5. 긍부정 점수 계산 코드

```python
#이어진 문장쪼개기
# 1. SF, SP 기준으로 쪼개기 -> 2. EC, EF 기준으로 쪼개기 -> 3. jks/jx
+...+etn으로 쪼개기
def seperator_sentence_final(s):
  ls = Seperator_part(s)
  #print('원문장: ',ls)

  stack = [] #쪼개지는 조건 만날 때 까지 계속 쌓임
  final_ls = [] #쪼개져서 묶여진 형태가 쌓이는 스택

  J_in = False

  for k in range(len(ls)):
    if ls[k][1] in ['SF', 'SP']:…


    elif ls[k][1] in ['EC', 'EF']:…
```

```python
def Calculator_review (reviews, df):
    #덧셈품사
    plus_pumsa = ['NNG', 'NNP', 'NR', 'IC', 'XR', 'VCN',
    'VCP']

    #곱셈품사
    mul_pumsa = ['VV', 'MAG', 'VX']

    #기타품사
    etc_pumsa = ['JKS', 'ETM', 'ETN', 'EC']
        #VA인 경우
        elif p == 'VA':
          # VA + ETM + NNP -> VA * NNP
          if (len(filter_pumsa[length-1:]) >= 3) and
          (filter_pumsa[length] == 'ETM') and ((filter_pumsa
          [length+1] == 'NNP') or (filter_pumsa[length+1]
          == 'NNG')):
            df, score_num = bring_word_score2(filter_word
            [length-1], p, df, False)
            cal_ls.append(score_num)
            cal_ls.append('*')
```

주요 함수 소개
5) 이어진 문장 쪼개기 함수(코드가 길어 이미지에서 중간 생략)
6) 점수 계산 함수(코드가 길어 이미지에서 중간 생략)

5. 개선 필요 리뷰 추출 및 카테고리 분류 코드

```python
# 5점 리뷰 제거
bank = bank[(bank['score'] < 5)]
bank = bank.reset_index(drop=True)
print(len(bank))
bank.head()

# comment가 비어있는 경우와 10자 미만인 리뷰 제거
bank = bank.loc[[i for i in range(len(bank['comment'])) if
(type(bank['comment'][i]) == str  and len(bank['comment'][i]
) >= 10)]]

print(len(bank))
bank.head()

problems = bank.query('comment.str.contains("개선|부탁|필요|
해주세|해주시면|해줘|해주셨으면|해주실|좋겠|좋을|나을")',
engine='python')
print(len(problems))
problems.to_csv(bank_name+'_problems.csv')
```

1) 5점 리뷰는 만족하는 내용일 것이라 예상하고, 개선 요구 리뷰에서는 제외함
2) 전처리 후 내용이 없는 리뷰와 10자 미만의 리뷰는 제거함
3) 개선 관련 단어들만이 포함된 리뷰를 추출

```python
# 카테고리 분류(오류, 속도, 사용성)
error_list = ['오류','장애','통신장애','에러','먹통','설치','인식','알람','꺼지','끄',
'켜','멈추','삭제','재설치',
'안되','안 되','통신','부팅','재부팅','종료','접속']
reaction_list = ['반응속도','속도','느리','반응','로딩']
usability_list = ['불편','편의','편리','어렵','힘들','UX','ux','UI','ui','UIUX',
'uiux','UXUI','uxui',
'아이콘','디자인','버튼','심플','한눈','한눈에','편의성','통합','안내',
'통일','인터페이스',
'배너','글씨체','글씨','폰트','메뉴','복잡','답답']

error = 0
reaction = 0
usability = 0

for i in range(len(word_list)):
    for word in word_list[i]:
        if error <= i and word in error_list:
            error += 1
        elif reaction <= i and word in reaction_list:
            reaction += 1
        elif usability <= i and word in usability_list:
            usability += 1

print('오류:', error)
print('속도:', reaction)
print('사용성:', usability)
```

1) 오류, 속도, 사용성으로 카테고리를 분류하고, 각 카테고리에 연관되는 단어들의 리스트를 생성함.
2) 각 카테고리 단어 리스트의 단어가 포함되어 있는 리뷰들을 카운트함.

06

DICTIONARY

산출된 사전

6. 사용자 정의 사전

```
본인인증 NNG
본인 인증 NNG
본인확인 NNG
본인 확인 NNG
공동인증서 NNP
공동 인증서 NNP
스마트 간편결재 NNP
스마트간편결재 NNP
간편결재 NNP
스마트알림 NNP
스마트 알림 NNP
스마트결제 NNP
스마트 결제 NNP
```

```
간편결제 NNP
스마트 간편결제 NNP
스마트간편결제 NNP
스마트 간편 결제 NNP
씨티스마트간편결제 NNP
간편 스마트 결제 NNP
스마트뱅킹 NNP
스마트앱 결제 NNP
스마트 로그인 NNP
```

모바일 뱅킹 앱의 기능과 관련된 단어

```
아주 MAG
정말 MAG
진짜 MAG
무한정 NNG
오류 NNG
되 VX
열 IC
구뜨 IC
쩔어 IC
졸라 MAG
구리 VA
```

```
한 방 NNG
그지 NNG
팝업 NNP
버퍼링 NNP
대박 MAG
겁나 MAG
콕 NNP
콕 뱅크 NNP
넘넘 MAG
작작 MAG
어마어마 MAG
```

비속어 및 부사

```
별 하나 NNG
별하나 NNG
별 하나도 아깝 VA
별점 1점 NNG
별점1점 NNG
1점도 아깝 VA
1점 주기도 아깝 VA
별1개 NNG
별 1개 NNG
별 다섯개 NNG
5점 NNG
다섯개도 부족 VA
5점도 부족 VA
5점 주기엔 NNG
5개 주기엔 NNG
5점 주긴 NNG
5개 주긴 NNG
```

리뷰 점수와 관련된 단어, 관용구

```
장난이 아닙 VV
장난이 아니 VV
장난아니 VV
장난 아니 VV
장난치는것도아니 VA
장난치는것도 아니 VA
장난치는 것도 아니 VA
장난 치는 것도 아니 VA
장난하는 것도 아니 VA
장난하는것도 아니 VA
장난하시는것도아니 VA
장난하는것도아니 VA
```

'장난'도 부정, '아니'도 부정이라
긍정으로 잘못 계산되어 나오는 문구

6. 점수 계산 사전(감성 사전)

스마트	NNG	1	1
스트레스	NNG	1	-2
시답잖	VA	-1	-1
시원찮	VA	-1	-1.5
신선	XR	1	1
신속	NNG	1	1
실망	NNP	1	-3
실용	NNG	1	1
실패	NNG	1	-3
싫	VA	-1	-2
심각	XR	1	-3
심지어	MAG	1.5	0
심플	NNP	1	3
심하	VA	1.5	-2
싸우	VV	-0.5	0
싹	MAG	1.5	0
쓰레기	NNG	1	-5
쓸데없	VA	-1	-2
아깝	VA	0.5	-0.5

탁월	XR	1	3
탈퇴	NNG	1	-3
터지	VV	-1	0
퇴행	NNG	1	-2
튕기	VV	-2	0
특별	XR	1	1
특별히	MAG	1.5	0
특출	XR	1	4
특히	MAG	1.2	0
틀리	VV	-1	0
페미니즘	NNG	1	-3
편리	NNG	1	3
편안	NNG	1	3
편하	VA	2	3
편히	MAG	2	0
평이	XR	1	1
포기	NNG	1	-1

*열의 순서는 왼쪽부터 word, tag, reinforcer, score 임.
1) 팀원 간 합의로 점수를 산정함. (-5부터 5까지)
2) 반의어는 절댓값 점수가 같도록 함.
3) 동의어, 비슷한 의미를 가진 단어는 동일한 점수를 매김.
4) 애매한 경우, 단어가 포함된 리뷰 원문들을 살펴보고 점수를 산정함.
5) 사전에 등록되지 않은 단어는 점수 계산 시, 더하기 점수 0, 곱하기 점수 1을 default 값으로 설정함.
5) 총 463개의 단어로 구성됨.

감사합니다