

LDA Topic Modeling을 통한 Toss 리뷰 분석

빅데이터경영통계전공 20200209 이현지



Contents

01. 연구배경
02. 선행연구
03. 제안방법론
04. 연구절차
05. 과정설명
06. 결론



연구배경

핀테크?

finance + technology

기존의 복잡했던 금융을 사용하기 편리하게 제공합니다.

Z세대 핀테크 앱 1순위 [토스](#).

토스의 어떤 차별화된 서비스가 1위로 만들어주었을까?

어떻게 하면 더 나은 핀테크 앱이 될 수 있을까?

리뷰들을 수집, 토픽 모델링

-> 성공요인과 개선해야 할 점 분석



선행연구

Xianghua et al. (2013) : LDA를 통해 중국 내의 소설 리뷰들을 효과적으로 분석할 수 있는 방법을 연구

Chevalier (2006) : 아마존과 반디앤루니스의 온라인 도서 리뷰들에서 주로 논의되는 숨겨진 토픽들을 찾아내기 위해 LDA 기법을 활용.

Jin et al. (2013) : LDA를 통해 트위터 리뷰들의 토픽 변화를 추적하는 연구를 수행

Park (2015) : Q&A 서비스에서의 고객 리뷰에 대한 LDA 토픽분석을 연구.

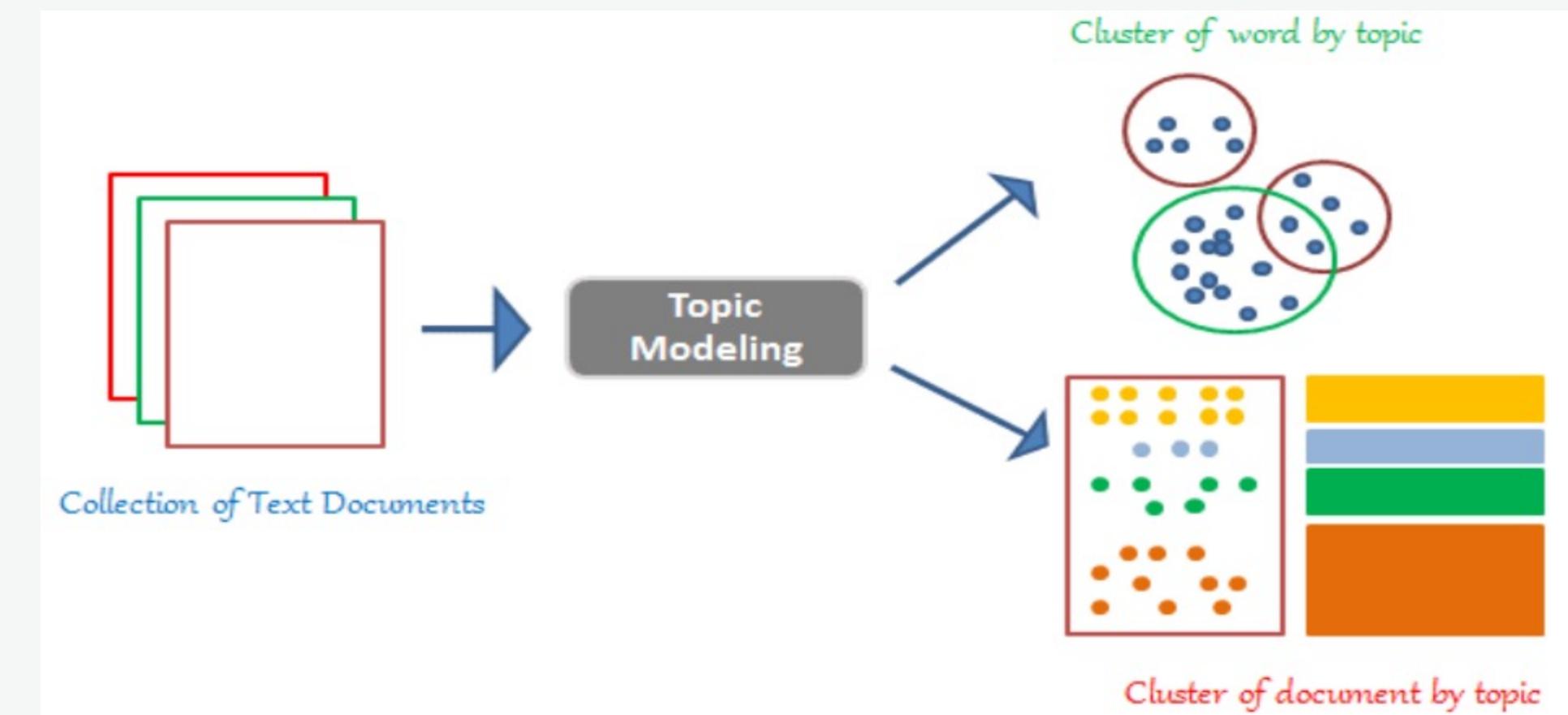
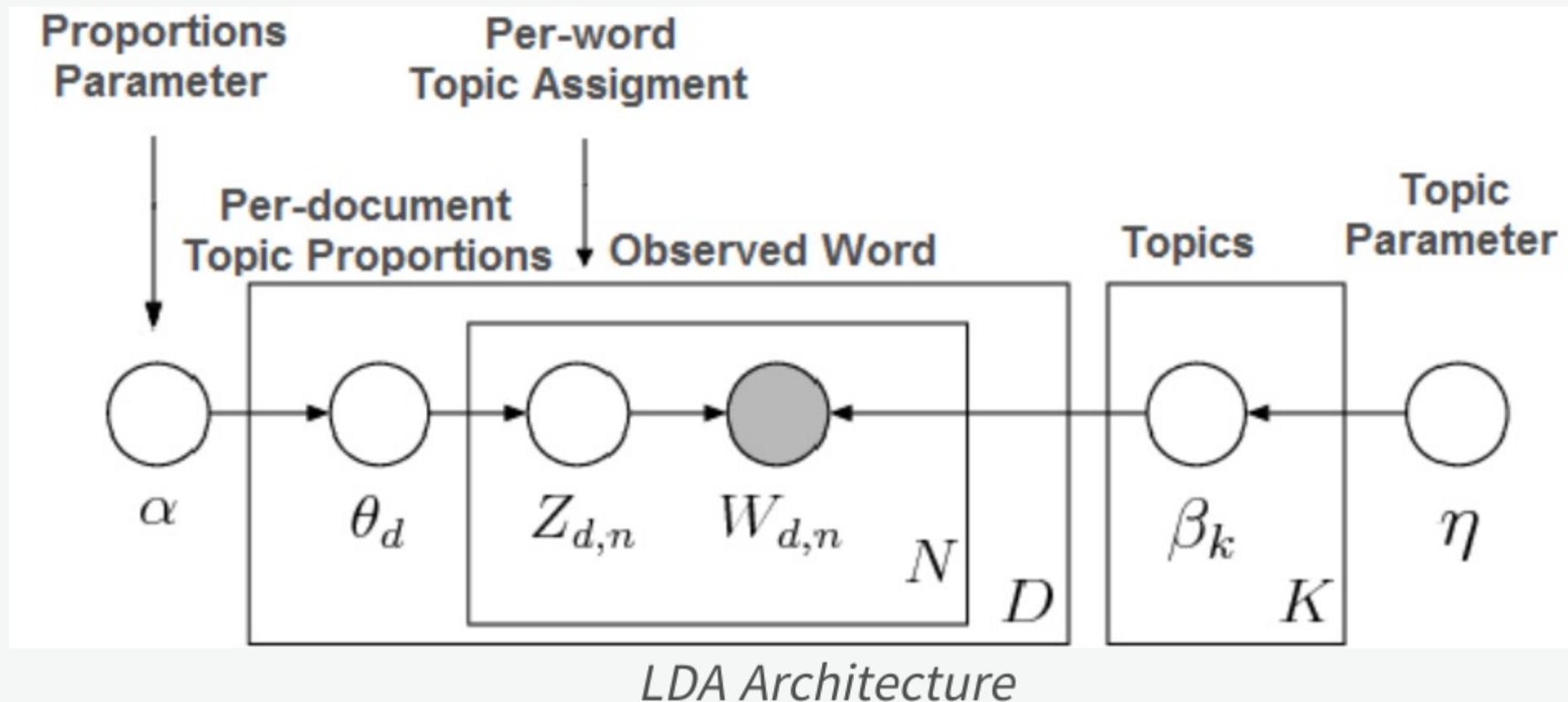


제안방법론

LDA Topic Modeling

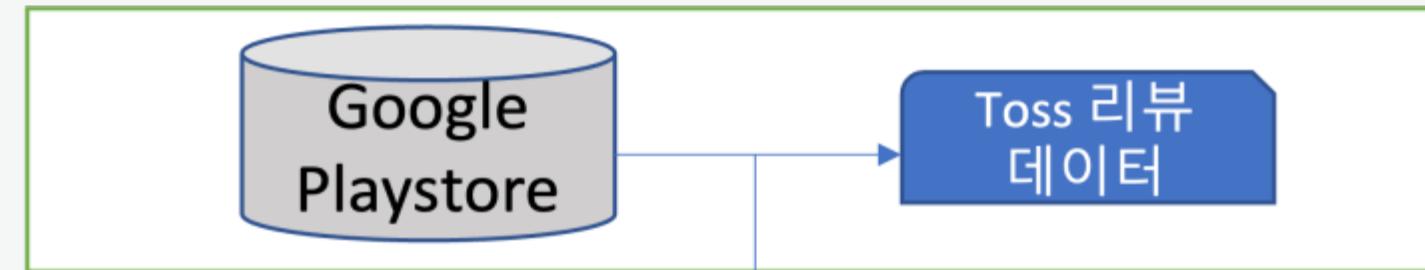
방대한 양의 문서 내의 숨겨져 있는 주제(topic)들을 효과적으로 찾아낼 수 있는 토픽 모델링 기법

- 문서를 여러 토픽들의 확률적인 집합으로 간주
- 주요 토픽들과 토픽 별 비율, 각 단어들이 토픽에 포함될 확률들을 알아낼 수 있다

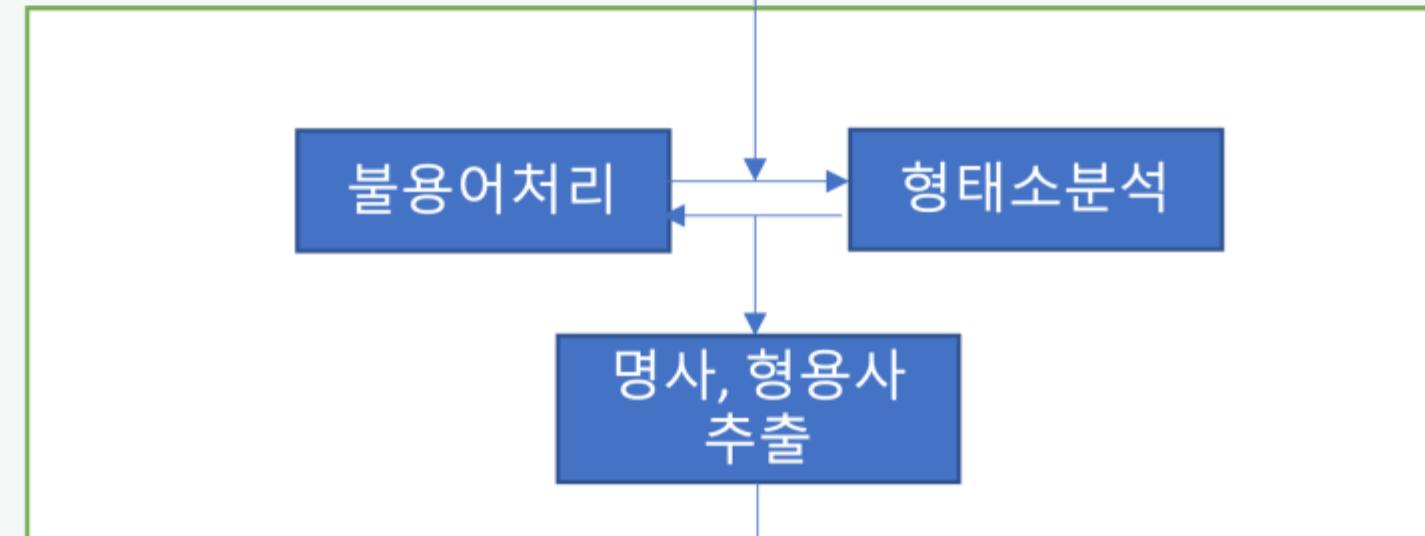


연구절차

1. 데이터수집



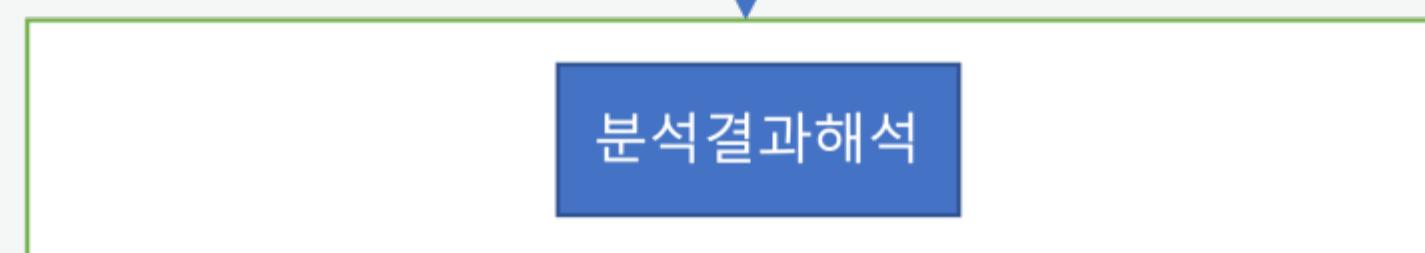
2. 데이터 전처리



3. 데이터 분석



4. 결과해석



과정 설명

1. 데이터수집 (crawling)

- 더보기 버튼 클릭

- 전체리뷰 버튼 클릭

- 자동 스크롤 내리기

- 단문 리뷰와 장문 리뷰 합치기

- 리뷰, 별점, 날짜 추출

```
● ● ●  
data = pd.DataFrame(data=[], columns=['리뷰', '별점', '날짜'])  
import time  
url = 'https://play.google.com/store/apps/details?id=viva.republica.toss&hl=ko&gl=US&showAllReviews=true' #크롤링주소  
# 페이지 열기  
driver.get(url)  
# 페이지 로딩 대기  
wait = WebDriverWait(driver, 5)  
  
SCROLL_PAUSE_TIME = 2  
SCROLL_TIMES = 4 #4번 스크롤 후 더보기 버튼 생성됨  
CLICK_PAUSE_TIME = 2  
  
# 스크롤 높이 받아오기  
last_height = driver.execute_script("return document.body.scrollHeight")  
for i in range(5):  
    for i in range(SCROLL_TIMES):  
        driver.execute_script("window.scrollTo(0, document.body.scrollHeight);")# 최하단까지 스크롤  
        time.sleep(CLICK_PAUSE_TIME) #짧은 시간동안 여러번 반복되지 않도록 중간에 쉬어줌  
        driver.find_element_by_xpath("//span[@class='RveJvd snByac'][0].click() # '더보기' 버튼 클릭  
        new_height = driver.execute_script("return document.body.scrollHeight") # 스크롤 높이 새롭게 받아오기  
  
        if new_height == last_height:# 스크롤 완료!  
            break  
  
    last_height = new_height  
  
spread_review = driver.find_elements_by_xpath("//button[@jsaction='click:TiglPc']") # 전체리뷰 버튼 누르기  
for i in range(len(spread_review)):  
    isTrue = spread_review[i].is_displayed()  
    if isTrue :  
        driver.execute_script("arguments[0].click()",spread_review[i])  
        time.sleep(CLICK_PAUSE_TIME)  
reviews = driver.find_elements_by_xpath("//span[contains(@jsname, 'bN97Pc')]") #짧은 리뷰  
for i in range(len(reviews)):  
    print(str(i) + "\t" + reviews[i].text)  
long_reviews = driver.find_elements_by_xpath("//span[@jsname='fbQN7e']") #긴 리뷰  
for i in range(len(reviews)):  
    print(long_reviews[i].text)  
  
merged_review = [t.text if t.text!='' else long_reviews[i].text for i,t in enumerate(reviews)] #긴리뷰와  
짧은리뷰 합치기  
star_grades = driver.find_elements_by_xpath('//div[@class="pf5lIe"]/div[@role="img"]') #별점  
dates = driver.find_elements_by_xpath("//div[@class='bAhLNe kx8XBd']/div/span[@class='p2Tk0b']") #날짜
```





```
# 리뷰를 수집
for i in range(len(reviews)):
    tmp = []
    tmp.append(merged_review[i])
    tmp.append(star_grades[i].get_attribute('aria-label'))
    tmp.append(dates[i].text)

# 수집한 1명의 리뷰를 결과 프레임에 추가합니다.
tmp = pd.DataFrame(data=[tmp], columns=data.columns)
data = pd.concat([data,tmp])

print("토스앱 리뷰 수집 완료")

data.reset_index(inplace=True, drop=True)
data.head()

tmp = data.copy()
#별점에서 숫자만 가져오기
tmp['별점'] = tmp['별점'].apply(lambda x : x[5:])

import random
m = re.compile('[0-9][\.0-9]*')
tmp['별점'] = tmp['별점'].apply(lambda x : m.findall(x)[0])
tmp.head(3)
```



	리뷰	별점	날짜
0	한국에서 이용하다가 해외 거주자라 다시 돌아왔는데요. 폰번호 해지되어 이용이 안됐어...	3.5	2022년 3월 23일
1	업데이트 후 해외사용, 인증불가/ 안녕하세요. 토스를 몇 년 째 잘 사용해오고 있습...	2	2022년 1월 4일
2	언제부턴가 앱을 열때 지문인식창이 두번 겹쳐나오는데 이럴땐 지문인식이 되지 않을때가...	4	2022년 5월 20일

- 별점에서 숫자만 추출



```

# 데이터 불러오기
DF_high = pd.read_csv("/content/drive/MyDrive/텍데분/토스리뷰_상위.txt")
print(len(DF_high))

# 불러온 데이터의 값이 비어 있는지 확인
print('Null값이 있는지 확인합니다.', DF_high.isnull().values.any()) # Null 값이 존재하는지 확인 (False=정상)
print('')
DF_high = DF_high.dropna(how = 'any') # Null 값이 존재하는 행 제거
print("고유리뷰수:", DF_high['리뷰'].nunique())
print('')

DF_high = DF_high.drop_duplicates() # 중복 데이터 프레임 제거
DF_high = DF_high.reset_index(drop=True) # 데이터 프레임 재생성
print('중복 및, NULL값을 제거한 후, 다시 NULL값을 확인 합니다.', DF_high.isnull().values.any()) # Null 값이 존재하는지 확인 (False=정상)
print('')
print("처리할 데이터수 : ",)

# 데이터 불러오기
DF_low = pd.read_csv("/content/drive/MyDrive/텍데분/토스리뷰_하위.txt")
print(len(DF_low))

# 불러온 데이터의 값이 비어 있는지 확인
print('Null값이 있는지 확인합니다.', DF_low.isnull().values.any()) # Null 값이 존재하는지 확인 (False=정상)
print('')
DF_low = DF_low.dropna(how = 'any') # Null 값이 존재하는 행 제거
print("고유리뷰수:", DF_low['리뷰'].nunique())
print('')

DF_low = DF_low.drop_duplicates() # 중복 데이터 프레임 제거
DF_low = DF_low.reset_index(drop=True) # 데이터 프레임 재생성
print('중복 및, NULL값을 제거한 후, 다시 NULL값을 확인 합니다.', DF_low.isnull().values.any()) # Null 값이 존재하는지 확인 (False=정상)
print('')
print("처리할 데이터수 : ")

```



- | | |
|----|---|
| 0 | 업데이트 후 해외사용, 인증불가/ 안녕하세요. 토스를 몇 년 째 잘 사용해오고 있습... |
| 1 | 토스 질사용하고잇습니다. 한가지 불편한점이 있어서 글남김니다. 다름이 아닌 만보기에... |
| 2 | 미션을 바꾸겠읍니까? 미션을 바꾸겠읍니까? 미션을 바꾸겠읍니까? 도데체 뭘 자꾸 미... |
| 3 | 건의사항 송금을 한다거나 이자를 받을때 발생하는 본인확인하는 기능을 횟수를 줄여 주... |
| 4 | 토스 잘 쓰고 있습니다. 한 가지 아쉬운게, 자산 편집 기능이 온전히 자유롭게 되지... |
| 5 | 너무 잘 이용하고 있는데 한가지 개선해주셨으면 하는 사항이 있어서 글 남겨요 토스뱅... |
| 6 | 이번 달 사용 금액을 확인하는데, 제가 지출하지 않은 내역이 떠있습니다. 해당 내역... |
| 7 | 연금저축이 뜨네요! 그런데 연금탭에 안가고 투자탭에 있더라고요 그래서 옮기고 싶은데... |
| 8 | 다음달 낼 카드값 보여주는 기능 아주 좋습니다! 모든 카드를 다 합산해서 보여주니 ... |
| 9 | 기변하고 앱을 새로 등록하는 과정에서 정보를 입력 하라길래 앱을 변경한 기기에서 다... |
| 10 | 너무 간편하고 너무 좋습니다 계좌 한번에 볼수 있고 대출 주식 장학금 지원금 등등 ... |

- null / 중복 행 제거

- 최종 리뷰 : 200 개

- 별점 3점을 기준으로 상위리뷰와 하위리뷰로 나누었으나 분석방향에 맞게 다시 전체로 합침
(코드로 첨부)



2. 토크나이징 (Tokenizing)

```
tokenizer = Twitter() # 토큰나이저 지정
stopword_vocab = DATA_STOP_WORD_FILE_NAME # 불용어 파일 불러오기
sep = "\n" # 불용어 처리 인자

def build_vocab(data_frame,stopword_vocab, separate):
    # 불용어 데이터를 가져와 리스트로 변환합니다.
    with open(stopword_vocab, encoding = 'utf-8') as f:
        temp1 = []
        for i in f:
            temp1.append(i)
    globals()['stopword_vocab'] = []

    # 불용어 데이터는 전역변수 stopword_vocab 선언합니다.
    # 구분자에 따라 stopword_vocab에 추가하여 불용어 사전을 구축합니다.
    for j in range(len(temp1)):
        temp2 = temp1[j].rstrip(separate)
        globals()['stopword_vocab'].append(temp2)

    # 품사를 달고 token을 출력합니다.
    globals()['list_sent2words'] = []
    for i in range(len(data_frame)) :
        num_list=[]
        temp = tokenizer.pos(data_frame[i])
        for j in range(len(temp)):
            num_list.append(temp[j])
        globals()['list_sent2words'].append(num_list)

    return [[word for word in doc if word not in globals()['stopword_vocab']] for doc in globals()['list_sent2words']]

result_data =build_vocab(temp['리뷰'],stopword_vocab, sep)
```



result_data

```
[[ ('업데이트', 'Noun'),
  ('후', 'Noun'),
  ('해외', 'Noun'),
  ('사용', 'Noun'),
  (',', 'Punctuation'),
  ('인증', 'Noun'),
  ('불가', 'Noun'),
  ('/', 'Punctuation'),
  ('안녕하세요', 'Adjective'),
```

- KoNLPy 패키지의 Twitter 형태소 분석기 사용
- Twitter().pos : 품사를 달고 token 추출



3. 전처리 및 불용어처리

```
#얻은 토큰에서 명사, 형용사만 가져오기  
newL=[ ]  
N_POS = ['Noun']  
V_POS = ['Adjective']  
for n in range(len(result_data)):  
    temp = []  
    for k in result_data[n]:  
        if k[1] in V_POS:  
            temp.append(k[0])  
        elif k[1] in N_POS:  
            temp.append(k[0])  
    newL.append(temp)  
print(newL)
```

```
#불용어 제거  
def remove_stopwords(text):  
    temp2 = []  
    for j in range(len(text)):  
        meaningful_words = [w for w in text[j] if not w in stopword_vocab]  
        temp2.append(meaningful_words)  
    return temp2  
  
finalL=remove_stopwords(newL)  
  
#길이가 1이상인 단어만 가져온다  
num_list = []  
for j in range(len(finalL)):  
    temp = []  
    for k in range(len(finalL[j])):  
        if len(finalL[j][k]) > 1:  
            temp.append(finalL[j][k])  
    num_list.append(temp)  
  
print(num_list)
```

```
#추가 불용어제거  
stopword = ['입니다', '입니다', '있어요', '있어서', '있는', '같아요', '있으나', '있으면']  
LL=[ ]  
for j in range(len(num_list)):  
    temp = [i for i in num_list[j] if i not in stopword]  
    LL.append(temp)  
print(LL)
```

- 명사, 형용사만 추출

- 불용어 제거 및 길이가 1이하인 단어 제거

- 추가적인 불용어 제거



```
[ ['업데이트', '해외', '인증', '불가', '안녕하세요', '이민', '한국', '잠시', '한국', '번호', '여태', '번호', '업데이트',  
'한국', '번호', '없어서' ] ] #중략
```



4. 단어빈도 확인 및 시각화 (Term Frequency / WordCloud)

```
● ● ●  
# 전체에 대한 워드 카운트 계수 확인  
  
def word_corpus(result_data):  
    # 전체 단어의 갯수 파악  
    words = list(itertools.chain(*result_data))  
    print('전체 워드의 개수 : {}'.format(len(words)))  
  
    # 단어의 빈도수를 확인 후 추가할 불용어 확인 작업  
    vocab = Counter(words)  
    vocab_size = len(words)  
    vocab = vocab.most_common(vocab_size) # 등장 빈도수가 높은 상위 n개의 단어만 저장 vocab  
    return vocab  
  
vocab=word_corpus(LL)  
print(vocab)  
# 전체 워드의 빈도 계수  
df_corpus=pd.DataFrame(columns=["text","count"])  
tmp_list=[]  
tmp_list1=[]  
for word, num in vocab:  
    tmp_list.append(word)  
    tmp_list1.append(num)  
df_corpus['text']=tmp_list  
df_corpus['count']=tmp_list1  
# 상위 20개의 워드 카운트 계수만 출력  
a=df_corpus.head(1000)  
print(df_corpus.head(20))  
  
# 빈도분석한 결과를 별도의 파일에 저장함  
a.to_excel(RESULT_corpus_EXCEL ,sheet_name = "sheet2low")
```



```
전체 워드의 개수 : 4442  
[('기능', 97), ('계좌', 75), ('업데이트', 56), ('송금', 44), ('카드', 39), ('주식', 39), ('추가', 32), ('은행', 31), ('개설', 28), ('소비', 28), ('이용', 27), ('입력', 27), ('포인트', 26)], #증권  
text count  
0 기능 97  
1 계좌 75  
2 업데이트 56  
3 송금 44  
4 카드 39  
5 주식 39  
6 추가 32  
7 은행 31  
8 개설 28  
9 소비 28  
10 이용 27  
11 입력 27  
12 포인트 26  
13 설정 25  
14 증권 25  
15 인증 24  
16 데이터 24  
17 문제 21  
18 자동 21  
19 좋겠습니다 20
```

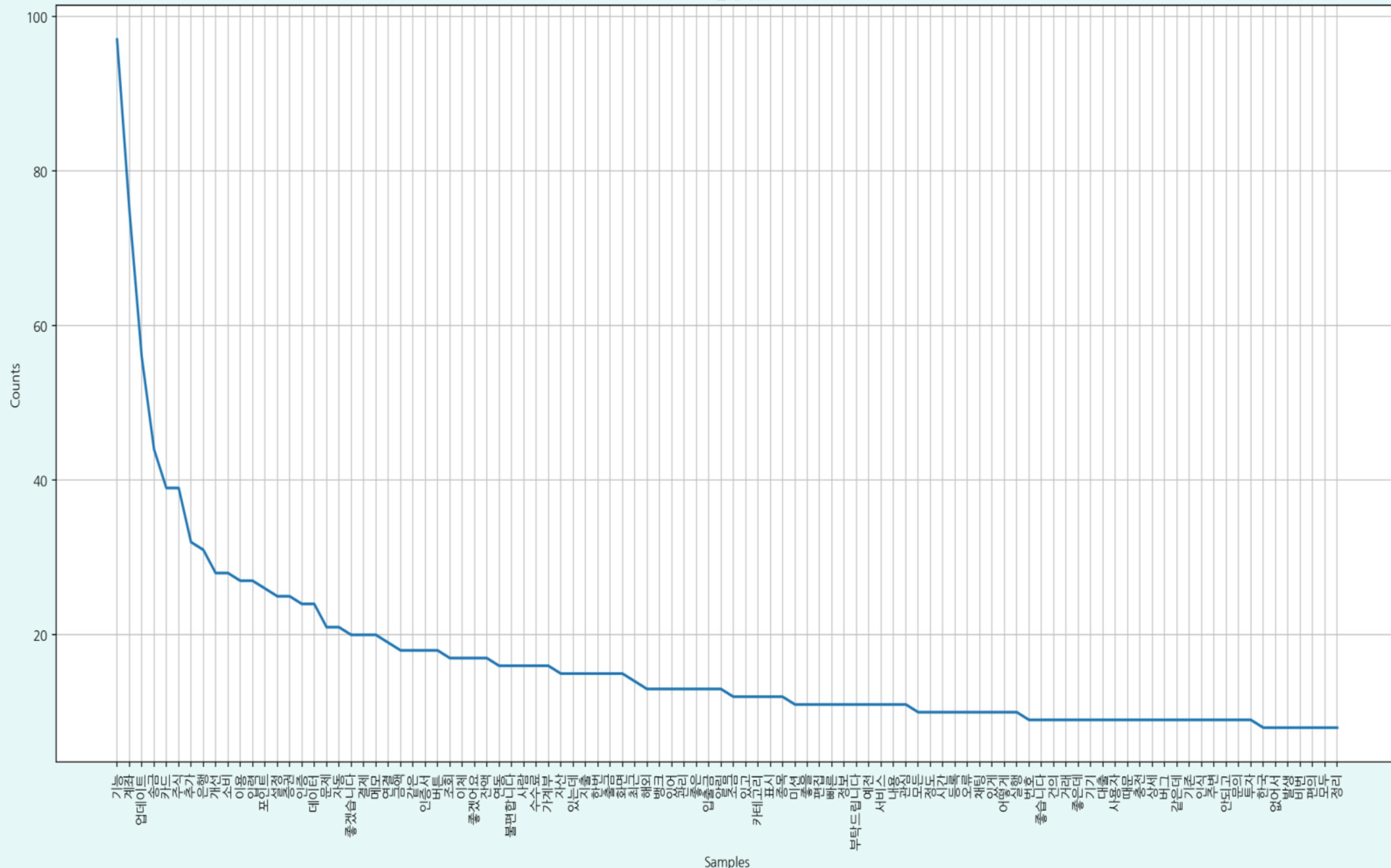
- 전체 워드의 개수 : 4442개

- 출현빈도

- 1) 기능 : 97
- 2) 계좌 : 75
- 3) 업데이트 : 56
- 4) 송금 : 44



토스전체리뷰



Word Cloud



5. LDA 토픽모델링



```
#documents를 받아서 문서 단어 행렬 만들기
def build_doc_term_mat(documents):

    id2word = corpora.Dictionary(documents) #id2word : 토픽 모델링 딕셔너리
    corpus = [id2word.doc2bow(document) for document in documents] #corpus : 용어-문서 빈도

    return corpus, id2word

#단어를 입력하면 단어가 여러 토픽에 해당될 수 있는데 각 토픽에서 갖는 가중치를 표시
def print_term_topics(term, dictionary, model):
    word_id = dictionary.token2id[term] #단어의 아이디 구함
    print(model.get_term_topics(word_id))

# 문서에 대한 토픽가중치를 반복하면서 전체 문서에 대해서 표시
def print_doc_topics(model, corpus):

    for doc_num, doc in enumerate(corpus):
        topic_probs = model[doc]
        print("Doc num: {}".format(doc_num))

        for topic_id, prob in topic_probs:
            print("\t{}\t{}".format(topic_id, prob))

        if doc_num == 2: # 시간 관계상 2번 문서까지만 출력, "0번문서, 1번문서, 2번문서"에 대해서만 해당문서의 토픽가중치를 표시
            break

    print("\n")

#모델링 후 각 토픽별로 중요한 단어들을 표시
def print_topic_words(model) :
    for topic_id in range(NUM_TOPICS):
        topic_word_probs = model.show_topic(topic_id, TOPICS_W_NUM)
        print("Topic ID: {}".format(topic_id))

        for topic_word, prob in topic_word_probs:
            print("\t{}\t{}".format(topic_word, prob))
        print("\n")

#LDA 시각화
def create_vis(model):
    pyLDAvis.enable_notebook()
    vis = pyLDAvis.gensim_models.prepare(model, corpus, id2word, sort_topics=False)
    pyLDAvis.save_html(vis, RESULT_SAVE_LDAVIS)
    return vis
```

- 문서 단어 행렬 만들기

- 토픽별 단어 가중치 갱신

- pyLDAvis : LDA 결과 시각화



```

● ● ●

# Perplexity와 Coherence Score 을 판단
# Perplexity는 작을 수록 Coherence Score는 높을 수록 좋다.

TOPICS_W_NUM =10 #토픽 당 단어수는 10개
save_lda_model=0
UPDATE_EVERY = 1
CHUNKSIZE = 100
PASSES = 10

for i in range(1,30):
    NUM_TOPICS=i

    #해당 셀은 토픽모델링(LDA)에 대해 모델을 정의하는 셀
    lda_model = gensim.models.ldamodel.LdaModel(corpus=corpus, id2word=id2word,
                                                num_topics=NUM_TOPICS, random_state=100,
                                                update_every=UPDATE_EVERY, chunksize=CHUNKSIZE,
                                                passes=PASSES, alpha='auto', per_word_topics=True)

    doc_lda = lda_model[corpus]

    # Perplexity | Coherence Score
    coherence_model_lda = CoherenceModel(model=lda_model, texts=LL, dictionary=id2word, coherence='c_v')
    coherence_lda = coherence_model_lda.get_coherence()

    print('NUM_TOPICS',NUM_TOPICS,'Perplexity: ', lda_model.log_perplexity(corpus),'Coherence: ',
coherence_lda)

```



NUM_TOPICS	Perplexity	Coherence
1	-7.060601889255185	0.30202677436604575
2	-7.086849524064651	0.270336405675204
3	-7.131684110913793	0.33793632128274415
4	-7.207959328452272	0.3238951171764439
5	-7.258335444604905	0.37669554761706453
6	-7.279465471729401	0.3991637614989227
7	-7.323815653636508	0.39751678754176334
8	-7.348520679832954	0.43146892277532845
9	-7.350629059300129	0.4129028994347923
10	-7.372638991170507	0.41546121040785355
11	-7.383842048909094	0.42048327296335103
12	-7.418442720647118	0.3973580641363392
13	-7.429866696091076	0.4036862983346316
14	-7.441872994234733	0.40712379105179936
15	-7.448787418304398	0.41333838345756335
16	-7.462325705393968	0.40788161479815693
17	-7.4889953393004385	0.3867070559579678
18	-7.478498119953139	0.40177420899140004
19	-7.510693222029367	0.4109871338131904
20	-7.501996440508515	0.37373978729670915
21	-7.495388243016977	0.38995957335769094
22	-7.518738039653079	0.39694474215365055
23	-7.524087794671881	0.39162175910411434
24	-7.537194650232765	0.36421580610746895
25	-7.536648677134503	0.3572179548613726
26	-7.539852211063159	0.35844030976674995
27	-7.5275298341176144	0.36192274310115485
28	-7.574491638151819	0.3553604395291286
29	-7.571419637142029	0.3526705043426142

- Perplexity | Coherence Score 계산

- 낮은 Perplexity, 높은 Coherence





```
print('토픽 기본 모델링을 실시 합니다. 해당 모델은 "lda_model" 변수로 입력됩니다.')
print(' ')

NUM_TOPICS = int(input('토픽의 개수를 입력해 주세요. ')) #8개
TOPICS_W_NUM = int(input('출력할 토픽별 단어의 개수를 입력해 주세요 ')) #10개
save_lda_model= int(input("선택한 토픽 모델을 저장하시겠습니까? \n0 저장 \n1 미저장 ")) #0 : 저장

UPDATE_EVERY = 1
CHUNKSIZE = 100
PASSES = 10

# lda 모델 정의
lda_model = gensim.models.ldamodel.LdaModel(corpus=corpus, id2word=id2word,
                                              num_topics=NUM_TOPICS, random_state=100,
                                              update_every=UPDATE_EVERY, chunksize=CHUNKSIZE,
                                              passes=PASSES, alpha='auto', per_word_topics=True)

# 토픽 출력
pprint(lda_model.print_topics(num_words=TOPICS_W_NUM))
doc_lda = lda_model[corpus]

# 모델 저장
if save_lda_model == 0:
    lda_model.save(LDA_MODEL_SAVE_NAME)
```



Topic ID: 5

기능	0.0514703169465065
계좌	0.03229152411222458
송금	0.02705664373934269
추가	0.01872647926211357
설정	0.015620549209415913
가계부	0.012666907161474228
좋겠습니다	0.011603404767811298
메모	0.010753708891570568
사람	0.009412523359060287
관리	0.009186951443552971

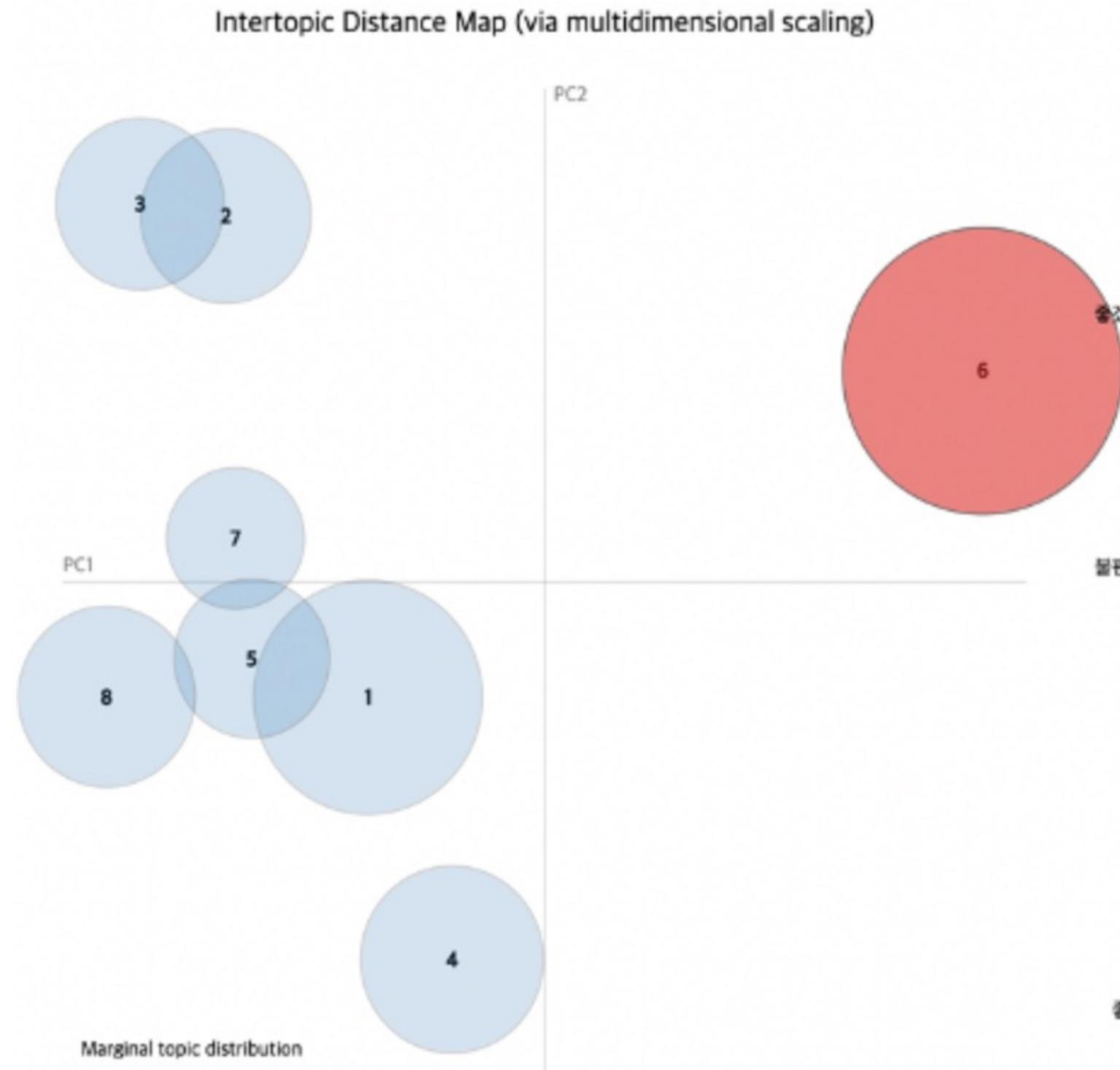
- 토픽 수 : 8

- 토픽 별 단어의 수 : 10



결론

Distance Map



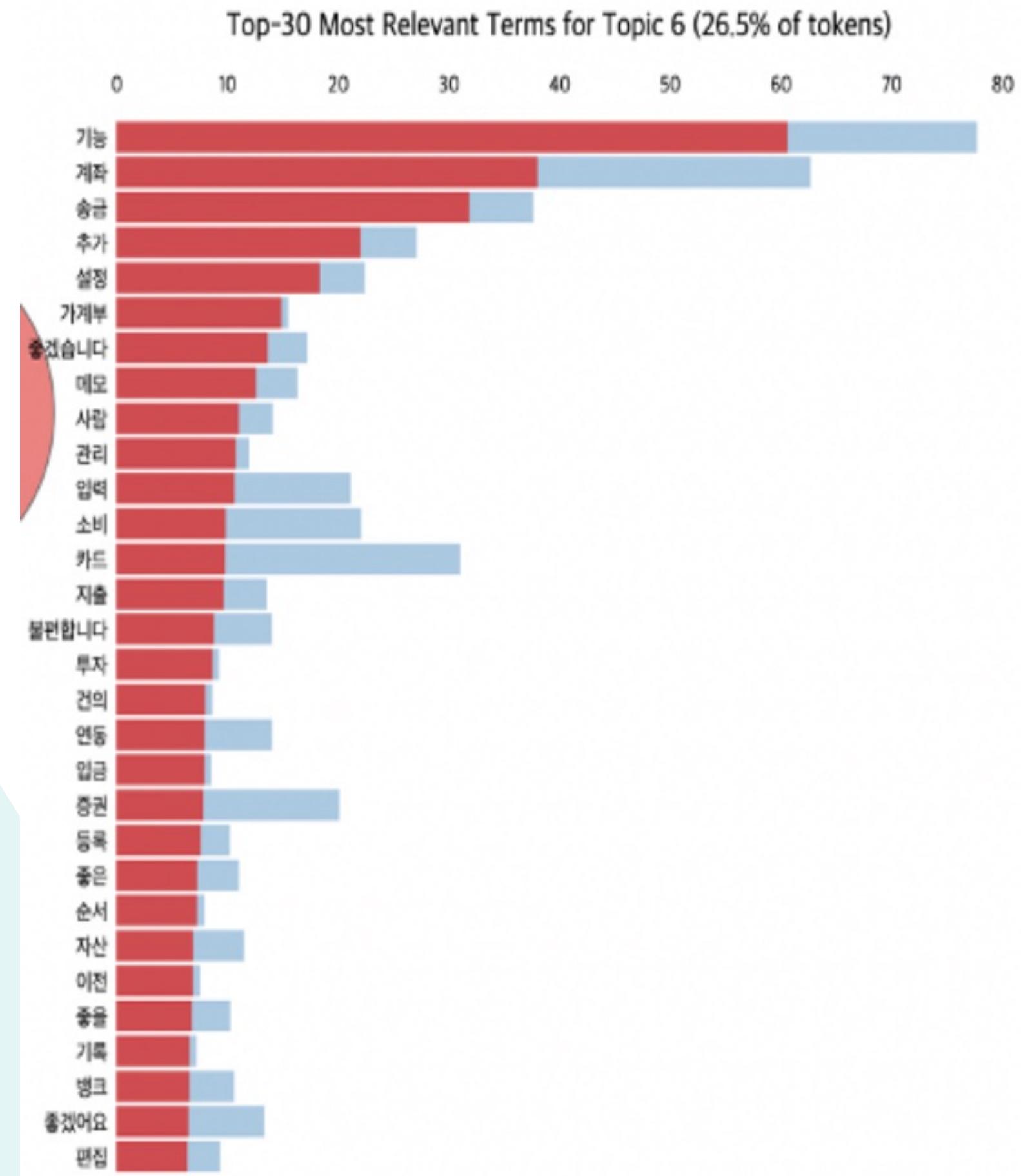
- 가까이 있을 수록 비슷한 토픽
=> Topic 2,3 / Topic 6/ Topic 1,4,5,7,8
- 원이 클 수록 문서 내 지배적인 토픽
=> 가장 지배적인 토픽은 Topic 6



Topic 6

▶ 토스의 다양한 기능들

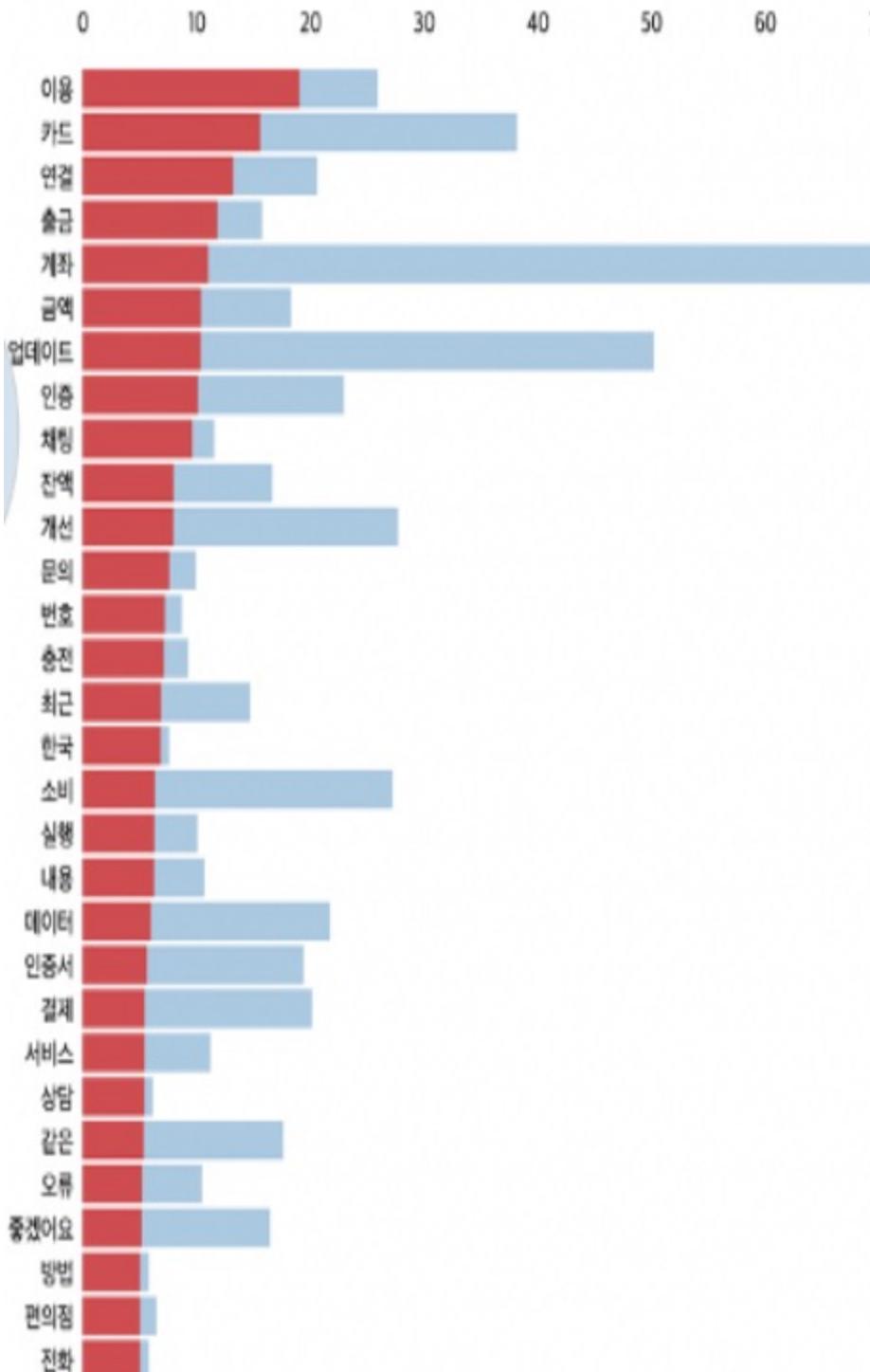
- 기능, 계좌, 송금, 가계부
- 토스의 강점 : 송금, 가계부, 메모, 주고받는 사람 표시, 투자, 증권 etc
- 좋겠습니다, 불편합니다, 좋겠어요 와 같은 감정표현 : 다양한 기능들의 강점을 더 부각하기 위해선 리뷰를 토대로 개선이 필요해보임



금융 서비스에서의 고충

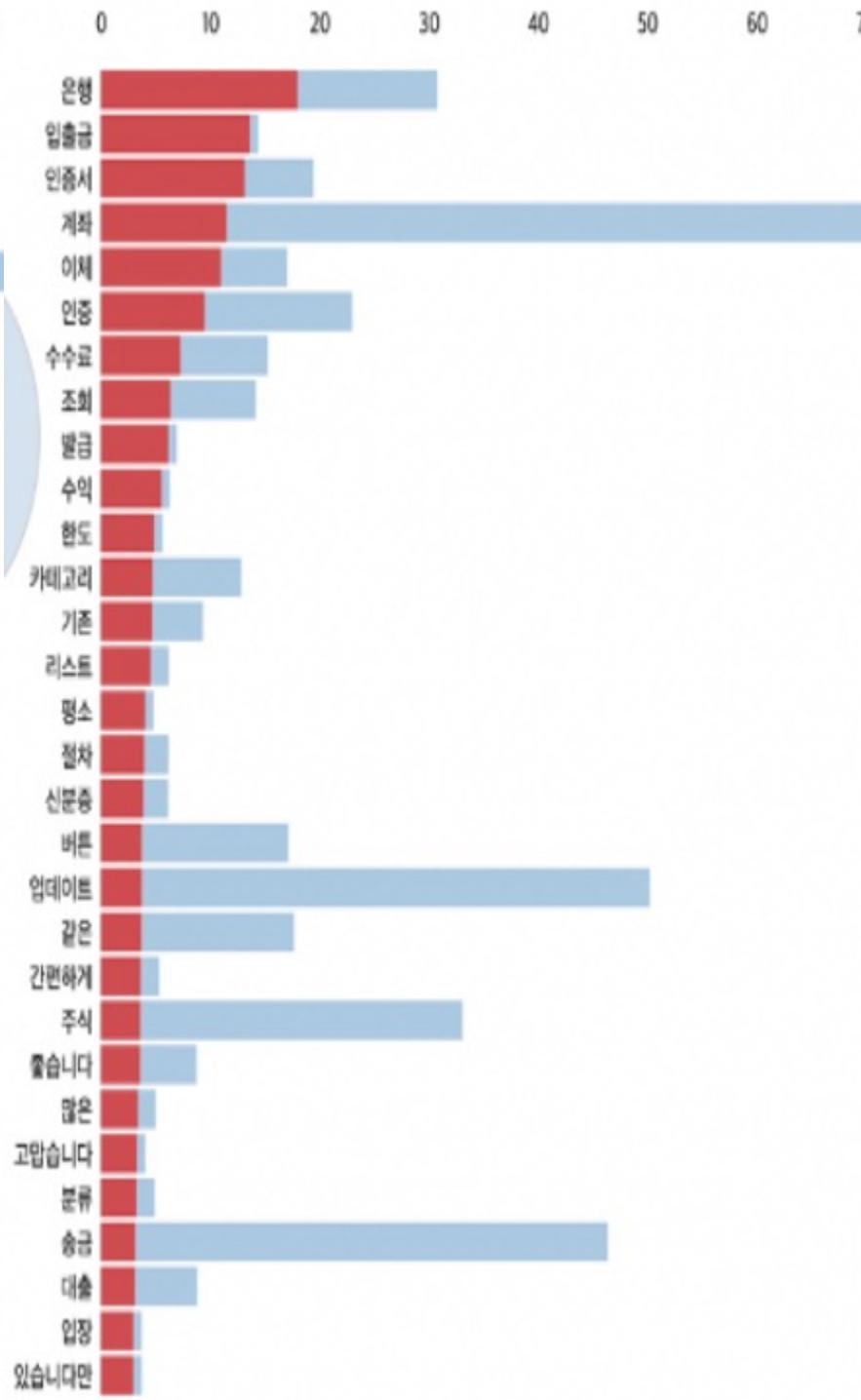
Topic 1

Top-30 Most Relevant Terms for Topic 1 (17.7% of tokens)



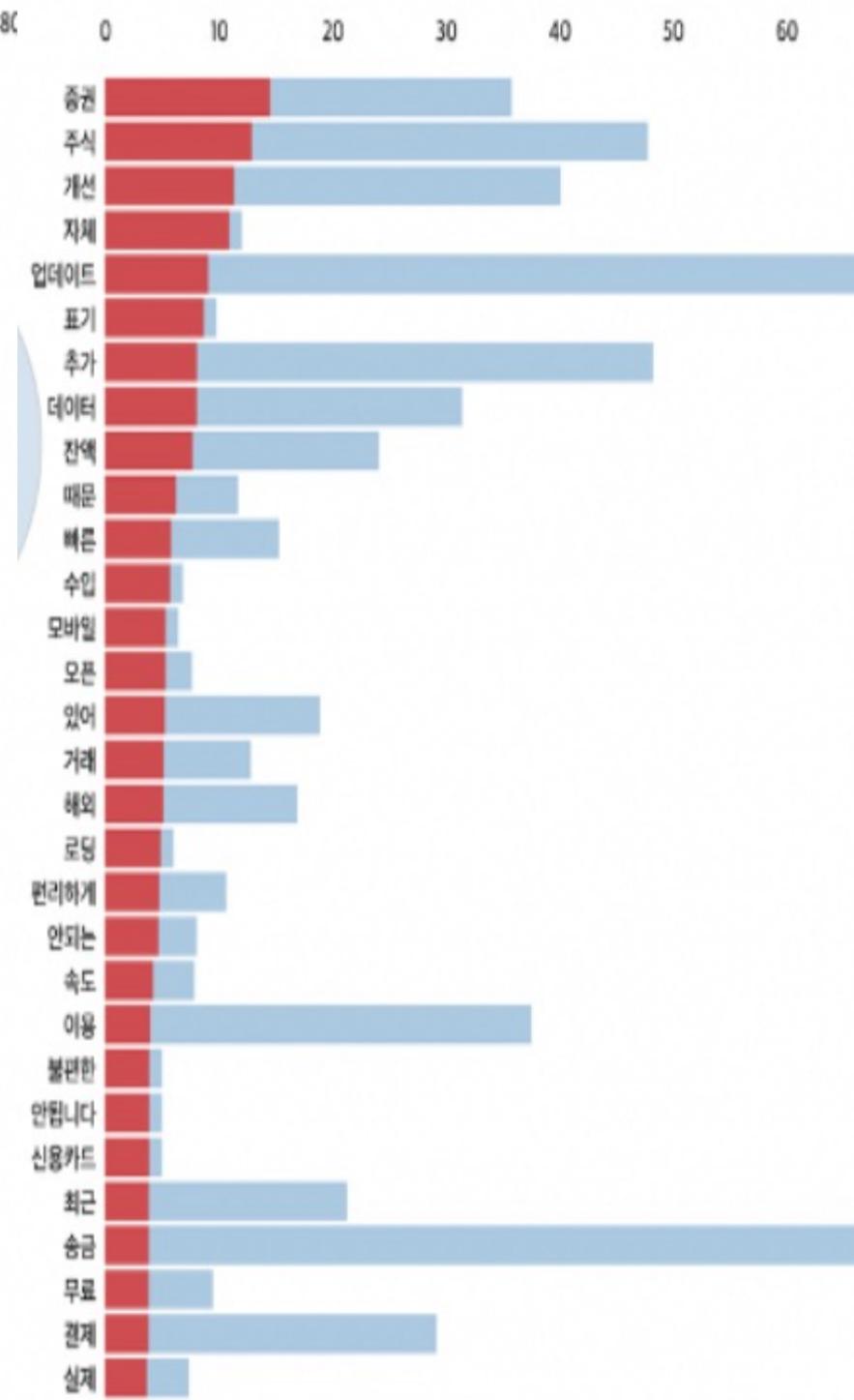
Topic 4

Top-30 Most Relevant Terms for Topic 4 (11.3% of tokens)



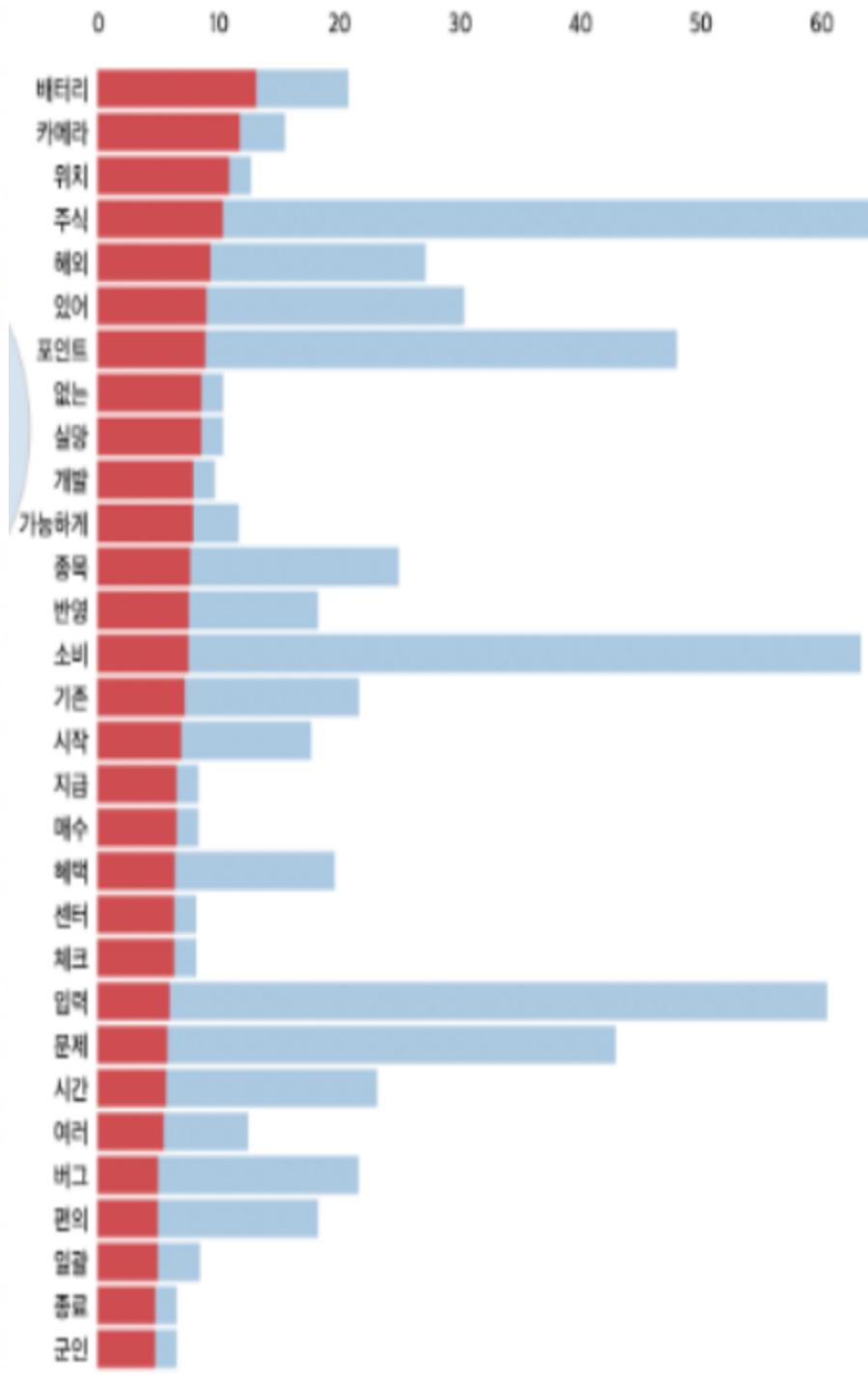
Topic 8

Top-30 Most Relevant Terms for Topic 8 (10.6% of tokens)



Topic 5

Top-30 Most Relevant Terms for Topic 5 (8.2% of tokens)



Topic 1

▶ 토스의 카드조회기능

- 이용, 카드, 연결
- 카드 연결 시 오류가 있음, 사용 금액이 앱에 즉각 반영되지 않음
- 대략 18% 의 비중을 차지하는 토픽, 토스의 카드조회기능에 대한 분석이 필요할 것으로 파악됨.
- 키워드만 보고선 어떤 토픽일지 한눈에 보이진 않는다.

Topic 5

▶ 의외의 부분에서의 불편함

- 배터리, 카메라, 핸드폰 위치 활성화
- 토스앱이 스마트폰 배터리 사용량에 미치는 영향이 매우 크다. 군인들은 카메라 사용이 불가하여 신분증 인증이 불가하다. 위치활성화 모드가 꺼지지 않는다.

Topic 8

▶ 증권,주식 부분의 개선

- 증권, 주식, 개선
- 불편한, 안됩니다
- 증권 주식부분의 개선이 필요함으로 파악됨.

Topic 4

▶ 토스의 입출금 기능 (기본적인 은행업무)

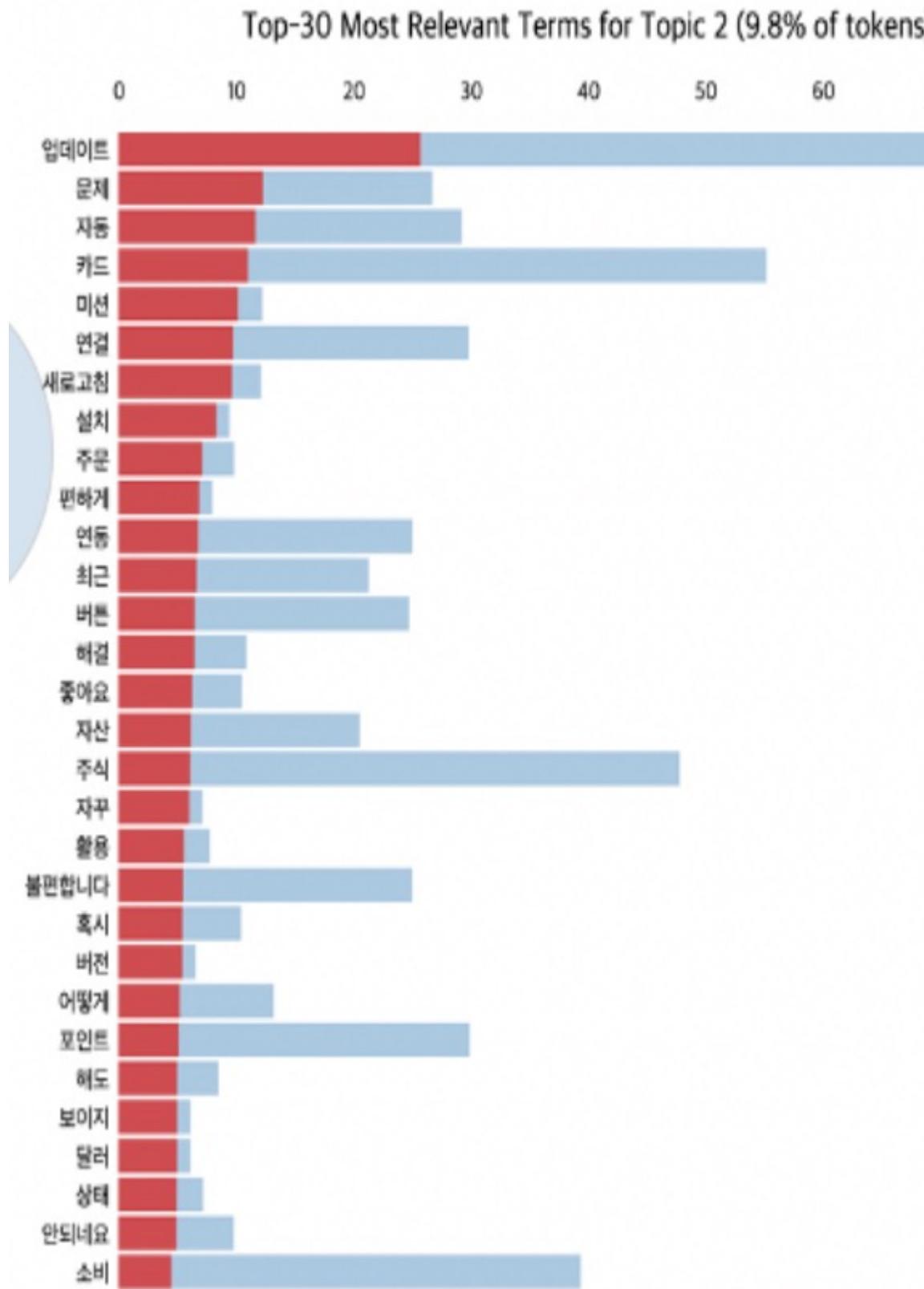
- 은행, 입출금, 인증서
- 한때 문제가 되었던 토스 인증서 인증문제.
- 그럼에도 수수료, 이체, 좋습니다, 간편하게라는 키워드가 보임
=>토스의 입출금기능이 호평 받고 있음



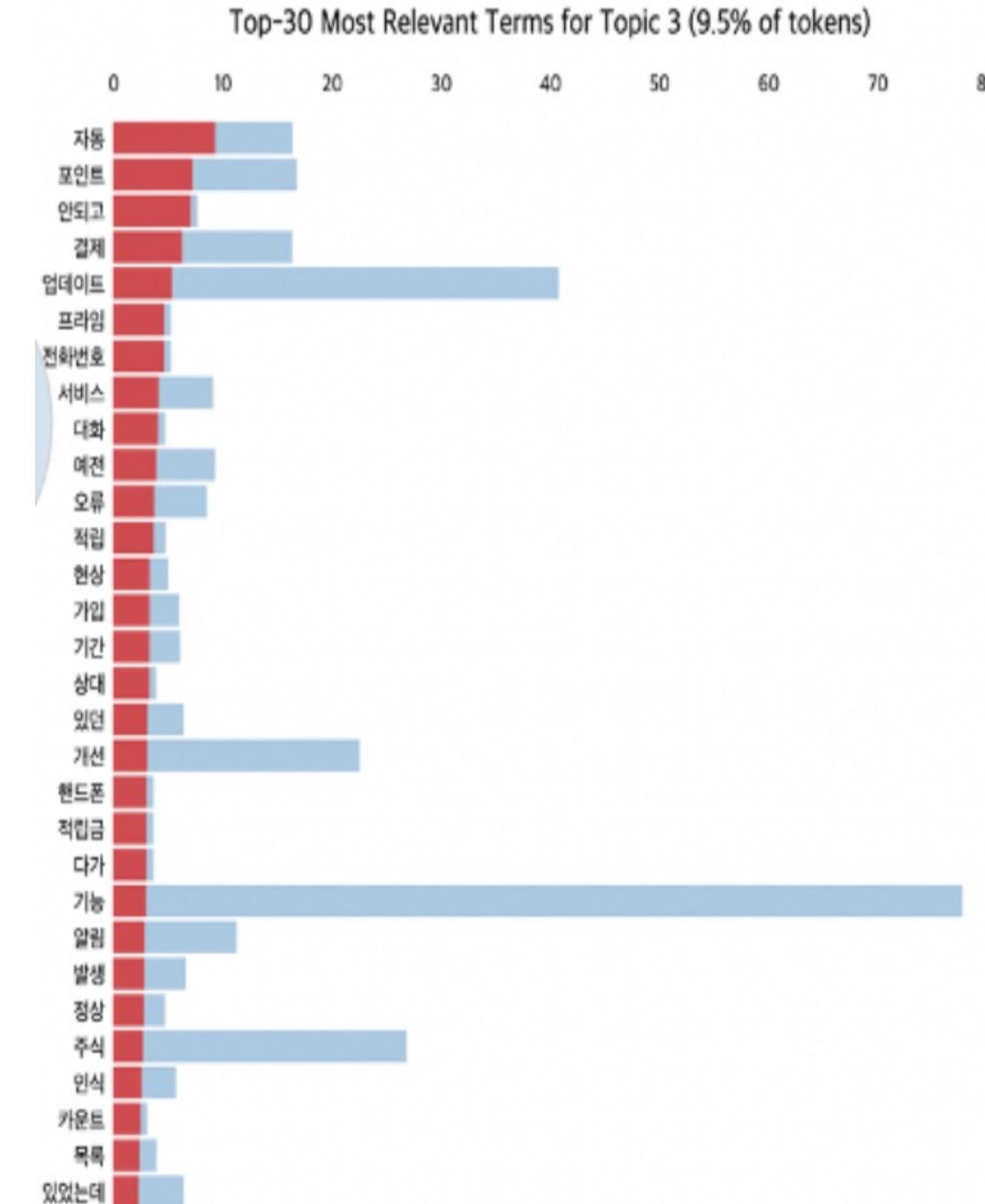
결론

업데이트

Topic 2



Topic 3



Topic 2

▶ 어플 업데이트 이슈

- 업데이트, 문제, 자동
- 업데이트 후 토스이용에 불편함이 더 많아졌다는 평이 업데이트 관련 리뷰에서 지배적이다.

Topic 3

▶ 업데이트 후 토스 포인트

- 자동, 포인트, 안되고
- 업데이트 후 포인트 적립에서 오류 발생
- 포인트 지급은 토스의 신박한 기능 중 하나. 보완을 통해 차별화 전략을 펼쳐야 할 것으로 분석됨



Pros

-> 신박하고 다양한 기능

ex) 토스만보기

Cons

-> 간단한 금융업무, 업데이트, 예상치 못한 곳에서의 불편함

**“토픽 모델링 분석 결과를 바탕으로
그들만의 장점을 살리고
보완해야 할 점을 보완한다면
더 성공적인 어플이 될 수 있을 것으로 판단됨”**

