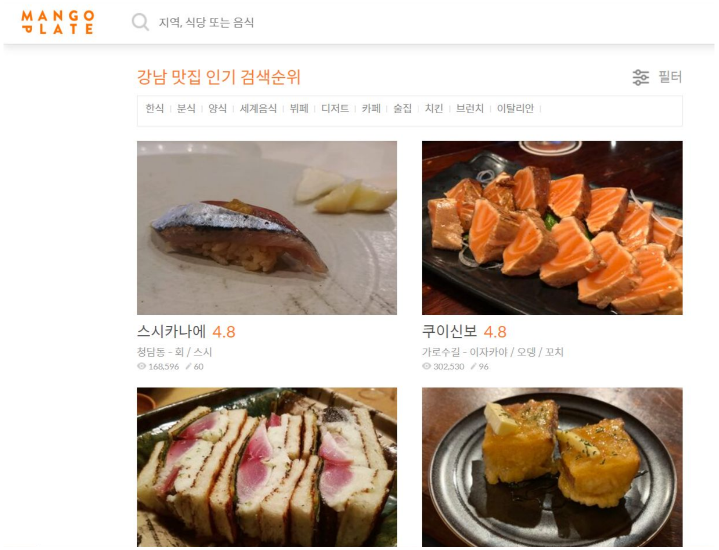
Project Presentation 2

18102082 Su Kyoung Oh
18102092 Won Ryeol Jeong
16102284 Sung Ho Lee

Contents

Crawling Result


MANGO PLATE

지역, 식당 또는 음식

강남 맛집 인기 검색순위

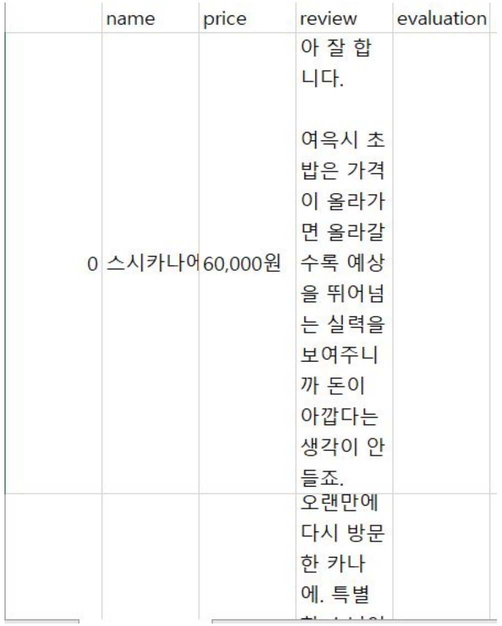
필터

한식 | 분식 | 양식 | 세계음식 | 뷔페 | 디저트 | 카페 | 술집 | 치킨 | 브런치 | 이탈리안 |

스시카나에 4.8
청담동 - 회 / 스시
168,596 60

쿠이신보 4.8
가로수길 - 이자카야 / 오뎅 / 꼬치
302,530 96

Crawling



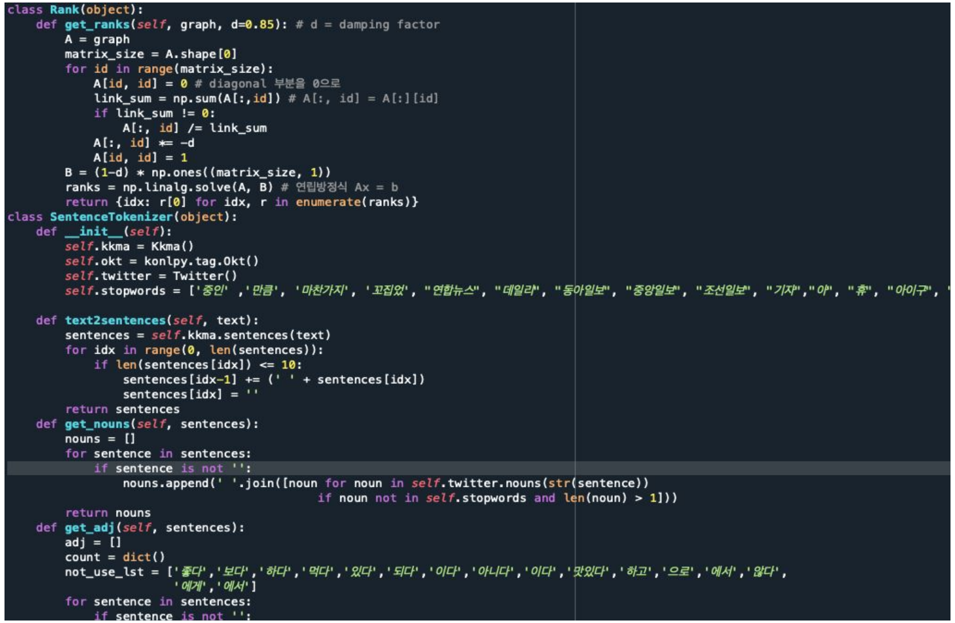
	name	price	review	evaluation
0	스시카나에	60,000원	아 잘 합니다. 여읏시 초밥은 가격이 올라가면 올라갈수록 예상을 뛰어넘는 실력을 보여주니까 돈이 아깝다는 생각이 안 들죠.	
			오랜만에 다시 방문한 카나에. 특별	

<GangnamGu Restaurant Review>

TextRank Algorithm

Text-rank algorithm was introduced for preprocessing of sentimental analysis.

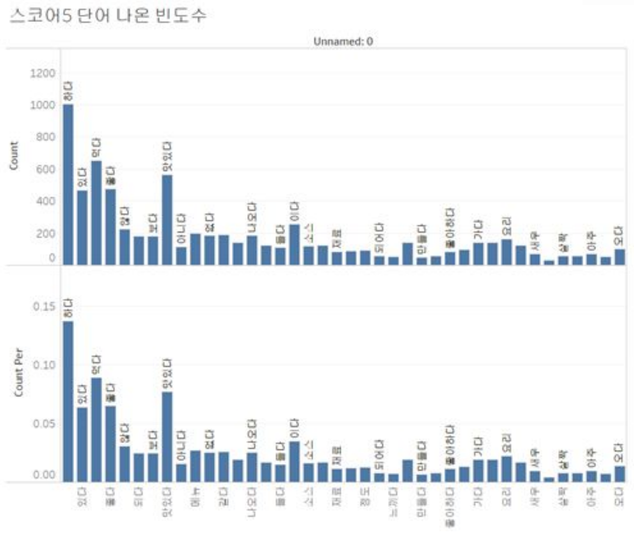
We tried to tokenize the reviews and classify the parts, and then applied the method of weighting the words such as nouns, adjectives, verbs, etc., which are expected to have a significant impact on the ratings according to their relative importance.



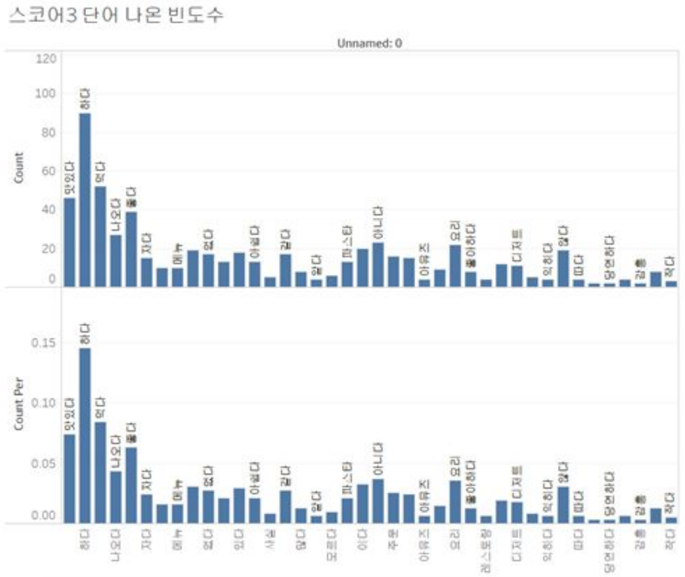
```
class Rank(object):
    def get_ranks(self, graph, d=0.85): # d = damping factor
        A = graph
        matrix_size = A.shape[0]
        for id in range(matrix_size):
            A[id, id] = 0 # diagonal 부분을 0으로
            link_sum = np.sum(A[:,id]) # A[:, id] = A[:][id]
            if link_sum != 0:
                A[:, id] /= link_sum
            A[:, id] *= -d
            A[id, id] = 1
        B = (1-d) * np.ones((matrix_size, 1))
        ranks = np.linalg.solve(A, B) # 연립방정식 Ax = b
        return {idx: r[0] for idx, r in enumerate(ranks)}
class SentenceTokenizer(object):
    def __init__(self):
        self.kkma = Kkma()
        self.okt = konlpy.tag.Okt()
        self.twitter = Twitter()
        self.stopwords = ['중인' ,'만큼', '마찬가지', '꼬집었', "연합뉴스", "데일리", "동아일보", "중앙일보", "조선일보", "기자","아", "휴", "아이구", '

    def text2sentences(self, text):
        sentences = self.kkma.sentences(text)
        for idx in range(0, len(sentences)):
            if len(sentences[idx]) <= 10:
                sentences[idx-1] += (' ' + sentences[idx])
                sentences[idx] = ''
        return sentences
    def get_nouns(self, sentences):
        nouns = []
        for sentence in sentences:
            if sentence is not '':
                nouns.append(' '.join([noun for noun in self.twitter.nouns(str(sentence))
                                       if noun not in self.stopwords and len(noun) > 1]))
        return nouns
    def get_adj(self, sentences):
        adj = []
        count = dict()
        not_use_lst = ['좋다','보다','하다','먹다','있다','되다','이다','아니다','이다','맛있다','하고','으로','에서','않다',
                       '에게','에서']
        for sentence in sentences:
            if sentence is not '':
```

Data Visualization

(※The graph`s order is descending from the left to the [‘rank’])



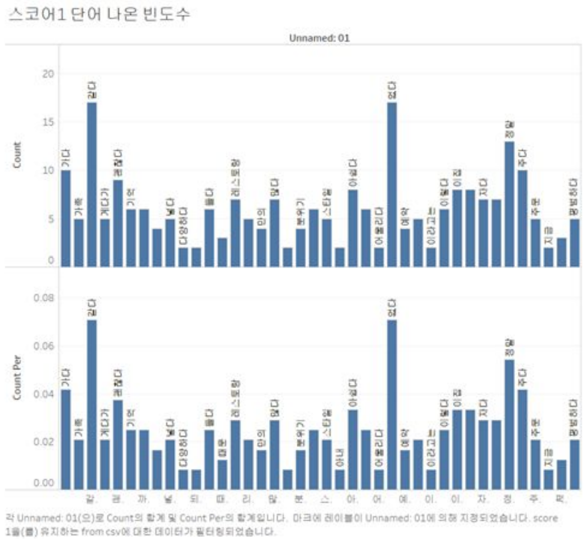
Frequency of word appearance of text rank algorithm results for 'score 5'



Frequency of word appearance of text rank algorithm results for 'score 3'

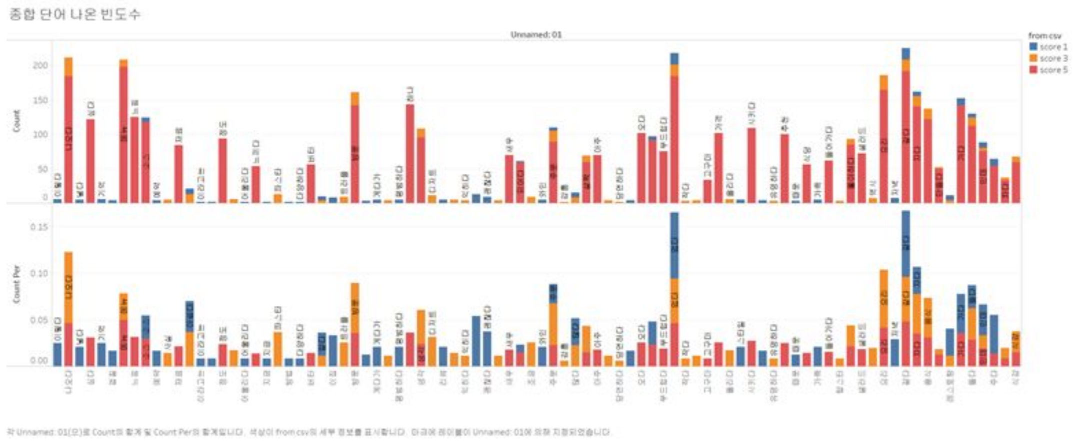
Data Visualization

(※The graph`s order is descending from the left to the [‘rank’])


스코어1 단어 나온 빈도수

각 Unnamed: 01(으)로 Count의 합계 및 Count Per의 합계입니다. 마크에 레이블이 Unnamed: 01에 의해 지정되었습니다. score 1을(를) 유지하는 from csv에 대한 데이터가 필터링되었습니다.

Frequency of word appearance of text rank algorithm results for 'score 1'

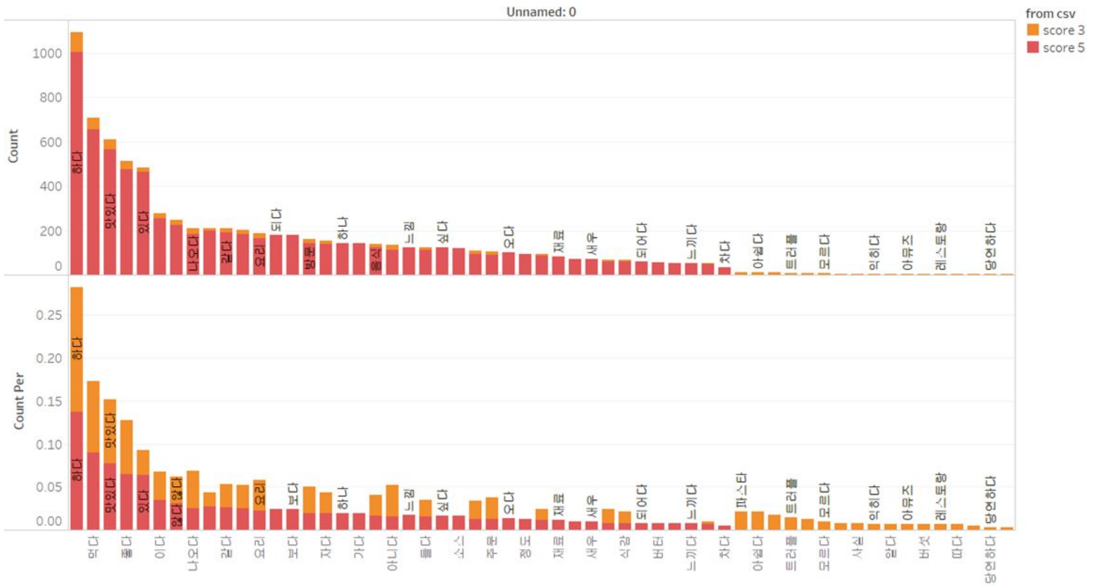

종합 단어 나온 빈도수

각 Unnamed: 01(으)로 Count의 합계 및 Count Per의 합계입니다. 색상이 from csv의 세부 정보를 표시합니다. 마크에 레이블이 Unnamed: 01에 의해 지정되었습니다.

Frequency of word appearance of text rank algorithm results for whole score class

Data Visualization

스코어3&5 단어 나온 빈도수

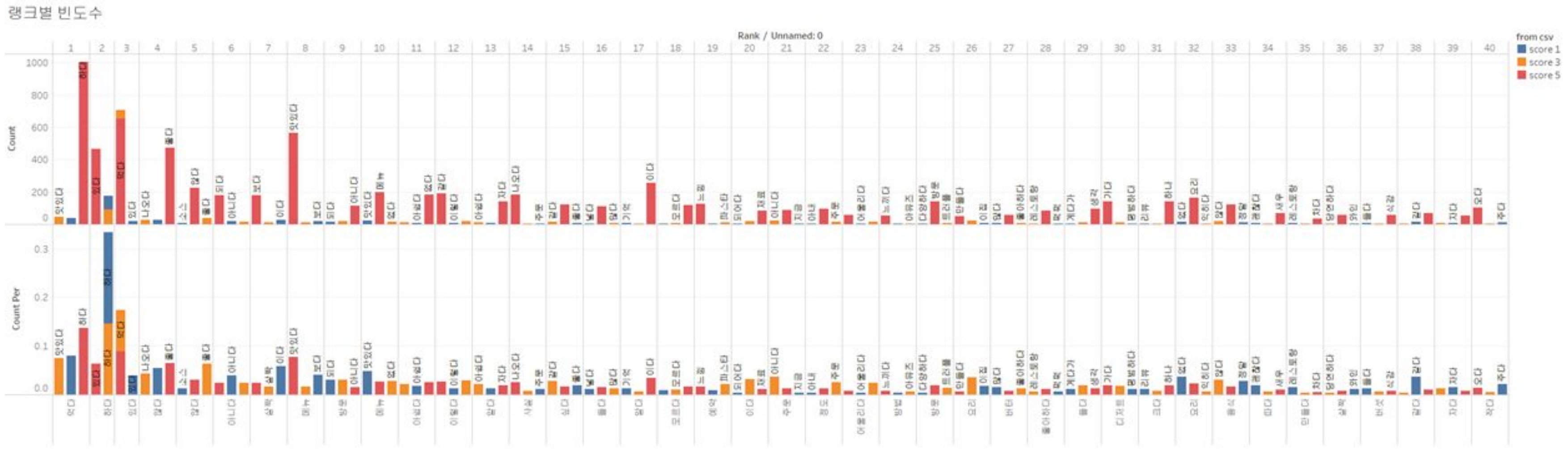


각 Unnamed: 0(으)로 Count의 합계 및 Count Per의 합계입니다. 색상이 from csv의 세부 정보를 표시합니다. 마크에 레이블이 Unnamed: 0에 의해 지정되었습니다. score 3 및 score 5을(를) 유지하는 from csv에 대한 뷰가 필터링되었습니다.

From earlier visualizations, we can clearly see a big difference between negative reviews (score 1) and others.

Therefore, we visualized scores 3 and 5 together as they are similar

Data Visualization



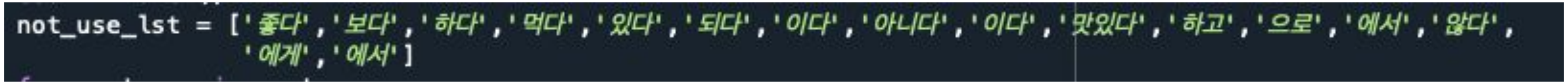
Rank별로 구분된 각 Unnamed: 0에 대해 Count의 합계 및 Count Per의 합계입니다. 색상은 from csv의 세부 정보를 표시합니다. 마크에 레이블이 Unnamed: 0에 의해 지정되었습니다. score 1, score 3 및 score 5을(를) 유지하는 from csv에 대한 뷰가 필터링되었습니다.

Frequency of words between each score class by ‘rank’

Data Visualization

As a result of applying the text ranking algorithm without any preprocessing, words with significant insights tend to be buried because of meaningless data such as '하다', 좋다' and '이다'.

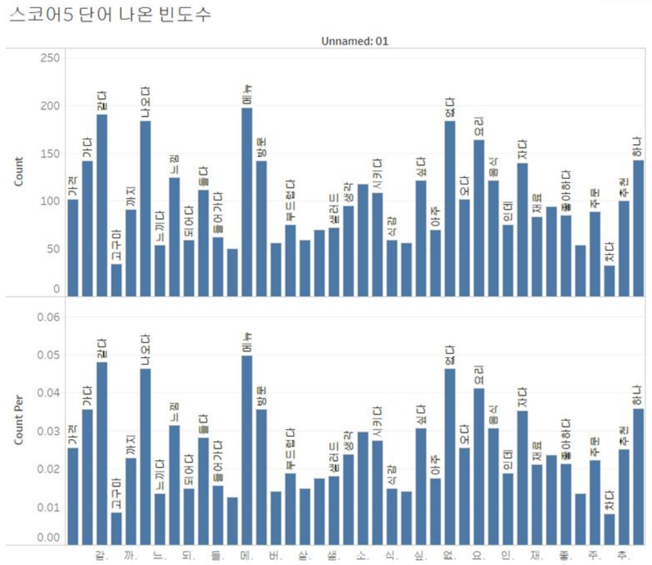
So we'll take these words out and reapply the text ranking algorithm again!



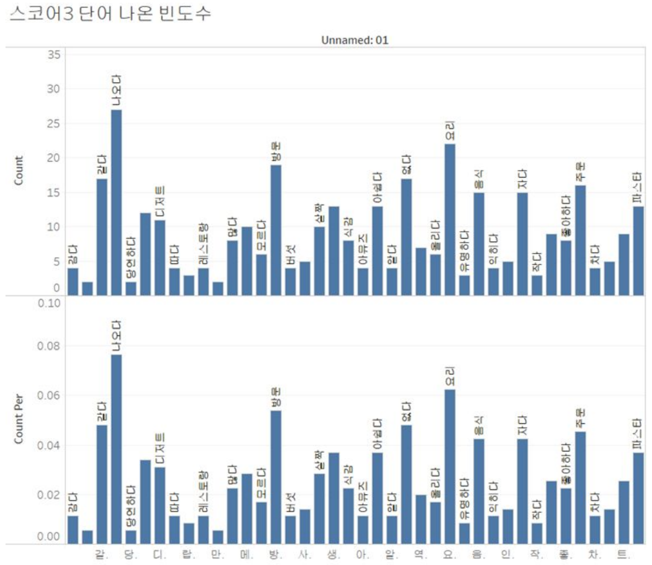
```
not_use_lst = ['좋다','보다','하다','먹다','있다','되다','이다','아니다','이다','맛있다','하고','으로','에서','않다',
               '에게','에서']
```

<Examples of Unnecessary words>

Data Visualization (※The graph`s order is descending from the left to the [‘rank’])

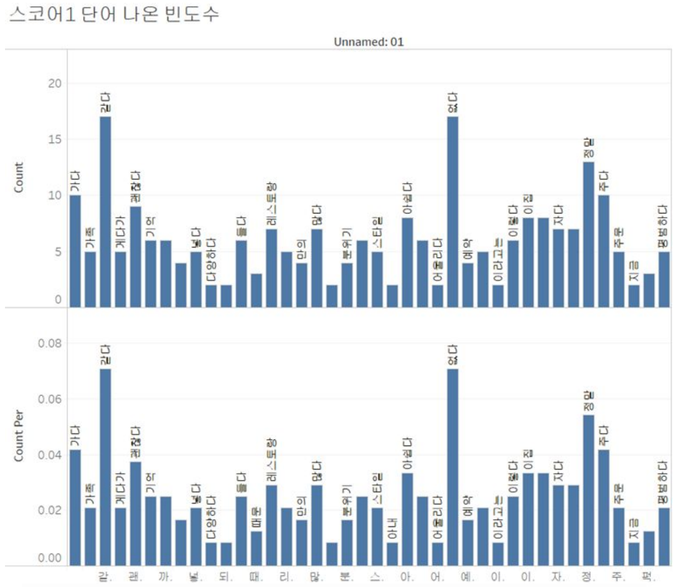


Frequency of word appearance of text rank algorithm results for 'score 5'

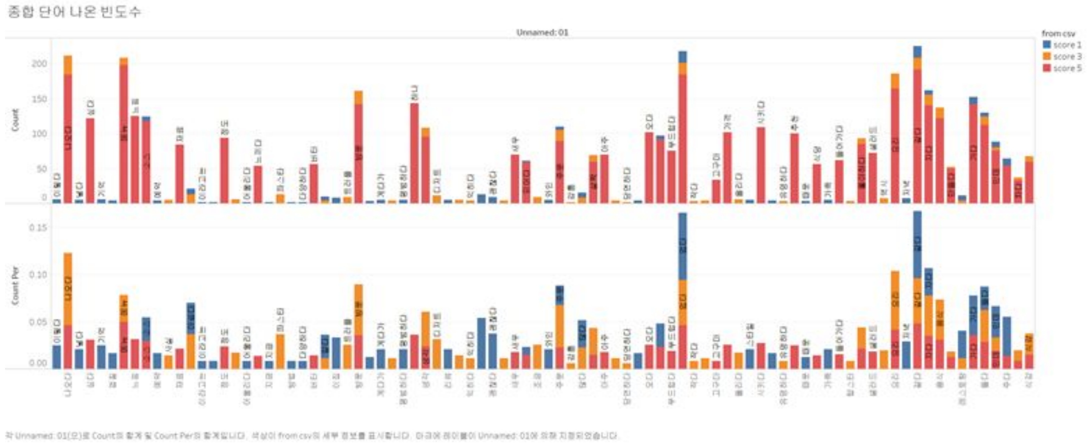


Frequency of word appearance of text rank algorithm results for 'score 3'

Data Visualization



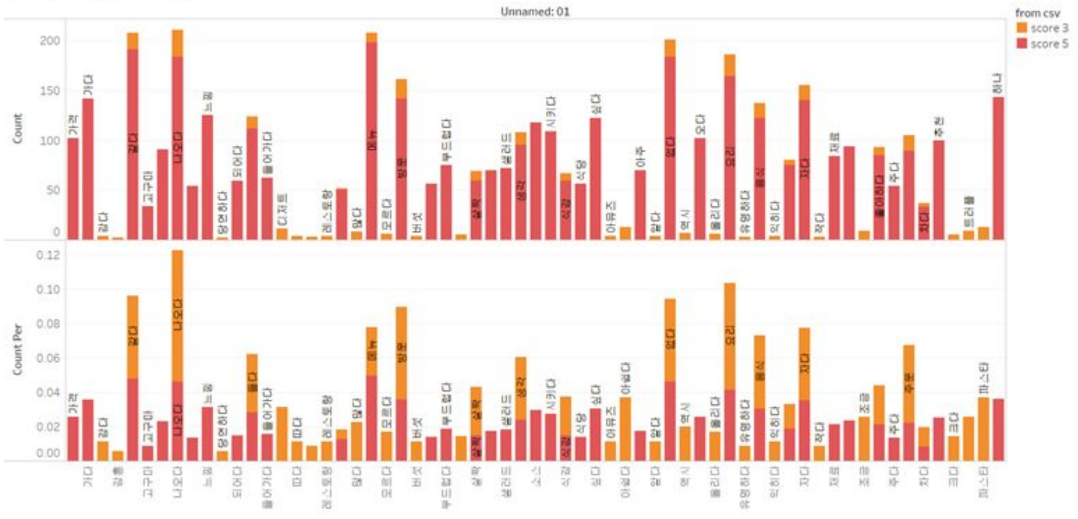
Frequency of word appearance of text rank algorithm results for 'score 1'



Frequency of word appearance of text rank algorithm results for whole score class

Data Visualization

스코어3&5 단어 나온 빈도수

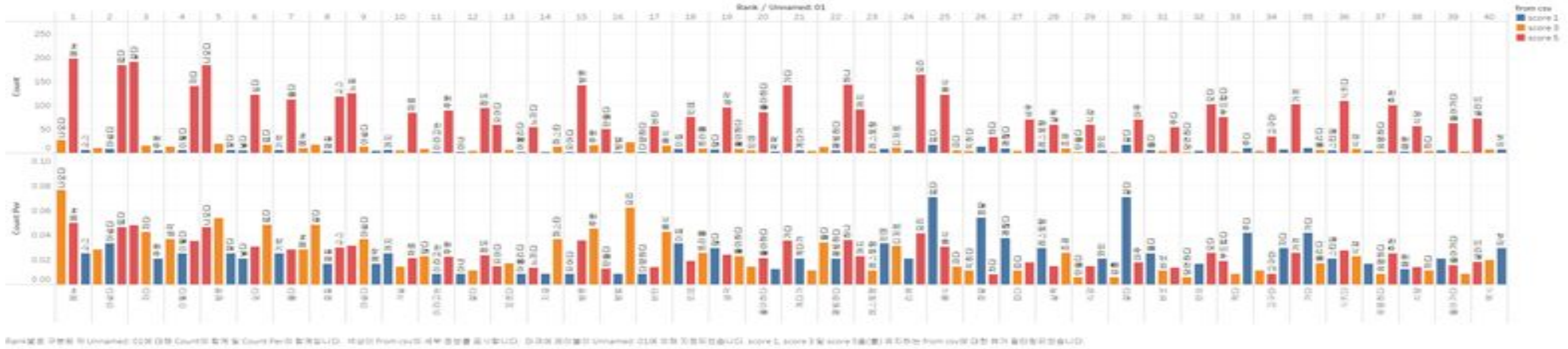


각 Unnamed: 01(으)로 Count의 합계 및 Count Per의 합계입니다. 색상이 from csv의 세부 정보를 표시합니다. 마크에 레이블이 Unnamed: 01에 의해 지정되었습니다. score 3 및 score 5을(를) 유지하는 from csv에 대한 뷰가 필터링되었습니다.

From earlier visualizations, we can compare each class of frequency of words.

However, after preprocessing, there is difference between ‘score 3’ and ‘score 5’, unlike the previous visualization.

Data Visualization



Frequency of words between each score class by ‘rank’

Feedback & Plan

- EDA on Korean reviews for sentiment analysis.
- As a result of data collection, there is a difference scale of data between the classes, so we will proceed with under-sampling or over-sampling to resolve this.
- From now on, we will try the same process for English reviews.
- When this is done, we will make several columns like the dummy data.
- Based on this preprocessing, the model will be built to find new insights by obtaining the necessary cut-off and coefficient to go to the corresponding review rate class.

THANKS
FOR
LISTENING