키워드 분석

한글 폰트 설정

- 실행 후 런타임 재시작 필요

```
[1] import matplotlib as mpl
    import matplotlib.pyplot as plt
    plt.style.use('seaborn-white')

    %config InlineBackend.figure_format='retina'

    !apt -qq -y install fonts-nanum

    import matplotlib.font_manager as fm
    fontpath= '/usr/share/fonts/truetype/nanum/NanumBarunGothic.ttf'
    font=fm.FontProperties(fname=fontpath,size=10)
    plt.rc('font',family='NanumBarunGothic')
    mpl.font_manager._rebuild()
```

한국어 자연어 처리 konlpy와 형태소 분석기 MeCab 설치

- https://raw.githubusercontent.com/konlpy/konlpy/master/scripts/mecab.sh

```
2분
    !set -x \
    && pip install konlpy \
    && curl -s https://raw.githubusercontent.com/konlpy/konlpy/master/scripts/mecab.sh | bash -x
```

네이버 영화 리뷰 데이터

- 키워드 분석에는 네이버 영화 리뷰 데이터를 사용
- 데이터 다운로드: https://raw.githubusercontent.com/e9t/nsmc/master/ratings.txt

[42]
```
import urllib.request

raw=urllib.request.urlopen('https://raw.githubusercontent.com/e9t/nsmc/master/ratings.txt').readlines()
print(raw[:5])
```

```
[b'id\tdocument\tlabel\n', b'8112052\t\xec\x96\xb4\xeb\xa6\xb4\xeb\x95\x8c\xeb\xb3\xb4\xea\xb3\xa0 \xec\xa7\x80\xea\xb8\x88\xeb\x8
```

[43]
```
raw = [x.decode() for x in raw[1:]]
print(raw[:5])

reviews=[]
for i in raw:
  reviews.append(i.split('\t')[1])

print(reviews[:5])
```

```
['8112052\t어릴때보고 지금다시봐도 재밌어요ㅋㅋ\t1\n', '8132799\t디자인을 배우는 학생으로, 외국디자이너와 그들이 일군 전통을 통해 발전해가는 문화산업이 부러웠는데.
['어릴때보고 지금다시봐도 재밌어요ㅋㅋ', '디자인을 배우는 학생으로, 외국디자이너와 그들이 일군 전통을 통해 발전해가는 문화산업이 부러웠는데. 사실 우리나라에서도 그 어려운
```

형태소 분석을 이용한 명사 추출

- 형태소 분석기 Mecab으로 명사만 추출

[8] 21초

```
from konlpy.tag import Mecab
tagger = Mecab()

nouns = []
for review in reviews:
  for noun in tagger.nouns(review):
    nouns.append(noun)

nouns[:10]
```

```
['때', '디자인', '학생', '외국', '디자이너', '그', '전통', '발전', '문화', '산업']
```

불용어(Stopwords) 사전 만들기

- 형태소 분석을 통해 조사, 접속사 등의 제거 가능
- 하지만 한국어는 명사에서도 상당히 많은 불필요한 단어들이 포함
- 사용자가 직접 불용어 사전을 유지하면서 불필요한 단어 제거 필요
- 불용어 예: `전 난 일 걸 뭐 줄 만 건 분 개 끝 잼 이거 번 중 듯 때 게 내 말 나 수 거 점 것`
- 빈도가 너무 커서 분석에 방해되는 단어도 제거 필요 (예: `영화`)

0초

```
[12] stop_words = '영화 전 난 일 걸 뭐 줄 말 건 분 개 끝 잼 이거 번 중 듯 때 게 내 말 나 수 거 점 것'
     stop_words = stop_words.split(' ')
     print(stop_words)
```

```
['영화', '전', '난', '일', '걸', '뭐', '줄', '말', '건', '분', '개', '끝', '잼', '이거', '번', '중', '듯', '때', '게', '내', '말', '나', '수', '거', '점', '것']
```

불용어를 제외하여 형태소 분석 수행

- 한글 텍스트에 대해서 형태소 분석 수행
- 분석으로 추출하는 명사 중에서 불용어에 포함되지 않은 텍스트만 추출하여 저장

```
[13] nouns = []
     for review in reviews:
       for noun in tagger.nouns(review):
         if noun not in stop_words:
           nouns.append(noun)

     nouns[:10]
```

21 초

```
['디자인', '학생', '외국', '디자이너', '그', '전통', '발전', '문화', '산업', '우리']
```

단어 빈도수 측정

- 단어 빈도수 측정에는 `collections` 라이브러리의 `Counter` 함수를 이용
- `collections` 라이브러리는 내장 라이브러리로 별도 설치가 필요없음
- `counter`를 이용하면 각 단어와 각 단어의 빈도 수를 딕셔너리로 편리하게 생성 가능

0초

```
from collections import Counter
nouns_counter=Counter(nouns)
top_nouns = dict(nouns_counter.most_common(50))
top_nouns
```

```
{'연기': 9175,
 '최고': 8813,
 '평점': 8514,
 '스토리': 7163,
 '생각': 6943,
 '드라마': 6896,
 '사람': 6742,
 '감동': 6489,
 '배우': 5893,
 '내용': 5731,
 '감독': 5629,
 '재미': 5479,
 '시간': 5320,
 '년': 4936,
 '사랑': 4741,
 '쓰레기': 4585,
 '작품': 3985,
 '하나': 3923,
 '정도': 3656,
 '이건': 3650,
 '마지막': 3647,
 '액션': 3568,
 '기대': 3465,
 '장면': 3262,
 '만': 3171,
 '이게': 3046,
 '편': 3044,
 '최악': 3019,
 '돈': 2980,
 '이야기': 2947,
 '이해': 2745,
 '애': 2730,
 '명작': 2685,
 '여자': 2678,
 '이상': 2676,
 '처음': 2673,
 '한국': 2640,
 '주인공': 2553,
 '우리': 2531,
 '연출': 2376,
 '때문': 2371,
 '기억': 2364,
 '현실': 2193,
 '마음': 2128,
 '굿': 2110,
 '남자': 2078,
 '결말': 2066,
 '인생': 2060,
 '공포': 2048,
 '전개': 2035}
```

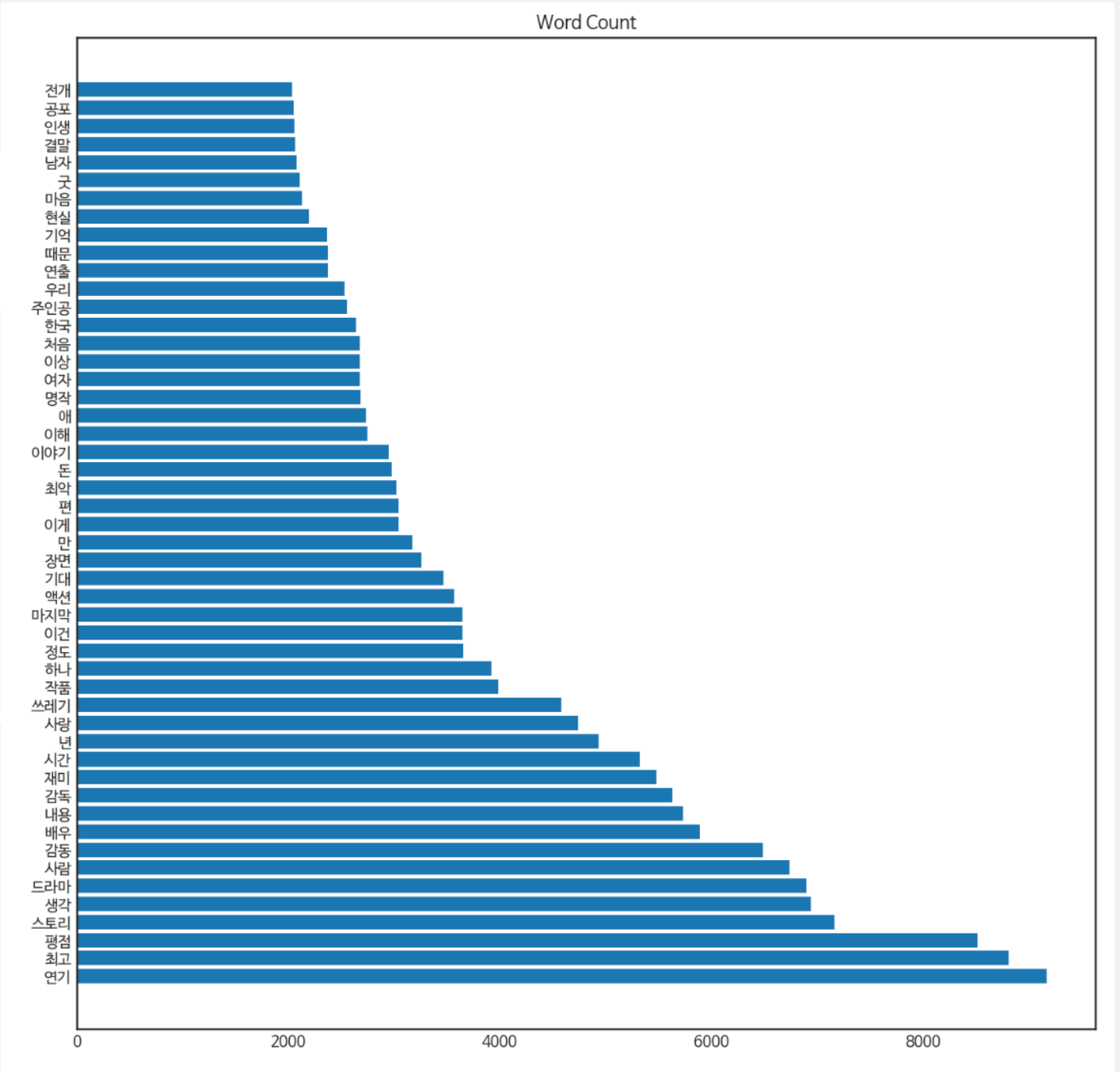
단어 빈도 시각화

1초

```
import numpy as np

y_pos = np.arange(len(top_nouns))

plt.figure(figsize=(12,12))
plt.barh(y_pos,top_nouns.values())
plt.title('Word Count')
plt.yticks(y_pos,top_nouns.keys())
plt.show()
```



워드클라우드(WordCloud)

- 텍스트에 담겨있는 여러 형태소들의 등장 빈도를 가장 직관적으로 시각화하는 방법
- 텍스트에 등장하는 단어를 그 등장 빈도에 따라 서로 크기가 다르게 구름 형태로 표현함으로써, 단어의 빈도 수를 한번에 알 수 있음
- 최근에 많은 서비스들이 어떤 핵심어가 많이 등장했는가를 워드클라우드 형식으로 시각화
- 빈도 수만을 시각적으로 표현한 것이기 때문에, 단어들 사이의 연관성이나 의미 구조 등을 분석하는 데는 한계가 있음

- 파이썬에서 워드 클라우드를 시각화하기 위해 `matplotlib` 라이브러리와 `WordCloud` 라이브러리를 `import` 해서 사용
- `WordCloud` 라이브러리는 `pip install wordcloud` 명령어를 통해 설치 필요

```
!pip install wordcloud
```

- WordCloud를 이용해 객체를 생성해주고, `generate_from_frequencies()` 함수로 빈도 수에 따라 워드클라우드 생성

```
from wordcloud import WordCloud

wc=WordCloud(background_color='white', font_path='./font/NanumBarunGothic.ttf')
wc.generate_from_frequencies(top_nouns)
```

+ 코드 + 텍스트

- 워드클라우드를 시각화할 때는 이미지 시각화 함수인 `imshow()` 함수를 사용해야 함

```
[25] figure = plt.figure(figsize=(12,12))
     ax=figure.add_subplot(1,1,1)
     ax.axis('off')
     ax.imshow(wc)
     plt.show()
```

squarify 트리맵 시각화

- `squarify`는 트리맵 생성을 지원해주는 파이썬 라이브러리
- `squarify` 라이브러리를 이용해 키워드와 키워드 빈도 수를 트리맵으로 나타냄

```
[14] !pip install squarify
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting squarify
  Downloading squarify-0.4.3-py3-none-any.whl (4.3 kB)
Installing collected packages: squarify
Successfully installed squarify-0.4.3
```

```
import squarify

norm = mpl.colors.Normalize(vmin=min(top_nouns.values()),
                            vmax=max(top_nouns.values()))

colors=[mpl.cm.Blues(norm(value)) for value in top_nouns.values()]


squarify.plot(label=top_nouns.keys(),
              sizes=top_nouns.values(),
              color=colors,
              alpha=.7);
```