

<7장> 텍스트 분석 및 시각화

학습 목표

- 자연어 처리를 위한 전처리 기법이 무엇인지 설명한다.
- 자연어 전처리에 필요한 라이브러리가 무엇인지 알고 설치한다.
- 자연어 전처리 기법을 학습하고, 실제 문제에 적용한다.
- 전처리 과정을 통해 얻은 결과를 워드클라우드로 시각화한다.

목차

01 텍스트 분석 이해

02 영문분석: NLTK(Natural Language Toolkit)

03 워드클라우드

04 한글분석:KoNLPy(Korean Natural Language Processing in Python)

05 자연어 처리 실습

01

텍스트 분석 이해

1. 텍스트 분석 이해

■ 텍스트 분석(Text Analysis)

- 텍스트 분석(Text Analysis) 또는 텍스트 마이닝(Text Mining)이라고 함
- 자연어로 작성된 비정형 데이터를 정형화된 방식으로 분석하여 유의미한 정보를 추출하는 과정
- 자연어 처리와 데이터 마이닝이 결합하여 발전된 분야로 비정형 텍스트 데이터에서 정보를 추출하여 분석하는 방법

■ 목적

- 텍스트에서 주요 키워드를 추출하고
- 감정, 주제, 패턴, 의미를 파악하여
- 의사 결정, 자동 분류, 추천, 의견 분석 등에 활용

1. 텍스트 분석 이해

■ 텍스트 분석의 주요 단계

- 텍스트 수집
 - 뉴스 기사, 소셜 미디어, 설문 응답, 이메일 등에서 텍스트를 수집
- 전처리 (Preprocessing) :
 - 텍스트를 분석하기 쉽게 정리하는 단계
 - 소문자 변환 (lowercase)
 - 특수문자 제거
 - 불용어(stopwords) 제거
 - 토큰화(tokenization) - 문장을 단어로 분리
 - 표제어 추출(lemmatization) 또는 어간 추출(stemming)
- 탐색적 분석 (EDA: Exploratory Data Analysis)
 - 단어 빈도 분석, 시각화 워드클라우드 생성
 - 주요 단어/구 표현 확인
 - 텍스트 길이 분포 등 통계적 특성 분석
- 모델링
 - 감정 분석 (Sentiment Analysis): 긍정/부정 등 판단
 - 주제 모델링 (Topic Modeling): 숨겨진 주제를 자동으로 찾음 (ex. LDA)
 - 텍스트 분류 (Text Classification): 카테고리 자동 분류
 - 군집화 (Clustering) 분석
 - 정보 추출 (NER, 관계 추출 등)
- 평가 및 해석 (Evaluation & Interpretation)
 - 모델 성능 평가 (정확도, F1 점수, 혼동 행렬 등)
 - 결과 해석 및 시각화
 - 비즈니스 인사이트 도출
- 응용 및 배포 (Application & Deployment)
 - 분석 결과를 서비스에 적용
 - 대시보드, API 등 형태로 결과 배포

02

영문 분석 :NLTK

2. 영문분석:NLTK

■ NLTK(Natural Language Toolkit)

- 정의: 자연어 처리 및 텍스트 분석을 위한 파이썬 라이브러리
- 목적: 언어 데이터의 분석, 전처리, 이해 등을 손쉽게 수행
- 지원: 토큰화, 형태소 분석, 품사 태깅, 파싱, 의미 분석, 감정 분석 등 다양한 기능 제공

■ NLTK 주요기능

기능	설명	
토큰화(Tokenization)	문장을 단어 또는 문장 단위로 분리	
형태소 분석 (Stemming & Lemmatization)	단어의 어간 또는 원형을 추출	
품사 태깅 (POS Tagging)	각 단어에 품사 정보 부여	
불용어 제거 (Stopword Removal)	분석에 불필요한 단어 제거	
구문 분석 (Parsing)	문장의 문법 구조 분석	
어휘 자원 (Lexical Resources)	워드넷(WordNet) 등 풍부한 언어 데이터셋 포함	
감정 분석 (Sentiment Analysis)	텍스트에서 감정 파악 (NLTK 자체보다는 추가 라이브러리 연동 활용)	

2. 영문분석:NLTK

■ NLTK 설치

```
pip install nltk
```

■ NLTK 주요 활용 분야

- 텍스트 전처리 자동화
- 뉴스, 리뷰 분석
- 챗봇 구축 전처리
- 교육 및 연구 목적의 언어 실험

■ NLTK 특징 요약

- 장점: 사용법이 쉽고 학습 자료가 풍부
- 단점: 대규모 텍스트 분석에서는 느릴 수 있으며, 최신 딥러닝 모델은 별도 라이브러리(BERT 등)가 필요

2. 영문분석:NLTK

■ NLTK 데이터 활용

- data 다운로드하여 설치

```
nltk.download('book', quiet=True)
```

```
nltk.corpus.gutenberg.fileids()
```

- 데이터 로드

```
emma_raw=nltk.corpus.gutenberg.raw('austen-emma.txt')  
emma_raw
```

2. 영문분석:NLTK

■ NLTK 토큰 생성

- 문장(sent) 토큰 생성

```
from nltk.tokenize import sent_tokenize  
print(sent_tokenize(emma_raw)[3])
```

- 단어(word)토큰 생성

```
emma_raw=nltk.corpus.gutenberg.raw('austen-emma.txt')  
emma_raw
```

- 정규식 토큰 생성

```
from nltk.tokenize import RegexpTokenizer  
regTokenizer=RegexpTokenizer("[\w]+")  
regTokenizer.tokenize(emma_raw[:1000])
```

2. 영문분석:NLTK

■ NLTK 형태소 분석기 활용(어간 추출)

- PorterStemmer : 보수적이고 일반적인 규칙기반 알고리즘 사용
- LancasterStemmer : 공격적이고 더 간단한 규칙기반 알고리즘 사용

```
from nltk.stem import PorterStemmer, LancasterStemmer
st1=PorterStemmer()
st2=LancasterStemmer()
words=['fly', 'flies', 'flying', 'flew', 'flown']

print("Porter Stemmer: ", [st1.stem(w) for w in words])
print("Lancaster Stemmer:", [st2.stem(w) for w in words])

list1=[]
for w in words:
    print(st1.stem(w), st2.stem(w))
    list1.append(st1.stem(w))
```

결과:

```
Porter Stemmer: ['fli', 'fli', 'fli', 'flew', 'flown']
Lancaster Stemmer: ['fly', 'fli', 'fly', 'flew', 'flown']
```

2. 영문분석:NLTK

■ NLTK 표제어 추출(lemmatization)

- WordNetLemmatize: 단어의 원형을 복원

```
from nltk.stem import WordNetLemmatizer  
nltk.download('omw-1.4')
```

```
lm=WordNetLemmatizer()  
v1=[lm.lemmatize(w,'v') for w in words]  
print(v1)  
  
list2=[]  
for w in words:  
    list2.append(lm.lemmatize(w,'v'))  
print(list2)
```

결과 :

```
[ ' fly ' , ' fly ' , ' fly ' , ' fly ' , ' fly ' ]  
[ ' fly ' , ' fly ' , ' fly ' , ' fly ' , ' fly ' ]
```

2. 영문분석:NLTK

■ NLTK 형태소 분석

▪ 품사 태깅

```
from nltk.tag import pos_tag
sentence=emma_raw[:1000]
tagged_list=pos_tag(regTokenizer.tokenize(sentence)) #(word_tokenize(sentence))
tagged_list
```

```
nltk.help.upenn_tagset('NNP') #품사 확인
```

결과

NNP: noun, proper, singular Motown Venneboerger
Czestochwa Ranzer Conchita Trumplane Christos
Oceanside Escobar Kreisler Sawyer Cougar Yvette Ervin
ODI Darryl CTCA Shannon A.K.C. Meltex Liverpool ...

결과

```
[('Emma', 'NN'),
 ('by', 'IN'),
 ('Jane', 'NNP'),
 ('Austen', 'NNP'),
 ('1816', 'CD'),
 ('VOLUME', 'NNP'),
 ('I', 'PRP'),
 ('CHAPTER', 'VBP'),
 ('I', 'PRP'),
 ('Emma', 'NNP'),
 ('Woodhouse', 'NNP'),
 ('handsome', 'VBD'),
 ('clever', 'NN'),
 ('and', 'CC'),
 ('rich', 'JJ'),
 ('with', 'IN'),
 ('a', 'DT'),
 ('comfortable', 'JJ'),
 ('home', 'NN'),
 ('and', 'CC'),
 ('happy', 'JJ')]
```

2. 영문분석:NLTK

■ NLTK 형태소 분석

■ 품사 태깅(품사태그 표기법)

태그	의미	예시
NNP	고유 명사	Emma, London
JJ	형용사	clever, young
VB	동사 원형	run, eat
VBD	과거 동사	ran, ate
NN	일반 명사	girl, apple
RB	부사	quickly

2. 영문분석:NLTK

■ NLTK 형태소 분석

■ 고유명사 및 명사 추출

```
noun_list=[t[0] for t in tagged_list if t[1]=='NNP']  
noun_list
```

■ 품사가 명사인 경우 word 그렇지 않는 경우 '-'로 표기

```
noun_list=[t[0] if t[1]=='NN' else '-' for t in tagged_list]  
noun_list
```

```
taglist=pos_tag(word_tokenize(emma_raw))  
nnp_list=[t[0] for t in taglist if t[1]=='NNP']  
nnp_list
```


2. 영문분석:NLTK

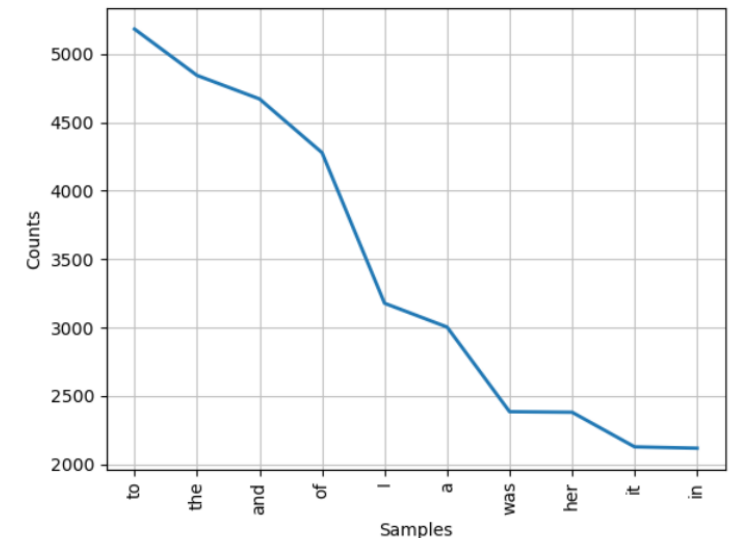
■ Text 객체

- 토큰화된 텍스트 데이터를 분석하고 탐색하기 위한 클래스
- 단어 리스트(토큰 리스트)를 감싸서, 다양한 자연어처리(NLP) 기능을 쉽게 사용할 수 있게 해주는 도구

- Text 객체 생성

```
from nltk import Text
text=Text(regTokenize.tokenize(emma_raw))
print(text)
```

```
import matplotlib.pyplot as plt
text.plot(10) #출현빈도가 가장 높은 단어 10개로 차트 작성
```



2. 영문분석:NLTK

■ Text 객체

■ 주요 메서드

메서드	설명
<code>text.concordance(word)</code>	특정 단어가 나온 문맥을 보여줍니다 (KWIC - keyword in context).
<code>text.similar(word)</code>	특정 단어가 사용된 문맥에 비슷하게 사용된 단어들을 보여줍니다.
<code>text.common_contexts(words)</code>	두 단어가 공통으로 사용된 문맥(예: bigrams)을 보여줍니다.
<code>text.count(word)</code>	단어의 등장 횟수를 셉니다.
<code>text.dispersion_plot([w1, w2, ...])</code>	단어들의 텍스트 내 분포를 시각화합니다.
<code>text.collocations()</code>	텍스트 내에서 자주 함께 등장하는 단어쌍(연어)을 찾아줍니다.

2. 영문분석:NLTK

■ Text 객체

- 주요 메서드 활용 예

```
emma_taglist=pos_tag(regTokenize.tokenize(emma_raw))  
emma_taglist
```

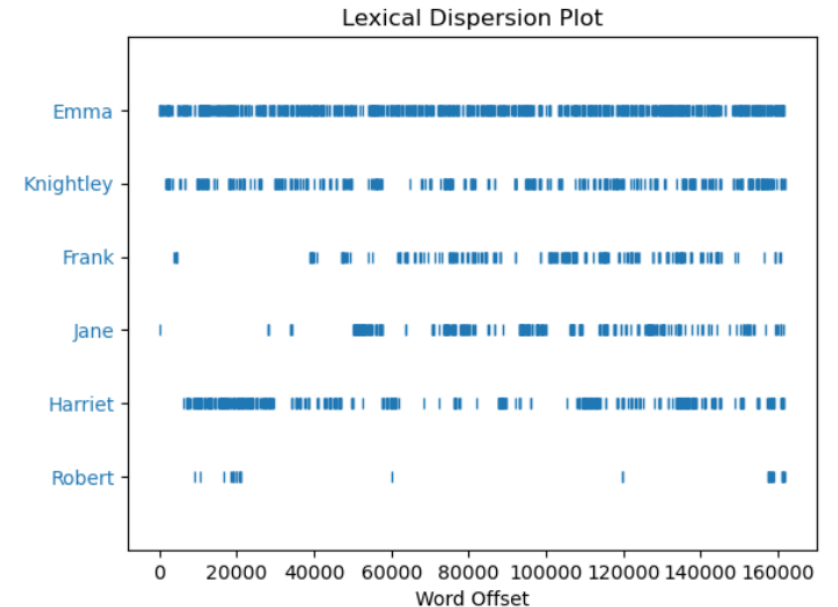
```
nnplist=[t[0] for t in emma_taglist if t[1]=='NNP']  
print(nnplist)
```

```
text.dispersion_plot(['Emma', 'Knightley', 'Frank',  
                      'Jane', 'Harriet', 'Robert'])
```

```
text.concordance("Robert")
```

Displaying 25 of 32 matches:

```
eye sufficiently acquainted with Mr Robert Martin His appearance was very neat  
he girl who could be gratified by a Robert Martin s riding about the country t  
to me I could not have visited Mrs Robert Martin of Abbey Mill Farm Now I am  
from a most unexceptionable quarter Robert Martin is the man Her visit to Abbe  
hear better sense from any one than Robert Martin He always speaks to the purp  
eaning of this Harriet Smith refuse Robert Martin madness if it is so but I ho
```



2. 영문분석:NLTK

■ FreqDist()

■ 토큰 빈도수 구하기

```
# 불용어(stopwords)제거
from nltk import FreqDist
stopwords=['Mr.','Mrs.','Miss','Mr','Mrs','Dear','A','No','Ah','Oh',
           'VOLUME','Woodhouse','Between','Sixteen']
emma_token=pos_tag(regTokenize.tokenize(emma_raw))
name_list=[t[0] for t in emma_token if t[1]=='NNP' and t[0] not in stopwords]
name_list
```

```
# 단어 출현빈도수 구하기
fd_name=FreqDist(name_list)
fd_name
```

```
print(fd_name.N()) #전체 단어의 수
print(fd_name['Emma']) #주어진 단어(Emma)의 출현 빈도
print(fd_name.freq('Emma')) #주어진 단어(Emma)의 출현빈도 확률
print(fd_name.most_common(10)) #출현빈도가 가장 높은 단어 10개 표시
```

2. 영문분석:NLTK

■ 실습

1. Emma_raw를 사용하여 정규표현식으로 Token을 생성하고 token에 품사태깅을 한다.
2. 품사 중 명사(NN)를 선택하여 taglist를 작성한다.
3. 전체 단어의 수
4. 주어진 단어()의 출현 빈도
5. 주어진 단어(Emma)의 출현빈도 확률
6. 출현빈도가 가장 높은 단어 10개 표시

03

워드클라우드

3. 워드클라우드

■ 워드클라우드

- 단어 빈도 기반 시각화 도구
- 텍스트 마이닝, 데이터 탐색, 발표자료 등에서 자주 사용
- 핵심 키워드 파악 및 직관적인 이해에 매우 유용

특징

특징	설명
크기	단어의 빈도수가 높을수록 더 크게 표시됨
색상	강조, 분류 또는 단순한 디자인 요소
배치	무작위 또는 알고리즘에 의해 자동 배치 (보통 중앙에서 바깥쪽으로)
형태	사각형, 원형, 사용자 이미지 모양 등 다양한 형태로 가능

2. 워드클라우드

■ 라이브러리 설치

- 워드클라우드와 맷플롯립 라이브러리 설치

워드클라우드 설치

1	<pre>! pip install pillow !pip install wordcloud</pre>
---	--

맷플롯립 설치

2	<pre>import matplotlib.pyplot as plt</pre>
---	--

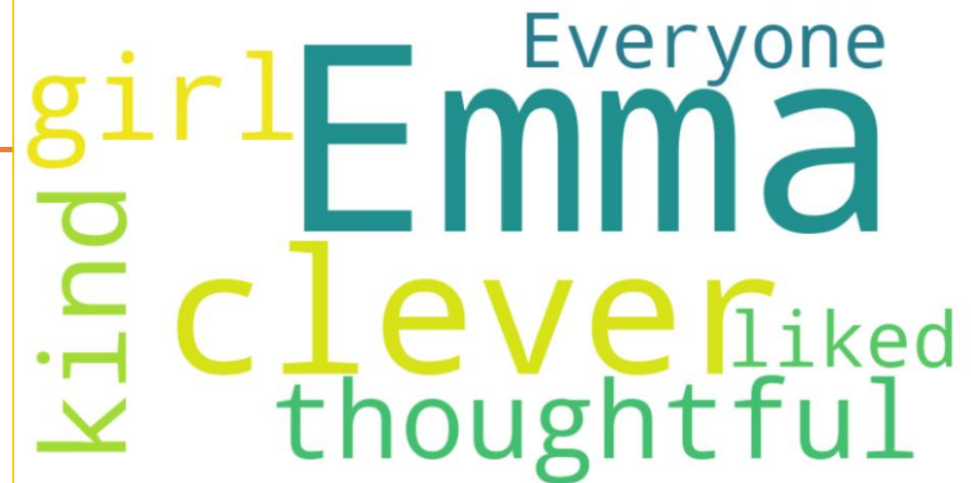
3. 워드클라우드

■ 예

```
from wordcloud import WordCloud
import matplotlib.pyplot as plt

text = "Emma was a clever girl. Emma was also kind and thoughtful. Everyone liked Emma."
wordcloud = WordCloud(width=800, height=400, background_color='white').generate(text)

plt.figure(figsize=(10, 5))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```



04

한글분석:KoNLPy

4. 한글분석:KoNLPy

■ KoNLPy(Korean Natural Language Processing in Python)

- 파이썬에서 한국어 자연어 처리를 쉽게 할 수 있도록 도와주는 오픈소스 라이브러리
- 한국어는 영어와 달리 **형태소 분석**이 중요하고 구조가 복잡하기 때문에,
- 일반적인 NLP 도구로는 분석하기 어려움
- 이런 문제를 해결하기 위해 다양한 **형태소 분석기(Morphological Analyzer)** 들을
- 파이썬에서 사용할 수 있도록 래핑(wrapping)한 라이브러리

4. 한글분석:KoNLPy

■ KoNLPy(Korean Natural Language Processing in Python) 주요 특징

- 다양한 형태소 분석기 지원
 - Hannanum (한나눔, KAIST 개발)
 - Kkma (꼬꼬마, 서울대 개발)
 - Komoran (코모란, Shineware 개발)
 - Okt (Open Korean Text, 트위터에서 개발; 예전 이름은 Twitter)
 - Mecab (고속 분석기로 유명; 별도로 설치 필요)
- 형태소 분석 기능 제공
 - 문장을 형태소 단위로 분해해서 품사 태깅을 붙여줍니다.
 - 예: ['학교', '/Noun', '에', '/Josa', '간다', '/Verb']
- 간단한 파이썬 API
 - 초보자도 쉽게 사용할 수 있도록 파이썬 인터페이스를 제공

4. 한글분석:KoNLPy

■ KoNLPy(Korean Natural Language Processing in Python) 활용분야

- 감정 분석
- 뉴스 기사 키워드 추출
- 챗봇 개발
- 토픽 모델링
- 텍스트 분류

4. 한글분석:KoNLPy

■ KoNLPy(Korean Natural Language Processing in Python)

■ 설치

```
! pip install konlpy
```

■ 말뭉치 가져오기

```
from konlpy.corpus import kolaw, kobill
```

■ 헌법 말뭉치 확인

```
kolaw.fileids()
```

■ 국회법안 말뭉치 확인

```
kobill.fileids()
```

4. 한글분석:KoNLPy

■ KoNLPy 활용

```
#대한민국헌법 데이터 100자 읽어오기
c=kolaw.open('constitution.txt').read()
print(c[:100])
```

```
#국회법안 데이터 100자 읽어오기
k=kobill.open('1809890.txt').read()
print(k[:100])
```

```
# 형태소 분석기 객체 생성
from konlpy.tag import *
hannanum=Hannanum()
kkma=Kkma()
komoran=Komoran()
okt=Okt()
```

```
# 명사추출
print(c[:100])
hannanum.nouns(c[:100])
kkma.nouns(c[:100])
komoran.nouns(c[:100])
okt.nouns(c[:100])
```


4. 한글분석:KoNLPy

■ KoNLPy 활용

```
s1='빅데이터 개념이 어렵네요 Oh ㅋㅋㅋㅋ'
print(hannanum.nouns(s1))
print(kkma.nouns(s1))
print(komoran.nouns(s1))
print(oka.nouns(s1))
```

```
#품사태깅
print(hannanum.pos(s1))
print(kkma.pos(s1))
print(komoran.pos(s1))
print(oka.pos(s1))
```

➡ 결과확인 해보기

```
#품사태그 확인
kkma.tagset
```

4. 한글분석:KoNLPy

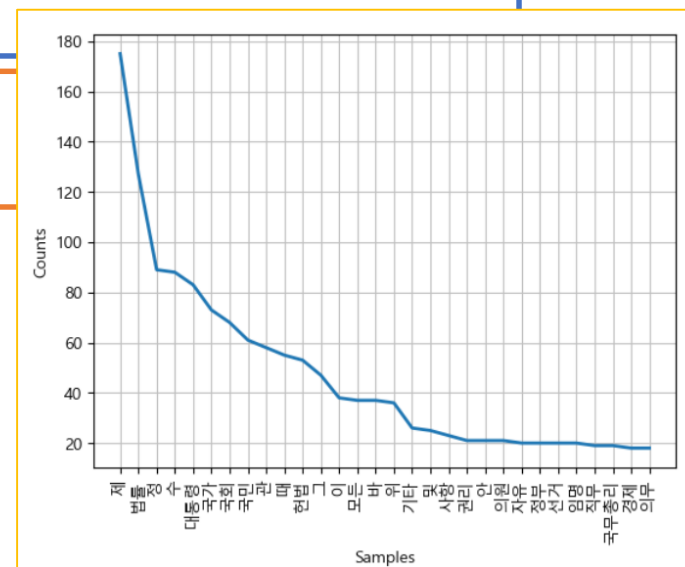
■ 한글 분석 NLTK 활용

```
from nltk import Text
import matplotlib.pyplot as plt
from matplotlib import font_manager, rc
```

#한글설정

```
font_path='data/fonts/malgun.ttf'
font_name=font_manager.FontProperties(fname=font_path).get_name()
rc('font', family=font_name)
```

```
kolaw_nouns=Text(oka.nouns(c))
kolaw_nouns.plot(30) #빈도수 높은 순 30개로 차트 그리기
```



4. 한글분석:KoNLPy

■ 한글 분석 NLTK 활용

```
# FreqDist 객체로 텍스트 빈도수 구함
kv=kolaw_nouns.vocab()
kv
```

```
kv2=dict()
for key, value in kv.items():
    # 명사의 글자수 1보다 크고, 출현빈도가 2보다 큰 것
    if(len(key)>1 and value>2):
        kv2[key]=value
kv2
```

결과:

```
FreqDist({'제': 175, '법률': 127, '정': 89, '수': 88, '대통령': 83, '국가': 73, '국회': 68, '국민': 61, '관': 58, '때': 55, ...})
```

결과

```
{'대한민국': 11, '헌법': 53, '우리': 3, '국민': 61, '민주': 6, '조국': 3, '평화': 6, '통일': 6, '로써': 5, '민족': 3, '공고': 3, '모든': 37, '사회': 8, '질서': 8, '정치': 9, '경제': 18, '문화': 4, '영역': 4, '기회': 3, '능력': 3, '자유': 20, '권리': 21, '책임': 5, '의무': 18, '생활': 8, ... '육성': 5, '정은': 3, '제안': 3, '당시': 6, '최초': 7}
```

4. 한글분석:KoNLPy

■ 한글 분석 NLTK 활용

```
# FreqDist 객체로 텍스트 빈도수 구함
kv=kolaw_nouns.vocab()
kv
```

```
kv2=dict()
for key, value in kv.items():
    # 명사의 글자수 1보다 크고, 출현빈도가 2보다 큰 것
    if (len(key)>1 and value>2):
        kv2[key]=value
kv2
```

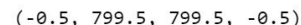
```
from wordcloud import WordCloud
font_path1='data/fonts/malgunbd.ttf'
wc=WordCloud(width=800, height=800,
background_color='white',font_path=font_path1,
random_state=0, colormap='Accent_r')
plt.imshow(wc.generate_from_frequencies(kv2))
plt.axis('off')
```

결과:

```
FreqDist({'제': 175, '법률': 127, '정': 89, '수': 88, '대통령': 83, '국가': 73, '국회': 68, '국민': 61, '관': 58, '때': 55, ...})
```

결과

{ '대한민국': 11, '헌법': 53, '우리': 3, '국민': 61, '민주': 6,
'조국': 3, '평화': 6, '통일': 6, '로써': 5, '민족': 3, '공고': 3,
'모든': 37, '사회': 8, '질서': 8, '정치': 9, '경제': 18, '문화': 4,
'영역': 4, '기회': 3, '능력': 3, '자유': 20, '권리': 21, '책임': 5,
'의무': 18, '생활': 8,...
'육성': 5, '정은': 3, '제안': 3, '당시': 6, '최초': 7}



4. 한글분석:KoNLPy

- collections.Counter 활용 텍스트 빈도 구하기

```
from collections import Counter
s1="aabbcccd"
list1=['abc','abc','aa','aa','bb','bb','bb']
counter1=Counter(s1)
counter2=Counter(list1)
print(counter1)
print(counter2)
```

결과:

```
Counter({'c': 4, 'd': 3, 'a': 2, 'b': 2}) Counter({'bb': 3, 'abc': 2, 'aa': 2})
```

4. 한글분석:KoNLPy

- collections.Counter 활용 텍스트 빈도 구하기

```
#대한민국헌법 말뭉치에 명사추출
nouns1=okt.nouns(c)
#추출된 명사들의 빈도 수 구함
noun_count1=Counter(nouns1)
#빈도수가 높은 순으로 100개 선택
noun_count2=noun_count1.most_common(100)
print(noun_count1.items())
print('counter2:',noun_count2)
count3=dict()
#단어의 수가 1보다크고, 빈도수도 1보다 큰것을 선택
for key, value in noun_count1.items():
    if(len(key)>1 and value>1):
        count3[key]=value
count3
```

4. 한글분석:KoNLPy

■ 텍스트 출현 빈도 시각화

```
from PIL import Image
import numpy as np
im=Image.open('data/images/img02.jpg')
mask_arr=np.array(im)
```

```
wc=WordCloud(width=1000, height=1000,
background_color='white', random_state=0,
font_path=font_path1,
colormap='PRGn_r', mask=mask_arr)
plt.imshow(wc.generate_from_frequencies(count3))
plt.axis('off')
plt.savefig('data/images/wc00.png')
plt.show()
```



05

자연어 처리 실습

5. 자연어 처리 실습

■ 영화 리뷰 자연어 처리 (1)

- 필요한 라이브러리 설치 및 불러오기
 - 코엔엘파이와 워크클라우드 라이브러리 설치
 - 판다스, 맵플롯립, 콜렉션즈, 워드클라우드 라이브러리 импорт

5. 자연어 처리 실습 (1)

■ 영화 리뷰 자연어 처리 (1)

- 데이터 로드하기
 - 네이버 영화 리뷰 데이터를 불러오기(ratings_train.txt)

〈실행결과〉

	id	document	label
0	9976970	아 더빙.. 진짜 짜증나네요 목소리	0
1	3819312	흠...포스터보고 초딩영화인줄....오버연기조차 가볍지 않구나	1
2	10265843	너무재밌었다그래서보는것을추천한다	0
3	9045019	교도소 이야기구먼 ..솔직히 재미는 없다..평점 조정	0

5. 자연어 처리 실습 (1)

■ 영화 리뷰 자연어 처리 (1)

- 데이터 전처리
 - 중복과 결측치 제거

중복 제거 및 결측치 확인		
1 2	<p>〈코드〉</p> <div></div>	<p>〈실행결과〉</p> <pre>150000 id 0 document 5 label 0 dtype: int64</pre>
중복 제거 및 결측치 확인		
1 2	<p>〈코드〉</p> <div></div>	<p>〈실행결과〉</p> <pre>False</pre>

3. 자연어 처리 실습 (1)

■ 영화 리뷰 자연어 처리 (1)

- 데이터 전처리
 - 불용어 제거

중복 제거 및 결측치 확인

```
1 df['document']=df['document'].str.replace("[^ㄱ-ㅎㅏ-ㅣ가-힣 ]","")
```

〈실행결과〉


	id	document	label
0	9976970	아 더빙 진짜 짜증나네요 목소리	0
1	3819312	흠포스터보고 초딩영화인줄 오버연기조차 가볍지 않구나	1
2	10265843	너무 재밌었다 그래서 보는 것을 추천한다	0
3	9045019	교도소 이야기구면 솔직히 재미는 없다 평점 조정	0
4	6483659	사이몬페그의 익살스런 연기가 돋보였던 영화 스파이더맨 에서 늘어보이기만 했던 커스틴 던..	1
...
149995	622902	인간이 문제지 소는 원죄인가	0
149996	8549745	평점이 너무 낮아서	0

3. 자연어 처리 실습 (1)

■ 영화 리뷰 자연어 처리 (1)

■ 데이터 전처리

- 데이터프레임(df)의 document 필드의 값을 형태소 분석기 Okt를 사용하여 품사 태깅을 수행한다.
- 태깅된 품사들 중 명사(Noun)와 형용사(Adjective)를 단어를 추출한다
- 추출된 단어를 collections.Counter()를 사용하여 빈도수를 구한다.
- 빈도수가 높은 50개의 tag를 추출한다

1	
2	
3	
4	
5	
6	
7	
8	
9	

3. 자연어 처리 실습 (1)

■ 영화 리뷰 자연어 처리 (1)

- 데이터 전처리
 - 아래 그림과 같이 워드클라우드 시각화 수행



워드클라우드 시각화

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11

```
plt.figure(figsize=(10,8))
plt.imshow(cloud)
```

〈실행결과〉



3. 자연어 처리 실습 (1)

■ 영화 리뷰 자연어 처리 (1)

- 데이터 전처리
 - 상관없는 단어(불용어) 삭제하기

불용어 처리

```
1 list=[]  
2 stopword=['점','정말','왜','말','그','없다','정도','걸','뭐','이건','영화',  
3 '완전','좋','있는','거','나','이','볼','입니다','것','이런','더','수','때']  
4  
5  
6  
7  
8  
9  
10  
11
```

〈실행결과〉

```
[('진짜', 8338),  
 ('연기', 6328),  
 ('평점', 6315),  
 ('최고', 6040),  
 ('스토리', 5335),  
 ('생각', 5315),  
 ('드라마', 5063),
```

.....중략.....

```
( '개', 1944),  
 ('재밌게', 1913),  
 ('이해', 1898),  
 ('안', 1843),  
 ('이영화', 1816),  
 ('또', 1781)]
```

3. 자연어 처리 실습 (1)

■ 영화 리뷰 자연어 처리 (1)

- 데이터 전처리
 - 워드 클라우드 시각화

워드클라우드 시각화

- 1
- 2
- 3
- 4
- 5
- 6

〈실행결과〉



Q&A