3장. 텍스트 분석 및 시각화

1. 영문분석(NLTK)+ 워드 클라우드
2. 한글분석(KoNLPy) + 워드 클라우드

1. 영문분석

목표 설정

- 목표: 'Big data'와 관련된 키워드를 도출하여 분석

핵심 개념 이해

① 영문 데이터에서 분석할 특징을 선정
② 컴퓨터가 처리할 수 있는 벡터 형태로 변환
③ 분석 기법을 적용하여 필요한 정보를 추출

- 텍스트 분석
  - 자연어 처리와 데이터 마이닝이 결합하여 발전된 분야로 비정형 텍스트 데이터에서 정보를 추출하여 분석하는 방법
  - 단어에 대한 분석을 기본으로 함
  - 텍스트 분류, 텍스트 군집화, 감성 분석 등

1. 영문분석

핵심 개념 이해

전처리

- 분석 작업의 정확도를 높이기 위해 분석에 사용할 데이터를 먼저 정리하고 변환하는 작업

전처리에서 수행하는 작업

종류	설명
정제cleaning	불필요한 기호나 문자를 제거하는 작업으로 주로 정규식을 이용하여 수행한다.
정규화normalization	정제와 같은 의미지만 형태가 다른 단어를 하나의 형태로 통합하는 작업으로 대/소문자 통합, 유사 의미의 단어 통합 등이 있다.
토큰화tokenization	데이터를 토큰으로 정한 기본 단위로 분리하는 작업이다. 문장을 기준으로 분리하는 문장 토큰화, 단어를 기준으로 분리하는 단어 토큰화 등이 있다.
불용어stopword 제거	의미가 있는 토큰을 선별하기 위해 조사, 관사, 접미사처럼 분석할 의미가 없는 토큰인 불용어stopword를 제거한다.
어간 추출semming	단어에서 시제, 단/복수, 진행형 등을 나타내는 다양한 어간stem을 잘라내어 단어의 형태를 일반화한다.
표제어 추출lemmatization	단어에서 시제, 단/복수, 진행형 등을 나타내는 다양한 표제어lemma를 추출하여 단어의 형태를 일반화한다. 품사를 지정하여 표제어를 추출하는 것이 가능하다.

어간 추출과 표제어 추출 예

사용 단어	어간 추출	표제어 추출
am	am	be
the going	the go	the going
having	hav	have

1. 영문분석

핵심 개념 이해

워드클라우드

- 텍스트 분석에서 많이 사용하는 시각화 기법
- 문서의 핵심 단어를 시각적으로 돋보이게 만들어 키워드를 직관적으로 알 수 있게 하는 것
- 출현 빈도가 높을수록 단어를 크게 나타냄
- 방대한 양의 텍스트 정보를 다루는 빅데이터 분석에서 주요 단어를 시각화하기 위해 사용

1. 영문분석

- **NLTK(**NLTK(Natural Language Toolkit) **자연어 처리 패키지¶**
  - 교육용으로 개발된 자연어 처리 및 문서 분석용 파이썬 패키지.
  - 다양한 기능 및 예제를 가지고 있으며 실무 및 연구에서도 많이 사용.
  - NLTK 패키지가 제공하는 주요 기능.
    - 말뭉치
    - 토큰 생성
    - 형태소 분석
    - 품사 태깅

1. 영문분석

- **말뭉치 가져오기**

```
import nltk
nltk.download('book',quiet=True)
from nltk.book import *
nltk.corpus.gutenberg.fileids()  #쿠텐베르그 문서 리스트 확인
emma_raw=nltk.corpus.gutenberg.raw('austen-emma.txt')  #쿠텐베르그 문서 중 오스틴 엠마 문서 로드
emma_raw[:1000]
```

토큰 생성

- 자연어 문서에서 분석을 위해 긴 문자열을 작은 단위로 나누는 것
- 문장 단위, 단어 단위, 정규표현식으로 나눌 수 있음

```
#문장단위 토큰 생성
from nltk.tokenize import sent_tokenize
print(sent_tokenize(emma_raw[:10000])[1])
```

```
#단어단위 토큰생성
from nltk.tokenize import word_tokenize
print(word_tokenize(emma_raw[50:100]))
```

```
#정규표현식 토큰생성
from nltk.tokenize import RegexpTokenizer
retTokenize=RegexpTokenizer("[₩w]+")
retTokenize.tokenize(emma_raw[50:100])
```

형태소 분석

- 형태소 : 언어학에서 일정한 의미가 있는 가장 작은 말의 단위
- 보통 자연어 처리에서 토큰으로 형태소를 이용
- 형태소 분석 : 단어로부터 어근, 접두사, 접미사, 품사 등 다양한 언어적 속성을 파악하고 이를 이용하여 형태소를 찾아내거나 처리하는 작업
- 형태소 분석의 예
  - 어간 추출
  - 원형 복원
  - 품사 부착

어간 추출

- PorterStemmer, LancasterStemmer 제공
- 어간추출은 단순히 어미만 제거함

```
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("LancasterStemmer:", [st2.stem(w) for w in words])
```

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


list2=[st1.stem(w) for w in words]
print(list2)
```

원형복원

- 같은 의미를 가지는 여러 단어를 사전형으로 통일하는 작업

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


lm=WordNetLemmatizer()
[lm.lemmatize(w, 'v') for w in words]
```

형태소 분석 품사 태깅

- 품사는 낱말을 문법적인 기능이나 형태, 뜻에 따라 구분한 것
- 품사의 예
  - NNP : 단순고유명사
  - VB : 동사
  - VBP : 동사현재형
  - NN : 명사
  - DT : 관형사

```
from nltk.tag import pos_tag
sentence='Emma refused to permit us to obtain the refuse permit'
tagged_list=pos_tag(word_tokenize(sentence))  # 품사태깅
tagged_list
```

형태소 분석

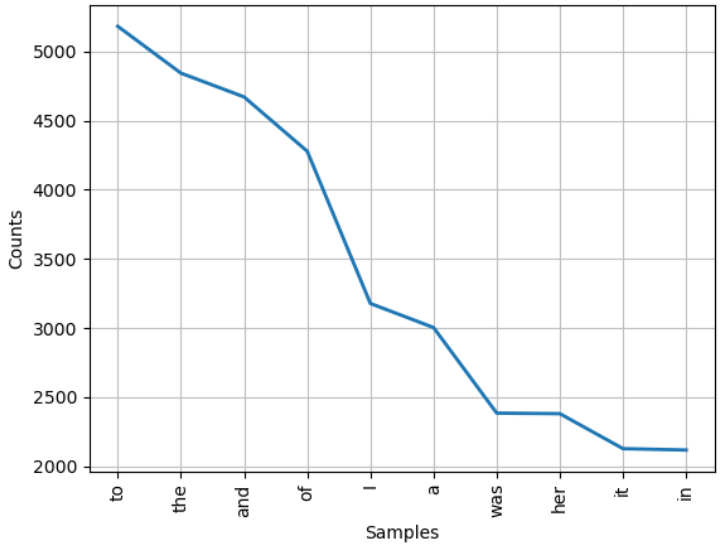
```
nltk.help.upenn_tagset('NNP') # 품사 정보 보기
#품사 중 명사만 추출
noun_list=[t[0] for t in tagged_list if t[1]=="NN"] noun_list

#품사가 명사일 때, 단어, 그렇지 않으면 '-' 표시
noun_list1=[t[0] if t[1]=='NN' else '-' for t in tagged_list]
noun_list1

list1=[]
for t in tagged_list:
    if t[1]=='NN':
        list1.append(t[0])
    else:
        list1.append('_')
list1
```

```
#부착된 품사 제거
from nltk.tag import untag
untag_list=untag(tagged_list)
untag_list
```

Text 클래스

- 문서 분석에 유용한 메소드 제공

```
from nltk import Text
text=Text(retTokenize.tokenize(emma_raw))  # 텍스트 객체 생성
print(text)

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



Text 클래스

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

# 단어가 사용된 위치를 직접표시하며, 문맥의 정보를 보여줌
text.concordance('Emma')

# similar : 같은 문맥에서 주어진 단어 대신 사용된 회수가 높은 단어를 찾아 줌
text.similar("Jane")
```

- **FreqDist**
  - 문서에 사용된 단어(토큰)의 사용빈도 정보를 담는 클래스
  - vocab() 메소드로 추출 `fd=text.vocab() ; type(fd)`

```
from nltk import FreqDist
stopwords=['Mr.','Mrs.','Miss','Mr','Mrs','Dear','A','No', 'Ah','Oh']  # 빈도수 구할 때 제거할 단어
emma_token=pos_tag(retTokenize.tokenize(emma_raw))
name_list=[t[0] for t in emma_token if t[1]=='NNP' and t[0] not in stopwords]

fd_name=FreqDist(name_list) ; fd_name

print(fd_name.N()) # 전체 단어의 수
print(fd_name['Emma']) # 주어진 단어에 해당하는 빈도수
print(fd_name.freq('Emma')) # 주어진 단어의 출현 확률
fd_name.most_common(10) # 출현빈도가 가장 높은 단어 10 표시
```

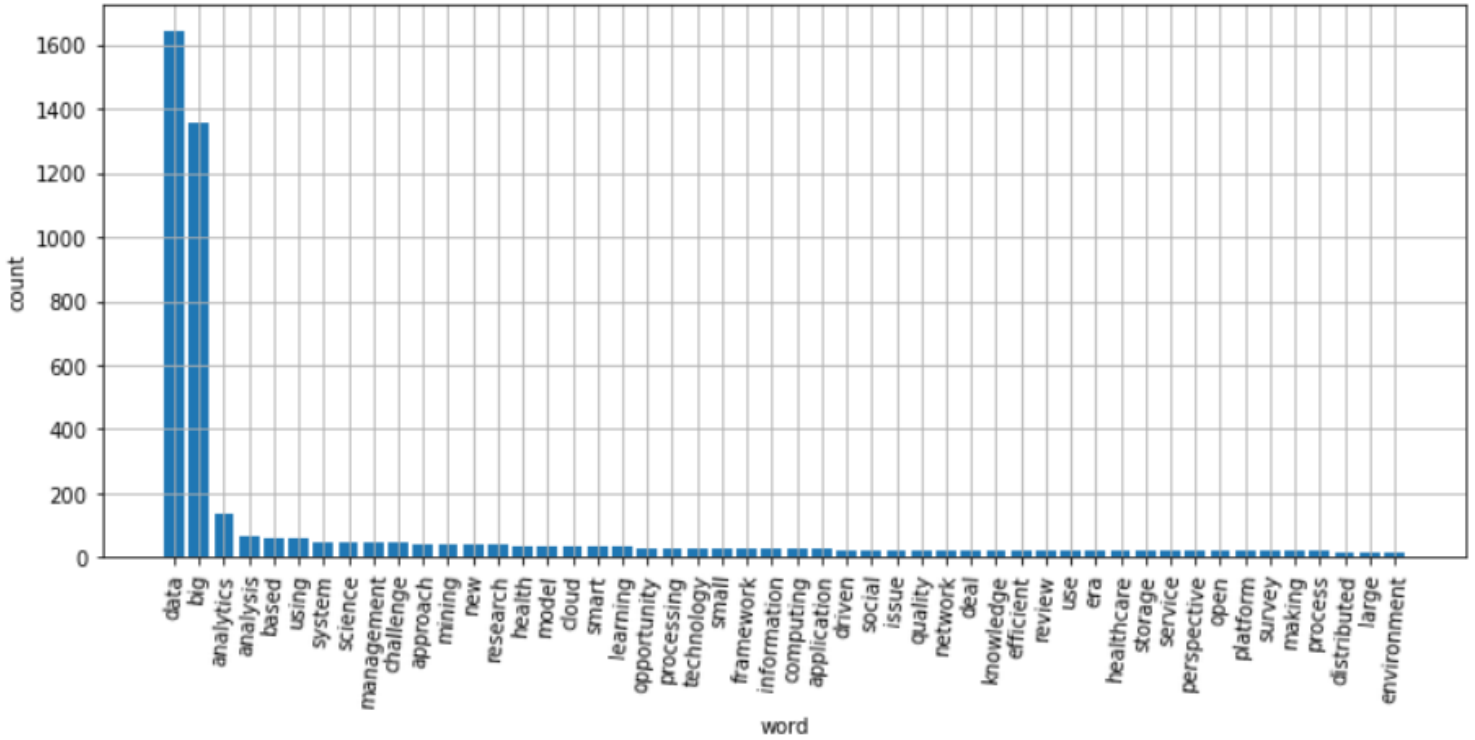
- **WordCloud**
  - 단어의 출현빈도 시각화

설치
```
! pip install pillow --upgrade
! pip install wordcloud
 conda install -c conda-forge wordcloud #console에서 설치
```

```
from wordcloud import WordCloud
from PIL import Image
import numpy as np

im=Image.open('data/img.png')
mask_arr=np.array(im)
mask_arr

wc=WordCloud(width=1000, height=800, background_color='white', mask=mask_arr, random_state=0)
plt.imshow(wc.generate_from_frequencies(fd_name))
plt.axis('off')
plt.savefig('data/wc.png')
plt.show()
```

1. 영문 분석

- **영문 문제 제목 키워드 분석하기**
  - 데이터 수집
  - 데이터 준비
  - 데이터 탐색 및 모델 구축
  - 결과 시각화



2. 한글 분석

- **KoNlpy**
  - **KoNLPy 한국어 처리 패키지**
  - KoNLPy(코엔엘파이라고 읽음)는 한국어 정보처리를 위한 파이썬 패키지

- **KoNLPy 설치**
  - Jpype 다운로드 : https://www.lfd.uci.edu/~gohlke/pythonlibs/#jpype
  - Jpype를 각자 파이썬 버전에 맞게(저의 경우에는 3.8버전) 다운로드
  - ' JPype1-1.2.0-cp38-cp38-win_amd64.whl '
  - pip install JPype1-1.2.0-cp38-cp38-win_amd64.whl
  - pip install konlpy

2. 한글 분석

- **KoNLPy**
  - 한국어 말뭉치¶
    - KoNLPy에서는 대한민국 헌법 말뭉치인 kolaw와 국회법안 말뭉치인 kobill을 제공한다.
    - 각 말뭉치가 포함하는 파일의 이름은 fields 메서드로 알 수 있고 open 메서드로 해당 파일의 텍스트를 읽어들인다.
  - 형태소 분석
    - KoNLPy는 다양한 형태소 분석, 태깅 라이브러리를 파이썬에서 쉽게 사용할 수 있도록 모아놓았다.
    - Hannanum: 한나눔. KAIST Semantic Web Research Center 개발. -http://semanticweb.kaist.ac.kr/hannanum/
    - Kkma: 꼬꼬마. 서울대학교 IDS(Intelligent Data Systems) 연구실 개발.
      - http://kkma.snu.ac.kr/
    - Komoran: 코모란. Shineware에서 개발.
      - https://github.com/shin285/KOMORAN
    - Open Korean Text: 오픈 소스 한국어 분석기. 과거 트위터 형태소 분석기.
      - https://github.com/open-korean-text/open-korean-text
    - 여기에서는 한나눔, 꼬꼬마, 오픈코리안텍스트 형태소만 예제로 포함하였다.

KoNLPy

형태소 분석기 생성

```
from konlpy.tag import *
hannanum = Hannanum()
kkma = Kkma()
komoran = Komoran()
okt = Okt()
```

명사추출

```
hannanum.nouns(c[:40])
kkma.nouns(c[:40])
komoran.nouns("₩n".join([s for s in c[:40].split("₩n") if s]))
```

형태소 추출

```
hannanum.morphs(c[:40])
komoran.morphs("₩n".join([s for s in c[:40].split("₩n") if s]))
okt.morphs(c[:40])
```

품사 부착

```
kkma.pos(c[:40])
komoran.morphs("₩n".join([s for s in c[:40].split("₩n") if s]))
okt.pos(c[:40])
```

부착된 품사태그의 기호와 의미 확인

```
kkma.tagset
komoran. tagset
okt. tagset
Hannanum.tagset
```

시각화(차트그리기)

```
from nltk import Text
import matplotlib.pyplot as plt

from matplotlib import font_manager, rc
font_path='c:/Windows/fonts/malgun.ttf'
font_name=font_manager.FontProperties(fname=font_path).get_name()
rc('font',family=font_name)

kolaw=Text(okt.nouns(c), name='kolaw')
kolaw.plot(30)
plt.show()
```

시각화(wordcloud)

```
kv=kolaw.vocab()
kv2=dict()
for tag, count in kv.items():
    if (len(tag))>1:
        kv2[tag]=count
kv2
#kv
```

```
from wordcloud import WordCloud
font_path='c:/Windows/fonts/malgun.ttf'
wc=WordCloud(width=1000, height=800, background_color='white',
font_path=font_path)
plt.imshow(wc.generate_from_frequencies(kv2))
plt.axis('off')
plt.show()
```

시각화(wordcloud)

```
from collections import Counter

nouns=okt.nouns(c)
counter=Counter(nouns)
counter1=counter.most_common(10)

counter2=dict()
for tag, count in counter.items():
    if (len(tag)>1) and (count>1):
        counter2[tag]=count
counter2
```

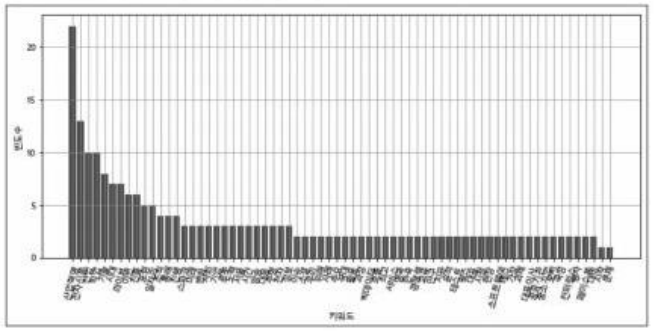
```
wc=WordCloud(width=1000,
             height=800,
             background_color='white',
             random_state=0,
             font_path=font_path)

plt.imshow(wc.generate_from_frequencies(counter2))
plt.axis('off')
plt.show()
```

네이버 뉴스 기사의 키워드 분석하기

한글 뉴스 기사의 키워드 분석하기	
목표	'4차 산업혁명'에 관한 한글 기사에서 명사 키워드를 분석한다.
핵심 개념	형태소 분석, 품사 태깅
데이터 수집	4차 산업혁명 기사: 페이스북 전자신문 페이지에서 크롤링하여 저장한 json 파일(예제소스로 제공)
데이터 준비	1. 데이터 추출: json 파일에서 message 항목만 추출, key() 2. 명사 단어 추출: Okt 품사 태깅 패키지로 명사 추출, from konlpy.tag import Okt
데이터 탐색 및 모델링	1. 단어 빈도 탐색 • Counter() 2. 단어 빈도 히스토그램 • font_manager.FontProperties() • matplotlib.pyplot

결과 시각화

1. 단어 빈도에 대한 히스토그램



2. 단어 빈도에 대한 워드클라우드