KoNLPy를 활용한 한국어 형태소 분석과 시각화

한국기술교육대학교
온라인평생교육원

학습목표

- KoNLPy 라이브러리를 활용하여 한국어 형태소를 분석할 수 있다.
- text 파일을 읽어서 형태소를 나누고, 워드클라우드를 만들 수 있다.

학습내용

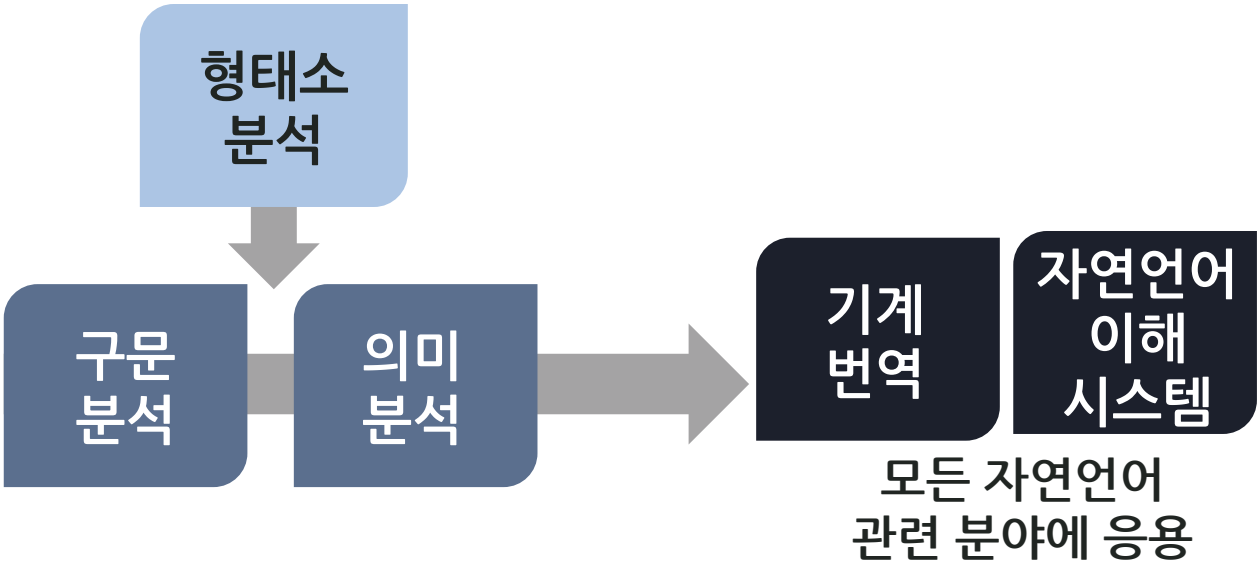
- KoNLPy를 활용한 한국어 형태소 분석
- 시각화

KoNLPy를 활용한 한국어 형태소 분석

1 형태소 분석 및 품사 태깅

1 형태소 분석의 정의

- 모든 자연언어 처리 분야에서 가장 중요하면서도 기본적으로 필요한 것
  ➡ 그 언어의 **형태소 분석**



KoNLPy를 활용한 한국어 형태소 분석

1 형태소 분석 및 품사 태깅

1 형태소 분석의 정의

형태소	▪ 언어에 있어서 최소 의미 단위
형태소 분석	▪ 형태소보다 언어 단위가 큰 어절, 혹은 문장을 최소 의미 단위인 형태소로 분절하는 과정 ▪ 형태소를 비롯하여, 어근, 접두사/접미사, 품사(POS, Part-of-Speech) 등 다양한 언어적 속성의 구조를 파악하는 것 ▪ 문장을 의미 있는 작은 단위로 나눠 주로 명사나 동사 중심으로 필요한 단어들을 추출하여 분석하는데 분석이 쉽지는 않음 ➡ 분석을 돕는 많은 라이브러리 존재

KoNLPy를 활용한 한국어 형태소 분석

1 형태소 분석 및 품사 태깅

2 품사 태깅 정의

품사 태깅	▪ 형태소의 뜻과 문맥을 고려하여 마크업을 하는 것

품사 태깅 예시

▪ 가방에 들어가신다

➡ 가방/NNG + 에/JKM + 들어가/VV + 시/EPH + ㄴ다/EFN

[출처 : https://KoNLPy-ko.readthedocs.io/ko/v0.4.3/morph/]

KoNLPy를 활용한 한국어 형태소 분석

1 형태소 분석 및 품사 태깅

3 형태소 분석툴의 종류

종류	특징
KoNLPy	▪ 대표적인 한국어 형태소 분석기 ▪ java 기반 → jdk 설치 필요 ▪ 품사 태깅, 내부에 Twitter, Kkma, Hannanum 등 형태소 분석기 사용 ▪ 기본 사전으로 세종말뭉치 사용 ▪ 사용자 사전 등록이 가능하나 오류가 많음 ▪ 윈도우 지원
SoyNLP	▪ 고유명사 추출에 유용 ▪ 품사 태깅 없이 토큰화 기능 우선 ▪ 윈도우 지원 ▪ 사용자 사전 등록 안 됨
MeCab	▪ 은전 한 닢 프로젝트 ▪ 오픈 소스 형태의 분석기 ▪ jdk 설치 ▪ 리눅스 기반, 윈도우 지원 어려움
Khaiii(카이)	▪ 카카오에서 만든 형태소 분석기

KoNLPy를 활용한 한국어 형태소 분석

2 KoNLPy 설치

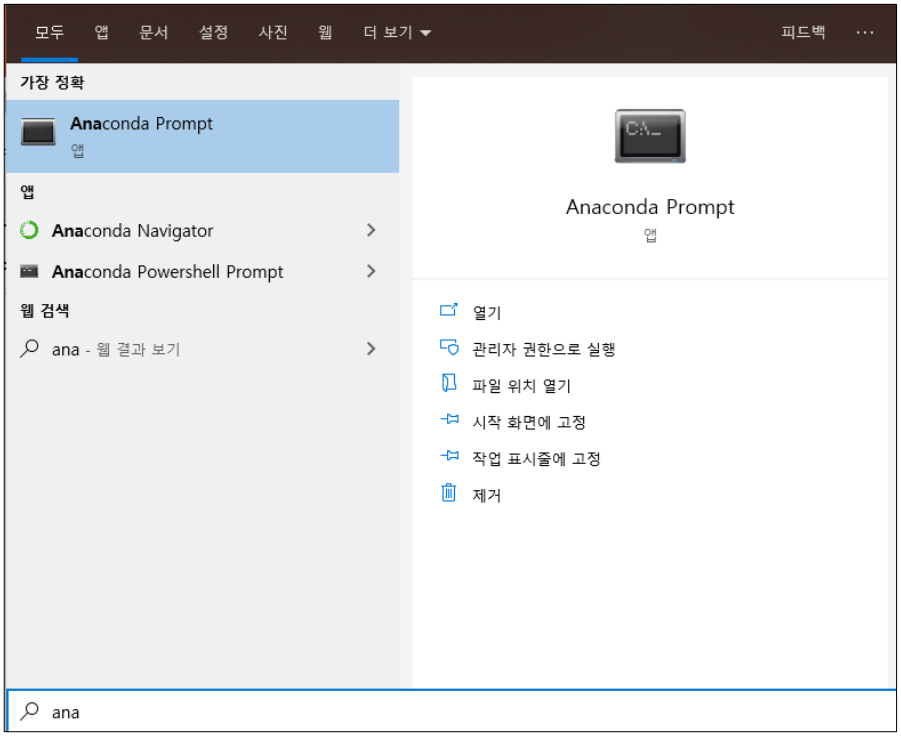
1 KoNLPy 설치

JDK 설치

검색 창에 anaconda를 검색하면
Anaconda Prompt라는 메뉴가 보임

메뉴 선택 후 마우스 오른쪽을 눌러
[관리자 권한으로 실행]

시스템 권한을 요하는 라이브러리도 있어
관리자 권한이 아닐 경우 오류가 발생할 수 있음

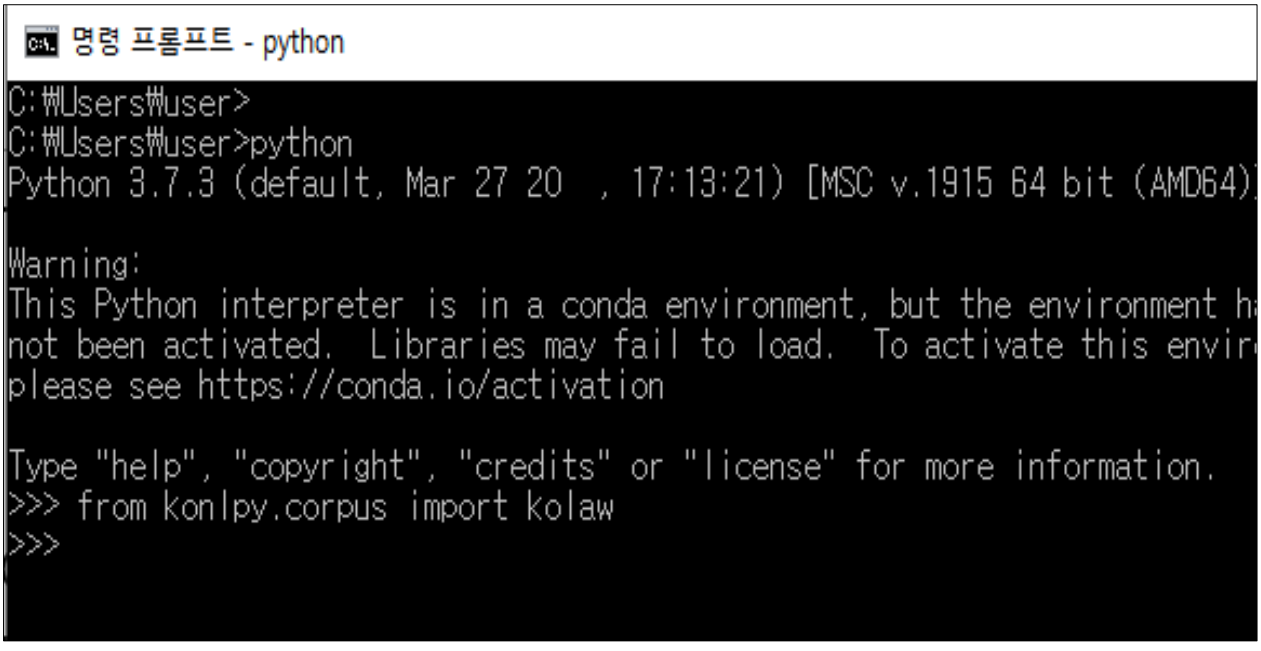


KoNLPy를 활용한 한국어 형태소 분석

2 KoNLPy 설치

2 설치 및 확인

- pip install konlpy
- cmd 창에서 python 입력
- from KoNLPy.corpus import kolaw라고 기술하고, 오류가 없다면 설치 완료



KoNLPy를 활용한 한국어 형태소 분석

3 형태소 분석 모듈 사용

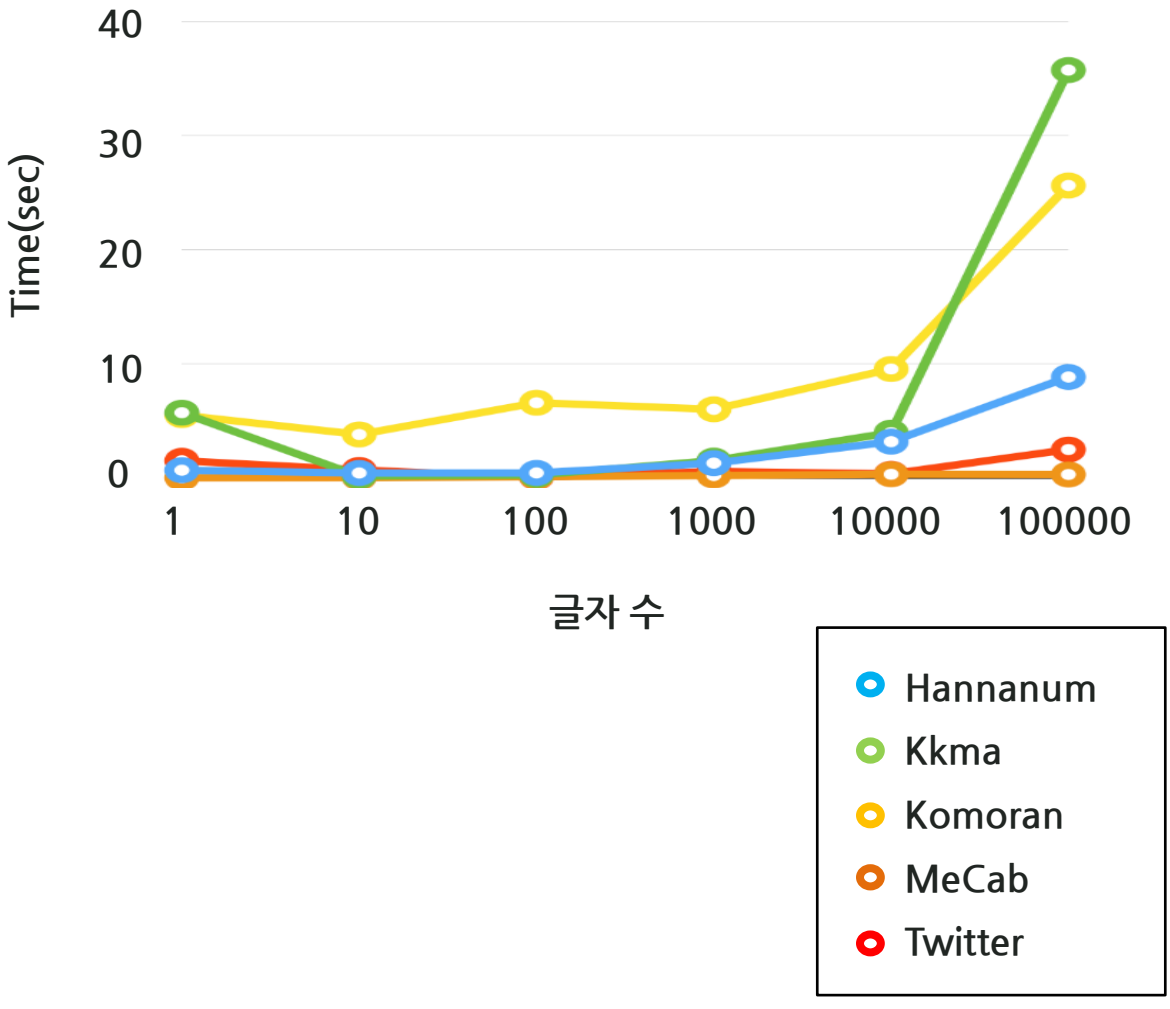
- KoNLPy 라이브러리에는 형태소 분석을 위해 지원하는 클래스가 여러 개 있음

- Hannanum Class(사용 가능)
- Kkma Class(사용 가능)
- Twitter Class(사용 가능)
- Komoran Class(현재 파이썬 버전에서는 작동 안 함)
- MeCab Class(Window 7에서 지원 안 함)

KoNLPy를 활용한 한국어 형태소 분석

3 형태소 분석 모듈 사용

- 클래스 간 성능 분석표



[출처 https://konlpy-ko.readthedocs.io/ko/v0.4.3/morph/#pos-tagging-with-konlpy]

KoNLPy를 활용한 한국어 형태소 분석

3 형태소 분석 모듈 사용

1 Kkma 모듈 사용

```
#파일명 : exam15_1.py
from KoNLPy.tag import Kkma
from KoNLPy.utils import pprint

msg = """
오픈 소스를 이용하여 형태소 분석을 배워 봅시다.
형태소 분석을 지원하는 라이브러리가 많습니다.
각자 어떻게 분석하는지 살펴보겠습니다.
이건 Kkma모듈입니다.
"""

kkma = Kkma()
print(kkma.sentences(msg))
print(kkma.morphs(msg) )
print(kkma.nouns(msg))
print(kkma.pos(msg))
```

```
['오픈 소스를 이용하여 형태소 분석을 배워 봅시다.', '형태소 분석을 지원하는 라이브러리가 많습니다.', '각자 어떻게 분석하는지 살펴보겠습니다.', '이건 Kkma 모듈입니다.
']
['오픈', '소스', '를', '이용', '하', '여', '형태소', '분석', '을', '배우', '어', '보', 'ㅂ시다', '.', '형태소', '분석', '을', '지원', '하', '는', '라이브러리', '가',
 '많', '습니다', '.', '각', '자', '어떻', '게', '분석', '하', '지', '는', '살펴보', '겠', '습니다', '.', '이건', 'Kkma', '모듈', '이', 'ㅂ니다', '.']
['오픈', '오픈소스', '소스', '이용', '형태소', '분석', '지원', '라이브러리', '이건', '모듈']
[('오픈', 'NNG'), ('소스', 'NNG'), ('를', 'JKO'), ('이용', 'NNG'), ('하', 'XSV'), ('여', 'ECS'), ('형태소', 'NNG'), ('분석', 'NNG'), ('을', 'JKO'), ('배우', 'VV'), (
'어', 'ECS'), ('보', 'VV'), ('ㅂ시다', 'EFA'), ('.', 'SF'), ('형태소', 'NNG'), ('분석', 'NNG'), ('을', 'JKO'), ('지원', 'NNG'), ('하', 'XSV'), ('는', 'ETD'), ('라이
브러리', 'NNG'), ('가', 'JKS'), ('많', 'VA'), ('습니다', 'EFN'), ('.', 'SF'), ('각', 'VV'), ('자', 'ECE'), ('어떻', 'VA'), ('게', 'ECD'), ('분석', 'NNG'), ('하', 'XS
V'), ('지', 'ECD'), ('는', 'JX'), ('살펴보', 'VV'), ('겠', 'EPT'), ('습니다', 'EFN'), ('.', 'SF'), ('이건', 'NNP'), ('Kkma', 'OL'), ('모듈', 'NNG'), ('이', 'VCP'), (
'ㅂ니다', 'EFN'), ('.', 'SF')]
```

KoNLPy를 활용한 한국어 형태소 분석

3 형태소 분석 모듈 사용

2 Hannanum 모듈 사용

```
#파일명 : exam15_2.py

from KoNLPy.tag import Hannanum
from KoNLPy.utils import pprint

msg = """
오픈 소스를 이용하여 형태소 분석을 배워 봅시다.
형태소 분석을 지원하는 라이브러리가 많습니다.
각자 어떻게 분석하는지 살펴보겠습니다.
이건 Hannanum 모듈입니다.
"""

hannanum = Hannanum()
print(hannanum.analyze(msg))
print(hannanum.morphs(msg))
print(hannanum.nouns(msg))
print(hannanum.pos(msg))
```

KoNLPy를 활용한 한국어 형태소 분석

3 형태소 분석 모듈 사용

2 Hannanum 모듈 사용

```
[[[('오픈', 'ncpa'), ('소스', 'ncn'), ('를', 'jco')]], [[('이용', 'ncpa'), ('하', 'xsva'), ('어', 'ecs')], [('이용', 'ncpa'), ('하', 'xsva'), ('어', 'ecx')], [('이용', 'ncpa'), ('하', 'xsva'), ('어', 'ef')]], [[('형태소', 'ncn')], [('형태', 'ncn'), ('소', 'ncn')]], [[('분석', 'ncpa'), ('을', 'jco')]], [[('배우', 'pvg'), ('어', 'ecx'), ('보', 'px'), ('ㅂ시다', 'ef')]], [[('.', 'sf')], [('.', 'sy')]], [], [[('형태소', 'ncn')], [('형태', 'ncn'), ('소', 'ncn')]], [[('분석', 'ncpa'), ('을', 'jco')]], [[('지원', 'ncpa'), ('하', 'xsva'), ('는', 'etm')], [('지원', 'ncpa'), ('하', 'xsva'), ('어', 'ecs'), ('는', 'jxc')], [('지원', 'ncpa'), ('하', 'xsva'), ('어', 'ef'), ('는', 'etm')]], [[('라이브러리', 'ncn'), ('가', 'jcc')], [('라이브러리', 'ncn'), ('가', 'jcs')]], [[('많', 'paa'), ('습니다', 'ef')]], [[('.', 'sf')], [('.', 'sy')]], [], [[('각자', 'mag')], [('각자', 'ncn')]], [[('어떻', 'pad'), ('게', 'ecs')], [('어떻', 'pad'), ('게', 'ecx')], [('어떻게', 'mag')]], [[('분석', 'ncpa'), ('하', 'xsva'), ('지는', 'ecs')], [('분석', 'ncpa'), ('하', 'xsva'), ('지는', 'ecx')], [('분석', 'ncpa'), ('하', 'xsva'), ('지는', 'etm')], [('분석', 'ncpa'), ('하', 'xsva'), ('지', 'ecs'), ('는', 'jxc')], [('분석', 'ncpa'), ('하', 'xsva'), ('지', 'ef'), ('는', 'etm')], [('분석', 'ncpa'), ('하', 'xsva'), ('어', 'ecx'), ('지', 'px'), ('는', 'etm')], [('분석', 'ncpa'), ('하', 'xsva'), ('어', 'ef'), ('지는', 'etm')]], [[('살펴보', 'pvg'), ('겠', 'ep'), ('습니다', 'ef')], [('살피', 'pvg'), ('어', 'ecx'), ('보', 'px'), ('겠', 'ep'), ('습니다', 'ef')]], [[('.', 'sf')], [('.', 'sy')]], [], [[('이', 'ncn'), ('건', 'xsnx')], [('이', 'ncn'), ('이', 'jp'), ('건', 'ecc')], [('이', 'ncn'), ('이', 'jp'), ('건', 'ecs')], [('이', 'nnc'), ('건', 'nbu')], [('이', 'nnc'), ('건', 'xsnx')], [('이', 'nnc'), ('이', 'jp'), ('건', 'ecc')], [('이', 'nnc'), ('이', 'jp'), ('건', 'ecs')], [('이', 'npd'), ('건', 'xsnx')], [('이', 'npd'), ('이', 'jp'), ('건', 'ecc')], [('이', 'npd'), ('이', 'jp'), ('건', 'ecs')], [('이', 'pvg'), ('건', 'ecc')], [('이', 'pvg'), ('건', 'ecs')]], [[('Hannanum', 'f')]], [[('모듈', 'ncn'), ('이', 'jp'), ('ㅂ니다', 'ef')], [('모듈입니', 'ncn'), ('이', 'jp'), ('다', 'ef')], [('모듈입니다', 'ncn')], [('모듈', 'nqq'), ('이', 'jp'), ('ㅂ니다', 'ef')], [('모듈입니', 'nqq'), ('이', 'jp'), ('다', 'ef')], [('모듈입니다', 'nqq')]], [[('.', 'sf')], [('.', 'sy')]]]
['오픈소스', '를', '이용', '하', '어', '형태소', '분석', '을', '배우', '어', '보', 'ㅂ시다', '.', '형태소', '분석', '을', '지원', '하', '는', '라이브러리', '가', '많', '습니다', '.', '각자', '어떻', '게', '분석', '하', '어', '지', '는', '살피', '어', '보', '겠습니다', '.', '이', '이', '건', 'Hannanum', '모듈입니', '이', '다', '.']
['오픈소스', '이용', '형태소', '분석', '형태소', '분석', '지원', '라이브러리', '각자', '분석', '이', '모듈입니']
[('오픈소스', 'N'), ('를', 'J'), ('이용', 'N'), ('하', 'X'), ('어', 'E'), ('형태소', 'N'), ('분석', 'N'), ('을', 'J'), ('배우', 'P'), ('어', 'E'), ('보', 'P'), ('ㅂ시다', 'E'), ('.', 'S'), ('형태소', 'N'), ('분석', 'N'), ('을', 'J'), ('지원', 'N'), ('하', 'X'), ('는', 'E'), ('라이브러리', 'N'), ('가', 'J'), ('많', 'P'), ('습니다', 'E'), ('.', 'S'), ('각자', 'N'), ('어떻', 'P'), ('게', 'E'), ('분석', 'N'), ('하', 'X'), ('어', 'E'), ('지', 'P'), ('는', 'E'), ('살피', 'P'), ('어', 'E'), ('보', 'P'), ('겠습니다', 'E'), ('.', 'S'), ('이', 'N'), ('이', 'J'), ('건', 'E'), ('Hannanum', 'F'), ('모듈입니', 'N'), ('이', 'J'), ('다', 'E'), ('.', 'S')]
```

KoNLPy를 활용한 한국어 형태소 분석

3 형태소 분석 모듈 사용

3 Twitter 모듈 사용

```
#파일명 : exam15_3.py

from KoNLPy.tag import Kkma, Hannanum, Komoran, MeCab,
 Twitter
from KoNLPy.utils import pprint

msg = """
오픈 소스를 이용하여 형태소 분석을 배워 봅시다.
형태소 분석을 지원하는 라이브러리가 많습니다.
각자 어떻게 분석하는지 살펴보겠습니다.
이건 Twitter 모듈입니다.
"""

twitter = Twitter()
print(twitter.morphs(msg))
print(twitter.nouns(msg))
print(twitter.pos(msg))
```

```
['\n', '오픈소스', '를', '이용', '하여', '형태소', '분석', '을', '배워', '봅시다', '.', '형태소', '분석', '을', '지원', '하는', '라이브러리', '가', '많습니다', '.',
'각자', '어떻게', '분석', '하', '지는', '살펴보겠습니다', '.', '이건', 'Twitter', '모듈', '입니다', '.', '\n']
['오픈소스', '이용', '형태소', '분석', '형태소', '분석', '지원', '라이브러리', '각자', '분석', '이건', '모듈']
[('\n', 'Foreign'), ('오픈소스', 'Noun'), ('를', 'Josa'), ('이용', 'Noun'), ('하여', 'Verb'), ('형태소', 'Noun'), ('분석', 'Noun'), ('을', 'Josa'), ('배워', 'Verb'),
 ('봅시다', 'Verb'), ('.', 'Punctuation'), ('형태소', 'Noun'), ('분석', 'Noun'), ('을', 'Josa'), ('지원', 'Noun'), ('하는', 'Verb'), ('라이브러리', 'Noun'), ('가', '
Josa'), ('많습니다', 'Adjective'), ('.', 'Punctuation'), ('각자', 'Noun'), ('어떻게', 'Adjective'), ('분석', 'Noun'), ('하', 'Suffix'), ('지는', 'Josa'), ('살펴보겠
습니다', 'Verb'), ('.', 'Punctuation'), ('이건', 'Noun'), ('Twitter', 'Alpha'), ('모듈', 'Noun'), ('입니다', 'Adjective'), ('.', 'Punctuation'), ('\n', 'Foreign')]
```

KoNLPy를 활용한 한국어 형태소 분석

3 형태소 분석 모듈 사용

4 파일을 읽어서 명사로 분리

```
#파일명 : exam15_4.py

from KoNLPy.tag import Kkma, Hannanum, Komoran, MeCab,
 Twitter
from KoNLPy.utils import pprint

file = open("./data/data1.txt")
msg = file.read()

print("--- Kkma 클래스 -----")
kkma = Kkma()
print("--- 문장으로 분해 --- ")
print(kkma.sentences(msg))

print("--- 명사 분해 --- ")
print(kkma.nouns(msg))

print("--- 품사 태깅 --- ")
print(kkma.pos(msg))
```

(계속)

KoNLPy를 활용한 한국어 형태소 분석

3 형태소 분석 모듈 사용

4 파일을 읽어서 명사로 분리

```
hannanum = Hannanum()
print("--- 문장으로 분해 --- ")
print(hannanum.analyze(msg))
print("--- 형태소로 분해 --- ")
print(hannanum.morphs(msg))
print("--- 명사 분해 --- ")
print(hannanum.nouns(msg))
print("--- 품사 태깅 --- ")
print(hannanum.pos(msg))

twitter = Twitter()
print("--- MeCab 클래스 -----")
twitter = Twitter()
print("--- 형태소로 분해 --- ")
print(twitter.morphs(msg))
print("--- 명사 분해 --- ")
print(twitter.nouns(msg))
print("--- 품사 태깅 --- ")
print(twitter.pos(msg))
```

KoNLPy를 활용한 한국어 형태소 분석

4 말뭉치와 사전

말뭉치	▪ 언어 연구를 위해 텍스트를 컴퓨터가 읽을 수 있는 형태로 모아 놓은 언어 자료

- 통계 분석 및 가설 검증 수행
- 특정한 언어 영역 내 언어 규칙 발생의 검사, 규칙의 정당성 입증에 사용
- 어떤 언어를 분석함에 있어서 기준이 되는 집합
  ➡ 집합을 어떤 것을 사용하는지에 따라 분석 결과가 달라짐
- 법과 관련된 언어 분석 ➡ 법률 관련 말뭉치 필요
- 예술과 관련된 언어 분석 ➡ 예술 관련 말뭉치 필요

KoNLPy 라이브러리에서 제공하는 말뭉치

- kolaw : 한국 법률 말뭉치, constitution.txt
- kobill : 대한민국 국회 의안 말뭉치
  파일 ID는 의안 번호를 의미,
  1809890.txt - 1809899.txt
- 세종 말뭉치 : 국립국악원에서 모아놓은 국어 말뭉치
- Kaist 말뭉치 : 카이스트에서 만듦

KoNLPy를 활용한 한국어 형태소 분석

4 말뭉치와 사전

- 사전은 말뭉치를 이용해 구축됨
- 사전은 형태소 분석이나 품사 태깅에 사용됨

Hannanum 사전	▪ Kaist 말뭉치를 이용해 구축
Kkma 사전	▪ 세종 말뭉치를 이용해 구축
MeCab 사전	▪ 세종 말뭉치를 이용해 구축

KoNLPy를 활용한 한국어 형태소 분석

4 말뭉치와 사전

```
#파일명 : exam15_5.py
from KoNLPy.corpus import kolaw, kobill
wordList = kolaw.open('constitution.txt').readlines()
print(wordList[:3])     #3줄만 출력
print()
wordList2 = kobill.open('1809890.txt').readlines()
print(wordList2[:30])   #30줄만 출력
```

```
['대한민국헌법\n', '\n', '유구한 역사와 전통에 빛나는 우리 대한국민은 3·1운동으로 건립된 대한민국임시정부의 법통과 불의에 항거한 4·19민주이념을 계승하고, 조국의 민주개혁과 평화적 통일의 사명에 입각하여 정의·인도와 동포애로써 민족의 단결을 공고히 하고, 모든 사회적 폐습과 불의를 타파하며, 자율과 조화를 바탕으로 자유민주적 기본질서를 더욱 확고히 하여 정치·경제·사회·문화의 모든 영역에 있어서 각인의 기회를 균등히 하고, 능력을 최고도로 발휘하게 하며, 자유와 권리에 따르는 책임과 의무를 완수하게 하여, 안으로는 국민생활의 균등한 향상을 기하고 밖으로는 항구적인 세계평화와 인류공영에 이바지함으로써 우리들과 우리들의 자손의 안전과 자유와 행복을 영원히 확보할 것을 다짐하면서 1948년 7월 12일에 제정되고 8차에 걸쳐 개정된 헌법을 이제 국회의 의결을 거쳐 국민투표에 의하여 개정한다.\n']

['지방공무원법 일부개정법률안\n', '\n', '(정의화의원 대표발의 )\n', '\n', ' 의 안\n', ' 번 호\n', '\n', '9890\n', '\n', '발의연월일 : 2010. 11. 12. \n', '\n', '발 의 자 : 정의화.이명수.김을동 \n', '\n', '이사철.여상규.안규백\n', '\n', '황영철.박영아.김정훈\n', '\n', '김학송 의원(10인)\n', '\n', '제안이유 및 주요내용\n', '\n', ' 초등학교 저학년의 경우에도 부모의 따뜻한 사랑과 보살핌이 필요\n', '\n', '한 나이이나, 현재 공무원이 자녀를 양육하기 위하여 육아휴직을 할 \n', '\n', '수 있는 자녀의 나이는 만 6세 이하로 되어 있어 초등학교 저학년인 \n', '\n', '자녀를 돌보기 위해서는 해당 부모님은 일자리를 그만 두어야 하고 \n', '\n', '이는 곧 출산의욕을 저하시키는 문제로 이어질 수 있을 것임.\n']
```

시각화

1 시각화란?

- 언어를 형태소로 분석 후 가장 많이 사용하는 시각화 도구는 워드클라우드(WordCloud)임

- 파이썬과 웹 프로그램 언어인 JavaScript 지원
- 텍스트 데이터를 전달하면 워드클라우드를 만들어서 다운받게 해주는 웹 사이트도 있음
- 파이썬에서도 워드클라우드를 지원하는 라이브러리가 여러 종류임

시각화

2 필요한 모듈 설치

- pip install pytagcloud ← 워드클라우드 지원 라이브러리
- pip install pygame
- pip install wordcloud ← 워드클라우드 지원 라이브러리 (만든 사람이 다름)

시각화

3 pytagcloud 라이브러리

1 pytagcloud 라이브러리란?

pytagcloud 라이브러리	▪ 데이터의 단어와 단어의 빈도 수를 튜플로 묶어 리스트 형태로 전달해야 함

- 예 : [('school',30), ('rainbow',10), ('cloud',23), ('peach',10), ('pink',20)]

시각화

3 pytagcloud 라이브러리

2 워드클라우드 작성

```
#파일명 : exam15_6.py
import pytagcloud      #워드클라우드 라이브러리
import webbrowser     #브라우저 제어 라이브러리

tag = [( ' school ' ,30), ( ' rainbow ' ,10), ( ' cloud ' ,23), ( ' peach ' ,10),
( ' pink ' ,20)]      #데이터 준비
#print(tag)
taglist = pytagcloud.make_tags(tag, maxsize=50)
#각 단어를 {'color': (131, 194, 49), 'size': 68, 'tag': 'school'} 색, 크기, tag 형태의 dict 타입으로 만듦

#print(taglist)
pytagcloud.create_tag_image(taglist,
                    'wordcloud.jpg',     #wordcloud.jpg 이름으로 저장
                    size=(300, 300),     #이미지 크기
                    fontname= ' Nobile',   #폰트명
                     rectangular=True)
                     #이미지 모양, true일 경우 사각형으로 그림

#브라우저에 이미지를 보여줌
webbrowser.open('wordcloud.jpg')
```

시각화

3 pytagcloud 라이브러리

3 한글 폰트

- pytagcloud는 한글 폰트를 지원하지 않으므로 한글을 쓰려면 폰트를 별도로 설치해야 함

- C:₩ProgramData₩Anaconda3₩Lib₩site-packages₩pytagcloud₩fonts 경로에 한글 폰트가 있어야 함

한글 폰트

- 웹 사이트에서 원하는 폰트 다운
- C:/Windows/Fonts에서 찾음
- 위 폴더에서 한글을 지원하는 폰트 파일을 복사하여 C:₩ProgramData₩Anaconda3₩Lib₩site-packages₩pytagcloud₩fonts 경로에 붙여 넣기
- ProgramData 폴더는 중요한 시스템 폴더이기 때문에 c:/에서 이 폴더가 보이지 않으므로 위 경로를 직접 쳐야 함

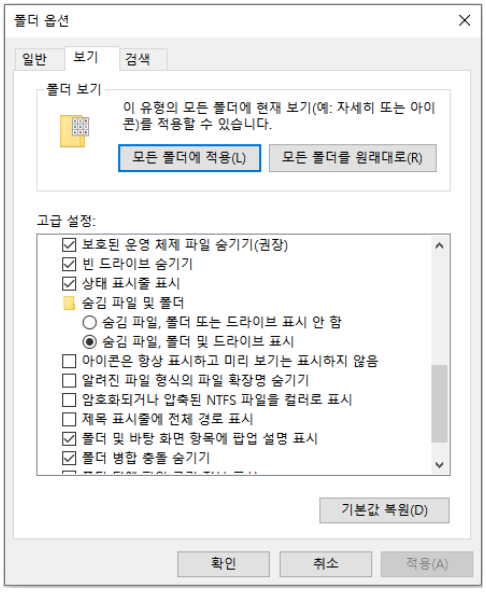
시각화

3 pytagcloud 라이브러리

3 한글 폰트

폴더 보기를 원하는 경우

- 탐색기의 '보기' 메뉴에서 - 옵션 - 폴더 및 검색 옵션 선택
- '보기' 탭을 누른 후 '숨김 파일, 폴더 및 드라이브 표시' 선택



시각화

3 pytagcloud 라이브러리

3 한글 폰트

- C:₩ProgramData₩Anaconda3₩Lib₩site-packages₩pytagcloud₩fonts에 있는 fonts.json 파일을 Visual Studio Code에서 열고 맨 앞에 한글 폰트 추가

```
{
        "name": "HangleFont1",
        "ttf": "H2GTRE.TTF",
        "web":
http://fonts.googleapis.com/css?family=Nobile
},
```

- 이름은 임의로 정하고, ttf에는 복사해 온 한글 폰트 파일명 작성
- 컴퓨터에 해당 폰트가 없으면 가지고 있는 한글 폰트 사용

시각화

3 pytagcloud 라이브러리

3 한글 폰트

```
#파일명 : exam15_7.py
import pytagcloud
import webbrowser

tag = [('학교',30), ('무지개',10), ('구름',23), ('복숭아',14),
       ('분홍색',20)]
#print(tag)
taglist = pytagcloud.make_tags(tag, maxsize=40)

#print(taglist)
pytagcloud.create_tag_image(taglist,
                    'wordcloud.jpg',
                    size=(300, 300),
                    fontname='HangleFont1', rectangular=True)

#브라우저에 이미지를 보여줌
webbrowser.open('wordcloud.jpg')

#이미지나 색은 계속해서 랜덤으로 바꿈
```

시각화

4 파일에서 읽은 데이터 시각화

```
import pytagcloud
import webbrowser
from KoNLPy.tag import Twitter
from collections import Counter

#1. txt 파일 읽기
file = open("./data/data1.txt")
text = file.read()

#2. 파일로부터 명사 추출
twitter = Twitter()
nouns = twitter.nouns(text)

#3. Counter 클래스를 사용하여 단어와 단어의 빈도 수 카운트
nounsCount = Counter(nouns)
tag = nounsCount.most_common(100)
#빈도수가 많은 것 100개만 추출
taglist = pytagcloud.make_tags(tag, maxsize=100)

pytagcloud.create_tag_image(taglist,      'wordcloud2.jpg',
                    size=(400, 400),        fontname='HangleFont1',
 rectangular=False)

#브라우저에 이미지를 보여줌
webbrowser.open('wordcloud2.jpg')
```

시각화

5 wordcloud 라이브러리

1 wordcloud 라이브러리란?

- pytagcloud 모듈보다 사용하기 편함
- 별도로 Counter 클래스를 사용하지 않고, text를 전달하면 형태소를 분석하여 시각화 진행

- 우선 라이브러리를 설치해야 함

- pip install wordcloud

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

wordcloud = WordCloud().generate(text)

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

시각화

5 wordcloud 라이브러리

2 워드클라우드 작성

```
#파일명 : exam15_9.py
import matplotlib.pyplot as plt
from wordcloud import WordCloud

#1. txt 파일 읽기
file = open("./data/alice.txt")
text = file.read()

#2. text를 그대로 전달
wordcloud = WordCloud().generate(text)

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

시각화

5 wordcloud 라이브러리

2 워드클라우드 작성

```
#파일명 : exam15_10.py
from os import path
from wordcloud import WordCloud
import nltk
from matplotlib import font_manager, rc, matplotlib_fname
import matplotlib.pyplot as plt

#한국 법률 말뭉치
from KoNLPy.corpus import kolaw
from KoNLPy.tag import  Twitter
```

시각화

5 wordcloud 라이브러리

2 워드클라우드 작성

```
#한국 법률 말뭉치로부터 헌법 읽어오기
c = kolaw.open('constitution.txt').read()
print(c)

wordcloud = WordCloud(font_path="c:/Windows/Fonts/malg
un.ttf").generate(c)
wordcloud.to_file("image4.png")

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

시각화

5 wordcloud 라이브러리

2 워드클라우드 작성

- 이미지에 맞춰 다양한 형태의 워드클라우드를 만들 수 있음

- 이미지를 준비하고, 이미지를 Numpy 배열로 전환

```
mask_image = np.array(Image.open("./1.jpg"))
```

- WordCloud 생성자 함수의 mask 속성에 배열 전달

```
wordcloud = WordCloud(font_path="c:/Windows/Fonts/
malgun.ttf",
                      relative_scaling=0.2,
                      background_color='white',
                      mask=mask_image,).generate_from_
                      frequencies(temp_data)
```

시각화

6 다양한 모양의 워드클라우드 만들기


```
from wordcloud import WordCloud
import nltk
from collections import Counter
#한국 법률 말뭉치
from KoNLPy.corpus import kolaw
from KoNLPy.tag import  Twitter
import matplotlib.pyplot as plt
import numpy as np
from wordcloud import  ImageColorGenerator
from PIL import Image
import random

c = open("./data/data1.txt", 'r').read()
#명사만 골라냄
t = Twitter()
tokens_ko = t.nouns(c)
print(tokens_ko)

#제거하고 싶은 단어들 제거
stop_words = ['다만', '그', '곳', '나','일', '패', '달리', '의']
tokens_ko = [each_word for each_word in tokens_ko if each_
word not in
stop_words]
#print(tokens_ko)
```

(계속)

시각화

6 다양한 모양의 워드클라우드 만들기

```
data = Counter(tokens_ko).most_common(100)
temp_data = dict(data)

mask_image = np.array(Image.open("./image/1.jpg"))
#image 폴더 아래의 1.jpg 이미지를 불러옴

wordcloud = WordCloud(font_path= " c:/Windows/Fonts/
malgun.ttf " ,
                    relative_scaling=0.2,
                    background_color= ' white ' ,
                    mask=mask_image,).generate_from_frequencies
                    (temp_data)

plt.figure(figsize=(16,8))  #차트 화면의 크기 지정
random.seed(1234)
#색 정보가 랜덤하게 생성되기 때문에 계속 바뀌는걸 방지하기 위해
값을 줌, 1234는 임의로 준 값임
#별도의 색을 부여하려면 사용자 정의 함수를 작성하여 전달
def custom_color_func(word, font_size, position, orientation,
random_state=None, **kwargs):

    color= " rgb({},{},{}) " .format( random.randint(0, 255),
    random.randint(0, 255), random.randint(0, 255))
#rgb 함수는 red, green, blue 의 세가지 색을 섞어서 색을 만듦.
값은 0~255까지만 부여가 가능
    return color
```

(계속)

시각화

6 다양한 모양의 워드클라우드 만들기

```
#사용자 정의 함수를 전달
plt.imshow(wordcloud.recolor(color_func=custom_color_func),
 interpolation= " bilinear " )

plt.axis("off")
plt.show()
```

학습정리

1. KoNLPy를 활용한 한국어 형태소 분석

- 형태소 분석이란 형태소보다 단위가 큰 언어 단위인 어절, 혹은 문장을 최소 의미 단위인 형태소로 분절하는 과정임
- KoNLPy 라이브러리는 한글 형태소를 분석하기 위해 카이스트에서 만든 라이브러리임
- KoNLPy는 Kkma, Twitter, Hannanum, Komoran 등의 태그 패키지를 지원함

학습정리

2. 시각화

- 워드클라우드는 단어를 분석해 시각화하는 차트로 가장 인기 있는 차트임
- pytagcloud, wordcloud 등의 모듈이 있음
- 두 모듈 모두 문장을 형태소 분석하여 단어와 빈도수의 데이터 타입을 만들어 전달해야 함
- wordcloud 라이브러리는 이미지 마스킹 기능을 이용해 특정한 이미지 모양에 맞는 워드클라우드를 만들 수 있고 사용자가 원하는 색상을 부여할 수 있음