한글 자연어 처리 (KoNLPy)

2019.10

"코엔엘파이"

1. NLP란?

- 텍스트에서 의미있는 정보를 분석, 추출하고 이해하는 일련의 기술
- NLP 응용사례
  - 텍스트 요약 (ex: Summly)
  - 자동 질의응답 시스템 (ex: Wolfram Alpha)
  - 대화 시스템 (ex: Apple Siri)
  - 기계 번역 (ex: Google Translate)
  - 검색엔진과 같은 정보검색 시스템

- 준비 사항
  - 언어에 대한 깊은 관심과 한국어에 대한 어느 정도의 이해
  - 기본적인 파이썬 프로그래밍 방법
  - 좋은 텍스트 에디터 또는 파이썬 IDE

2. NLTK 자연어 처리 패키지

■ 개요

- NLTK(Natural Language Toolkit) 패키지는 교육용으로 개발된 자연어 처리 및 문서 분석용 파이썬 패키지
- 주요 기능
  - 말뭉치
  - 토큰 생성
  - 형태소 분석
  - 품사 부착

■ 말뭉치

- 자연어 분석 작업을 위해 만든 샘플 문서 집합
- 사용시 download 하여 사용

■ 토큰 생성

- 문장 분석을 위해 나뉘어진 작은 문자열
- 토큰 생성 함수(tokenizer)를 통해 만들어짐

■ 형태소 분석

- 형태소(morpheme) : 일정한 의미가 있는 가장 작은 말의 단위
- 단어로부터 어근, 접두사, 접미사, 품사 등 다양한 언어적 속성을 파악하고 이를 이용하여 형태소를 찾아내거나 처리하는 작업
- 어간 추출(stemming)
  - 변화된 단어의 접미사나 어미를 제거하여 같은 의미를 가지는 형태소의 기본형을 찾는 방법
  - 단순히 어미를 제거
- 원형 복원(lemmatizing)
  - 같은 의미를 가지는 여러 단어를 사전형으로 통일하는 작업
  - 품사(part of speech)를 지정하는 경우 좀 더 정확한 원형을 찾을 수 있음
- 품사 부착(Part-Of-Speech tagging)
  - 낱말을 문법적인 기능이나 형태, 뜻에 따라 구분
  - NLTK에서는 펜 트리뱅크 태그세트(Penn Treebank Tagset)라는 것을 이용

■ 품사 부착(Part-Of-Speech tagging)

- 낱말을 문법적인 기능이나 형태, 뜻에 따라 구분
- 품사의 구분은 언어마다 그리고 학자마다 다름
- NLTK에서는 펜 트리뱅크 태그세트(Penn Treebank Tagset)라는 것을 이용
- 품사의 예
  - NNP: 단수 고유명사
  - VB: 동사
  - VBP: 동사 현재형
  - TO: to 전치사
  - NN: 명사(단수형 혹은 집합형)
  - DT: 관형사

■ Text 클래스에서 사용할 수 있는 유용한 메소드

- plot() - 각 단어(토큰)의 사용 빈도를 그래프로 보여줌
- dispersion_plot() - 단어가 사용된 위치를 시각화
- concordance() - 해당 단어의 앞과 뒤에 사용된 단어(문맥)를 보여줌
- similar() - 같은 문맥에서 주어진 단어 대신 사용된 횟수가 높은 단어들을 찾아줌
- common_contexts() - 두 단어의 공통 문맥

■ FreqDist 클래스

- FreqDist 클래스는 문서에 사용된 단어(토큰)의 사용빈도 정보를 담는 클래스
- Text 클래스의 vocab 메서드로 추출 가능
- 직접 클래스 생성 가능
- most_common() - 가장 출현 횟수가 높은 단어를 찾아줌

3. KoNLPy 설치

- **Ubuntu(Linux)**

```
$ apt-get update
$ apt-get install g++ openjdk-8-jdk
$ pip3 install konlpy

$ apt-get install curl
$ bash <(curl -s
https://raw.githubusercontent.com/konlpy/konlpy/master/scripts/mecab.sh)
```

- **Windows**
  - JDK 설치, JAVA_HOME 셋팅
  - pip upgrade: `pip install –upgrade pip`
  - KoNLPy 설치 : `pip install konlpy`
  - JPype1 설치 : `pip install JPype1-py3`

4. 형태소 분석 및 품사 태깅

■ 개요

- 형태소 분석 이란 형태소를 비롯하여, 어근, 접두사/접미사, 품사(POS, part-of-speech) 등 다양한 언어적 속성의 구조를 파악하는 것
- 품사 태깅 은 형태소의 뜻과 문맥을 고려하여 그것에 마크업을 하는 일

  가방에 들어가신다 -> 가방/NNG + 에/JKM + 들어가/VV + 시/EPH + ㄴ다/EFN

■ KoNLPy로 품사 태깅하기

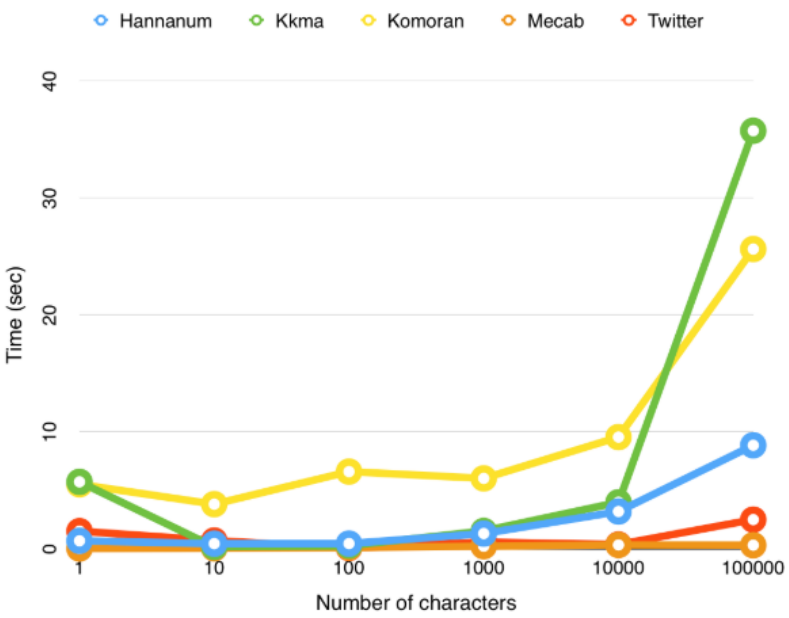
- KoNLPy에는 품사 태깅을 하기 위한 옵션이 여럿 있음
- 문구(phrase)를 입력받아 태깅된 형태소를 출력하는 동일한 입출력 구조
- 대표적인 품사 태깅 클래스

  Kkma, Komoran, Hannanum, Okt(구 Twitter), Mecab

■ 품사 태깅 클래스 간 비교 – 처리 성능

태깅 클래스	로딩 시간[1)]	실행 시간[2)]
Kkma	5.6988 *secs*	35.7163 *secs*
Komoran	5.4866 *secs*	25.6008 *secs*
Hannanum	0.6591 *secs*	8.8251 *secs*
Okt(구 Twitter)	1.4870 *secs*	2.4714 *secs*
Mecab	0.0007 *secs*	0.2838 *secs*

1) 로딩 시간: 사전 로딩을 포함하여 클래스를 로딩하는 시간
2) 실행시간: 10만 문자의 문서를 대상으로 각 클래스의 pos 메소드를 실행하는데 소요되는 시간.



■ 품사 태깅 클래스 간 비교 - 분석 성능

- 예1) "*아버지가방에들어가신다*"

Hannanum	Kkma	Komoran	Mecab	Twitter
아버지가방에들어가 / N	아버지 / NNG	아버지가방에들어가신다 / NNP	아버지 / NNG	아버지 / Noun
이 / J	가방 / NNG		가 / JKS	가방 / Noun
시ㄴ다 / E	에 / JKM		방 / NNG	에 / Josa
	들어가 / VV		에 / JKB	들어가신 / Verb
	시 / EPH		들어가 / VV	다 / Eomi
	ㄴ다 / EFN		신다 / EP+EC	

■ 품사 태깅 클래스 간 비교 – 분석 성능

- 예2) "*나는 밥을 먹는다*" vs "*하늘을 나는 자동차*"

Hannanum	Kkma	Komoran	Mecab	Twitter
나 / N	나 / NP	나 / NP	나 / NP	나 / Noun
는 / J	는 / JX	는 / JX	는 / JX	는 / Josa
밥 / N	밥 / NNG	밥 / NNG	밥 / NNG	밥 / Noun
을 / J	을 / JKO	을 / JKO	을 / JKO	을 / Josa
먹 / P	먹 / VV	먹 / VV	먹 / VV	먹는 / Verb
는다 / E	는 / EPT	는다 / EC	는다 / EC	다 / Eomi
	다 / EFN			

하늘 / N	하늘 / NNG	하늘 / NNG	하늘 / NNG	하늘 / Noun
을 / J	을 / JKO	을 / JKO	을 / JKO	을 / Josa
나 / N	날 / VV	나 / NP	나 / NP	나 / Noun
는 / J	는 / ETD	는 / JX	는 / JX	는 / Josa
자동차 / N	자동차 / NNG	자동차 / NNG	자동차 / NNG	자동차 / Noun

■ 말뭉치(corpus)

- 자연언어 연구를 위해 특정한 목적을 가지고 언어의 표본을 추출한 집합
- 사용 가능한 말뭉치
  - kolaw: 한국 법률 말뭉치(예: constitution.txt – 헌법)
  - kobill: 대한민국 국회 의안 말뭉치. 파일 ID는 의안 번호를 의미 (예: 1809890.txt - 1809899.txt)

```
>>> from konlpy.corpus import kolaw
>>> c = kolaw.open('constitution.txt').read()
>>> print c[:10]
대한민국 헌법

유구한 역사와
>>> from konlpy.corpus import kobill
>>> d = kobill.open('1809890.txt').read()
>>> print d[:15]
지방공무원법 일부개정법률안
```

■ 사전

- Hannanum 시스템 사전 - KAIST 말뭉치를 이용해 생성된 사전. (4.7MB)
- Kkma 시스템 사전 - 세종 말뭉치를 이용해 생성된 사전. (32MB)
- Mecab 시스템 사전 - 세종 말뭉치로 만들어진 CSV 형태의 사전. (346MB)

■ KoNLPy 예제

- 문서 탐색하기
- 말뭉치 탐색하기
- 연어(collocation) 찾기
- 구문 분석