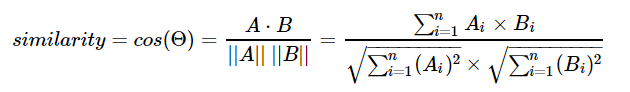
**한글 문장의 유사도 계산 (4)**

사용된 대용량 말뭉치: KCCq28\_Q01.txt (31.29MB, 20만줄)

사용된 입력 문장: 안녕하세요 이것은 테스트 문장입니다.

사용된 유사도 계산식: 코사인 유사도



**Ex. 코사인 유사도를 예시를 들어 해석**

1. 나는 사과 좋아요 : 나는(1), 사과(1), 좋아요(1)
2. 나는 좋아요 좋아요: 나는(1), 좋아요(2)

코사인 유사도 분모:

sqrt(나는 개수^2 + 사과 개수^2 + 좋아요 개수^2) \* sqrt(나는 개수^2 + 좋아요 개수^2)

= sqrt((1^2+1^2+1^2) \* (1^2+2^2)

= sqrt(3 \* 5)

= sqrt(15)

코사인 유사도 분자:

((문장1: 나는 개수) \* (문장2: 나는 개수)) + ((문장1: 사과 개수) \* (문장2: 사과 개수))

+ ((문장1: 좋아요 개수) \* (문장2: 좋아요 개수))

= 1\*1 + 1\*0 + 1\*2

= 1+2

= 3

* 코사인 유사도 3/sqrt(15)
* 문장 유사도 100 \* 3/sqrt(15)

코드 구현:

* 멀티 프로세싱을 이용해 형태소 분석 및 유사도 측정을 하기 위해 코드 내부에서 대용량 말뭉치 문장들을 코어 개수에 맞게 분할

Ex. CPU 코어 개수가 4개, 말뭉치 문장이 20만줄이 있다고 가정

* 200000/4 = 50000 : 5만줄짜리 텍스트 리스트 4개를 생성
* 각 코어당 분할된 문장 리스트를 주고 형태소 분석 및 문장 유사도 검사를 시킨다.
* 유사도 검사가 끝나면 각 코어에서 유사도가 높은 상위 2문장씩 출력해준다.

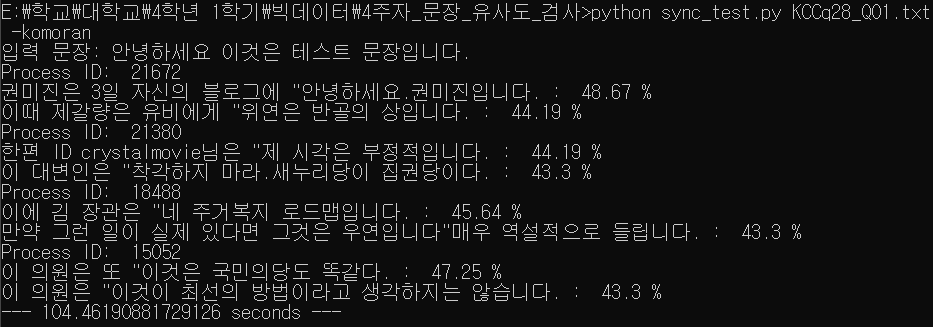
실행 방법:

python sync\_test.py [input-text] [-option]

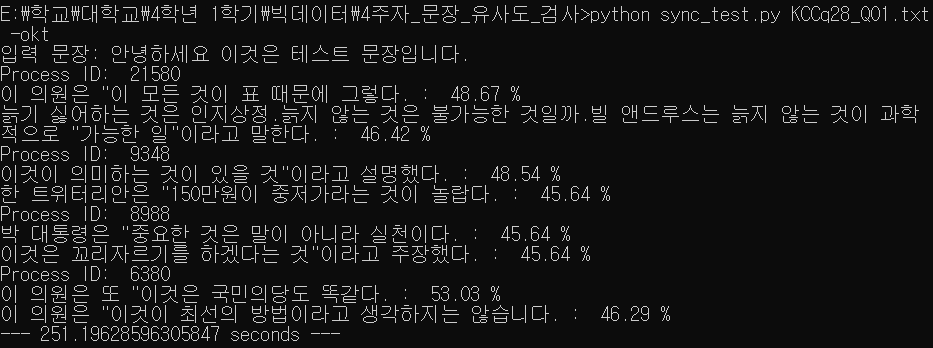
* Option: -okt , -komoran , -kkma

어떤 형태소 분석기를 사용할 지 옵션으로 선택

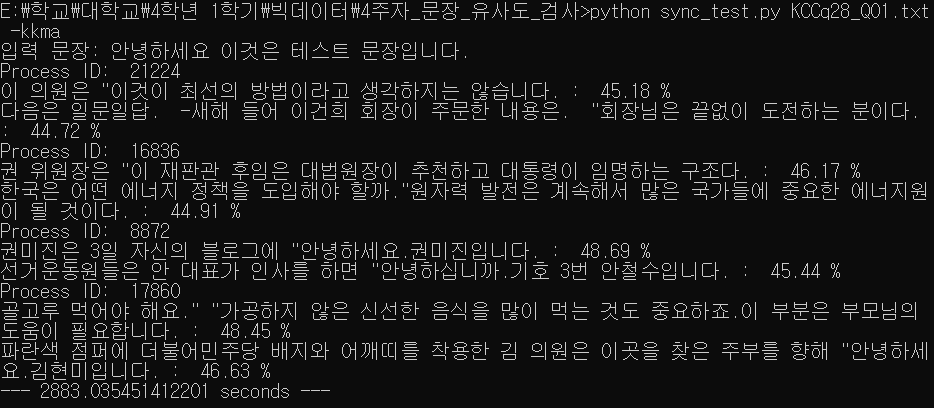
* 입력 문장으로는 “안녕하세요 이것은 테스트 문장입니다.” 를 터미널에 타이핑하였다.
* python sync\_test.py KCCq28\_Q01.txt -komoran
* 실행시간: 약 104초 -> 약 1분 44초



* python sync\_test.py KCCq28\_Q01.txt -okt
* 실행시간: 약 251초 -> 약 4분 11초



* python sync\_test.py KCCq28\_Q01.txt -kkma
* 실행시간: 약 2883초 -> 약 45분



결론:

* 형태소 분석기마다 다르게 형태소를 나누기 때문에 유사도가 다 다르게 나온다.
* 가장 빠른 형태소 분석기는 실행시간에서 알 수 있듯이 komoran이다.