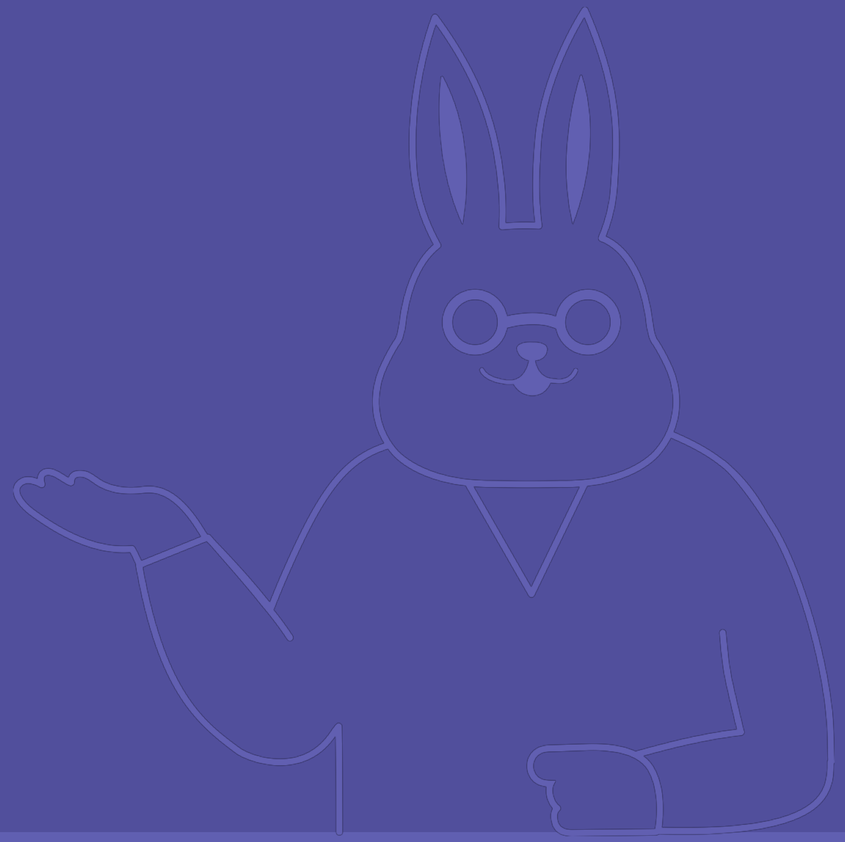
딥러닝을 이용한 자연어 처리

03 한국어 자연어 처리 및 문장 유사도

목차

01

한국어 자연어 처리



자연어 처리의 기본 요소

[텍스트 1] : Hello Elice, how are you today?

[텍스트 1] : Hello | Elice | how | are | you | today

자연어 처리의 기본은 **단어 추출**에서 시작

자연어 처리의 기본 요소

[텍스트 1] : Hello Elice, how are you today?

[텍스트 1] : Hello | Elice | how | are | you | today

인물 시간

텍스트의 단어를 통해 **문장의 의미, 구성 요소 및 특징**을 파악 가능

한국어에서 단어란

국어국문학자료사전

단어

[單語]

구분 어학

문법 단위 중 기본이 되는 언어 단위의 하나. 그 정의는 쉽지 않으며, 아직도 일정하게 내려지지 못하고 있다. 단일한 의미를 가지는 음 결합체'라 하여 의미를 기준으로 [illegible]

한국어에서 단어의 기준은 **명확하지 않음**

한국어에서 단어란

엘리스는

엘리스가

엘리스에게

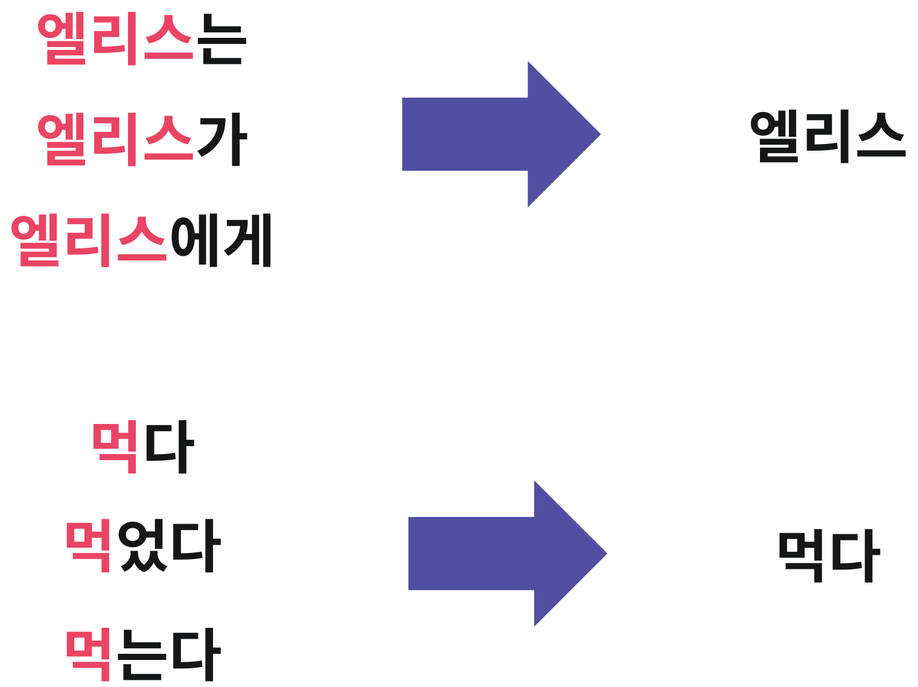
먹다

먹었다

먹는다

교착어인 한국어에서 단어는 **의미적 기능**을 하는 부분과
문법적인 기능을 하는 부분의 조합으로 구성

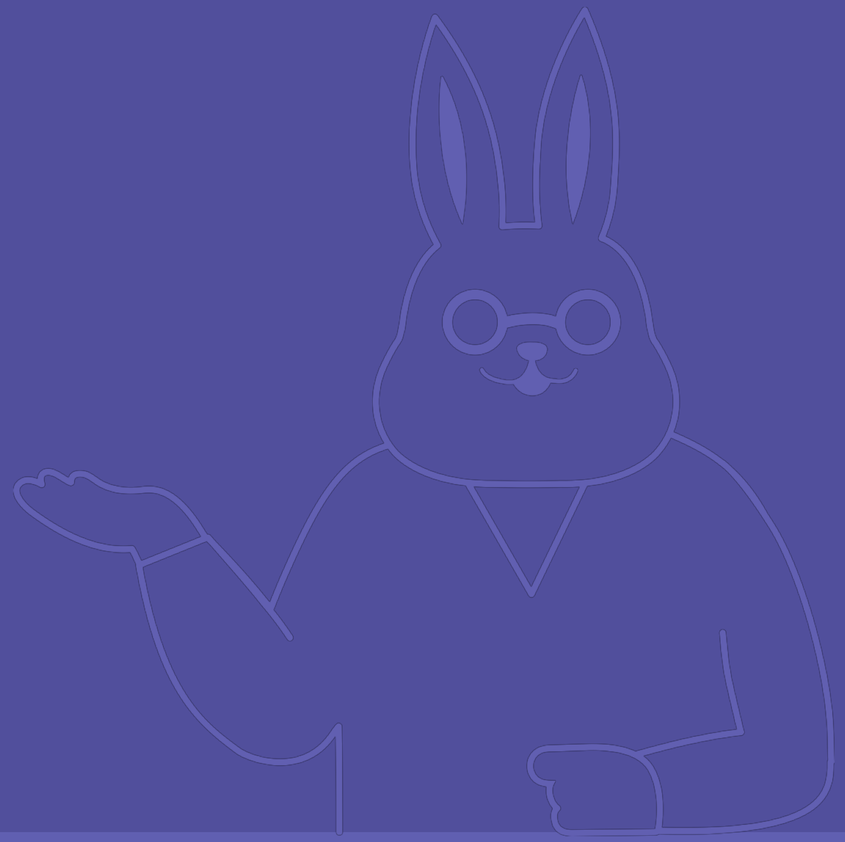
한국어에서 단어란



한국어 자연어 처리에서는 단어의 **의미적 기능**과 **문법적인 기능**을 **구분**하는 것이 중요

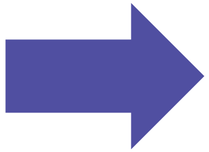
02

KoNLPy



형태소 분석

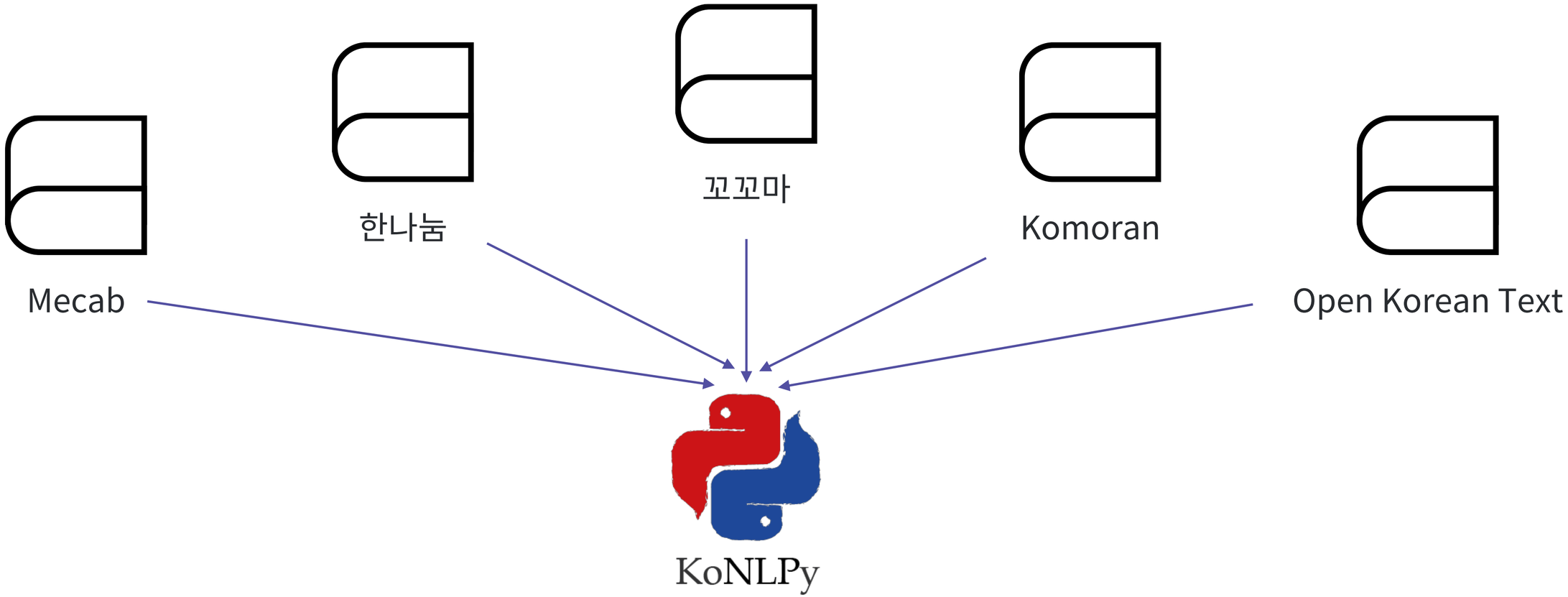
엘리스는
엘리스가
엘리스에게



엘리스 + 는
엘리스 + 가
엘리스 + 에게

형태소 분석이란 주어진 한국어 텍스트를 단어의 원형 형태로 분리해 주는 작업

KoNLPy



KoNLPy는 여러 한국어 형태소 사전을 기반으로
한국어 단어를 추출해 주는 파이썬 라이브러리

KoNLPy

Example

```
from konlpy.tag import Kkma, Okt

kkma = Kkma()
okt = Okt()
```

각 형태소 분석기 호출 방식:

- Hannanum( )
- Kkma( )
- Komoran(userdict=경로)
- Mecab( )
- Okt( )

KoNLPy

Example

```
from konlpy.tag import Kkma

sent = "안녕 나는 엘리스야 반가워. 너의 이름은 뭐야?"

kkma = Kkma()

print(kkma.nouns(sent)) # ['안녕', '나', '엘리스', '너', '이름', '뭐']
print(kkma.pos(sent)) # [('안녕', 'NNG'), ('나', 'NP'), ('는', 'JX'),
                      ('엘리스', 'NNG'), ('야', 'JX'), ...
print(kkma.sentences(sent)) # ['안녕 나는 엘리스야 반가워. 너의 이름은 뭐야?']
```

KoNLPy

Example

```
from konlpy.tag import Okt

sent = "안녕 나는 엘리스야 반가워. 너의 이름은 뭐야?"

okt = Okt()

print(okt.nouns(sent)) # ['안녕', '나', '엘리스', '너', '이름', '뭐']
print(okt.pos(sent)) # [('안녕', 'Noun'), ('나', 'Noun'), ('는', 'Josa'),
                        ('엘리스', 'Noun'), ...
print(okt.pos(sent, stem = True)) # ... ('반갑다', 'Adjective') ...
```

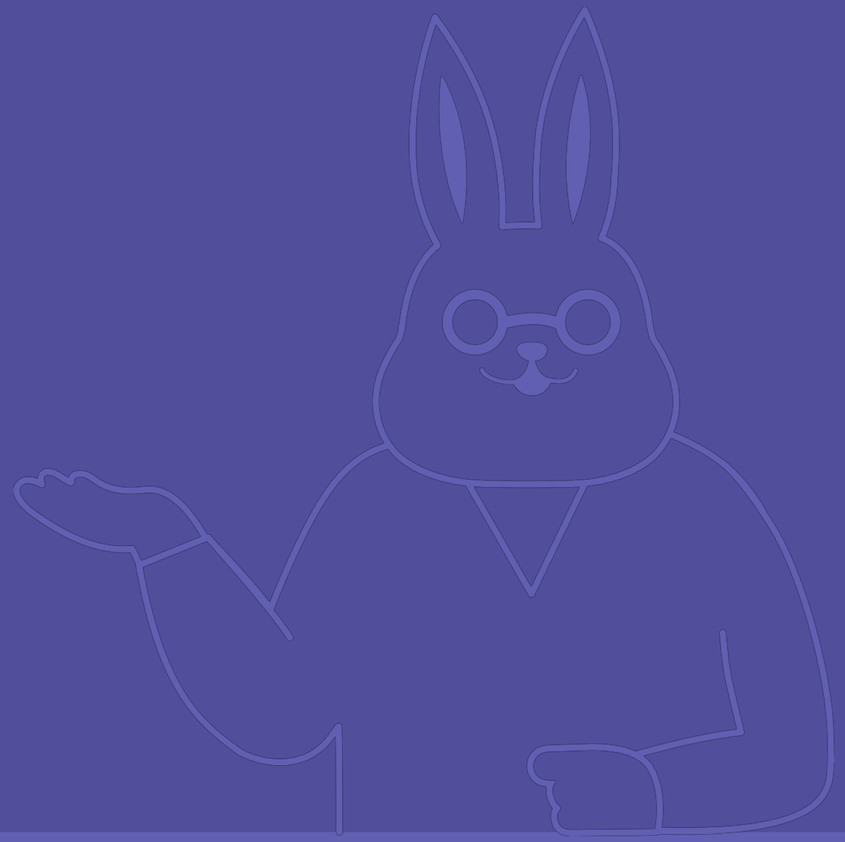
KoNLPy

Kkma (ntags=10)	Kkma (ntags=30)	Kkma (ntags=56)		Hannanum (ntags=9)	
Tag	Tag	Tag	Description	Tag	Description
		NNG	보통명사		
		NNP	고유명사		
		NNB	일반 의존 명사		
	NN	NNM	단위 의존 명사		
	NR	NR	수사		
N	NP	NP	대명사	N	체언

각 형태소 사전별 형태소 **표기 방법 및 기준의 차이**가 존재

03

soynlp



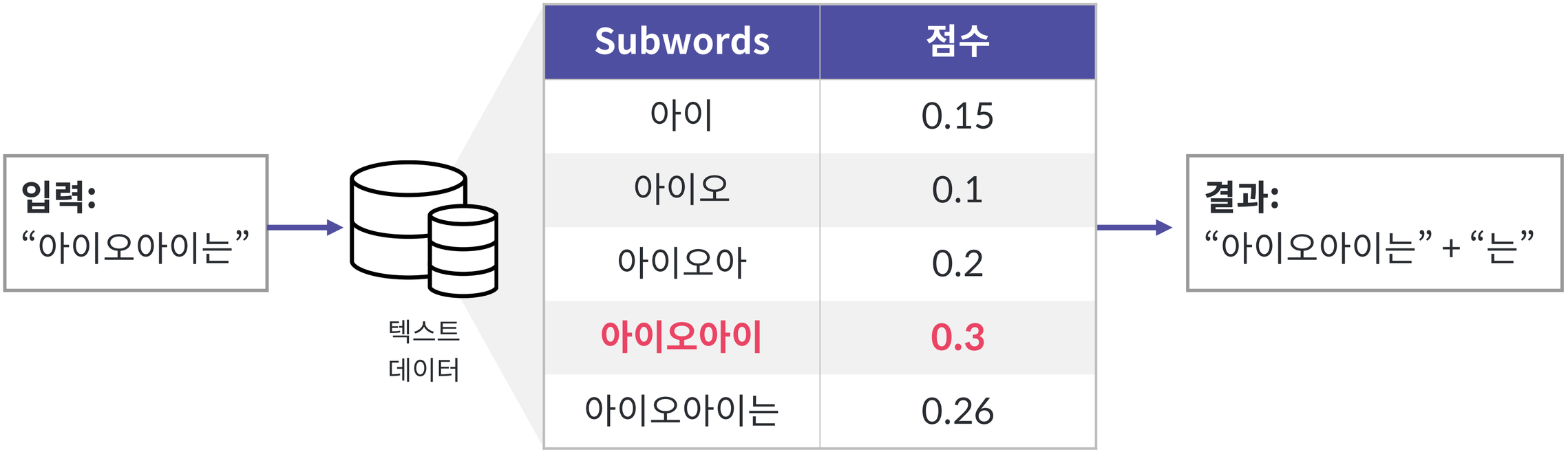
soynlp

[문장 1]: 보코하람 테러로 소말리아에서 전쟁이 있었어요

꼬꼬마 기준 추출된 명사: [보, 보코, 코, 테러, 소말리, 전쟁]
OKT 기준 추출된 명사: [보코하람, 테러, 소말리아, 전쟁]

사전 기반의 단어 처리의 경우, **미등록 단어 문제**가 발생할 수 있음

soynlp



soynlp는 학습 데이터 내 자주 발생하는 패턴을 기반으로 단어의 경계선을 구분

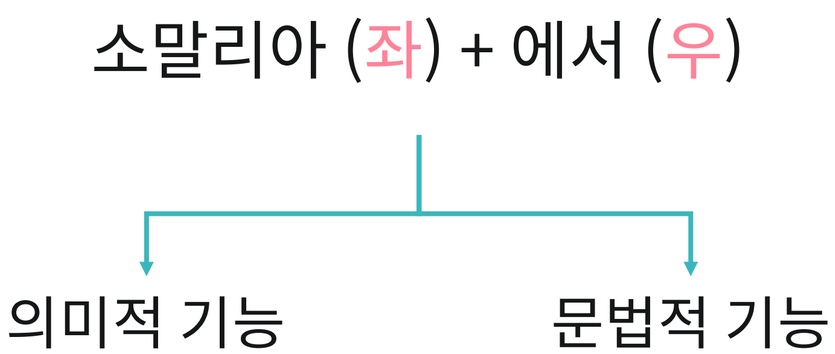
soynlp

[문장 1]: 보코하람 테러로 소말리아에서 전쟁이 있었어요

단어는 **연속으로 등장하는 글자의 조합**이며 글자 간 연관성이 높다는 가정

soynlp

[문장 1]: 보코하람 테러로 소말리아에서 전쟁이 있었어요



한국어의 어절은 **좌 – 우 구조**로 2등분 할 수 있다

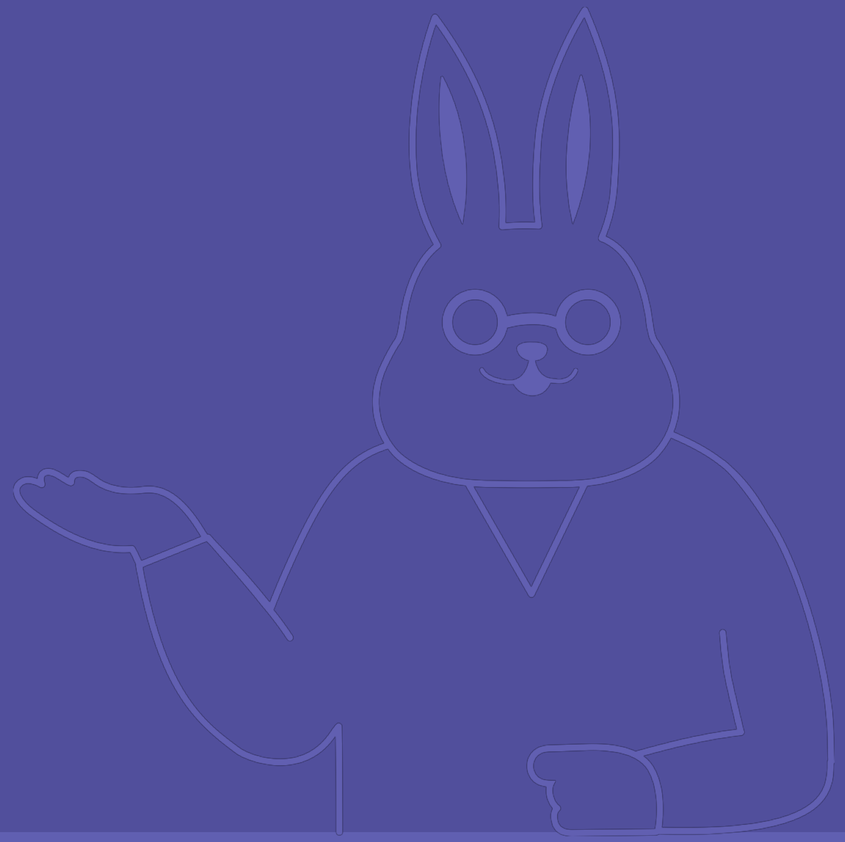
soynlp

Example

```
from soynlp.utils import DoublespaceLineCorpus
from soynlp.word import WordExtractor
from soynlp.noun import LRNounExtractor_v2

train_data = DoublespaceLineCorpus(학습데이터의 경로) # 데이터 기반 패턴 학습

noun_extractor = LRNounExtractor_v2()
nouns = noun_extractor.train_extract(train_data) # [할리우드, 모바일게임 ...

word_extractor = WordExtractor()
words = word_extractor.train_extract(train_data) # [클린턴, 트럼프, 프로그램 ...
```

04

문장 유사도



문장 유사도

[문장 1] : 오늘은 중부지방을 중심으로 소나기가 예상됩니다.

[문장 2] : 오늘은 전국이 맑은 날씨가 예상됩니다.

[문장 3] : 앞으로 접종 속도는 빨라질 것으로 예상됩니다.

문장 간 유사도는 **공통된 단어** 혹은 **의미**를 기반으로 계산

자카드 지수

$$\text{문장 1과 문장 2의 유사도} = \frac{\text{(두 문장 내 공통된 단어의 종류)}}{\text{(두 문장 내 모든 단어의 종류)}}$$

자카드(Jaccard) 지수는 문장 간 **공통된 단어**의 비율로 문장 간 유사도를 정의

자카드 지수

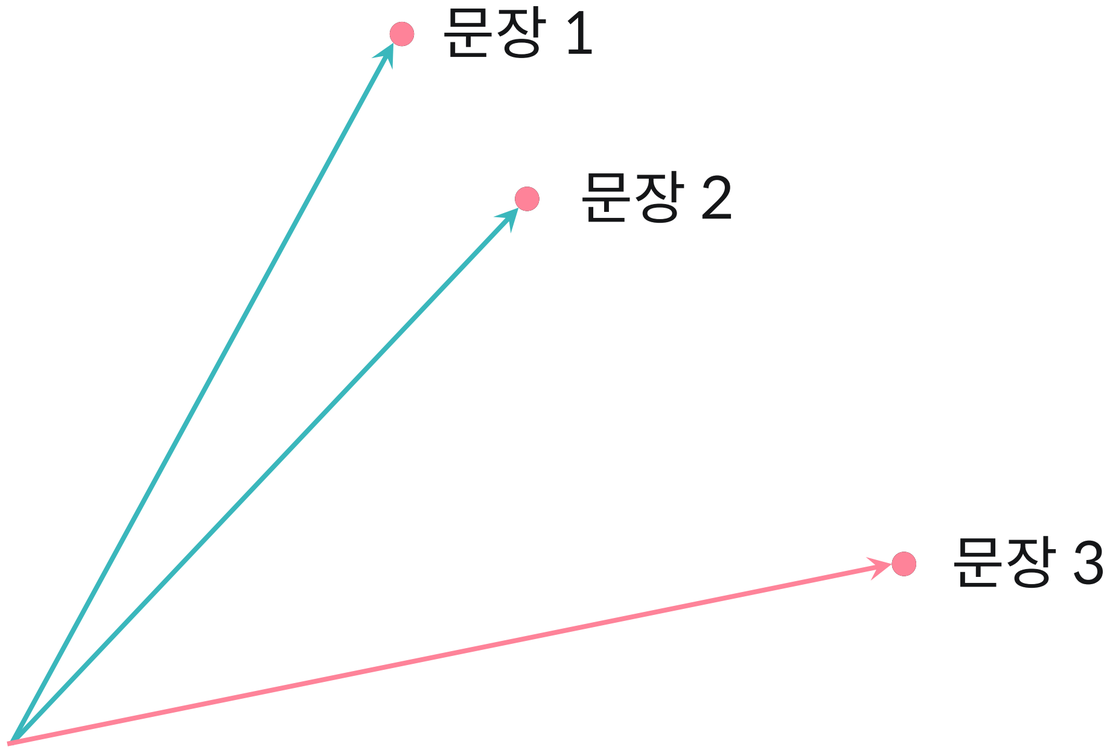
[문장 1] : 오늘은 중부지방을 중심으로 소나기가 예상됩니다.

[문장 2] : 오늘은 전국이 맑은 날씨가 예상됩니다.

$$\text{문장 1과 문장 2의 유사도} = \frac{2}{8} = 0.25$$

자카드 지수는 문장 간 유사도를 **0 ~ 1** 사이로 정의

코사인 유사도



코사인 유사도는 **문장 벡터 간의 각도**를 기반으로 계산

코사인 유사도

$$A = [1, 3], B = [0, 2]$$

$$\text{A와 B의 코사인 유사도} = \frac{A \cdot B}{\|A\|\|B\|} = \frac{(1 \times 0) + (3 \times 2)}{\sqrt{1^2 + 3^2} \times \sqrt{0^2 + 2^2}} = \frac{6}{2\sqrt{10}} \approx 0.9487$$

벡터 간의 각도는 **벡터 간 내적**을 사용해서 계산

코사인 유사도

$$d(\mathbf{p}, \mathbf{q}) = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}$$

유클리드 거리와 같은 다양한 거리 지표가 존재

코사인 유사도

$$A = [1, 3], B = [0 ,2]$$

$$A = [1, 3, 23, 12, 3, \ldots], B = [0 ,2, 32, 78, 65, \ldots]$$

코사인 유사도는 고차원의 공간에서 **벡터 간의 유사성**을 잘 보존하는 장점이 있음