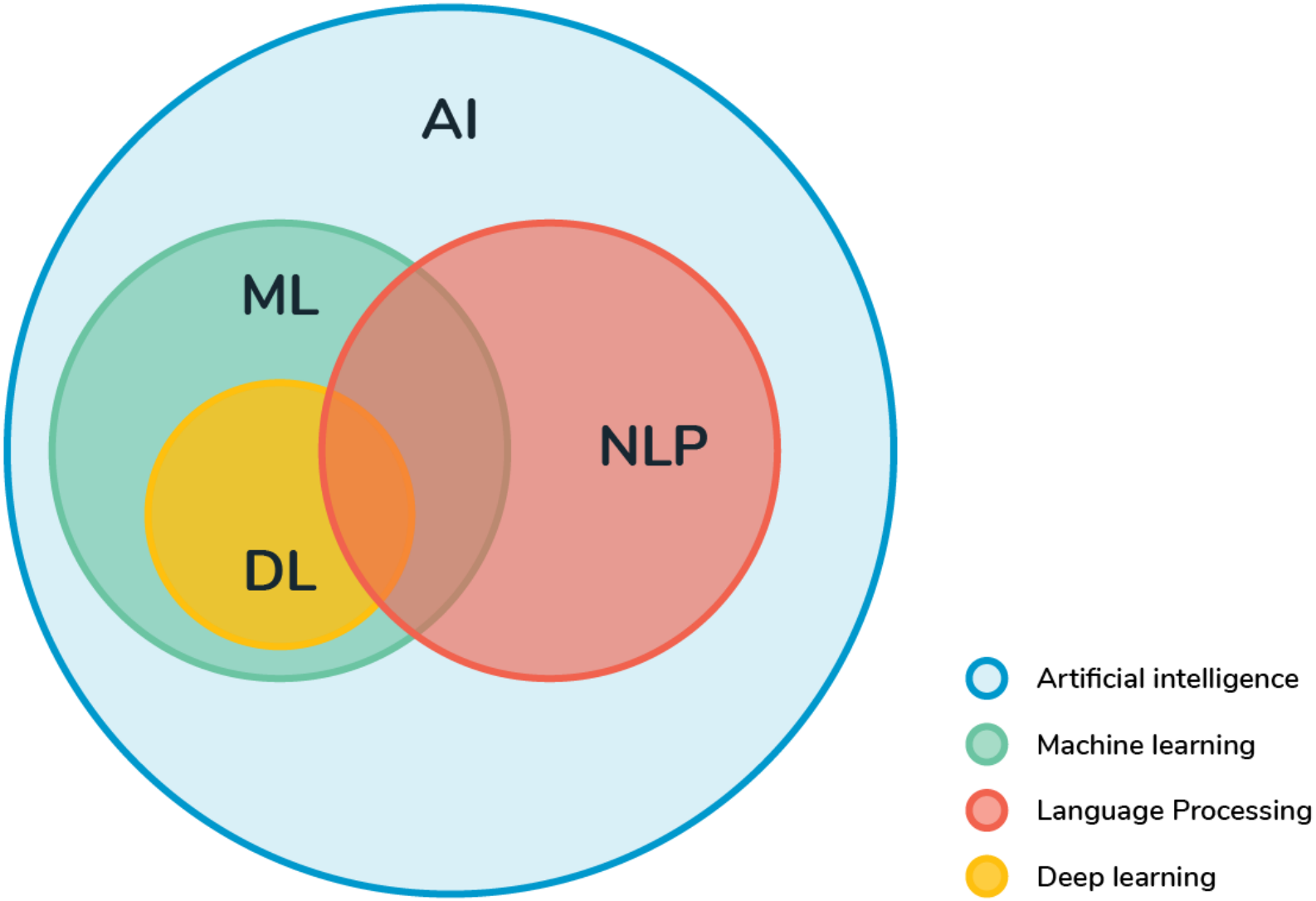
Preprocessing


AI
ML
NLP
DL
Artificial intelligence
Machine learning
Language Processing
Deep learning

자연어 분석

- 형태소 분석
- 구문 분석
- 의미 분석
- 담화 분석
- 중의성 해소

뭘 할 수 있을까..?

응용 기술

- 검색
- 온라인 광고
- 자동번역
- 감정분석
- 음성인식
- 맞춤법검사

토큰 분리, 어간 추출, 품사 부착, 색인, 벡터화

문장 경계 인식, 구문분석, 공기어, 개체명 사전 구축(PLOT, 수치, 외국어 한글 표기), 개체명 인식

대용어 해소(대명사, 두문자어, 약어, 수치), 의미 중의성 해결(동명이인, 이명동인)

분류, 군집, 중복, 요약, 가중치, 순위화, 토픽 모델링, 이슈 트래킹, 평판분석, 감성분석, 복합논증분석,

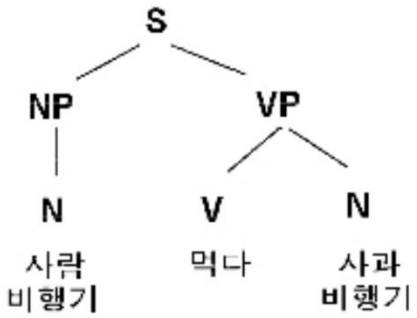
형태소 분석 → 구문 분석 → 의미 분석 → 담화 분석

입력된 문장을 형태소 단위로 분할하고 품사를 부착

1) 나는
- 나+는
- 날(다)+는
- 나(다)+는

2) 과학자들에게
- 과학자 + 들 + 에게

주어, 목적어, 서술어와 같은 구문단위를 찾음



문장이 의미적으로 올바른 문장인지를 판단

1) 사람이 사과를 먹는다. (o)
2) 사람이 비행기를 먹는다. (x)
3) 비행기가 사과를 먹는다. (x)

대화 흐름상 어떤 의미를 가지는지를 찾음
- 문맥구조 분석
  (문장들의 연관 관계),
- 의도분석
  (전후관계를 통한 실제 의도)

1) 철수는 어항을 떨어뜨렸다.
   그는 울고 말았다.

2) 철수는 우승을 했다.
   그는 울고 말았다.

담화 문장이 연속되어 이루어지는 말의 단위, 문단

나는 급하게 옷을 주워 입었다. 개미 한 마리가 방바닥을 내 발이 있는 쪽으로 기어오고 있었다. 그 개미가 내 발을 붙잡으려고 하는 것 같은 느낌이 들어서 나는 얼른 자리를 옮겨 디디었다.

밖의 이른 아침에는 싸락눈이 내리고 있었다. 우리는 할 수 있는 한 빠른 걸음으로 여관에서 떨어져 갔다.

"난 그 사람이 죽으리라는 걸 알고 있었습니다."

안이 말했다.

"난 짐작도 못했습니다."

라고 나는 사실대로 얘기했다.

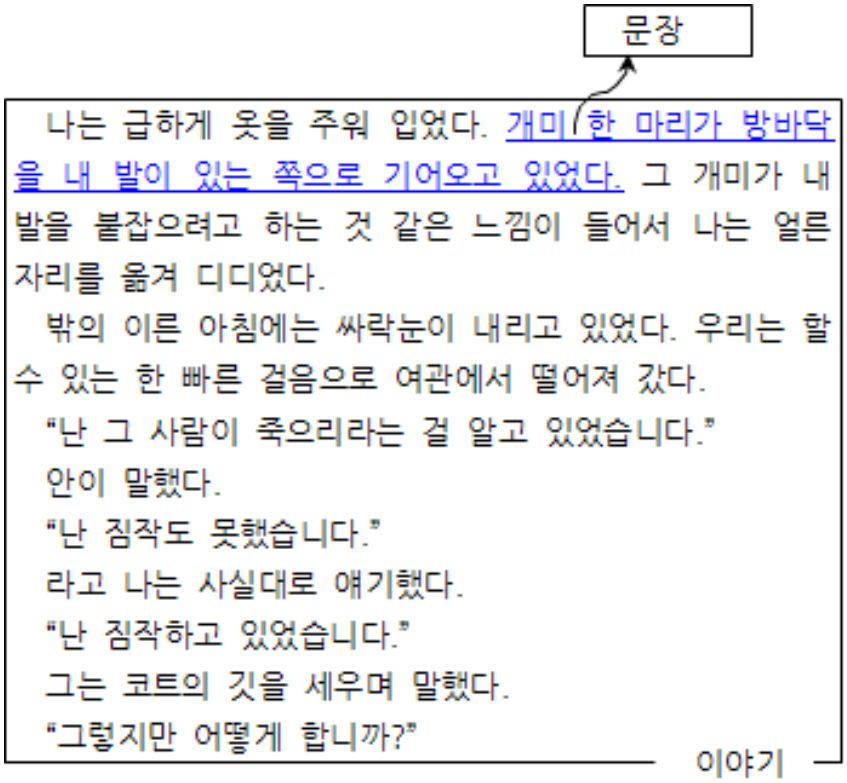
"난 짐작하고 있었습니다."

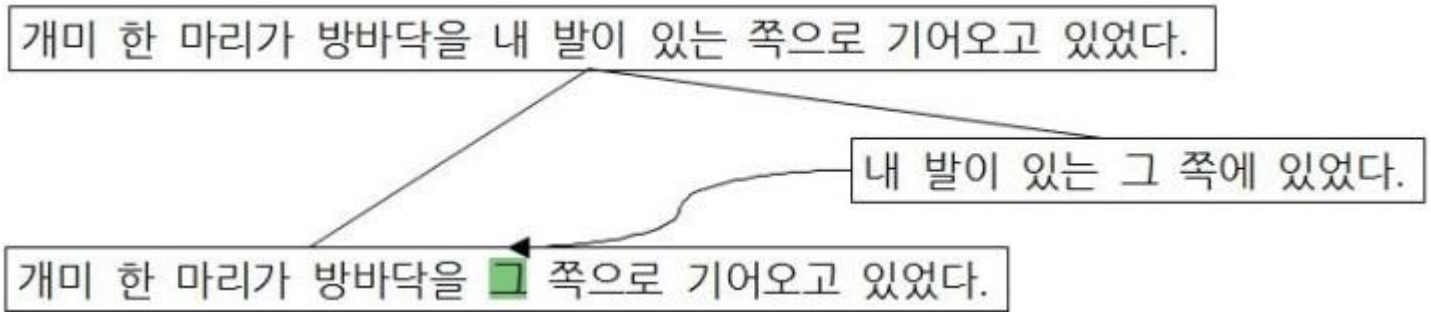
그는 코트의 깃을 세우며 말했다.

"그렇지만 어떻게 합니까?"

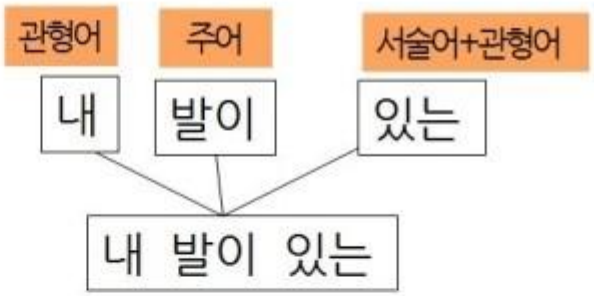
— 이야기

문장　　완결된 내용을 나타내는 최소 단위

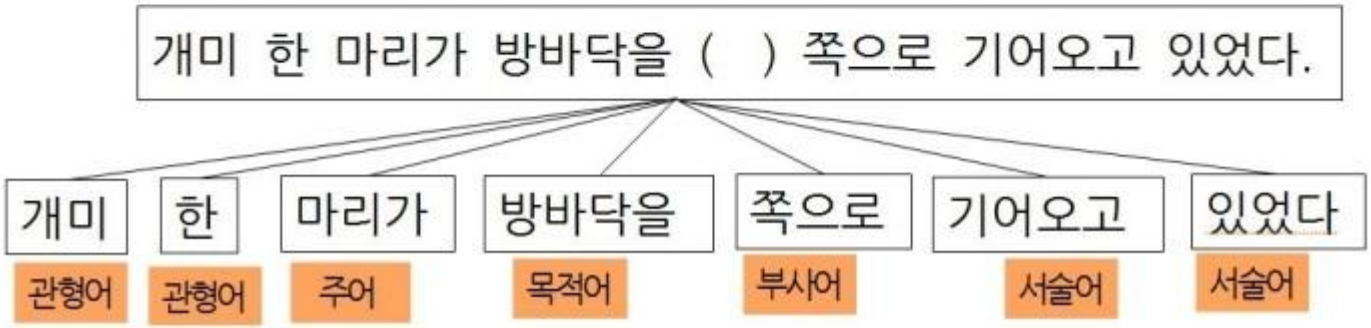




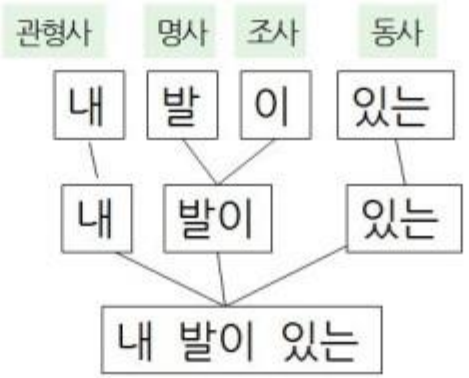
어절 문장을 구성하는 단위



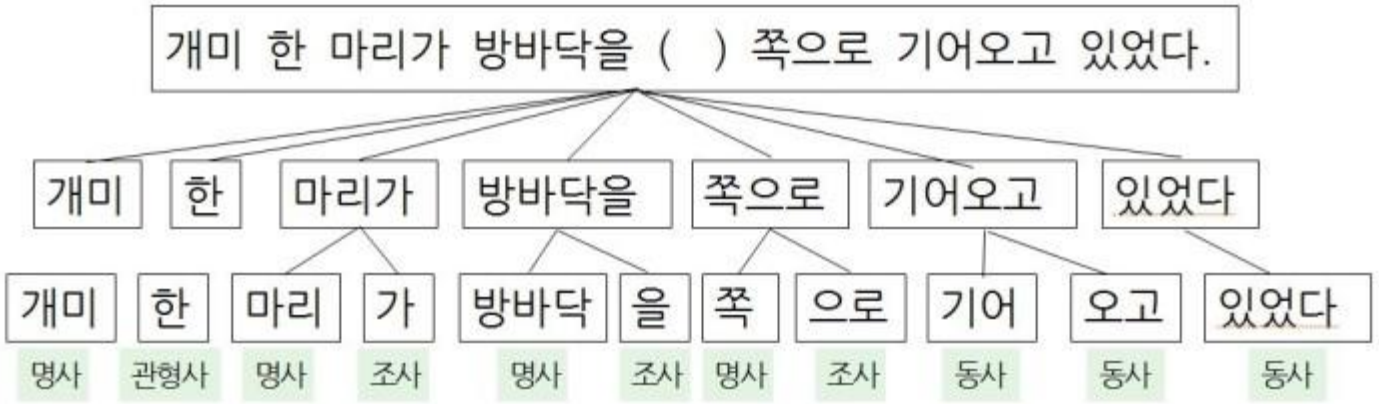
개미 한 마리가 방바닥을 내 발이 있는 쪽으로 기어오고 있었다.



단어 어절을 구성하는 요소

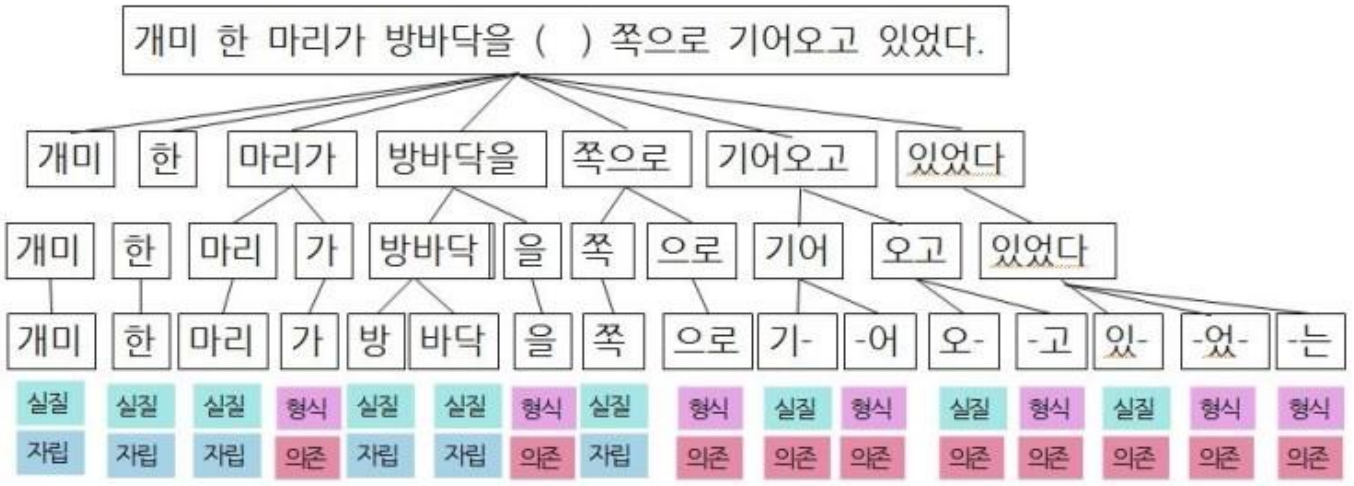


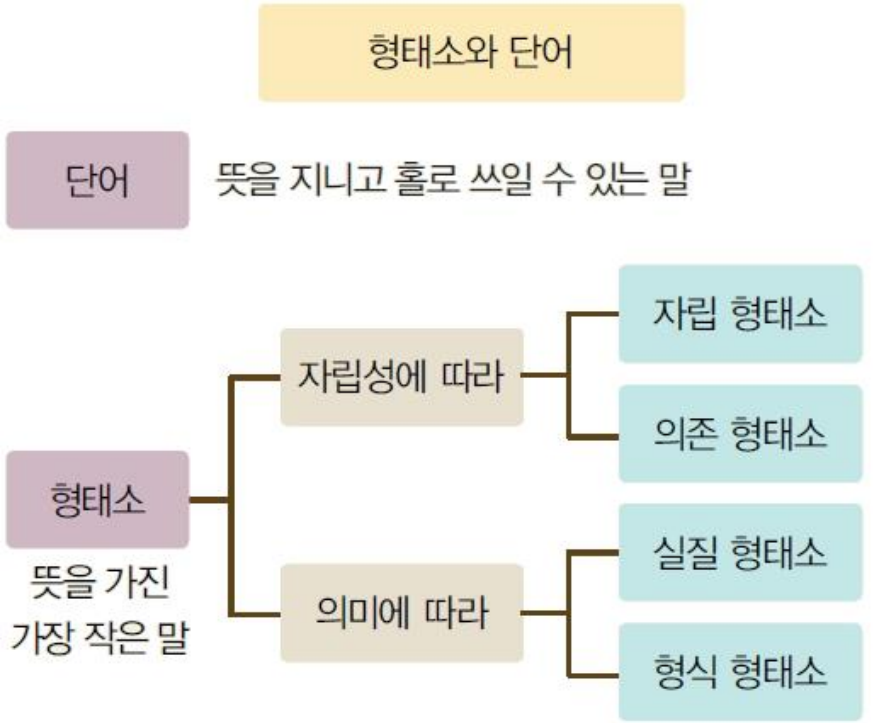
개미 한 마리가 방바닥을 내 발이 있는 쪽으로 기어오고 있었다.

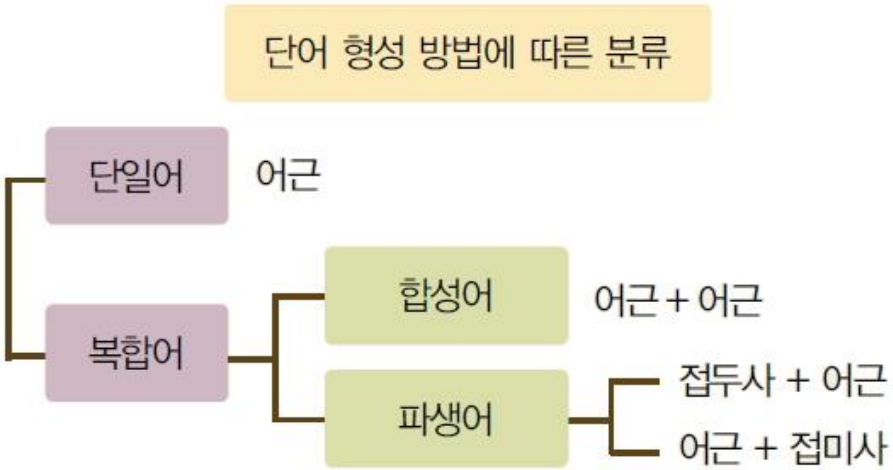


형태소 의미를 가진 문법의 최소 단위

개미 한 마리가 방바닥을 내 발이 있는 쪽으로 기어오고 있었다.




형태소와 단어
단어
뜻을 지니고 홀로 쓰일 수 있는 말
형태소
뜻을 가진
가장 작은 말
자립성에 따라
의미에 따라
자립 형태소
의존 형태소
실질 형태소
형식 형태소


단어 형성 방법에 따른 분류
단일어
어근
복합어
합성어
어근 + 어근
파생어
접두사 + 어근
어근 + 접미사

KoNLPy, NLTK

KoNLPy

KoNLPy 한국어 정보처리를 위한 파이썬 패키지

Corpus, Morpheme Analyzer, POS Tagging, …

http://konlpy.org/ko/latest/

JDK 1.8 이상

JPype1 0.5.7 이상

Acknowledgement

박은정, 조성준, "KoNLPy: 쉽고 간결한 한국어 정보처리 파이썬 패키지",
제 26회 한글 및 한국어 정보처리 학술대회 논문집, 2014

Corpus

다음의 말뭉치(corpus)를 사용할 수 있습니다:

1. `kolaw`: 한국 법률 말뭉치.
    - constitution.txt
2. `kobill`: 대한민국 국회 의안 말뭉치. 파일 ID는 의안 번호를 의미합니다.
    - 1809890.txt - 1809899.txt

```
!pip install konlpy
```

```
from konlpy.corpus import kolaw

kolaw.fieids()

corpus = kolaw.open('constitution.txt').read()

print(len(corpus.split()))
print(corpus.splitlines()[:3])
```

NLTK

NLTK Building Python programs to work with human language data

Provides easy-to-use interfaces to over 50 corpora and lexical resources

classification, tokenization, stemming, tagging, parsing, and semantic reasoning

Natural Language Analysis
with Python NLTK

Corpus

```
!pip install nltk
```

```
import nltk

nltk.download('brown')
nltk.download('gutenberg')
```

```
from nltk.corpus import brown, gutenberg

corpus = brown.open('ca01').read()
print(len(corpus.split()))
print(corpus.splitlines()[:3])

corpus = gutenberg.open('austen-emma').read()
print(len(corpus.split()))
print(corpus.splitlines()[:3])
```

Tokenizing

`nltk.tokenize.sent_tokenize(text, language='english')`

Return a sentence-tokenized copy of *text*, using NLTK's recommended sentence tokenizer (currently `PunktSentenceTokenizer` for the specified language).

Parameters:

- **text** – text to split into sentences
- **language** – the model name in the Punkt corpus

```
from nltk.tokenize import sent_tokenize

nltk.download('punkt')

sentences = sent_tokenize(corpus)
print(len(sentences.split()))
print(sentences [:3])
```

`nltk.tokenize.word_tokenize(text, language='english', preserve_line=False)`

Return a tokenized copy of *text*, using NLTK's recommended word tokenizer (currently an improved `TreebankWordTokenizer` along with `PunktSentenceTokenizer` for the specified language).

Parameters:

- **text** (*str*) – text to split into words
- **language** (*str*) – the model name in the Punkt corpus
- **preserve_line** – An option to keep the preserve the sentence and not sentence tokenize it.

```
from nltk.tokenize import word_tokenize

words = word_tokenize(sentences[0])
print(len(words))
```

Empirical Law

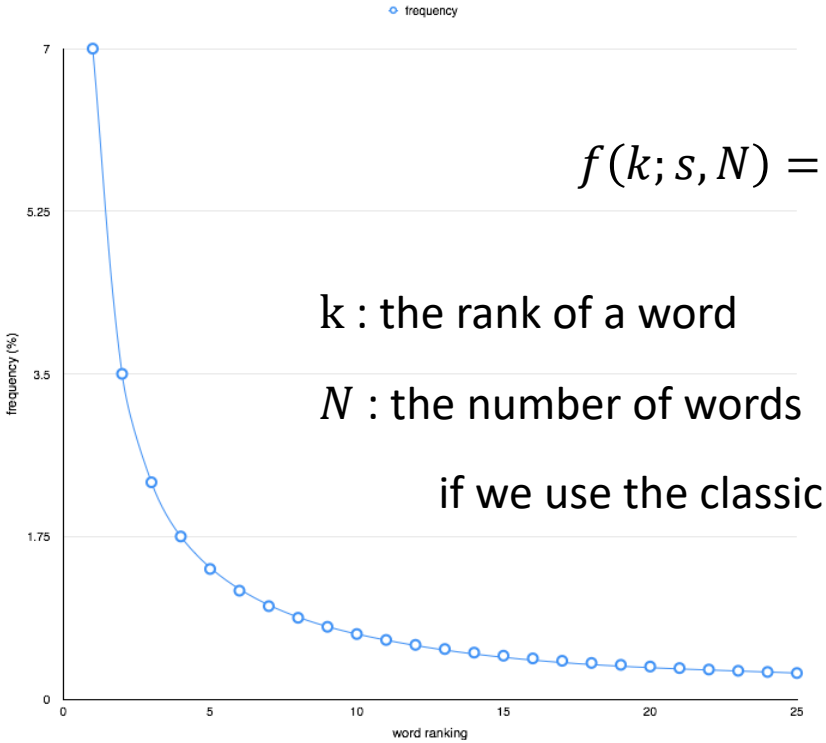
Zipf's Law

WHAT **An empirical law**

frequency of word is inversely proportional to its rank in frequency table

most frequent word will occur approximately twice

as often as the second most frequent word

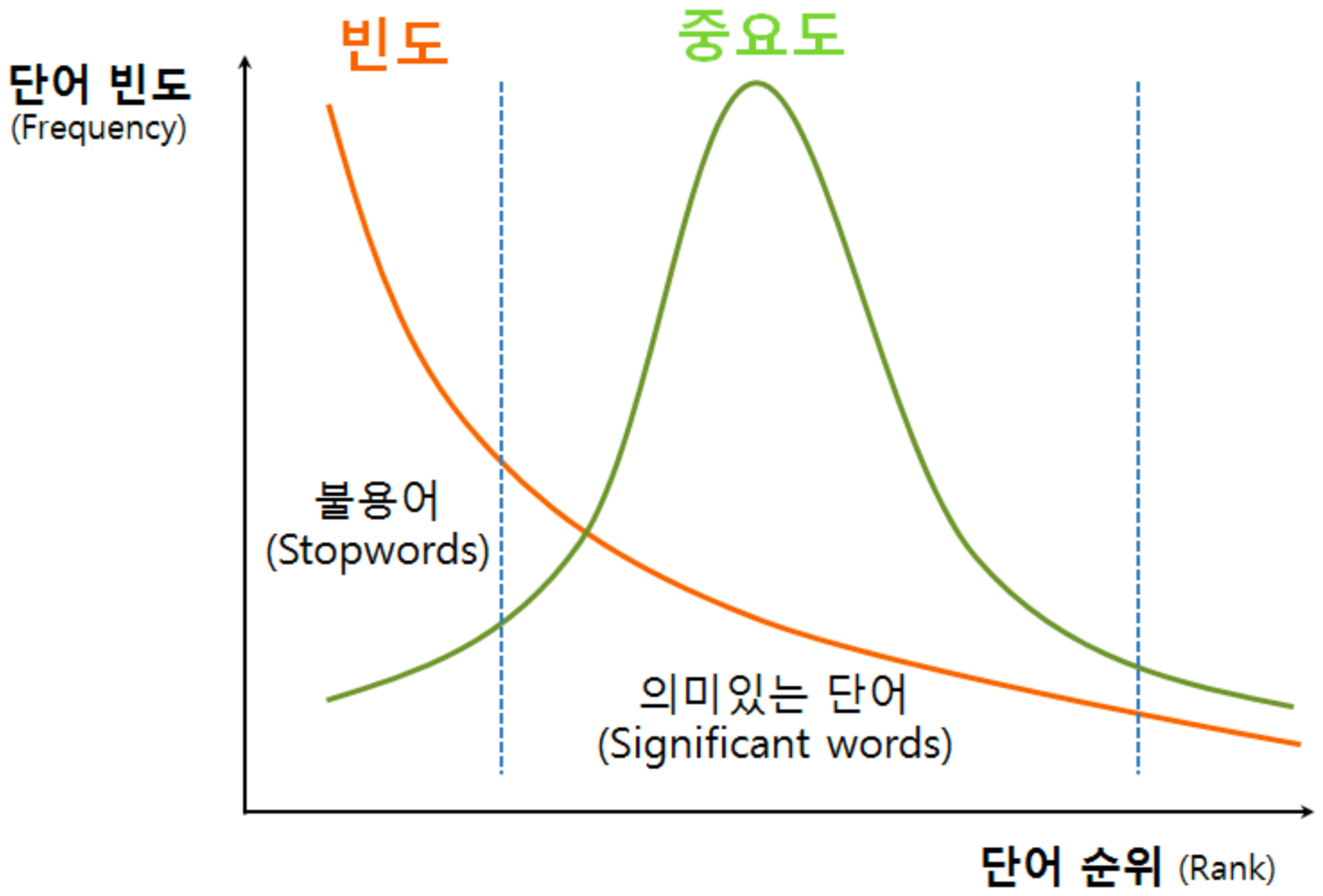


$$f(k;s,N) = \frac{1/k^s}{\sum_{n=1}^{N}(1/n^s)} \Leftrightarrow \frac{1}{k^s H_{n,s}}$$

k : the rank of a word

N : the number of words

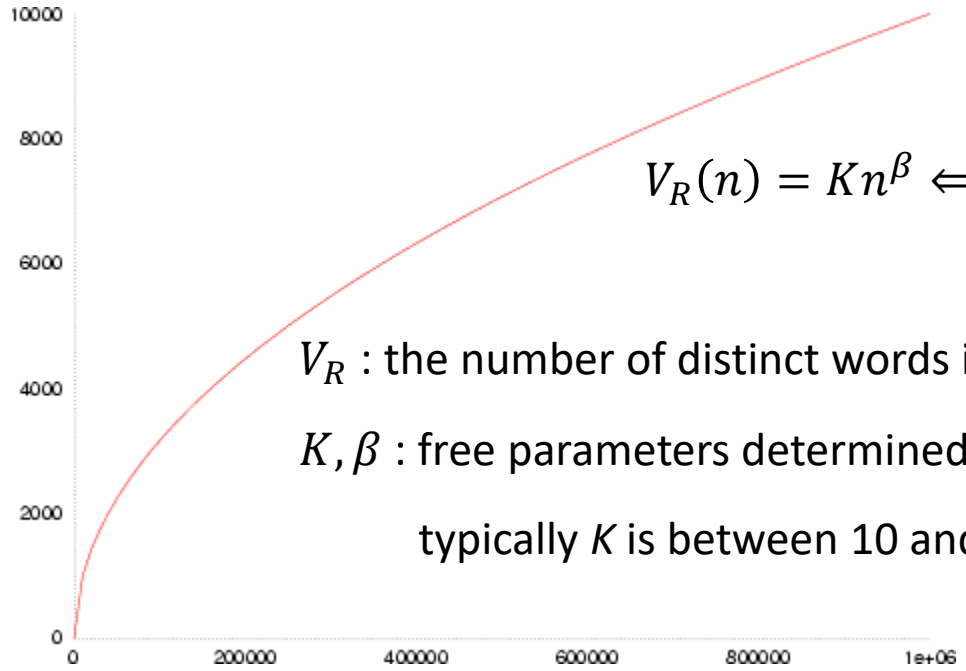
if we use the classic version of Zipf's law, the exponent s is 1


빈도
중요도
단어 빈도
(Frequency)
불용어
(Stopwords)
의미있는 단어
(Significant words)
단어 순위 (Rank)

Heaps' Law

WHAT **An empirical law**

# of distinct words in a document as a function of the document length



$$V_R(n) = Kn^{\beta} \Leftrightarrow M = kT^b$$

V_R : the number of distinct words in an instance text of size *n*

K, β : free parameters determined empirically

typically *K* is between 10 and 100, and β is between 0.4 and 0.6

n-gram

n-gram

WHAT **Contiguous sequence** of ***n*** items from a given sample of text or speech

phonemes, ***syllables***, *letters*, ***words*** or *base pairs* according to the application

WHY **Predicts** a next letter

Given a sequence of letters, what is the likelihood of the next letter?

$x_{i-(n-1)}, \dots, x_{i-1}$ $\quad$ x_i

$$\boldsymbol{P(x_i | x_{i-(n-1)}, \dots, x_{i-1})}$$

WPM

Word Piece Model

하나의 단어를 내부 단어(Subword Unit)들로 분리하는 단어 분리 모델

Google's Neural Machine Translation System:
Bridging the Gap between Human and Machine Translation
Wo at el, 2016

BPE

Byte Pair Encoding (Digram Coding)

Simple form of data compression

The most common pair of consecutive bytes of data is replaced with a byte

Philip Gage, 1994

Neural Machine Translation of Rare Words with Subword Units
Sennrich at el, 2015