

Python 영문 형태소 분석기(NLTK) 소개 및 설치

NLTK 패키지

- 교육용으로 개발된 자연어 처리 및 문서 분석용 파이썬 패키지

NLTK 패키지 주요 기능

- 코퍼스(Corpus), 토큰화(Tokenizing), 형태소 분석 등의 기능 제공
- 텍스트 분석에 유용한 다양한 메서드를 포함하는 Text 클래스 제공

NLTK 패키지 설치

- `pip install nltk`

영문 형태소 분석기(NLTK) 사용

- 저작권이 말소된 문학작품을 포함하는 gutenberg 코퍼스 다운로드

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

- gutenberg 코퍼스 목록

```
from nltk.corpus import gutenberg  
print(gutenberg.fileids())
```

```
['austen-emma.txt',  
'austen-persuasion.txt',  
'austen-sense.txt',  
'bible-kjv.txt',  
'blake-poems.txt',  
'bryant-stories.txt',  
'burgess-busterbrown.txt',  
'carroll-alice.txt',  
'chesterton-ball.txt',  
'chesterton-brown.txt',  
'chesterton-thursday.txt',  
'edgeworth-parents.txt',  
'melville-moby_dick.txt',  
'milton-paradise.txt',  
'shakespeare-caesar.txt',  
'shakespeare-hamlet.txt',  
'shakespeare-macbeth.txt',  
'whitman-leaves.txt']
```

영문 형태소 분석기(NLTK) 사용

- hamlet 코퍼스 로드

```
hamlet = nltk.corpus.gutenberg.raw('shakespeare-hamlet.txt')  
print(hamlet[:1000])
```

[The Tragedie of Hamlet by William Shakespeare 1599]

Actus Primus. Scoena Prima.

Enter Barnardo and Francisco two Centinels.

Barnardo. Who's there?

Fran. Nay answer me: Stand & vnfold
your selfe

Bar. Long liue the King

Fran. Barnardo?

Bar. He

Fran. You come most carefully vpon your houre

Bar. 'Tis now strook twelue, get thee to bed Francisco

Fran. For this releefe much thanks: 'Tis bitter cold,
And I am sicke at heart

Barn. Haue you had quiet Guard?

Fran. Not

토큰화 (Tokenizing)

코퍼스를 문장(Sentence),
단어(Word) 단위 등의 작은 단위로
분리하는 과정



문장 토큰화

코퍼스를 문장 단위로 토큰화

```
from nltk.tokenize import sent_tokenize  
print(sent_tokenize(hamlet[:100]))
```

단어 토큰화

코퍼스를 단어 단위로 토큰화

```
from nltk.tokenize import word_tokenize  
print(word_tokenize(hamlet[:100]))
```

영문 형태소 분석

- Tagger 다운로드

```
import nltk
nltk.download('averaged_perceptron_tagger')
```

- 품사 태깅

```
from nltk.tag import pos_tag
sentence = "You come most carefully vpon your houre"
tagged_list = pos_tag(word_tokenize(sentence))
print(tagged_list)
```

```
[('You', 'PRP'),
 ('come', 'VBP'),
 ('most', 'RBS'),
 ('carefully', 'RB'),
 ('vpon', 'VB'),
 ('your', 'PRP$'),
 ('houre', 'NN')]
```

- 품사 태깅 제거

```
from nltk.tag import untag
untag(tagged_list)
```

```
['You', 'come', 'most', 'carefully', 'vpon', 'your', 'houre']
```

한글 형태소 분석기 설치(Python, Konlpy)

- Java JDK 설치
 - dJDK 1.8 설치
 - 이미 설치되어 있는 경우 생략 가능



- JPytype1 설치

```
pip install JPytype1
```

- Konlpy 설치

```
pip install konlpy
```

한글 형태소 분석

한글 품사태깅 클래스 종류

Konlpy의 한글 품사태깅 클래스 종류

Kkma

Komoran

Hannanum

Okt
(previous Twitter)

Mecab



Windows 환경에서는 Mecab 미지원, 별도 설치 필요

품사태깅 클래스 품사태깅 성능 비교

Hannanum	Kkma	Komorán	Mecab	Twitter
아버지가방에 들어가 / N	아버지 / NNG	아버지가방에 들어가신다 / 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	



<출처> <http://asq.kr/3AqqEJFYQsIK>

수행시간 성능 비교

품사 태깅 클래스	클래스 로딩 시간 (secs)	10만 문자 품사태깅 실행시간 (secs)
Kkma	5.6988	35.7163
Komoran	5.4866	25.6008
Hannanum	0.6591	8.8251
Twitter	1.487	2.4714
Mecab	0.0007	0.2838



Konlpy를 이용한 한글 형태소 분석

- Konlpy 클래스 import
 - `from konlpy.tag import Okt, Kkma, Komoran, Hannanum`
- 사용하고자 하는 클래스 객체 생성
 - `tagger = Okt()`
 - `tagger = Kkma()`
 - `tagger = Komoran()`
 - `tagger = Hannanum()`

Konlpy를 이용한 한글 형태소 분석

- 형태소 분석 수행
 - `tagger.taggerhs(txt)`
- 명사만 추출
 - `tagger.nouns(txt)`
- 품사 태깅
 - `tagger.pos(txt)`

한글 형태소 분석

- 파이썬에서 Konlpy 사용 실습

```
from eunjeon import Mecab
```

```
tagger = Mecab()
```

```
txt = '나는 하늘을 나는 비행기를 보았습니다.'
```

```
tagger.morphs(txt)
```

```
['나', '는', '하늘', '을', '나', '는', '비행기', '를', '보', '았', '습니다', '.']
```

```
tagger.nouns(txt)
```

```
['나', '하늘', '나', '비행기']
```

```
tagger.pos(txt)
```

```
[('나', 'NP'),  
 ('는', 'JX'),  
 ('하늘', 'NNG'),  
 ('을', 'JKO'),  
 ('나', 'NP'),  
 ('는', 'JX'),  
 ('비행기', 'NNG'),  
 ('를', 'JKO'),  
 ('보', 'VV'),  
 ('았', 'EP'),  
 ('습니다', 'EF'),  
 ('.', 'SF')]
```

Mecab 형태소 분석기

- pyeunjeon(python + eunjeon)은 은전한닢 프로젝트와 mecab 기반의 한국어 형태소 분석기의 독립형 python 인터페이스
- 윈도우즈 환경에서 Mecab 설치

```
pip install eunjeon
```

- eunjeon 패키지 사용

```
from eunjeon import Mecab  
tagger = Mecab()
```

- Mecab 형태소 분석기를 이용한 한글 형태소 분석 실습

```
from konlpy.tag import Okt, Kkma, Komoran, Hannanum
```

```
# tagger = Kkma()  
# tagger = Komoran()  
# tagger = Hannanum()  
tagger = Okt()
```

```
txt = '나는 하늘을 나는 비행기를 보았습니다.'
```

```
tagger.morphs(txt)
```

```
['나', '는', '하늘', '을', '나', '는', '비행기', '를', '보았습니다', '.']
```

```
tagger.nouns(txt)
```

```
['나', '하늘', '나', '비행기']
```

```
tagger.pos(txt)
```

```
[('나', 'Noun'),  
 ('는', 'Josa'),  
 ('하늘', 'Noun'),  
 ('을', 'Josa'),  
 ('나', 'Noun'),  
 ('는', 'Josa'),  
 ('비행기', 'Noun'),  
 ('를', 'Josa'),  
 ('보았습니다', 'Verb'),  
 ('.', 'Punctuation')]
```



Word2Vec

희소표현

(Sparse Representation)

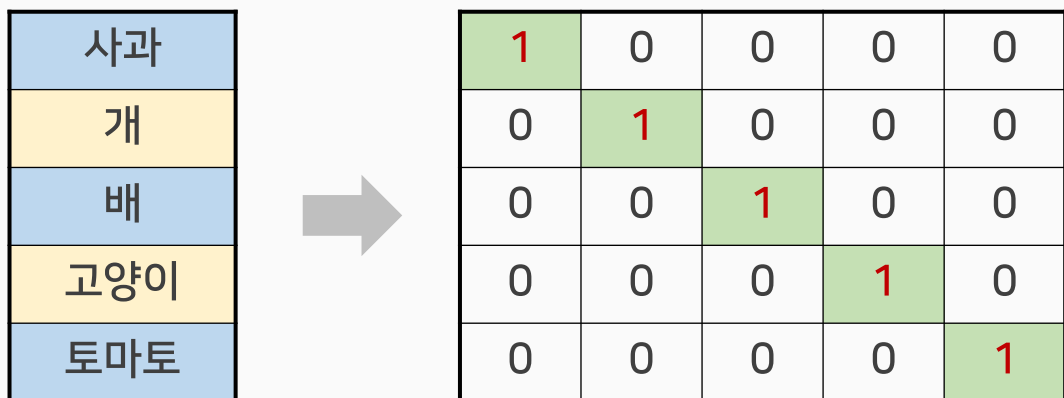
표현하고자 하는 단어의
인덱스의 값만 1으로 표현하고
나머지 인덱스에는 모두 0으로
표현되는 벡터 표현 방법
(One Hot Encoding)



One Hot Encoding

전체 레이블(label) 수에 대해 표현하고자 하는 특정 단어(Word)만
'1'로 활성화한 벡터로 변환하는 방법

- 단어의 의미와 관계를 전혀 고려하지 않음
- 벡터의 크기는 총 단어 수만큼의 Sparse Vector를 생성
- 정답(label)을 인식하게 하는 용도로 주로 사용



사과	1	0	0	0	0
개	0	1	0	0	0
배	0	0	1	0	0
고양이	0	0	0	1	0
토마토	0	0	0	0	1

분산 표현

(Distributed Representation)

분포 가설(Distributional Hypothesis) 가정하에
문자열을 표현하는 방법



예쁘다!

분포가설은 **비슷한 위치**에서 등장하는 단어들은
비슷한 의미를 가질 확률이 높다는 가설

예 강아지는 주로
'귀엽다', '예쁘다', '멍멍' 등의
단어와 함께 등장

귀엽다!

멍멍!

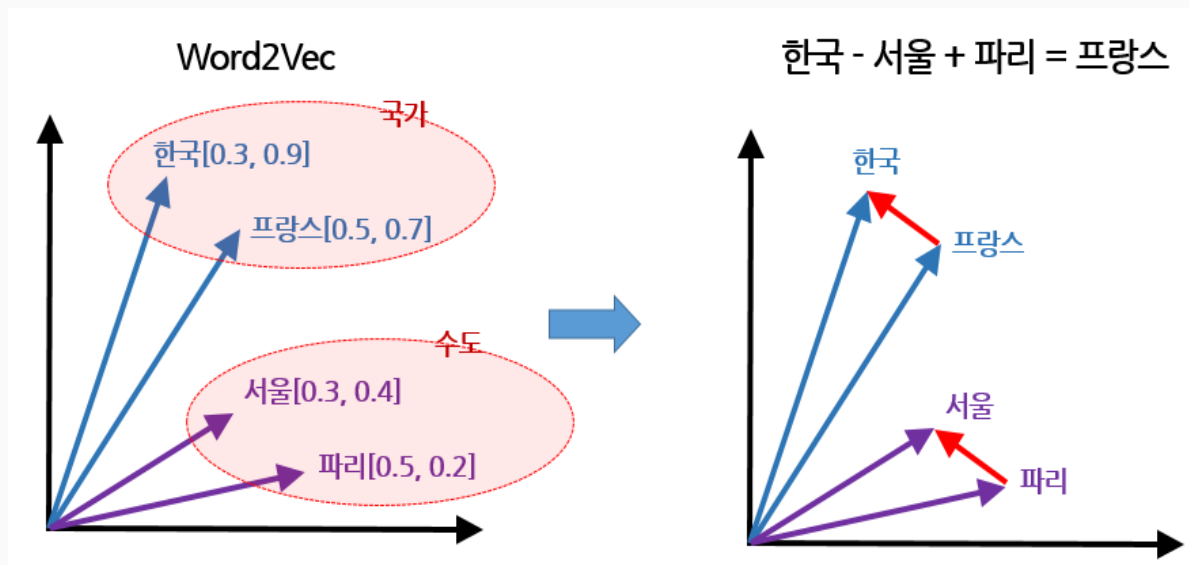
Word2Vec

대표적인 분산 표현 방법으로 분포 가설을 이용

- 코퍼스(말뭉치)에서 각 단어를 단어의 의미에 따라 여러 차원에 분산하여 벡터로 표현
- 단어 간 유사도 계산 등의 연산 가능

Word2Vec을 이용한 Word Embedding

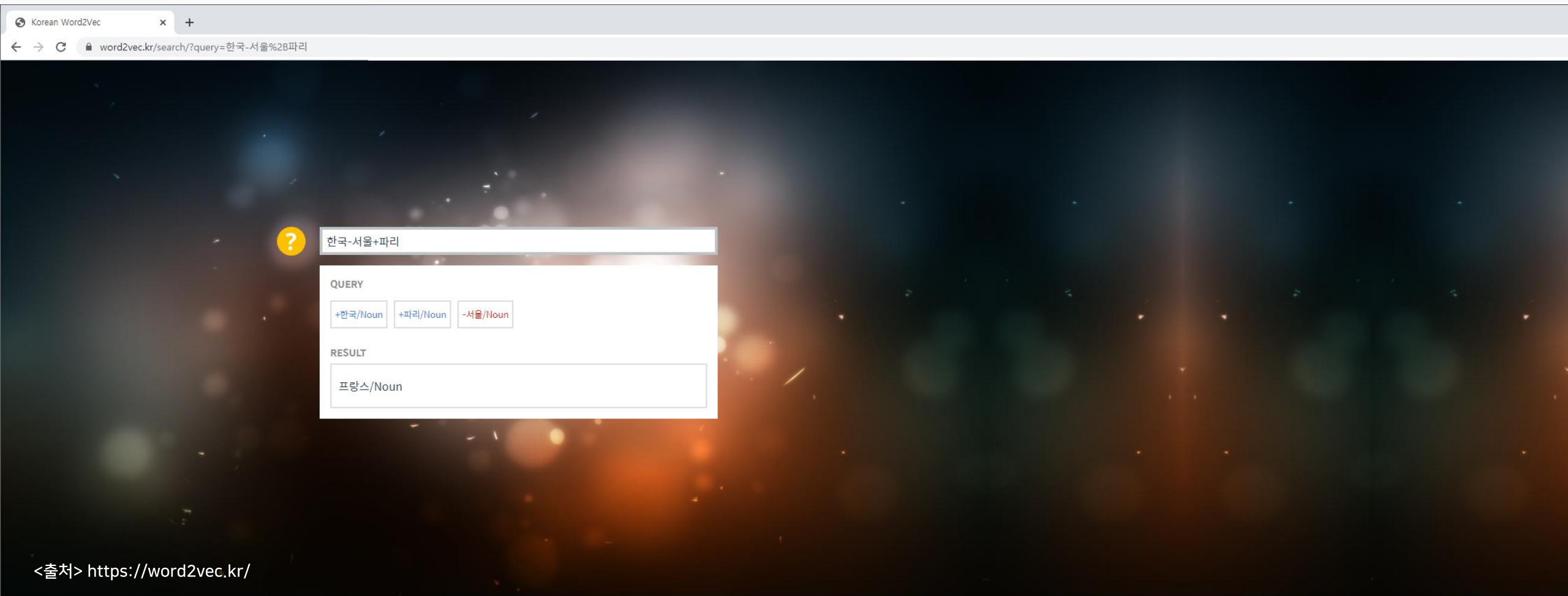
- Word2Vec은 유사한 단어가 유사한 위치에 Embedding
- 단어 벡터 간 연산을 통해 의미 파악 가능



- Word2Vec은 CBOW와 skip-gram 등 두가지 방법이 존재

Word2Vec을 이용한 Word Embedding

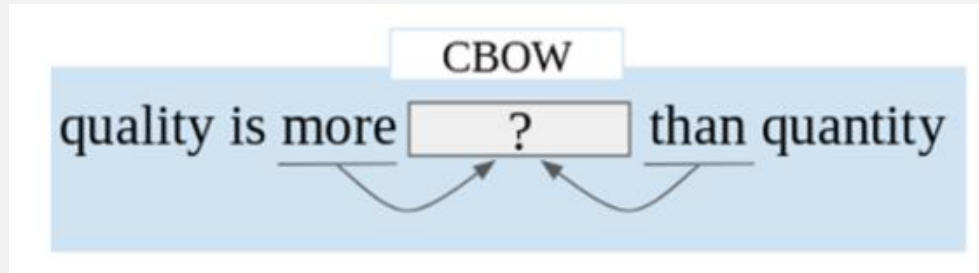
- Word2Vec은 CBOW와 skip-gram 등 두가지 방법이 존재



CBOW와 skip-gram

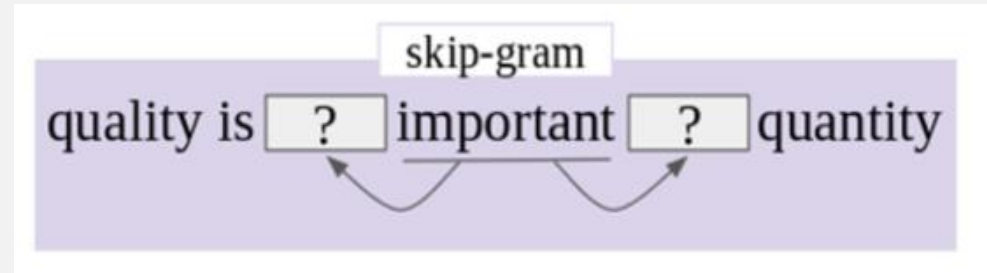
CBOW(Continuous Bag-of-Words)

주변 단어(context word)를
Embedding하여 중심 단어(target word)를
예측하기 위한 Word2Vec 방법



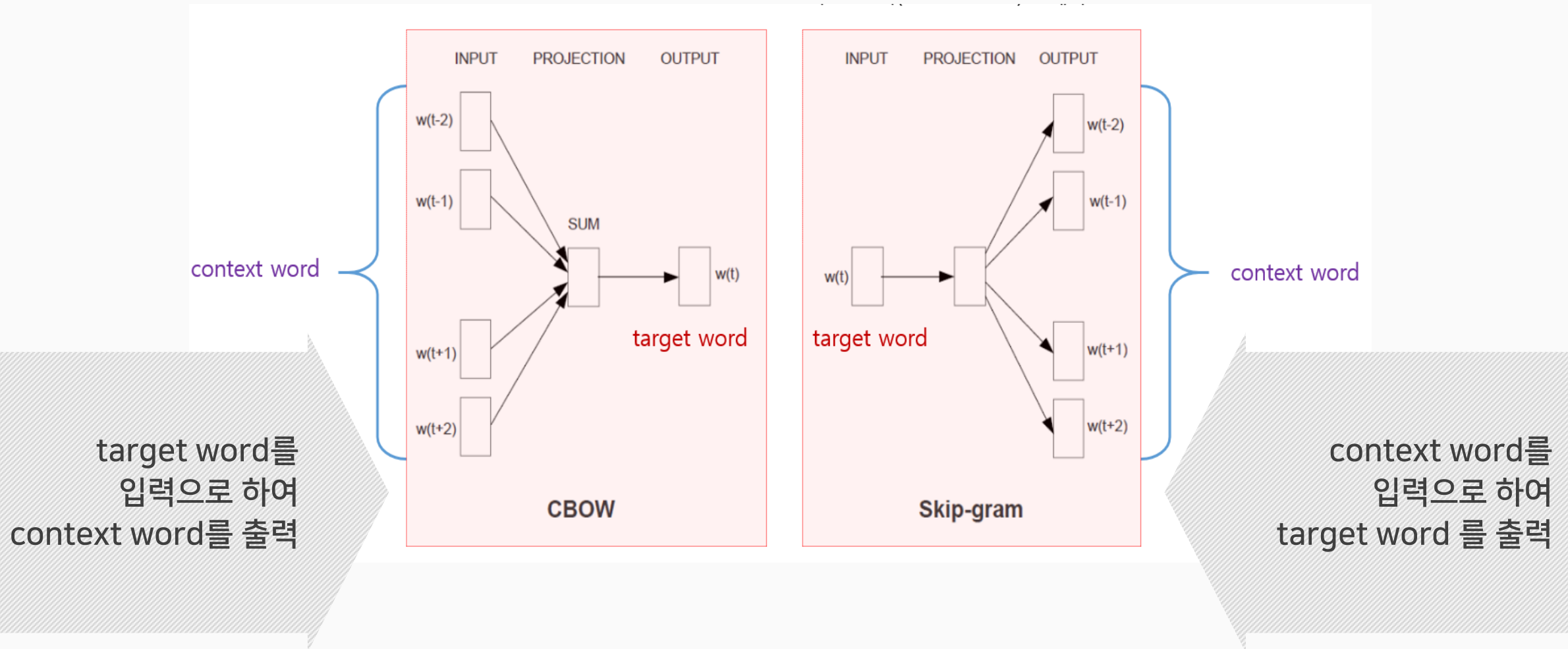
skip-gram

중심 단어(target word)를
Embedding하여 주변 단어(context
word)를 예측하기 위한 Word2Vec 방법



CBOW와 skip-gram

- CBOW와 skip-gram는 서로 대칭적 구조

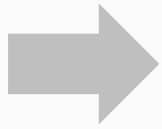


CBOW와 skip-gram을 이용한 단어 예측

Word2Vec 학습 문장

“quality is more important than quantity”의
one-hot encoding 표현

quality
is
important
than
quantity



1	0	0	0	0
0	1	0	0	0
0	0	1	0	0
0	0	0	1	0
0	0	0	0	1

CBOW와 skip-gram을 이용한 단어 예측

윈도우 (Window)

target word를 기준으로 참조하고자 하는 context word의 범위

Window size가 2인 경우 Window 예시

“quality is more important than quantity”

quality	is	more
---------	----	------

quality	is	more	important
---------	----	------	-----------

quality	is	more	important	than
---------	----	------	-----------	------

CBOW와 skip-gram을 이용한 단어 예측

슬라이딩 윈도우(Sliding Window)

- 전체 학습 문장에 대해 Window를 이동하며 학습을 위한 target word와 context word 데이터 셋을 추출하는 과정

Window size가 2인 경우 Sliding Window

1	quality	is	more	important	than	quantity
			↓			
2	quality	is	more	important	than	quantity
			↓			
3	quality	is	more	important	than	quantity
			↓			
4	quality	is	more	important	than	quantity
			↓			
5	quality	is	more	important	than	quantity
			↓			
6	quality	is	more	important	than	quantity

CBOW와 skip-gram을 이용한 단어 예측

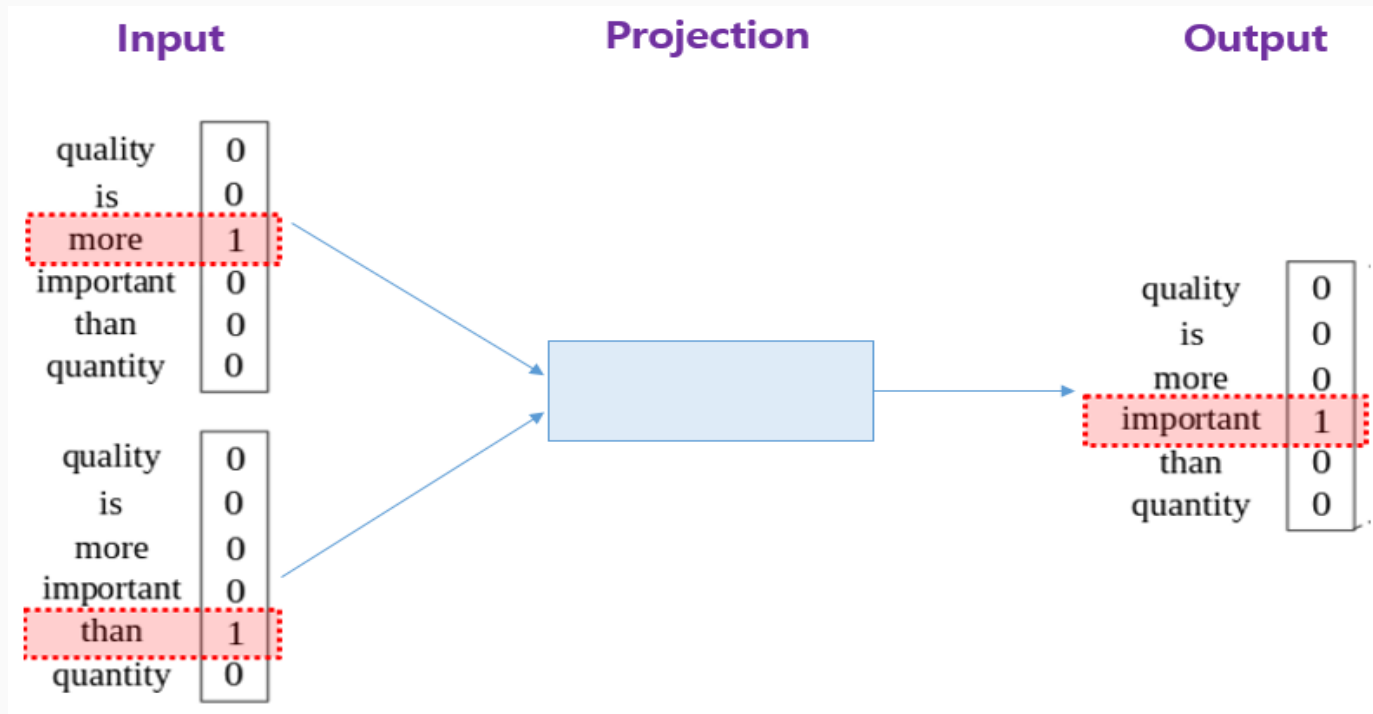
Sliding Window를 통해 추출된 target word와 context word 데이터 셋

Sliding	target word	context word
1	[1, 0, 0, 0, 0, 0]	[0, 1, 0, 0, 0, 0], [0, 0, 1, 0, 0, 0]
2	[0, 1, 0, 0, 0, 0]	[1, 0, 0, 0, 0, 0], [0, 0, 1, 0, 0, 0], [0, 0, 0, 1, 0, 0]
3	[0, 0, 1, 0, 0, 0]	[1, 0, 0, 0, 0, 0], [0, 1, 0, 0, 0, 0], [0, 0, 0, 1, 0, 0], [0, 0, 0, 0, 1, 0]
4	[0, 0, 0, 1, 0, 0]	[0, 1, 0, 0, 0, 0], [0, 0, 1, 0, 0, 0], [0, 0, 0, 0, 1, 0], [0, 0, 0, 0, 0, 1]
5	[0, 0, 0, 0, 1, 0]	[0, 0, 1, 0, 0, 0], [0, 0, 0, 1, 0, 0], [0, 0, 0, 0, 1, 0]
6	[1, 0, 0, 0, 0, 1]	[0, 0, 0, 0, 1, 0], [0, 0, 0, 1, 0, 0]

CBOW와 skip-gram을 이용한 단어 예측

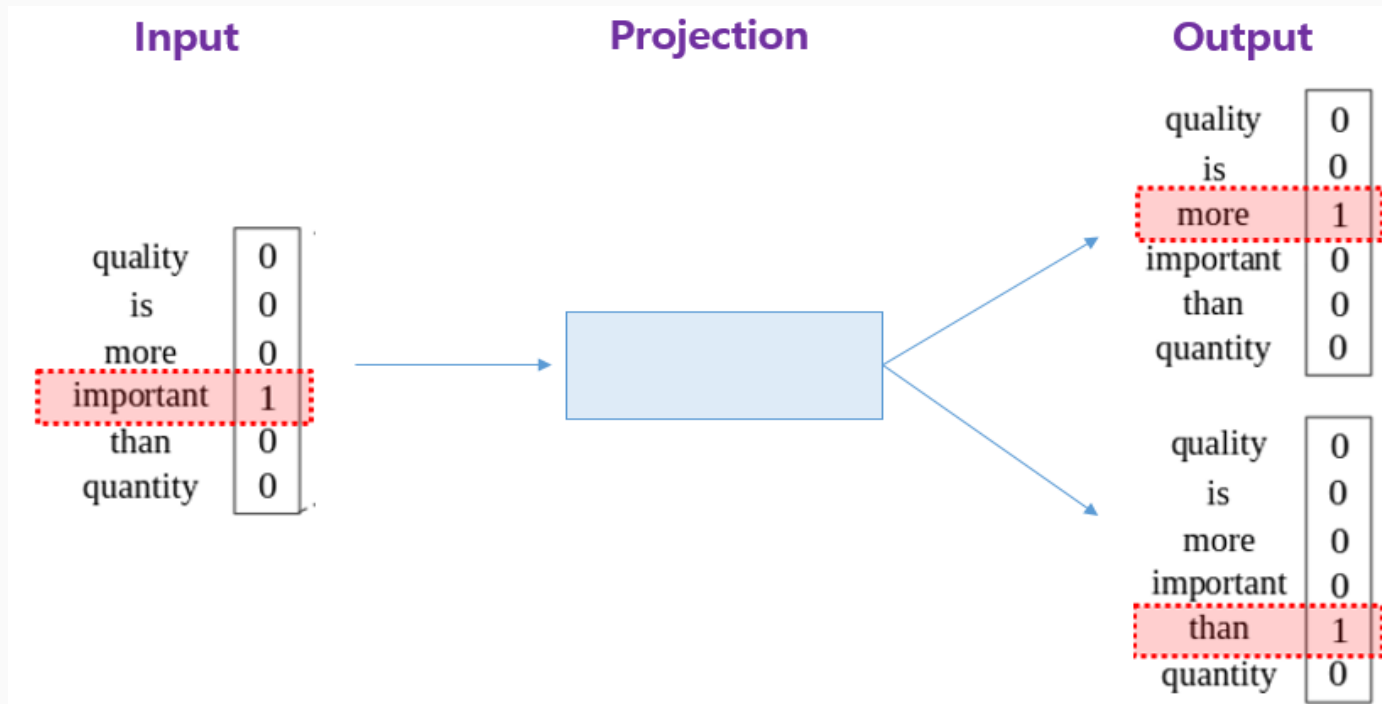
CBOW의 target word 예측

- Window size가 1인 경우
 - 입력 : more [001000], than [000010]
 - 출력 : important [000100]



skip-gram의 target word 예측

- Window size가 1인 경우
 - 입력 : important [000100]
 - 출력 : more [001000], than [000010]





Word2Vec의 활용

gensim 패키지

파이썬에서
CBOW, skip-gram 등의 Word2Vec
적용을 위한 라이브러리



- gensim 패키지 설치

```
pip install gensim
```

- 파이썬에서 gensim패키지를 이용한 Word2Vec 적용

```
from gensim.models import Word2Vec
model = Word2Vec([data],          # 리스트 형태의 데이터
                  sg=1,           # 0: CBOW, 1: Skip-gram
                  size=100,       # 벡터 크기
                  window=3,       # 고려할 앞뒤 폭(앞뒤 3단어)
                  min_count=3)    # 사용할 단어의 최소 빈도
                                (3회 이하 단어 무시)
```

genism 패키지를 이용한 유사도 분석

- 코퍼스에서 '게임'과 유사한 단어 및 단어별 유사도 분석

```
result = model.most_similar('게임')  
print(result)
```

```
[('이용', 0.17749272286891937),  
( '규정', 0.12711596488952637),  
( '적용', 0.11808653175830841),  
( '국내', 0.10660543292760849),  
( '질병', 0.10210900008678436),  
( '코드', 0.10153514891862869),  
( '지난', 0.09195291996002197),  
( '콘텐츠', 0.07838629186153412),  
( '게임중독', 0.05836869031190872),  
( '분류', 0.04828557372093201)]
```

genism 패키지를 이용한 유사도 분석

- 코퍼스에서 '게임'과 유사한 단어 및 단어별 유사도 분석

```
result = model.most_similar('게임')
print(result)

[('이용', 0.17749272286891937),
 ('규정', 0.12711596488952637),
 ('적용', 0.11808653175830841),
 ('국내', 0.10660543292760849),
 ('질병', 0.10210900008678436),
 ('코드', 0.10153514891862869),
 ('지난', 0.09195291996002197),
 ('콘텐츠', 0.07838629186153412),
 ('게임중독', 0.05836869031190872),
 ('분류', 0.04828557372093201)]
cos = model.wv.similarity('게임', '질병')

print(cos)

0.102109
```



pre-trained 모델

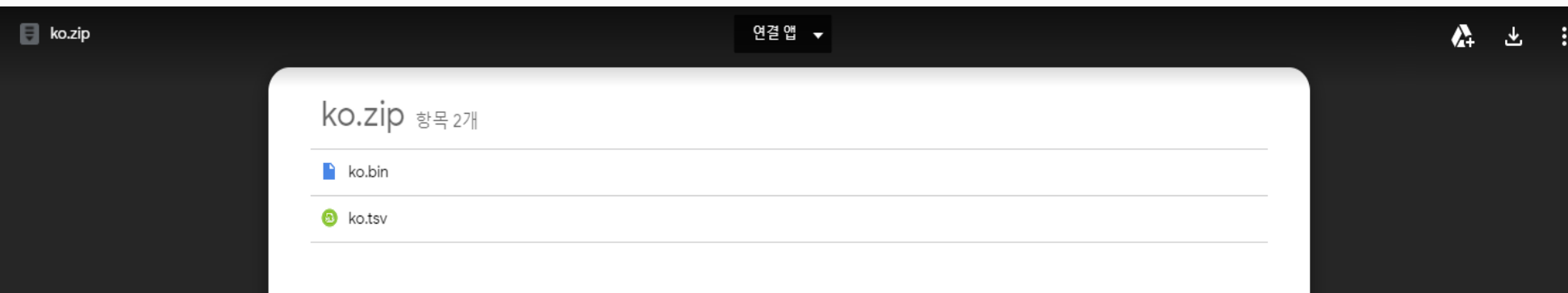
대용량 코퍼스 데이터를 이용하여
사전에 학습된 모델

한국어 pre-trained Word2Vec 모델을 활용한 유사도 분석 방법

1 pre-trained Word2Vec 모델 다운로드

- <https://github.com/Kyubyong/wordvectors>에 공개된 한국어 pre-trained의 다운로드 경로를 통해 한국어 pre-trained Word2Vec 모델 다운로드
- 한국어 pre-trained Word2Vec 모델 다운로드 URL
<https://drive.google.com/file/d/0B0ZXk88koS2KbDhXdWg1Q2RydIU/view>

2 압축해제 및 ko.bin 파일을 data 디렉토리로 이동



한국어 pre-trained Word2Vec 모델을 활용한 유사도 분석 방법

3 pre-trained 모델 로드 및 활용

```
import gensim
model = gensim.models.Word2Vec.load('./data/ko.bin')

result = model.most_similar("게임")
print(result)
[('액션', 0.6752270460128784),
 ('게임기', 0.6604887843132019),
 ('콘솔', 0.6558898687362671),
 ('슈팅', 0.6405965089797974),
 ('아케이드', 0.6360284090042114),
 ('퍼즐', 0.6304599046707153),
 ('어드벤처', 0.6224750876426697),
 ('닌텐도', 0.6071302890777588),
 ('애니메이션', 0.6063960790634155),
 ('온라인', 0.6031370162963867)]
```

```
result = model.most_similar("인공지능")  
print(result)
```

```
[('컴퓨팅', 0.6520194411277771),  
 ('가상현실', 0.6393702030181885),  
 ('심리학', 0.63037109375),  
 ('모델링', 0.625065267086029),  
 ('신경망', 0.6200423836708069),  
 ('로봇', 0.6109743118286133),  
 ('시뮬레이션', 0.6101070642471313),  
 ('지능', 0.6092982888221741),  
 ('기술', 0.6087721586227417),  
 ('기술인', 0.5957076549530029)]
```