NLP 스터디 1주차

한국어 임베딩

2.4 어떤 단어가 같이 쓰였는가
3.1 한국어 전처리 – 데이터 확보

이영현

어떤 단어가 같이 쓰였는가

분포 가정이란?

- 분포
  특정 범위(= 윈도우)내에 동시에 등장하는 이웃 단어 또는 문맥의 집합을 가리킴 in 자연어처리

- 분포 가정의 전제
  어떤 단어 쌍이 비슷한 문맥 환경에서 자주 등장한다면 그 의미 또한 유사할 것
  “단어의 의미는 곧 그 언어에서의 활용이다” –비트겐슈타인

분포 가정이란?

… 특기 는 자칭 청소 와 **빨래** 지만 요리 는 절망 적…
… 재 를 우려낸 물 로 **빨래** 할 때 나…
… 개울가 에서 속옷 **빨래** 를 하 는 남녀…

… 찬 물 로 옷 을 **세탁** 한다…
… **세탁**, 청소, 요리 와 가사 는…

- 예)
  타깃 단어: 빨래 -> 문맥 단어: 청소, 요리, 물, 속옷
  타깃 단어: 세탁 -> 문맥 단어: 청소, 요리, 물, 옷

- 빨래 VS 세탁

- 분포 정보 = 의미
  ???

분포와 의미

(1) 형태소
: 의미어휘적인 것뿐 아닌 문법적인 것도 포함를 가지는 최소 단위, 더 쪼개면 뜻을 잃어버림

- 철수가 밥을 먹었다

- 계열 관계 paradigmatic relation
  해당 형태소 자리에 다른 형태소가 대치돼 쓰일 수 있는가

  영희가 빵을 먹었다 -〉 철수와 밥 자리에 영희, 빵을 쓸 수 있음 =〉 철수, 밥은 형태소!

  언어학자들이 특정 타깃 단어 주변의 문맥 정보를 바탕으로 형태소를 확인
  즉, 말뭉치의 분포 정보와 형태소가 밀접한 관계를 이루고 있음

분포와 의미

(2) 품사
: 단어를 문법적 성질의 공통성에 따라 언어학자들이 몇 갈래로 묶어 놓은 것

품사 분류 기준 – 기능 function, 의미 meaning, 형식 form 등

- 이 샘의 깊이가 얼마냐?
- 저 산의 높이가 얼마냐?
- 이 샘이 깊다.
- 저 산이 높다.

- **기능: 깊이, 높이 – 문장의 주어, 깊다, 높다 – 문장의 서술어**

  한 단어가 문장 가운데서 다른 단어와 맺는 관계를 가리킴

분포와 의미

(2) 품사
: 단어를 문법적 성질의 공통성에 따라 언어학자들이 몇 갈래로 묶어 놓은 것

품사 분류 기준 – 기능 function, 의미 meaning, 형식 form 등

- 이 샘의 깊이가 얼마냐?
- 저 산의 높이가 얼마냐?
- 이 샘이 깊다.
- 저 산이 높다.

- 의미:
  어휘적 의미; 깊이, 깊다/ 높이, 높다
  형식적 의미; 깊이, 높이/ 깊다, 높다

어떤 단어가 사물의 이름을 나타내느냐, 그렇지 않으면 움직임이나 성질, 상태를 나타내느냐

분포와 의미

(2) 품사
: 단어를 문법적 성질의 공통성에 따라 언어학자들이 몇 갈래로 묶어 놓은 것

품사 분류 기준 – 기능 function, 의미 meaning, 형식 form 등

- 이 샘의 깊이가 얼마냐?
- 저 산의 높이가 얼마냐?
- 이 샘이 깊다.
- 저 산이 높다.

- **형식: 깊이, 높이** 변화X/**깊다, 높다** 변화O 깊었다/높았다, 깊겠다/높겠다

  단어의 형태적 특징 의미

분포와 의미

(2) 품사
: 단어를 문법적 성질의 공통성에 따라 언어학자들이 몇 갈래로 묶어 놓은 것

결정적인 품사 분류 기준

- 의미
  - 공부하다 움직임
  - 공부 사물의 이름

  공부라는 단어에는 움직임이라는 의미가 전혀 없는 것일까?
- 형식
  (a) 영수가 학교에 간다. 명사
  (b) 영수! 조용히 해. 감탄사

  형태는 같지만 기능과 의미가 달라졌음

-〉 결정적인 분류 기준 될 수 없음

분포와 의미

(2) 품사
: 단어를 문법적 성질의 공통성에 따라 언어학자들이 몇 갈래로 묶어 놓은 것

결정적인 품사 분류 기준

- 기능
  해당 단어가 문장 내에서 점하는 역할에 초점을 맞춰 품사를 분류
  -〉 결정적인 분류 기준 즉 어떤 단어의 기능이 그 단어의 분포와 매우 밀접한 관련을 맺고 있음

〈정리〉

기능 – 특정 단어가 문장 내에서 어떤 역할을 하는지

분포 – 그 단어가 어느 자리에 있는지

임베딩에 분포 정보를 함축하게 되면 해당 벡터에 해당 단어의 의미를 자연스레 내재시킬 수 있게 된다 -〉 분포 = 의미

점별 상호 정보량(PMI)

두 확률 변수 random variable 사이의 상관성을 계량하는 단위

두 단어의 등장이 독립일 때 대비해 얼마나 자주 같이 등장하는지를 수치화 한 것

$$PMI(A,B) = \log \frac{P(A,B)}{P(A) \times P(B)}$$

-〉 분포 가정에 따른 가중치 할당 기법

단어-문맥 행렬(co-occurrence matrix)

예 : k=1 의
문맥창으로 한 경우

|V| 는 어휘수

I enjoy technology.
I like eating.
I like to sleep.

"like" 앞에 "I"가 두번 출현
"like" 뒤에 "eating"이 한번 출현

|V|

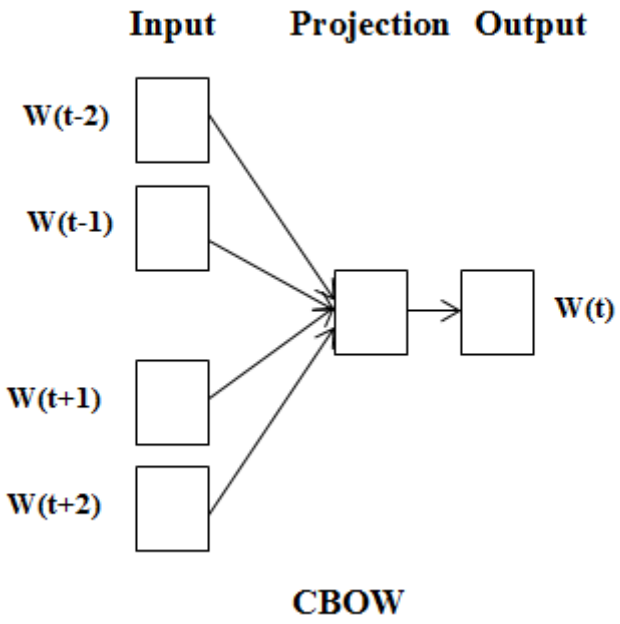
|V|

	I	enjoy	technology	like	eating	to	sleep	.
I	0	1	0	2	0	0	0	0
enjoy	1	0	1	0	0	0	0	0
technology	0	1	0	0	0	0	0	1
like	2	0	0	0	1	1	0	0
eating	0	0	0	1	0	0	0	1
to	0	0	0	1	0	0	1	0
sleep	0	0	0	0	0	1	0	1
.	0	0	1	0	1	0	1	0

Word2Vec 분포 가정의 대표적 모델, 임베딩 기법

CBOW

문맥 단어들을 가지고 타깃 단어 하나를 맞추는 과정에서 학습



center word context words

I like playing football with my friends

I like playing football with my friends

I like playing football with my friends

I like playing football with my friends

I like playing football with my friends

I like playing football with my friends

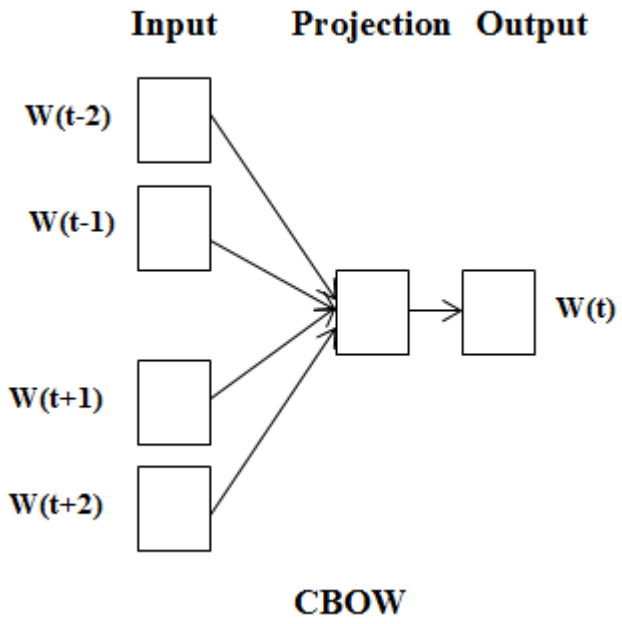
I like playing football with my friends

center word	context words
[1,0,0,0,0,0,0]	[0,1,0,0,0,0,0] [0,0,1,0,0,0,0]
[0,1,0,0,0,0,0]	[1,0,0,0,0,0,0] [0,0,1,0,0,0,0] [0,0,0,1,0,0,0]
[0,0,1,0,0,0,0]	[1,0,0,0,0,0,0] [0,1,0,0,0,0,0] [0,0,0,1,0,0,0] [0,0,0,0,1,0,0]
[0,0,0,1,0,0,0]	[0,1,0,0,0,0,0] [0,0,1,0,0,0,0] [0,0,0,0,1,0,0] [0,0,0,0,0,1,0]
[0,0,0,0,1,0,0]	[0,0,1,0,0,0,0] [0,0,0,1,0,0,0] [0,0,0,0,0,1,0] [0,0,0,0,0,0,1]
[0,0,0,0,0,1,0]	[1,0,0,1,0,0,0] [0,0,0,0,1,0,0] [0,0,0,0,0,0,1]
[0,0,0,0,0,0,1]	[0,0,0,0,1,0,0] [0,0,0,0,0,1,0]

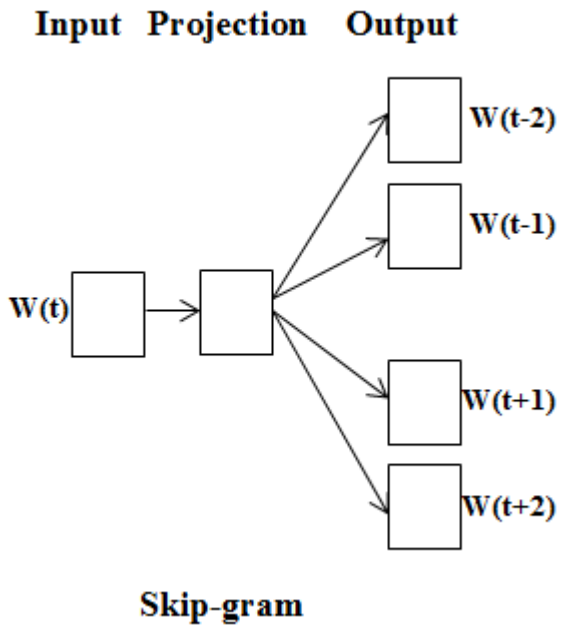
Word2Vec 분포 가정의 대표적 모델, 임베딩 기법

Skip-gram

타깃 단어들을 가지고 문맥 단어가 무엇일지 예측하는 과정에서 학습

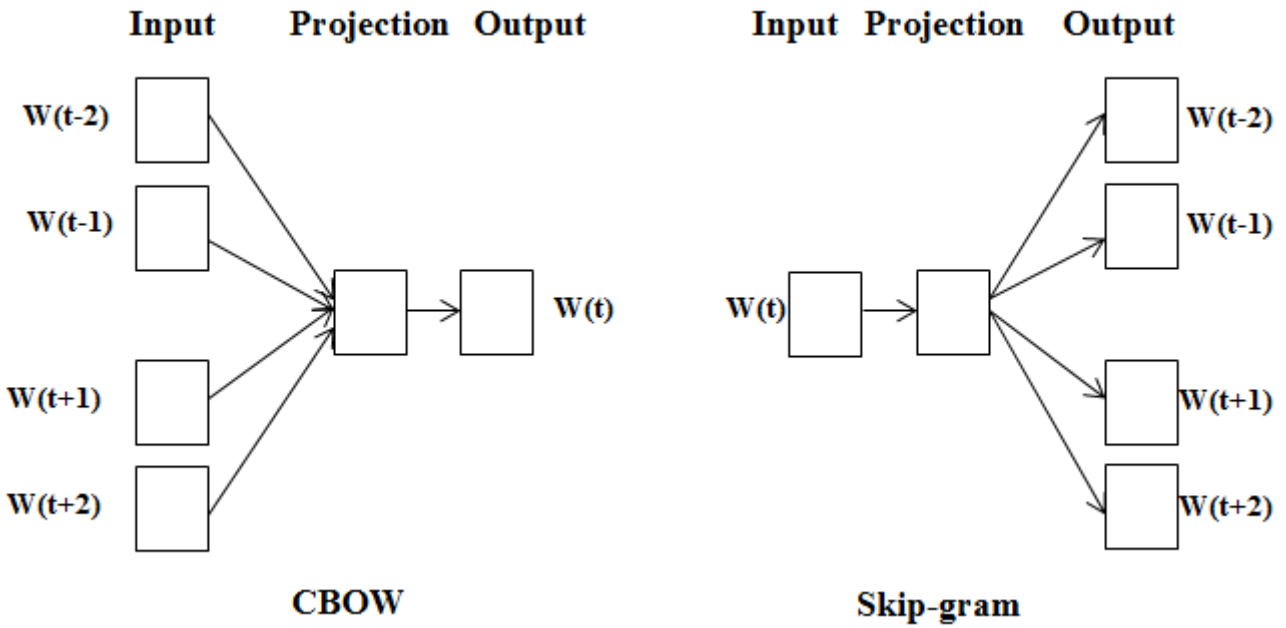


CBOW



Skip-gram

Word2Vec 분포 가정의 대표적 모델, 임베딩 기법



분포 정보를 임베딩에 함축-〉 문맥을 고려한다는 의미

요약

- 분포 가정에서는 문장에서 어떤 단어가 같이 쓰였는지를 중요하게 따진다.

- 백오브워즈 가정, 언어 모델, 분포 가정은 말뭉치의 통계적 패턴을 서로 다른 각도에서 분석하는 것이며 상호 보완적이다.

데이터 전처리

한국어 위키백과

```
from gensim.corpora import WikiCorpus, Dictionary
from gensim.utils import to_unicode
import re
```

```
WIKI_REMOVE_CHARS = re.compile("'+|(=+.{2,30}=+)|__TOC__|(ファイル:).+|:(en|de|it|fr|es|kr|zh|no|fi):|₩n", re.UNICODE)
WIKI_SPACE_CHARS = re.compile("(₩₩s|□|□|  )+", re.UNICODE)
EMAIL_PATTERN = re.compile("(^[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+₩.[a-zA-Z0-9-.]+$)", re.UNICODE)
URL_PATTERN = re.compile("(ftp|http|https)?://(?:[a-zA-Z]|[0-9]|[$-_@.&+]|[!*₩(₩),]|(?:%[0-9a-fA-F][0-9a-fA-F]))+", re.UNICODE)
WIKI_REMOVE_TOKEN_CHARS = re.compile("(₩₩*$|:$|^파일:.+|^;)", re.UNICODE)
MULTIPLE_SPACES = re.compile(' +', re.UNICODE)


def tokenize(content, token_min_len=2, token_max_len=100, lower=True):
    content = re.sub(EMAIL_PATTERN, ' ', content)  # remove email pattern
    content = re.sub(URL_PATTERN, ' ', content) # remove url pattern
    content = re.sub(WIKI_REMOVE_CHARS, ' ', content)  # remove unnecessary chars
    content = re.sub(WIKI_SPACE_CHARS, ' ', content)
    content = re.sub(MULTIPLE_SPACES, ' ', content)
    tokens = content.replace(", )", "").split(" ")
    result = []
    for token in tokens:
        if not token.startswith('_'):
            token_candidate = to_unicode(re.sub(WIKI_REMOVE_TOKEN_CHARS, '', token))
        else:
            token_candidate = ""
        if len(token_candidate) > 0:
            result.append(token_candidate)
    return result
```

한국어 위키백과

```
def make_corpus(in_f, out_f):
    """Convert Wikipedia xml dump file to text corpus"""
    output = open(out_f, 'w', encoding = "utf-8")
    wiki = WikiCorpus(in_f, tokenizer_func=tokenize, dictionary=Dictionary())
    i = 0
    for text in wiki.get_texts():
        output.write(bytes(' '.join(text), 'utf-8').decode('utf-8') + '₩n')
        i = i + 1
        if (i % 10000 == 0):
            print('Processed ' + str(i) + ' articles')
    output.close()
    print('Processing complete!')
```

```
i_f = "kowiki-latest-pages-articles.xml.bz2"
o_f = "processed_wiki_ko.txt"

make_corpus(i_f, o_f)
```

KorQuAD

한국어 기계 독해를 위한 데이터 셋

네이버 영화 리뷰 말뭉치