

15기 정규세션

ToBig's 14기 정세영

NLP 기초

Contents

Unit 01 | NLP

Unit 02 | Process

Unit 03 | Tokenizing

Unit 04 | Embedding

Unit 05 | Similarity

Contents

Unit 01 | NLP

Unit 02 | Process

Unit 03 | Tokenizing

Unit 04 | Embedding

Unit 05 | Similarity

Unit 01 | NLP

Q. 텍스트 기초에서 텍스트란?

A. **자연어(Natural Language)**를 가리킨다.

사람들이 일상생활에서 자연스럽게 사용하는 언어

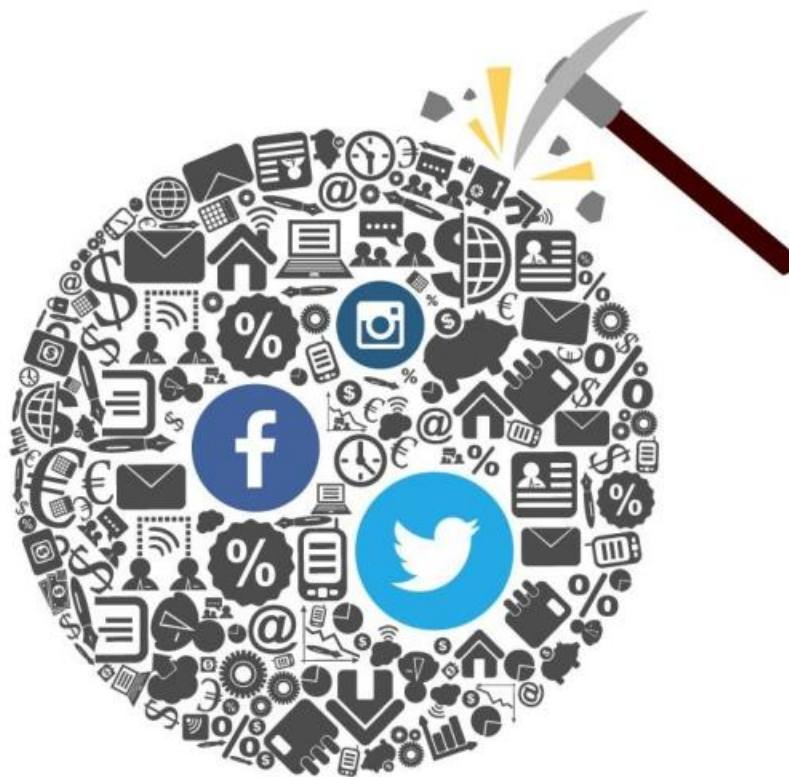


어떻게 컴퓨터가 자연어를 이해할 수 있을까?

자연어처리 (Natural Language Processing)

컴퓨터가 자연어를 이해하거나 생성할 수 있도록 하는 학문 분야

Unit 01 | NLP



현존하는 비정형 데이터 중 상당 부분이 **텍스트 데이터**

Unit 01 | NLP

문서 자동 요약

감성 및 주제 분석

기계 번역

맞춤법 검사



질의응답 (챗봇)

음성인식 스피커

Contents

Unit 01 | NLP

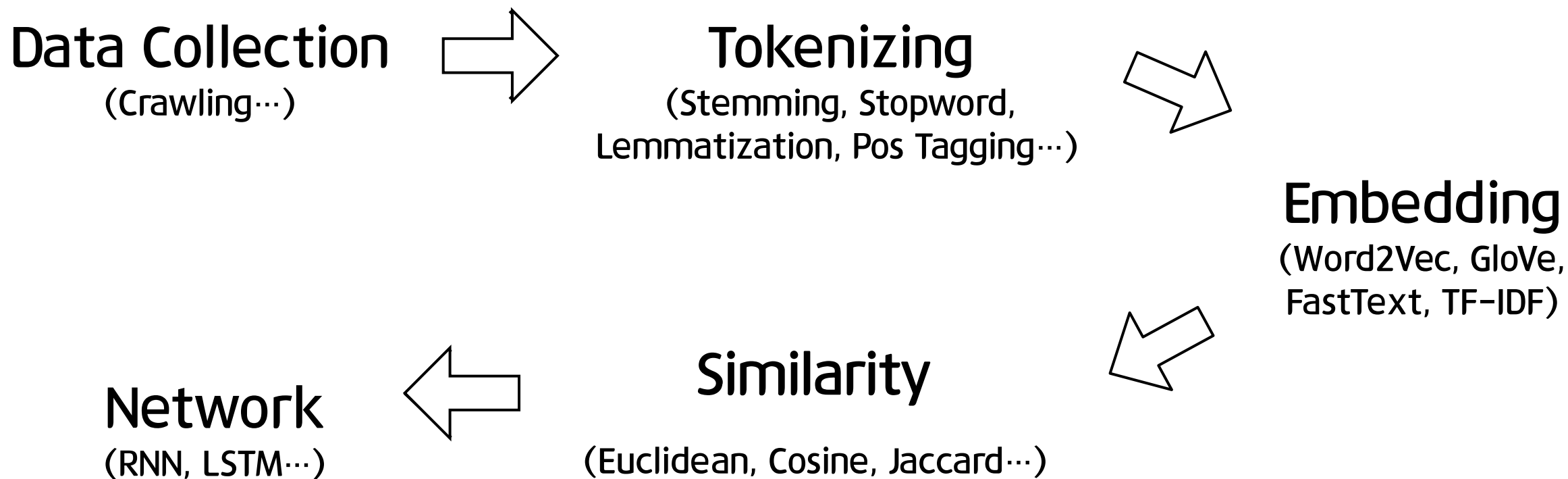
Unit 02 | Process

Unit 03 | Tokenizing

Unit 04 | Embedding

Unit 05 | Similarity

Unit 02 | Process



Unit 02 | Process

Step 1. Data Collection

로그인하지 않음 토론 기여 계정 만들기 로그인

문서 토론 읽기 편집 역사 보기 위키백과 검색

2019년 1차 언론학 에디터톤이 8월 31일에 열립니다. [숨기기]

빅 데이터

위키백과, 우리 모두의 백과사전.

빅 데이터(영어: big data)란 기존 데이터베이스 관리도구의 능력을 넘어서는 대량(수십 테라바이트)의 정형 또는 심지어 데이터베이스 형태가 아닌 비정형의 데이터 집합조차 포함한 데이터로부터 가치를 추출하고 결과를 분석하는 기술이다.

다양한 종류의 대규모 데이터에 대한 생성, 수집, 분석, 표현을 그 특징으로 하는 빅 데이터 기술의 발전은 다변화된 현대 사회를 더욱 정확하게 예측하여 효율적으로 작동케 하고 개인화된 현대 사회 구성원마다 맞춤형 정보를 제공, 관리, 분석 가능케 하며 과거에는 불가능했던 기술을 실현시키기도 한다.

이같이 빅 데이터는 정치, 사회, 경제, 문화, 과학 기술 등 전 영역에 걸쳐서 사회와 인류에게 가치있는 정보를 제공할 수 있는 가능성을 제시하며 그 중요성이 부각되고 있다.

위키백과의 편집 현황의 시각화 자료(BM 작성). 수 테라바이트의 용량을 지닌 위키백과의 텍스트 및 이미지 자료는 빅 데이터의 전형적 사례에 속한다.

위키백과

대문
사용자 모임
요즘 화제
최근 바뀐
모든 문서 보기
임의 문서로
도움말
기부

도구
여기를 가리키는 문서
가리키는 글의 최근 바뀐
파일 올리기
특수 문서 목록
고유 링크
문서 정보
위키데이터 항목

p Clear (41) Toggle Position XPath

Elements Console Sources Network

```
<doctype html>
<html class="client-js ve-available" lang="ko" dir="ltr">
html body div#content.mw-body h1#firstHeading.firstHeading
```

Styles Event Listeners DOM Breakpoints Properties Accessibility

Filter :hov .cls +

element.style { }

.mw-body load.php?la...in=vector:1
h1:lang(ja), .mw-body-content
h1:lang(ja), .mw-body-content
h2:lang(ja), .mw-body h1:lang(he),
.mw-body-content h1:lang(he), .mw-
body-content h2:lang(he), .mw-body
h1:lang(ko), .mw-body-content
h1:lang(ko), .mw-body-content
h2:lang(ko) {
font-family: sans-serif;
}

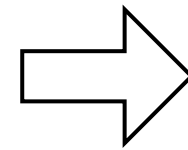
.mw-body load.php?la...in=vector:1
.firstHeading {
overflow: visible;
}

Console What's New

Highlights from the Chrome 76 update

Autocomplete with CSS keyword values
Typing a keyword value like "bold" in the Styles pane now autocompletes to "font-weight: bold".

A new UI for network settings
The "Use large request rows", "Group by frame", "Show overview", and "Capture screenshots" options have moved to the new Network Settings pane.

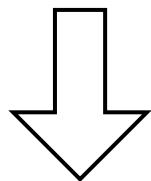


corpus
텍스트(문서)의 집합
“말뭉치”

Unit 02 | Process

Step 2. Tokenizing

나는 그 사람이 아프다



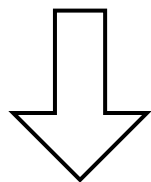
의미를 가진 가장 작은 말의 단위로 쪼개기

‘나’, ‘는’, ‘그’, ‘사람’, ‘이’, ‘아프’, ‘다’

Unit 02 | Process

Step 3. Embedding

나는 그 사람이 아프다



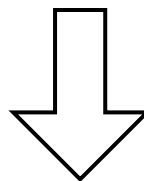
자연어를 숫자의 나열인 벡터로 변환하기

‘나’: [0.1234, 0.1234] ‘는’: [0.5678, 0.1234] ‘그’: [0.7890, 0.1567]
‘사람’: [0.9021, 0.4321] ‘이’: [0.0876, 0.3579] ‘아프’: [0.3456, 0.1764]
‘다’: [0.1234, 0.0399]

Unit 02 | Process

Step 4. Similarity

‘나’: [0.1234, 0.1234] ‘는’: [0.5678, 0.1234] ‘그’: [0.7890, 0.1567]
‘사람’: [0.9021, 0.4321] ‘이’: [0.0876, 0.3579] ‘아프’: [0.3456, 0.1764]
‘다’: [0.1234, 0.0399]

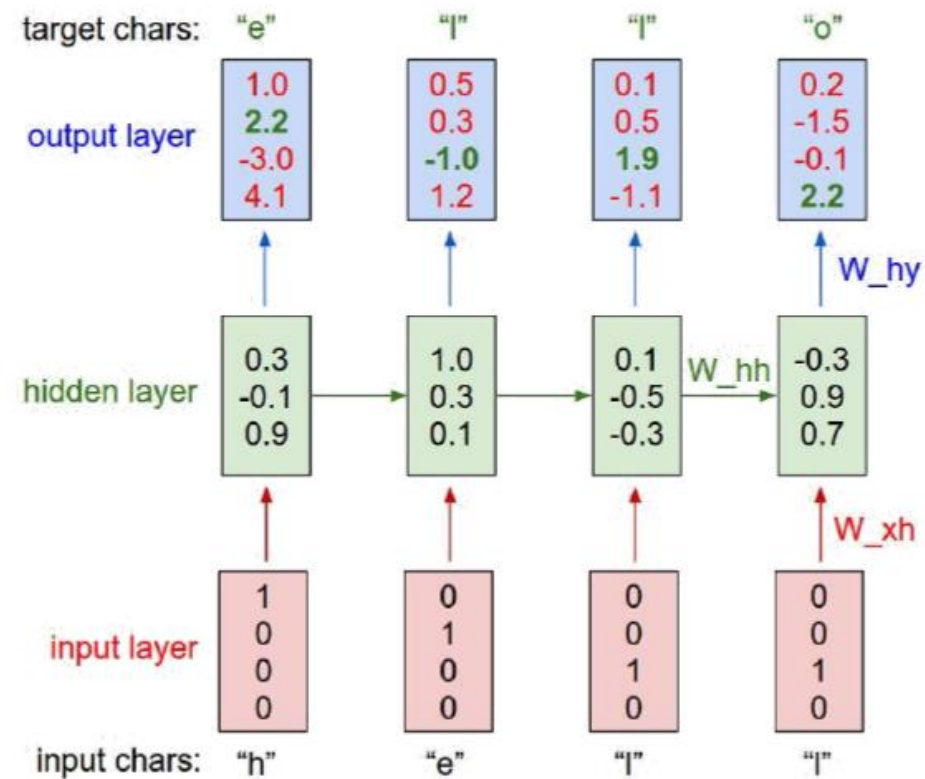
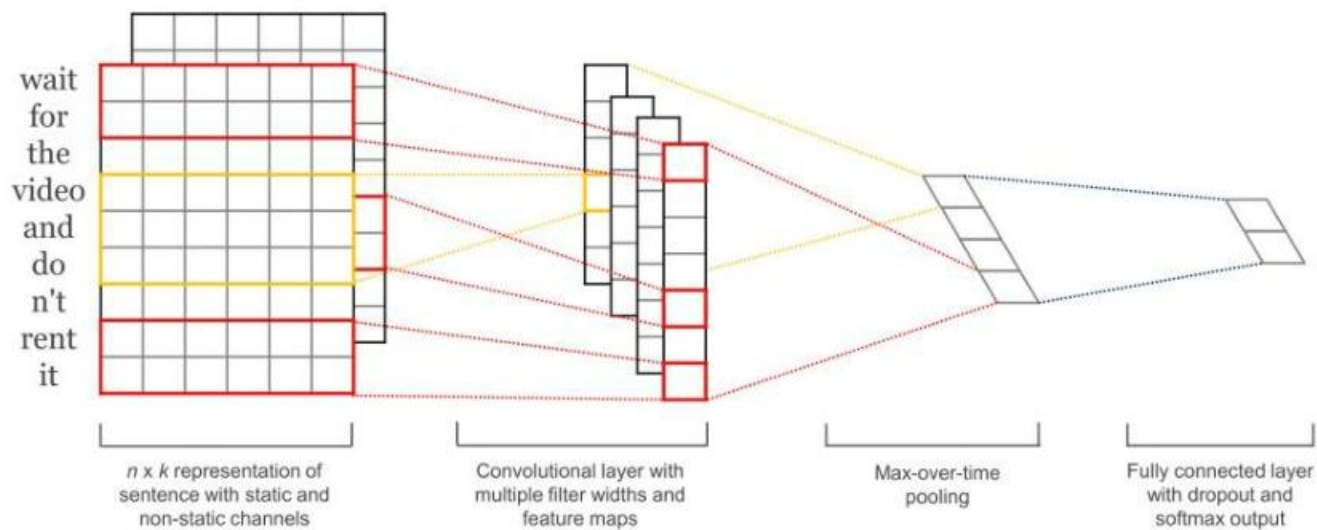


‘사람’: [0.9021, 0.4321] ‘아프’: [0.3456, 0.1764]

코사인 유사도에 따르면, 이 두 단어는 유사하다고 판단할 수 있다.

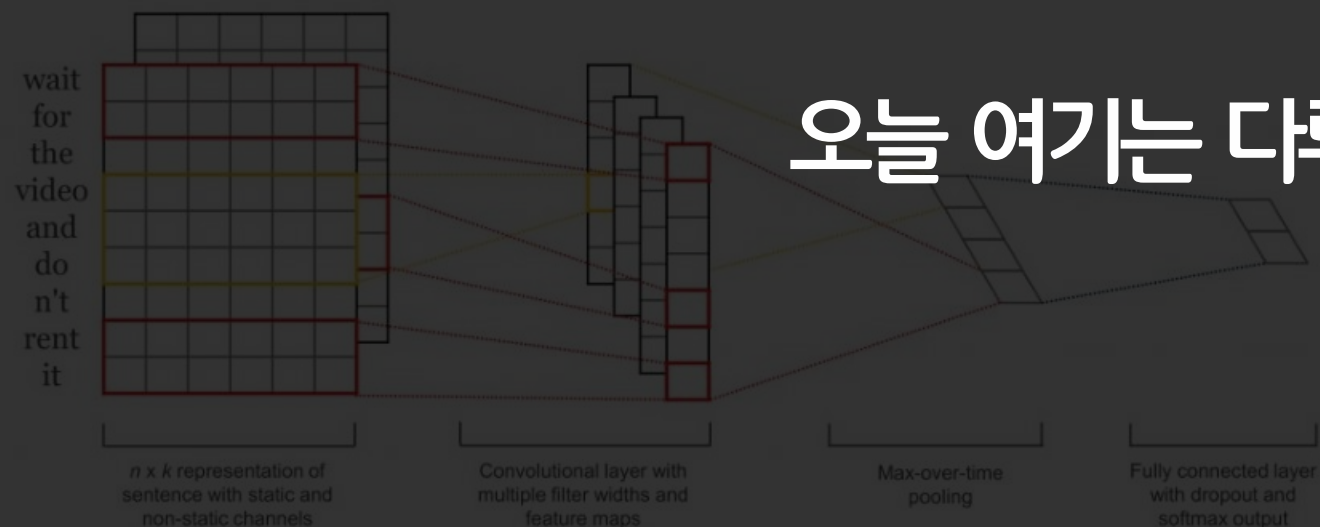
Unit 02 | Process

Step 5. Network

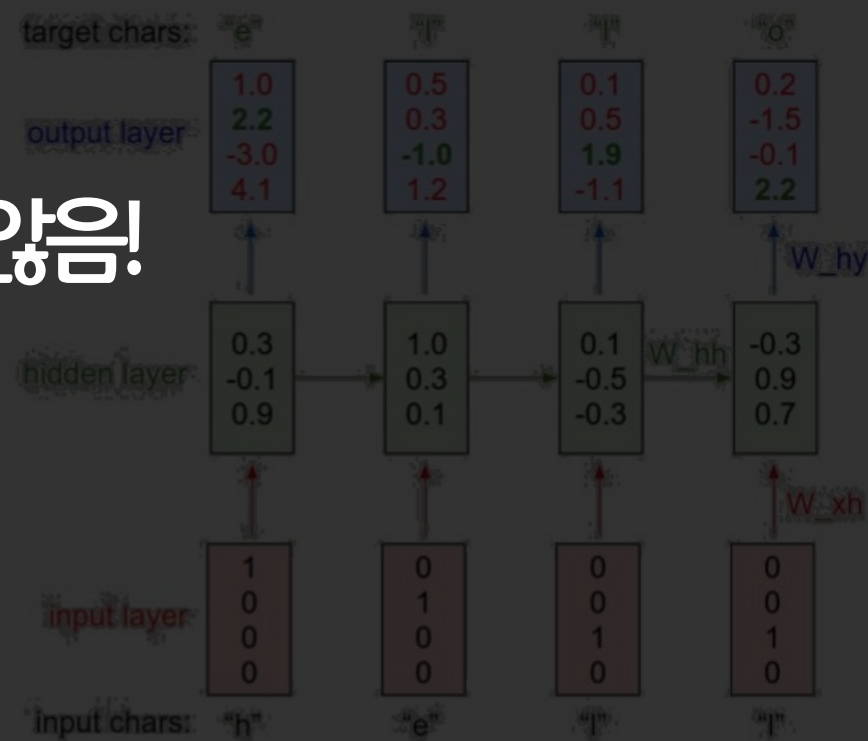


Unit 02 | Process

Step 5. Network



오늘 여기는 다루지 않음!



Unit 02 | Process

Q. 왜 오늘은 텍스트 관련 모델을 다루지 않아요?

A. 텍스트 기초 세션은

텍스트 분석의 준비과정을 배우는 시간으로

주된 학습 내용은 “모델 전 데이터를 **최적의 상태로 만들기**”

(추가) 임베딩만으로 **유의미한 해석 도출하기**

좋은 임베딩은 좋은 결과를 만듭니다.

Contents

Unit 01 | NLP

Unit 02 | Process

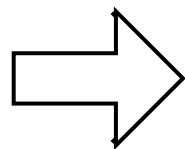
Unit 03 | Tokenizing

Unit 04 | Embedding

Unit 05 | Similarity

Unit 02 | Process

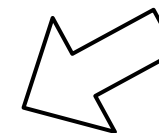
Data Collection
(Crawling...)



Tokenizing
(Stemming, Stopword,
Lemmatization, Pos Tagging...)

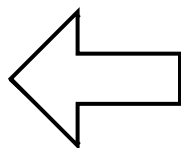


Embedding
(Word2Vec, GloVe,
FastText, TF-IDF)



Similarity

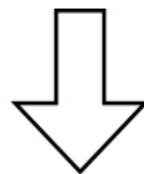
(Euclidean, Cosine, Jaccard...)



Network
(RNN, LSTM...)

Unit 03 | Tokenizing

나는 그 사람이 아프다



‘나’, ‘는’, ‘그’, ‘사람’, ‘이’, ‘아프’, ‘다’

Unit 03 | Tokenizing

나는 그 사람이 아프다
특정 기준에 의해
텍스트를 Token으로 변환

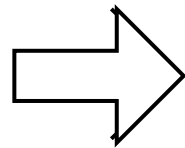


‘나’, ‘는’, ‘그’, ‘사람’, ‘이’, ‘아프’, ‘다’

Unit 03 | Tokenizing

Q. **Token** 이란?

A. 의미를 가지는 요소



뜻을 가진 최소 단위인 형태소

자립하여 쓸 수 있는 **최소 단위인 단어**

즉, **Tokenizing**이란 문서나 문장을 분석하기 좋도록 토큰으로 나누는 작업

Unit 03 | Tokenizing

English

NLTK

Tokenization 어절/문장 분리

Pos tagging 형태소 품사 태깅

Stopwords 불용어 처리

Lemmatization 표제어 추출

Stemming 어간 추출

Korean

KONLPY

kakao

Khaiii

Kkma

Komoran

Mecab

Okt(Twitter)

Hannaum

Unit 03 | Tokenizing

English

NLTK

word_tokenize : 어절/문장 분리 tokenizing

stopwords : 불용어 처리

WordNetLemmatizer : 표제어 추출

모듈 엄청 많음! 구글링 해보면서 적합한걸 찾자

Korean

KONLPY

kakao

Khaiii

Kkma

Komoran

Mecab

Okt(Twitter)

Hannaum

Unit 03 | Tokenizing

Stopwords 불용어

인터넷 검색 시 검색 용어로 사용하지 않는 단어

관사, 전치사, 조사, 접속사 등 검색 색인 단어로 의미가 없는 단어

자주 나타나지만 실제 의미에 큰 기여를 하지 못하는 단어

불용어가 의미 단위인 Token에 해당하지 않도록 제거해주어야 한다.

Unit 03 | Tokenizing

아버지가방에들어가신다

Hannanum	Kkma	Komoran	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	

Contents

Unit 01 | NLP

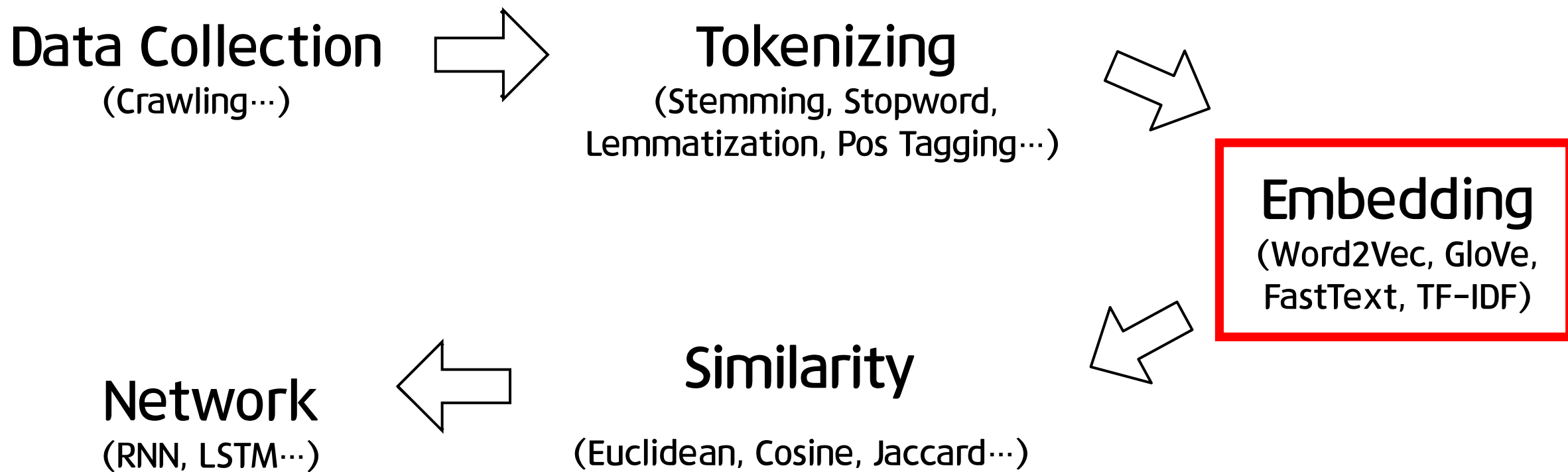
Unit 02 | Process

Unit 03 | Tokenizing

Unit 04 | Embedding

Unit 05 | Similarity

Unit 02 | Process

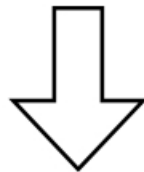


Unit 04 | Embedding

Q. 컴퓨터는 각 token을 어떻게 이해할 수 있을까?

A. 컴퓨터가 처리할 수 있는 것은 **수치** 뿐

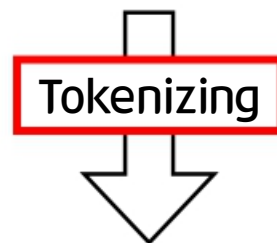
컴퓨터가 언어의 특성을 이해할 수 있도록 각 token마다 **수치를 부여!**



임베딩 Embedding

Unit 04 | Embedding


지금은 새벽 3시야 나는 강의를 준비하고 있지 자고 싶다



['지금', '은', '새벽', '3시', '야', '나', '는', '강의',
'를', '준비', '하', '고', '있지', '자', '고', '싶다']

Unit 04 | Embedding

지금은 새벽 3시야 나는 강의를 준비하고 있지 자고 싶다
이전 세션에서 우리는 이미
단어 임베딩의 한 방법을 배웠습니다!



['지금', '은', '새벽', '3시', '야', '나', '는', '강의',
'를', '준비', '하', '고', '있지', '자', '고', '싶다']

Unit 04 | Embedding

One-Hot Encoding

‘지금’:	[1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
‘은’:	[0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
‘새벽’:	[0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
‘3시’:	[0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
‘야’:	[0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
‘나’:	[0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
‘는’:	[0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
‘강의’:	[0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0]
‘를’:	[0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0]
‘준비’:	[0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0]
‘하’:	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0]
‘고’:	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0]
‘있지’:	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0]
‘자’:	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0]
‘고’:	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0]
‘싶다’:	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1]

Unit 04 | Embedding

One-Hot Encoding

‘지금’: [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
‘은’: [0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
‘새벽’: [0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
‘3시’: [0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
‘야’: [0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
‘나’: [0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
‘는’: [0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
‘강의’: [0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0]
‘를’: [0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0]
‘준비’: [0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0]
‘하’: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0]
‘고’: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0]
‘있지’: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0]
‘자’: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0]
‘고’: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0]
‘싶다’: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1]

차원이 너무 커지고
불필요한 계산이 많아지죠

Unit 04 | Embedding

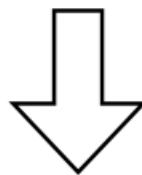
Q. One-Hot Encoding의 문제점?

A. n 개 token \rightarrow n 개 feature: 불필요한 계산이 너무 많다.

유사도 측정이 어려워 유의어, 반의어 등의 언어적 특성을 고려하기 힘들다.

Unit 04 | Embedding

단어를 좀 더 **조밀한 차원**에 **벡터**로 표현해보자!



단어 임베딩

Word2Vec, Glove, FastText

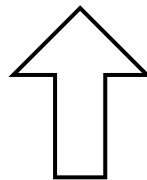
Unit 04 | Embedding

I . Word2Vec

말 그래도 Word to Vector

Word2Vec

CBOW, Skip-gram



“ 비슷한 위치에서 등장하는 단어들은
비슷한 의미를 가진다 ”

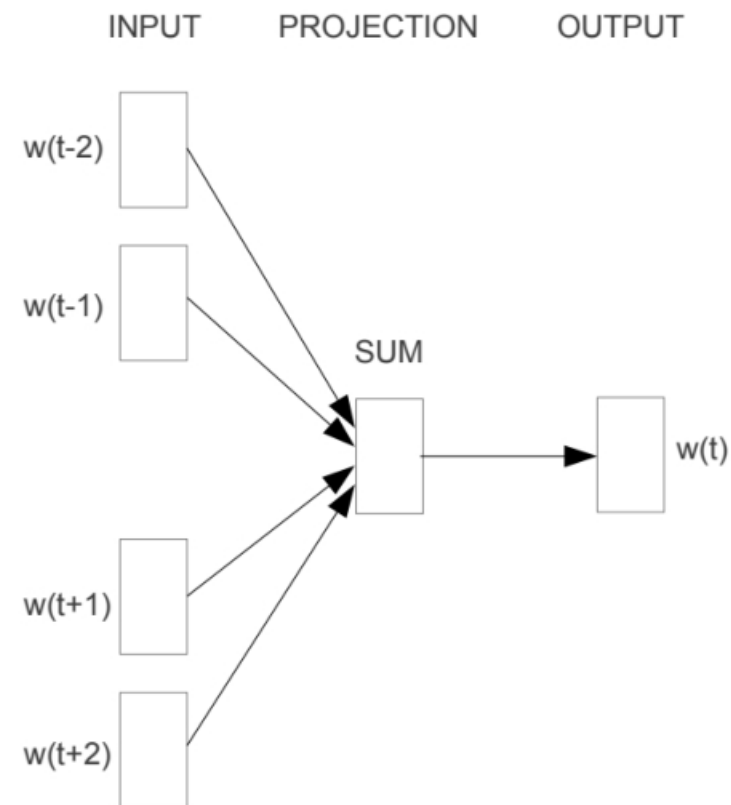
Unit 04 | Embedding

I . Word2Vec : CBOW

내가 어떻게 해야 잊을 수 있을 까

타킷

맥락

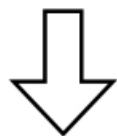


CBOW

Unit 04 | Embedding

1. Word2Vec : CBOW

‘내’, ‘가’, ‘어떻게’, ‘해야’, ‘그대’,
‘를’, ‘있을’, ‘수’, ‘있을’, ‘까’



몇 개의 단어를 맥락으로 볼 것인가?

윈도우(Window)

타깃 Center Word	맥락 Neighbor Words
‘내’	‘가’, ‘어떻게’
‘가’	‘내’, ‘어떻게’, ‘해야’
‘어떻게’	‘내’, ‘가’, ‘해야’, ‘그대’
‘해야’	‘가’, ‘어떻게’, ‘그대’, ‘를’
‘그대’	‘어떻게’, ‘해야’, ‘를’, ‘있을’
‘를’	‘해야’, ‘그대’, ‘있을’, ‘수’
‘있을’	‘그대’, ‘를’, ‘수’, ‘있을’
‘수’	‘를’, ‘있을’, ‘있을’, ‘까’
‘있을’	‘있을’, ‘수’, ‘까’
‘까’	‘수’, ‘있을’

Unit 04 | Embedding

I . Word2Vec : CBOW

Window size = 2

슬라이딩 윈도우

&

One-hot Vector

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

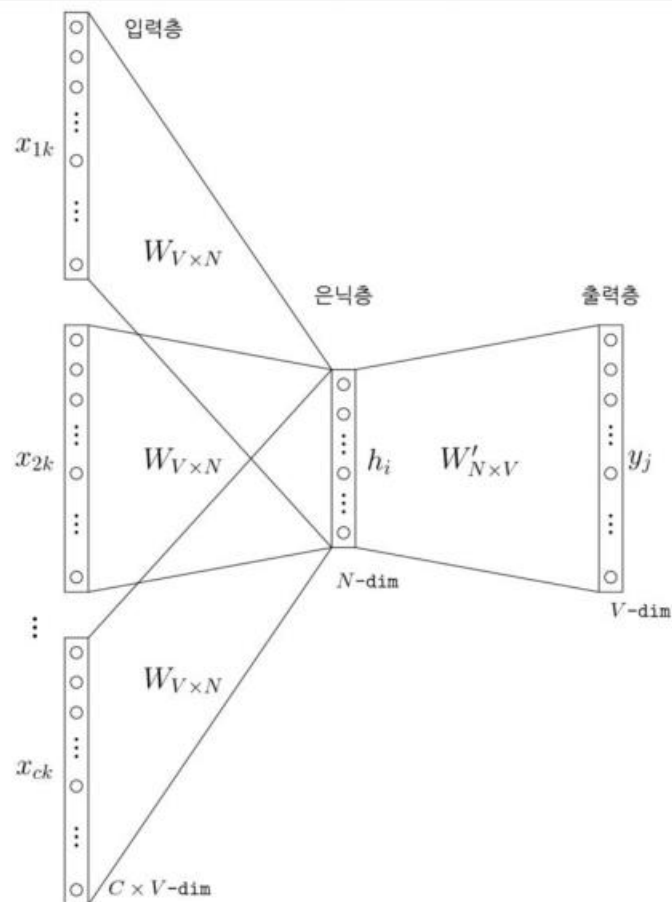
I like playing football with my friends

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]

Unit 04 | Embedding

1. Word2Vec : CBOW

입력으로
Neighbor Words의
One-hot Vector



출력으로
Center Word의
One-hot Vector

Unit 04 | Embedding

I . Word2Vec : CBOW

Q. 아까는 One-hot Encoding이 문제라면서요!

A. 여전히 문제가 되는게 맞습니다.

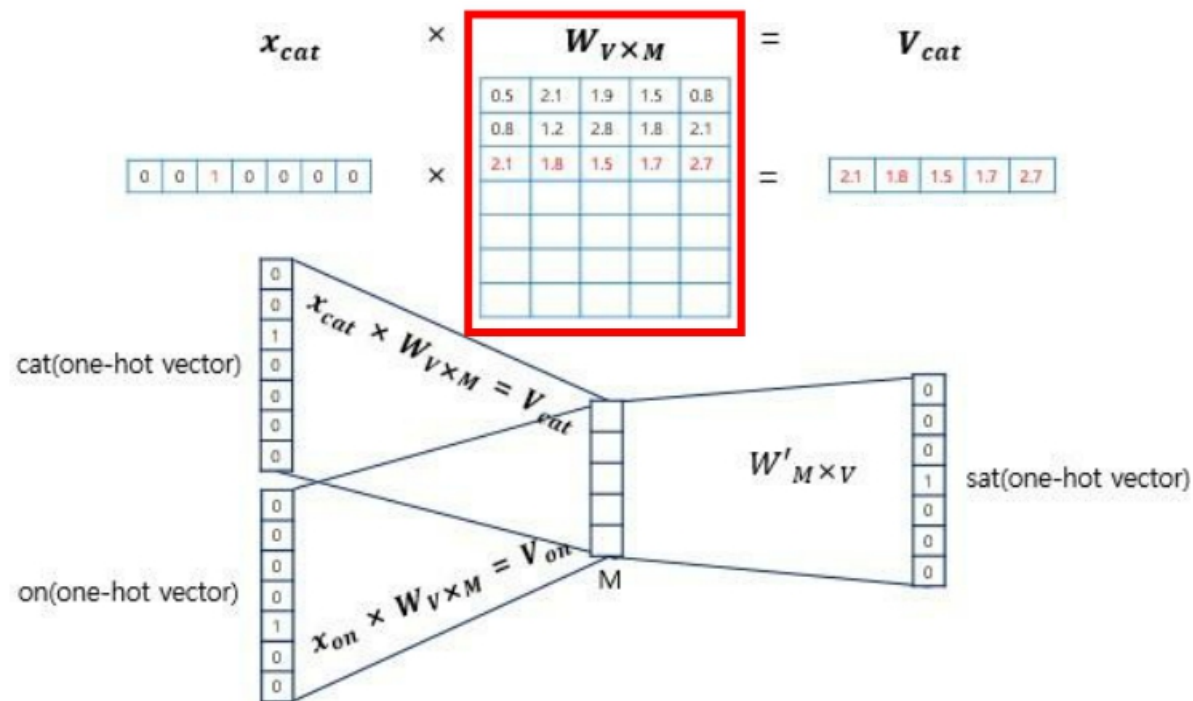
다만 Word2Vec에서 **특정 연산**을 위해 사용하고 있어요!

lookup

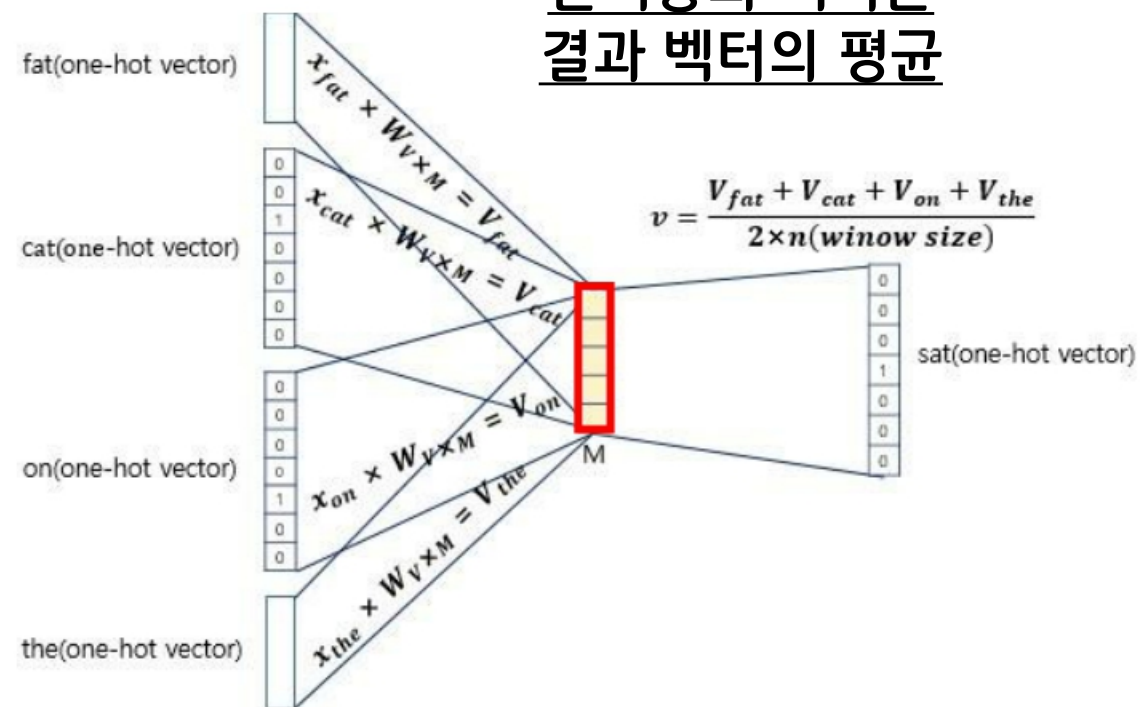
Unit 04 | Embedding

I . Word2Vec : CBOW

lookuptable

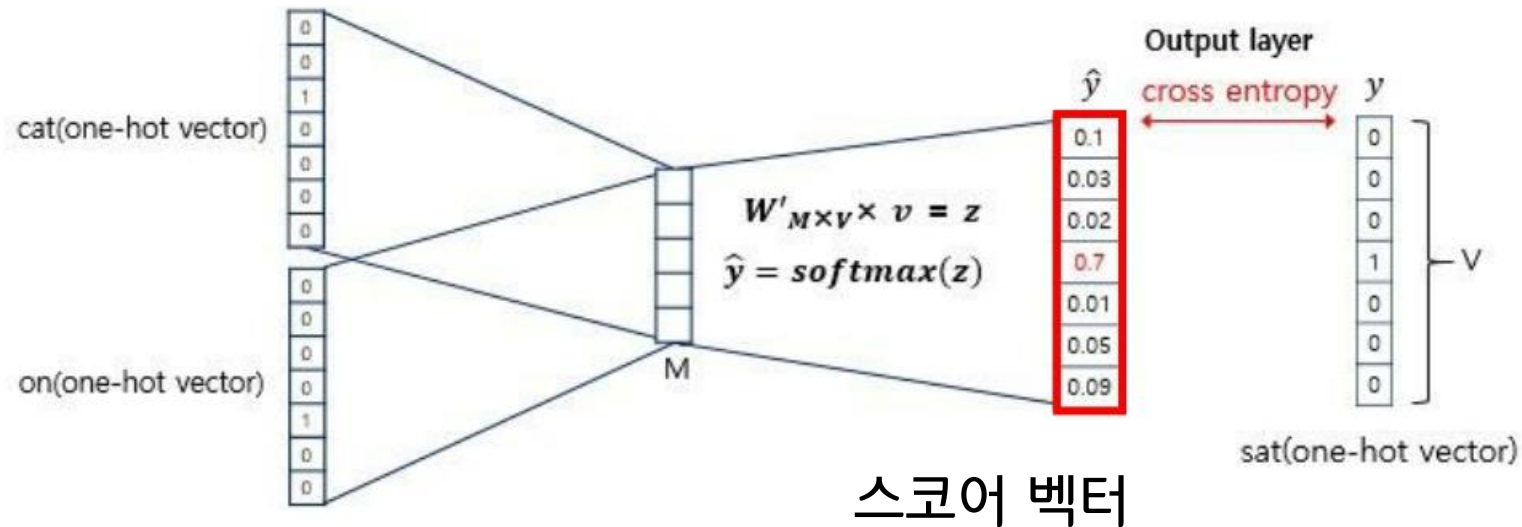


은닉층의 벡터는
결과 벡터의 평균



Unit 04 | Embedding

I . Word2Vec : CBOW



y_hat과 y 벡터값 오차를
줄이기 위한 손실함수

Cross-entropy

$$H(\hat{y}, y) = - \sum_{j=1}^{|V|} y_j \log(\hat{y}_j)$$

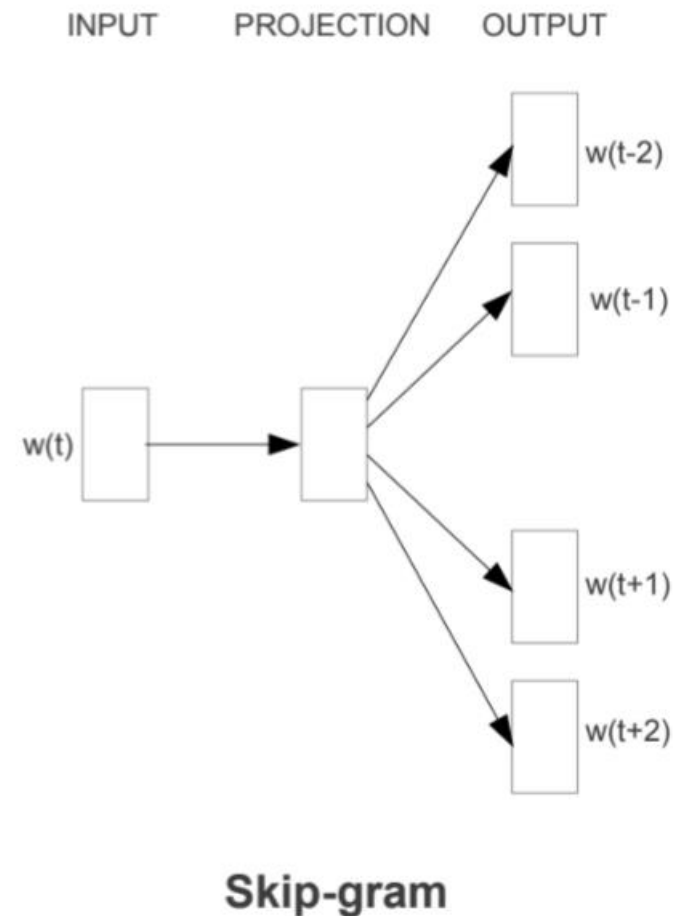
최소가 되는 방향으로 학습

Backpropagation!

Unit 04 | Embedding

1. Word2Vec : Skip-gram

내가 어떻게 맥락 그대를 맥락 수 있을까
타깃



Unit 04 | Embedding

1. Word2Vec : Skip-gram

Window size = 2

슬라이딩 윈도우

&

One-hot Vector

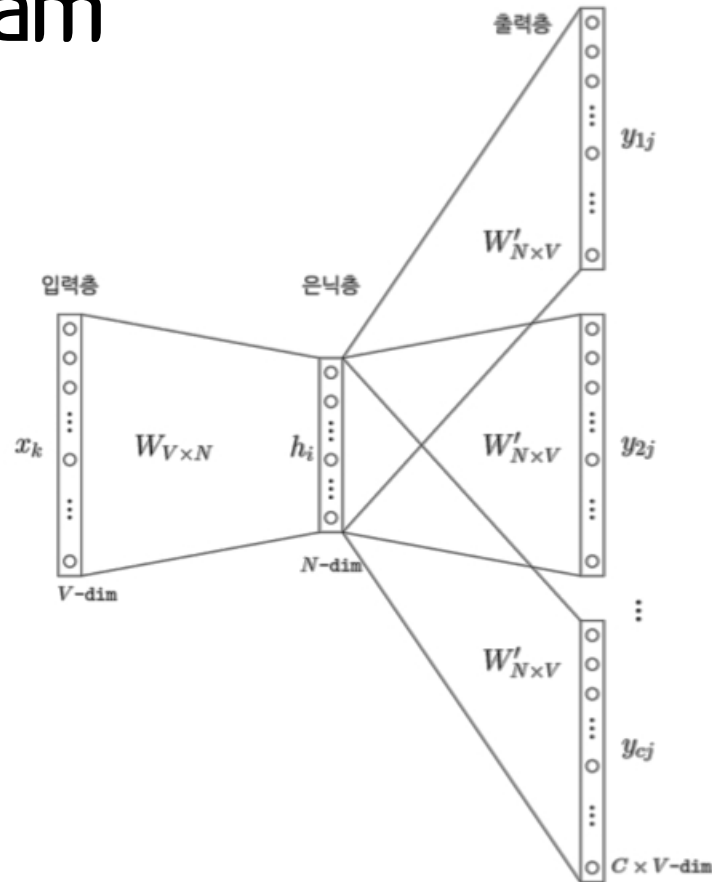
Source Text	Training Samples						
<table><tr><td>The</td><td>quick</td><td>brown</td></tr></table> fox jumps over the lazy dog. ➡	The	quick	brown	(the, quick) (the, brown)			
The	quick	brown					
<table><tr><td>The</td><td>quick</td><td>brown</td><td>fox</td></tr></table> jumps over the lazy dog. ➡	The	quick	brown	fox	(quick, the) (quick, brown) (quick, fox)		
The	quick	brown	fox				
<table><tr><td>The</td><td>quick</td><td>brown</td><td>fox</td><td>jumps</td></tr></table> over the lazy dog. ➡	The	quick	brown	fox	jumps	(brown, the) (brown, quick) (brown, fox) (brown, jumps)	
The	quick	brown	fox	jumps			
<table><tr><td>The</td><td>quick</td><td>brown</td><td>fox</td><td>jumps</td><td>over</td></tr></table> the lazy dog. ➡	The	quick	brown	fox	jumps	over	(fox, quick) (fox, brown) (fox, jumps) (fox, over)
The	quick	brown	fox	jumps	over		

Unit 04 | Embedding

I . Word2Vec : Skip-gram

입력으로
Target Word의
One-hot Vector

출력으로
Neighbor Words의
One-hot Vector



$$J = - \sum_{j=0, j \neq m}^{2m} \log P(u_{c-m+j} | v_c)$$

$$= \sum_{j=0, j \neq m}^{2m} H(\hat{y}, y_{c-m+j})$$

CBOW에 비해 중심 단어당
더 여러 번 업데이트할 수 있다
더 성능이 좋다!

Unit 04 | Embedding

I . Word2Vec의 문제점

- ① **한번에 하나**의 출현만 고려 → 전체적인 정보를 이용하지 못해 비효율적, 부정확성 증가
- ② train corpus에 **존재하지 않았던 단어**의 벡터를 만들어낼 수 없음
- ③ 단어 개수에 비례하는 계산량 → Hierarchical Softmax 및 **Negative Sampling**
SGNS (Skip-gram with Negative Sampling)
- ④ 자주 등장하지만 크게 의미가 없는 단어를 고려 → Subsampling

Unit 04 | Embedding

II. GloVe

전체 텍스트의 정보를 이용해보자!

train corpus에서 동시에 같이 등장한 단어의 빈도를 세어서
전체 corpus의 단어 개수로 나눠준 **동시 등장 확률**을 고려하자!

Unit 04 | Embedding

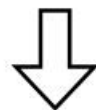
II. GloVe

Probability and Ratio	$k = solid$	$k = gas$	$k = water$	$k = fashion$
$P(k ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k ice)/P(k steam)$	8.9	8.5×10^{-2}	1.36	0.96

8.9배!

Ice가 주어졌을 때 solid가 등장할 확률 > steam이 주어졌을 때 solid가 등장할 확률

$$\text{따라서 } \frac{P(solid|ice)}{P(solid|steam)} > 1, \text{ 반대로 } \frac{P(gas|ice)}{P(gas|steam)} < 1$$



이러한 동시 등장 확률의 특징을 활용해 임베딩하자!

Unit 04 | Embedding

II. GloVe

“ 임베딩 된 중심 단어와 주변 단어 벡터의 내적이 ”

전체 코퍼스에서의 동시 등장 확률이 되도록 만드는 것

$$\text{dot product}(w_i \tilde{w}_k) \approx \log P(k | i) = \log P_{ik}$$

$$F((w_i - w_j)^T \tilde{w}_k) = \frac{P_{ik}}{P_{jk}}$$

단어 벡터들에게 어떠한 함수 F를 수행하면 그 단어들 간의 확률의 비가 나온다.

$$F(w_i^T \tilde{w}_k - w_j^T \tilde{w}_k) = \frac{F(w_i^T \tilde{w}_k)}{F(w_j^T \tilde{w}_k)}$$

준동형(Homomorphism)을 만족하도록 식을 변형한다.

$$\exp(w_i^T \tilde{w}_k) = P_{ik} = \frac{X_{ik}}{X_i}$$

F를 만족하는 함수? 지수함수!

$$\text{Loss function} = \sum_{m,n=1}^V f(X_{mn})(w_m^T \tilde{w}_n + b_m + \tilde{b}_n - \log X_{mn})^2$$

편향 추가, 가중치 함수 추가 등등하여 일반화 시킨 손실 함수

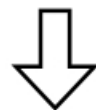
Unit 04 | Embedding

III. FastText

Word2Vec의 한계

Out-of-Vocabulary(OOV)

Word2Vec은 단어 단위로 어휘집(Vocabulary)를 구성하기 때문에
어휘집에 없는 새로운 단어가 등장하면 데이터 전체를 다시 학습시켜야 함



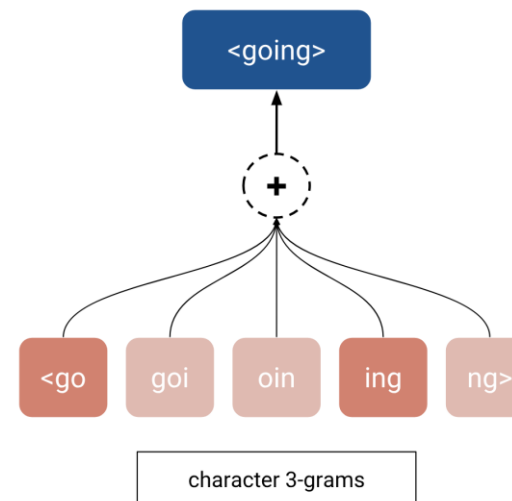
더 낮은 단위로 입력을 내려보자!

Unit 04 | Embedding

III. FastText

: SGNS의 확장판 ~

Subword Embedding

단어가 아닌 단어 내부의 n -gram이 최소 단위!어휘를 구성하는 모든 n -gram 벡터의 평균 벡터로 단어 임베딩!

Unit 04 | Embedding

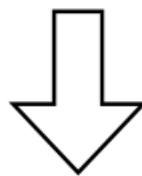
III. FastText

- ① train corpus에 **존재하지 않았던** 단어의 embedding이 가능함 (ex) disaster와 disastrous
- ② 희소한 단어에 대해 더 좋은 embedding이 가능함
- ③ 문법에 따라 **변화하는 패턴**도 학습하기 용이
- ④ 적은 데이터로도 높은 성능

Unit 04 | Embedding

단어를 넘어서 **문서**로 시선을 높여볼까요?

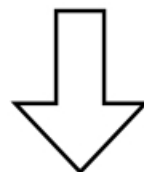
단어가 모이면 문장을 이루고, 문장이 모이면 문서가 이루어져요



문서 임베딩

Unit 04 | Embedding

문서에서 많이 나타나는 단어(빈도)로 임베딩을 하면 되지 않을까?



모든 문서에 많이 나오는 단어는 어떻게 처리할 수 있을까?

(관사, 전치사... 별로 중요하지 않은데 빈도수는 높네)

Unit 04 | Embedding

I . TF-IDF

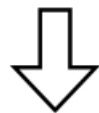
단어의 빈도와 역 문서 빈도를 사용하여 **DTM(Document Term Matrix)** 내의 각 단어들마다 중요한 정도를 가중치로 주는 방법

-	과일이	길고	노란	먹고	바나나	사과	싶은	저는	좋아요
문서1	0	0	0	1	0	1	1	0	0
문서2	0	0	0	1	1	0	1	0	0
문서3	0	1	1	0	2	0	0	0	0
문서4	1	0	0	0	0	0	0	1	1

Unit 04 | Embedding

I . TF-IDF

단어의 빈도와 **역 문서 빈도**를 사용하여 DTM(Document Term Matrix) 내의 각 단어들마다 중요한 정도를 가중치로 주는 방법



단어 빈도 (Term Frequency) = $tf(d, t)$ = 특정 문서 d 에서 특정 단어 t 의 등장 횟수

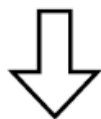
문서 빈도 (Document Frequency) = $df(t)$ = 특정 단어 t 가 등장한 문서의 수

역 문서 빈도 (Inverted Document Frequency) = $idf(d, t)$ = $df(t)$ 에 반비례하는 수

Unit 04 | Embedding

1. TF-IDF

단어의 빈도와 역 문서 빈도를 사용하여 DTM(Document Term Matrix) 내의 각 단어들마다 중요한 정도를 가중치로 주는 방법



역 문서 빈도(IDF)

$$w_{t,d} = \underbrace{tf_{t,d}}_{\text{단어의 빈도(TF)}} \times \underbrace{\log_{10}\left(\frac{n}{1 + df(t)}\right)}_{\text{역 문서 빈도(IDF)}}$$

Unit 04 | Embedding

I . TF-IDF

-	과일이	길고	노란	먹고	바나나	사과	싫은	저는	좋아요
문서1	0	0	0	1	0	1	1	0	0
문서2	0	0	0	1	1	0	1	0	0
문서3	0	1	1	0	2	0	0	0	0
문서4	1	0	0	0	0	0	0	1	1

DTM에서 문서별
Term Frequency
확인하기

Unit 04 | Embedding

1. TF-IDF

-	과일이	길고	노란	먹고	바나나	사과	싫은	저는	좋아요
문서1	0	0	0	1	0	1	1	0	0
문서2	0	0	0	1	1	0	1	0	0
문서3	0	1	1	0	2	0	0	0	0
문서4	1	0	0	0	0	0	0	1	1



문서 빈도 (Document Frequency)

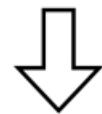
	과일이	길고	노란	먹고	바나나	사과	싫은	저는	좋아요
DF	1	1	1	2	2	1	2	1	1

Unit 04 | Embedding

I . TF-IDF

문서 빈도 (Document Frequency)

	과일이	길고	노란	먹고	바나나	사과	싫은	저는	좋아요
DF	1	1	1	2	2	1	2	1	1

역 문서 빈도 (Inverted Document Frequency) $\log\left(\frac{n}{1 + df(t)}\right)$

	과일이	길고	노란	먹고	바나나	사과	싫은	저는	좋아요
IDF	0.693	0.693	0.693	0.287	0.287	0.693	0.287	0.693	0.693

Unit 04 | Embedding

I . TF-IDF

 $tf_{t,d}$

×

 $log(\frac{n}{1 + df(t)})$

-	과일이	길고	노란	먹고	바나나	사과	싫은	저는	좋아요
문서1	0	0	0	1	0	1	1	0	0
문서2	0	0	0	1	1	0	1	0	0
문서3	0	1	1	0	2	0	0	0	0
문서4	1	0	0	0	0	0	0	1	1

	과일이	길고	노란	먹고	바나나	사과	싫은	저는	좋아요
IDF	0.693	0.693	0.693	0.287	0.287	0.693	0.287	0.693	0.693

Contents

Unit 01 | NLP

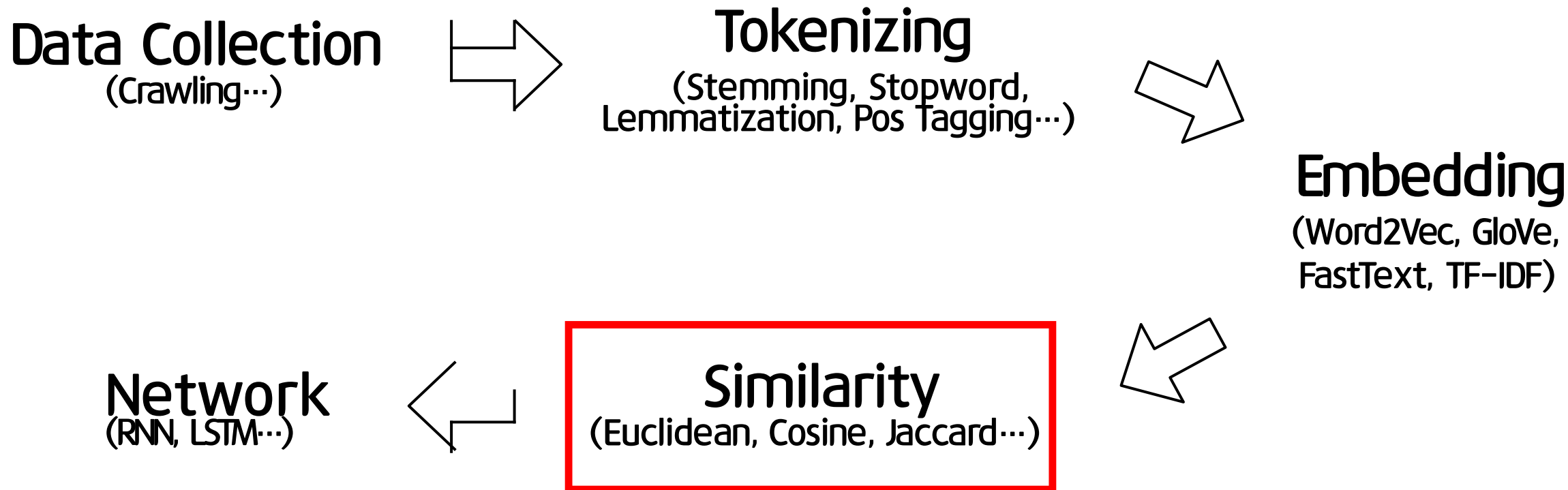
Unit 02 | Process

Unit 03 | Tokenizing

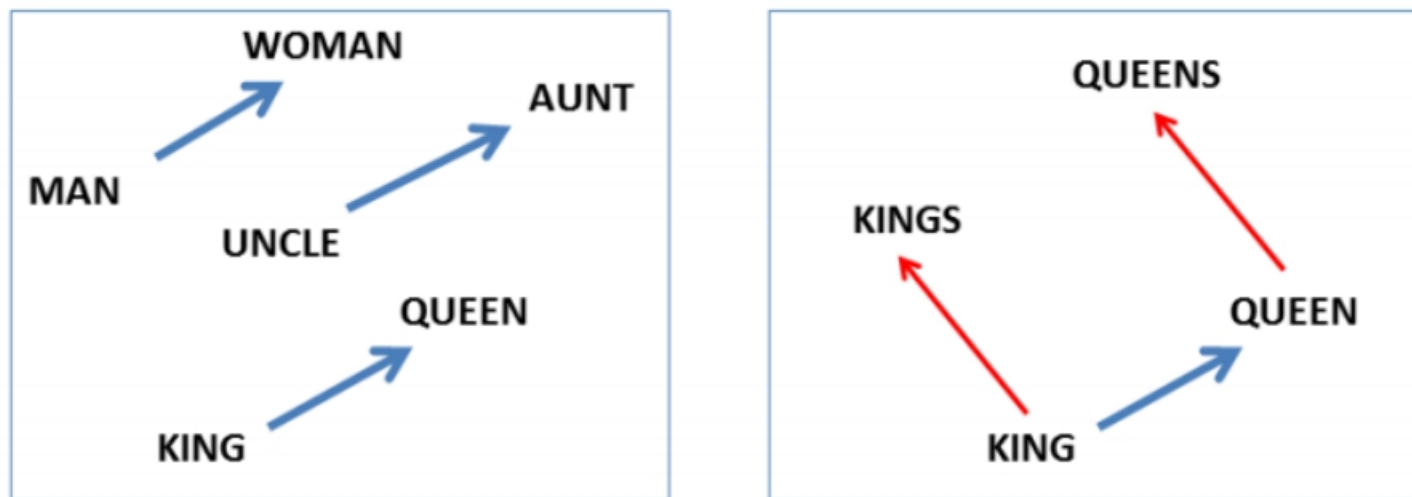
Unit 04 | Embedding

Unit 05 | Similarity

Unit 03 | Tokenizing



Unit 05 | Similarity



(Mikolov et al., NAACL HLT, 2013)

유사도를 구해 의미론적 해석을 이끌어낼 수 있다!

Unit 05 | Similarity

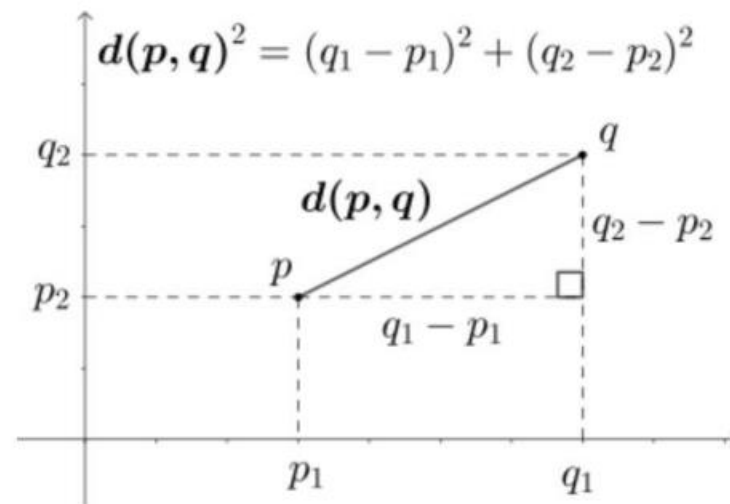
I . Euclidean Similarity

두 점 사이의 거리를 계산

-	바나나	사과	저는	좋아요
문서Q	1	1	0	1

문서Q와 가장 유사한 문서는?

-	바나나	사과	저는	좋아요
문서1	2	3	0	1
문서2	1	2	3	1
문서3	2	1	2	2



$$\sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \dots + (p_n - q_n)^2} = \sqrt{\sum_{i=1}^n (p_i - q_i)^2}$$

Unit 05 | Similarity

I . Euclidean Similarity

두 점 사이의 거리를 계산

-	바나나	사과	저는	좋아요
문서Q	1	1	0	1

-	바나나	사과	저는	좋아요
문서1	2	3	0	1
문서2	1	2	3	1
문서3	2	1	2	2

```
import numpy as np
def dist(x, y):
    return np.sqrt(np.sum((x-y)**2))

doc1 = np.array((2,3,0,1))
doc2 = np.array((1,2,3,1))
doc3 = np.array((2,1,2,2))
docQ = np.array((1,1,0,1))

print(dist(doc1, docQ))
print(dist(doc2, docQ))
print(dist(doc3, docQ))

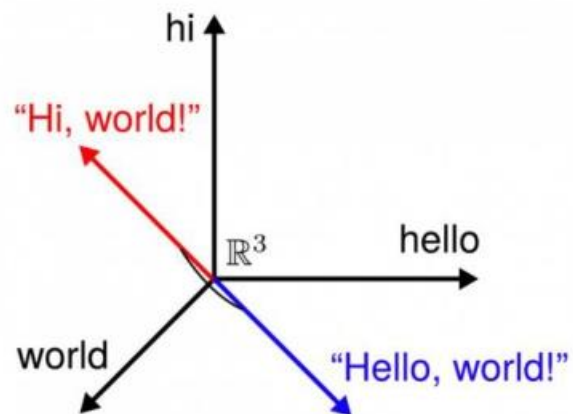
2.23606797749979
3.1622776601683795
2.449489742783178
```

Euclidean Similarity가 가장 낮은 문서1과 유사하다고 할 수 있다!
(= 문서 간의 거리가 가장 가까운)

Unit 05 | Similarity

II. Cosine Similarity

두 벡터 간의 코사인 각도로 유사도 측정
 1에 가까울 수록 유사도가 높다!
 -1에 가까울 수록 정반대의 유사도를 갖는다!



Cosine Similarity

-	바나나	사과	저는	좋아요
문서1	0	1	1	1
문서2	1	0	1	1
문서3	2	0	2	2



$$\text{similarity}(A,B) = \frac{A \cdot B}{\|A\| \times \|B\|} = \frac{\sum_{i=1}^n A_i \times B_i}{\sqrt{\sum_{i=1}^n A_i^2} \times \sqrt{\sum_{i=1}^n B_i^2}}$$

```
from numpy import dot
from numpy.linalg import norm
import numpy as np
def cos_sim(A, B):
    return dot(A, B)/(norm(A)*norm(B))

doc1=np.array([0,1,1,1])
doc2=np.array([1,0,1,1])
doc3=np.array([2,0,2,2])

print(cos_sim(doc1, doc2)) #문서1과 문서2의 코사인 유사도
print(cos_sim(doc1, doc3)) #문서1과 문서3의 코사인 유사도
print(cos_sim(doc2, doc3)) #문서2과 문서3의 코사인 유사도

0.67
0.67
1.00
```

Unit 05 | Similarity

III. Jaccard Similarity

- 두 집합의 교집합의 크기를 합집합의 크기로 나눈 값으로 두 문서(집합)의 유사도를 측정
- 0에서 1사이의 값을 가지며 두 집합 사이에 교집합이 없으면 0, 두 집합이 동일하면 1의 값을 가짐

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

문서 A: 그대 제품에 안겨 눈을 감아요

문서 B: 그대 제품에 안겨 사랑의 꿈 나눠요

	그대	제품에	안겨	눈을	감아요	사랑의	꿈	나눠요
문서 A	0	0	0	0	0	X	X	X
문서 B	0	0	0	X	X	0	0	0

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} = \frac{3}{8}$$

Unit 05 | Similarity

III. Jaccard Similarity

- 두 집합의 교집합의 크기를 합집합의 크기로 나눈 값으로 두 문서(집합)의 유사도를 측정
- 0에서 1사이의 값을 가지며 두 집합 사이에 교집합이 없으면 0, 두 집합이 동일하면 1의 값을 가짐

실습 코드!

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

문서 A: 그대 내품에 안겨 눈을 감아요

문서 B: 그대 내품에 안겨 사랑의 꿈 나뉘요

	그대	내품에	안겨	눈을	감아요	사랑의	꿈	나뉘요
문서 A	O	O	O	O	O	X	X	X
문서 B	O	O	O	X	X	O	O	O

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} = \frac{3}{8}$$

Assignment

과제: NLP 제대로 맛보기

Step1. 데이터 확인 (데이터가 어떻게 생겼는지 먼저 확인해봐야겠죠?)

Step2. Tokenizing (불용어 처리, 특수 문자 제거 등의 전처리 포함)

Step3. 임베딩 (One-hot encoding, CBOW, Skip-gram, GloVe, FastText 등)

Step4. 유의미한 해석 도출 (유사도, Wordcloud, 이진 분류 모델, 그래프 해석 등)

Assignment

[주의사항] 정답이 없는 과제인 만큼, 여러분이 하기 나름입니다!

- 임베딩 모델을 적어도 2개 이상 적용해본 후, 해석에 따라 가장 좋은 모델을 선택해주세요.
(ex) CBOW, Skip-gram, GloVe, FastText 등
- 유의미한 해석을 도출하는 것이 핵심입니다. 워드클라우드, 유사도, 이진분류모델 등을 활용하여 세 가지 이상의 인사이트를 도출해주세요.
(ex) 유사도 하나 말하고 그걸로 인사이트 끝? 글썄요 그게 정녕 인사이트일까요
- 토큰나이저 및 임베딩 모델 선택 과정이나 인사이트 해석은 주석으로 설명 필수

Assignment

[우수과제 선정 기준]

- ① 토큰나이저 및 임베딩을 선택한 판단 근거가 명확한가 (파라미터 포함)
- ② NLP에 대해 스스로 공부하고 고민한 흔적이 보이는가
- ③ **인사이트의 창의성**
- ④ 전처리를 얼마나 꼼꼼히 진행하였는가
- ⑤ 주석이 가득한 정성이 담긴 과제

Reference

참고자료

- ToBig's 14기 정규세션 NLP 기초 강의 (정주원님)
 - <https://wikidocs.net/22660>
 - <https://ratsgo.github.io/from%20frequency%20to%20semantics/2017/03/30/word2vec/>
 - <https://wikidocs.net/22885>
 - <https://brunch.co.kr/@learning/7>
 - <https://eda-ai-lab.tistory.com/428>
 - <https://wikidocs.net/24603>
 - <http://solarisailab.com/archives/959>
- 자연어처리 바이블 (임희석 | 고려대학교 자연어처리연구실 저)
- 밑바닥부터 시작하는 딥러닝. 2 (사이토 고키 지음)
- 2019년 2학기 고려대학교 이상근 교수님 정보검색 수업, 2020년 1학기 고려대학교 임희석 교수님 자연어처리 수업

Q&A

들어주셔서 감사합니다.