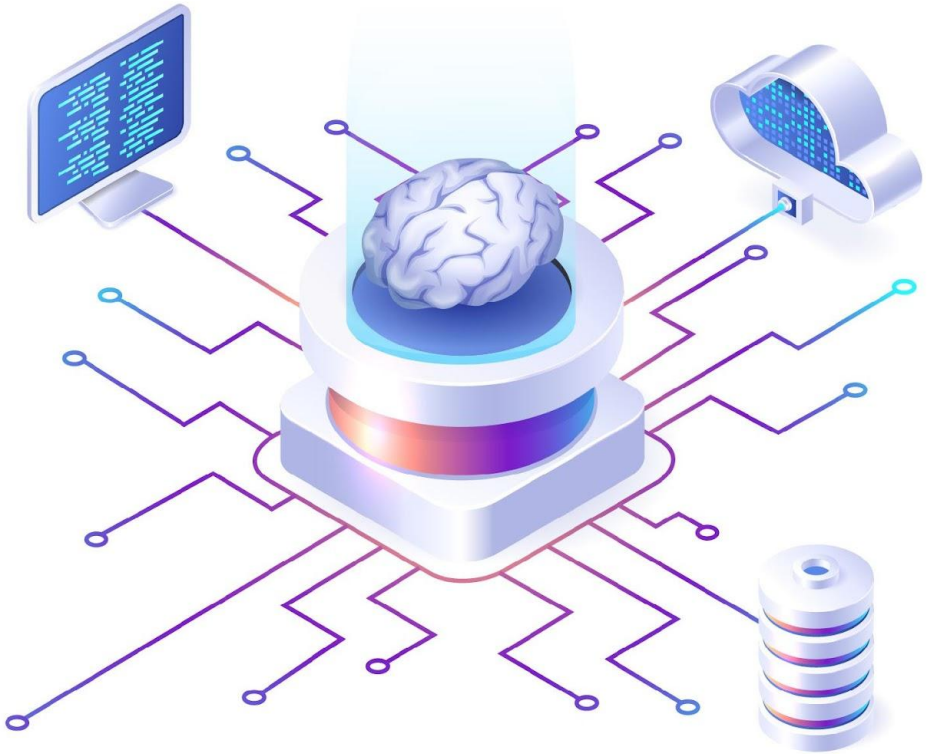
Insight campus

자연어 처리란?

실무형 인공지능 자연어처리



Insight campus

텍스트 전처리란?

자연어 처리를 위해

용도에 맞도록

사전에 표준화 하는 작업

Insight campus

텍스트 전처리 왜 필요한가?

텍스트 내 정보를 유지하고,

분석의 효율성을 높이기 위해

텍스트 전처리 개요

- 분석 하기 전 텍스트를 분석에 적합한 형태로 변환하는 작업
- 전처리 단계는 텍스트를 토큰화하고 자연어 처리에 필요없는 조사, 특수문자, 단어(불용어)의 제거과정을 포함
- 전처리는 분석결과와 모델 성능에 직접 영향을 미치기 때문에 전처리 단계는 매우 중요

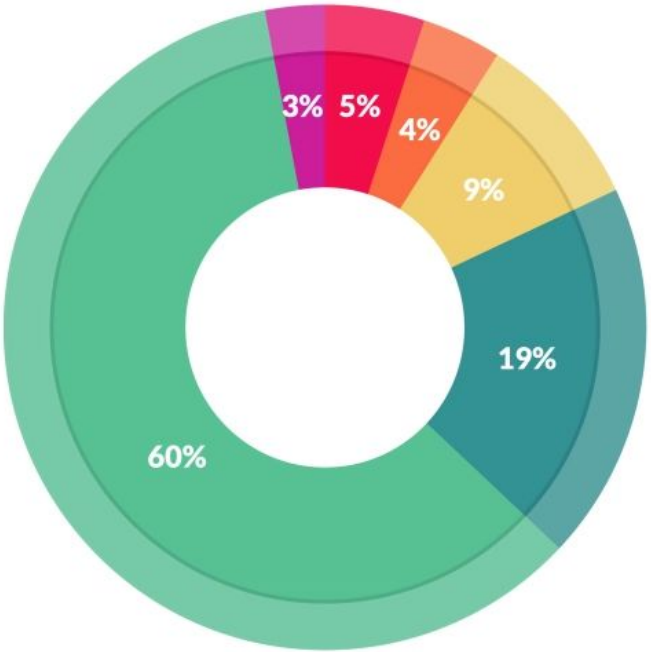
1. 토큰화	구두점으로 문서를 문장으로 분리하는 문장 토큰화과 단어 단위로 분리하는 단어 토큰화로 구성된다.
2. 형태소 분석	뜻을 가진 가장 작은 단위인 형태소로 분리하는 과정이다.
3. 품사 태깅	분리된 토큰에 품사를 태깅하는 과정이다.
4. 원형 복원	단어의 원형을 복원하여 표준화하는 과정이다. Stemming방식과 Lemmatization방식이 있다.
5. 불용어 처리	분석에 불필요한 단어나 방해되는 단어를 제거하는 과정이다.

Insight campus

전처리 중요성

Insight campus

전처리 중요성



What data scientists spend the most time doing

- *Building training sets: 3%*
- *Cleaning and organizing data: 60%*
- *Collecting data sets; 19%*

79%

- *Mining data for patterns: 9%*
- *Refining algorithms: 4%*
- *Other: 5%*

https://whatsthebigdata.com/2016/05/01/data-scientists-spend-most-of-their-time-cleaning-data/

텍스트 전처리 (Text Preprocessing)

텍스트 전처리 (Text Preprocessing)

1

토큰화
(Tokenization)

Insight campus

토큰화 (Tokenization)

- 텍스트를 자연어 처리를 위해 분리 하는 것
- 토큰화는
  문장별로 분리하는 "문장 토큰화(Sentence Tokenization)"와
  단어별로 분리하는 "단어 토큰화(Word Tokenization)"로 구분

Insight campus

문장 토큰화 (Sentence Tokenization)

- 문장(Sentence)를 기준으로 토큰화
- 온점(.), 느낌표(!), 물음표(?) 등으로 분류하면 해결 될 것으로 생각됨
- 하지만 단순하게 분리할 경우 정확한 분리가 어려움

Barack Obama likes fried chicken. He don't like spicy chicken.

Barack Obama likes fried chicken.

He don't like spicy chicken.

Insight campus

단어 토큰화(Word Tokenization)

- 단어(word)를 기준으로 토큰화
- 영문의 경우 공백을 기준으로 분리하면 유의미한 토큰화가 가능
- 반면 한글의 경우 품사를 고려한 토큰화(=형태소분석)가 필요

단어 토큰화 고려사항

- 특수문자가 있는 경우
  (구두점 및 특수문자를 단순하게 제외해서는 안됨)

특수문자	원문	토큰화 예제1	토큰화 예제2
‘	Don’t	Do / n’t	Don / ‘ / t
-	State-of-the-art	State / of / the / art	State-of-the-art

- 단어 내 띄어쓰기가 있는 경우

	원문	토큰화 예제1	토큰화 예제2
공백	New York	New / York	New York

2

품사태깅
(PoS Tagging)

Insight campus

품사 부착(PoS Tagging)

- 각 토큰에 품사 정보를 추가
- 분석시에 불필요한 품사를 제거하거나 (예. 조사, 접속사 등) 필요한 품사를 필터링 하기 위해 사용

Barack Obama likes fried chicken very much.

Barack / NNP 명사	Obama / NNP 명사	likes / VBZ 동사	fried / VBN 동사	chicken / JJ 형용사	very / RB 부사	much / RB 부사	.

3

개체명 인식
(NER, Named Entity Recognition)

Insight campus

개체명 인식 (NER, Named Entity Recognition)

- 사람, 조직, 지역, 날짜, 숫자 등 개체 유형을 식별
- 검색 엔진 색인에 활용

Barack Obama likes fried chicken very much.

Barack / NNP **PERSON**	Obama / NNP **ORGANIZATION**	likes / VBZ	fried / VBN	chicken / JJ	very / RB	much / RB	.

4

원형 복원
(Stemming & Lemmatization)

어간 추출 (Stemming)

- 각 토큰의 원형 복원을 함으로써 토큰을 표준화하여 불필요한 데이터 중복을 방지 (=단어의 수를 줄일수 있어 연산을 효율성을 높임)
- 어간 추출(Stemming) : 품사를 무시하고 **<u>규칙에 기반하여 어간을 추출</u>**
  규칙 : https://tartarus.org/martin/PorterStemmer/def.txt

원문	Stemming
running	run
beautiful	beauti
believes	believ
using	use
conversation	convers
organization	organ
studies	studi

표제어 추출 (Lemmatization)

- 각 토큰의 원형 복원을 함으로써 토큰을 표준화하여 불필요한 데이터 중복을 방지 (=단어의 수를 줄일수 있어 연산을 효율성을 높임)
- 표제어 추출 (Lemmatization) : **품사정보를 유지하여 표제어 추출 (사전 기반)**

원문	Lemmatization
running	running
beautiful	beautiful
believes	belief
using	using
conversation	conversation
organization	organization
studies	study

5

불용어 처리
(Stopwords)

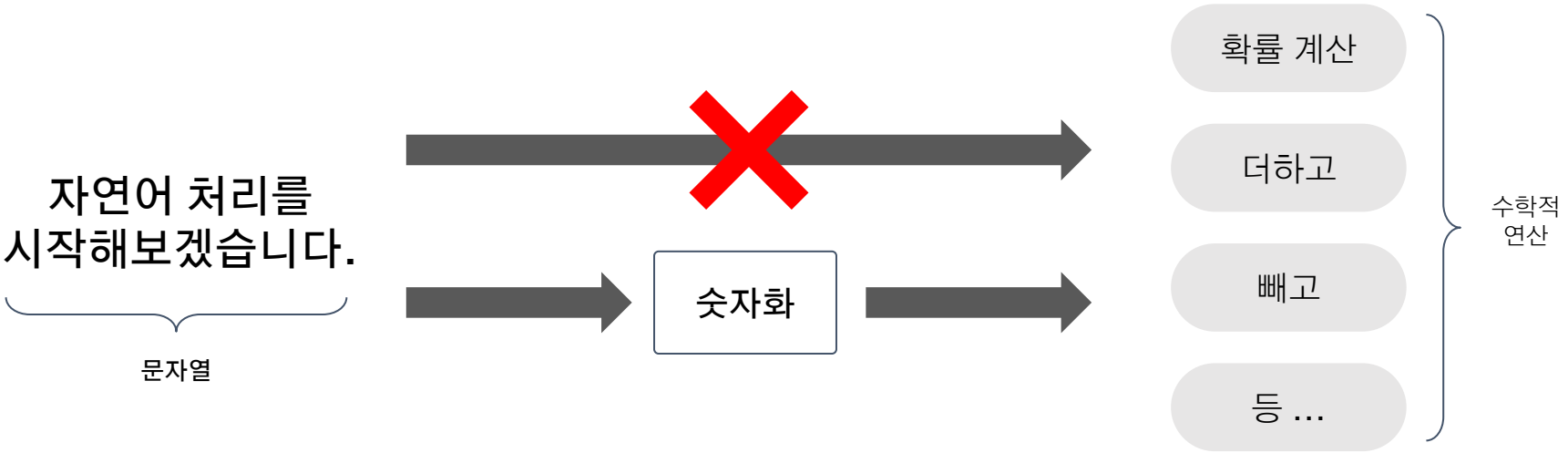
Insight campus

불용어 처리(Stopwords)

- 불필요한 토큰을 제거 하는 작업
- 불필요한 품사를 제거 하기도 함

Insight campus

단어의 표현이 필요한 이유



단어의 표현

표현(Representation)

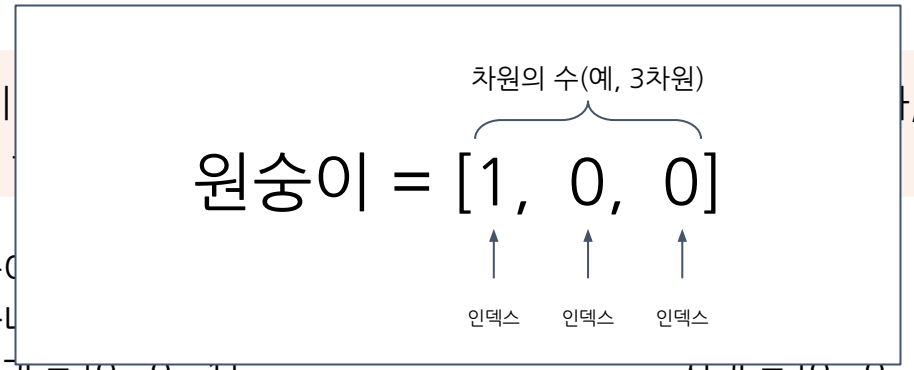
1 원핫-인코딩 (One-Hot-Encoding)

Insight campus

원핫-인코딩(One-Hot-Encoding)

원핫-인코딩은 단어(word)를 숫자로 표현하고자 할 때 적용할 수 있는 간단한 방법론

원숭이



, 코끼리를

원숭이
바나나
사과 = [0, 0, 1]

0, 0]
0, 0]
사과 = [0, 0, 1, 0]
코끼리 = [0, 0, 0, 1]

Insight campus

원핫-인코딩(One-Hot-Encoding) 한계점

차원 크기의 문제

원숭이, 바나나, 사과를
표현할 때

원숭이 = [1, 0, 0]
바나나 = [0, 1, 0]
사과 = [0, 0, 1]

단어의 수만큼 차원이 필요함

단어수가 많아진다면?

2017년 표준국어대사전에 등재된 단어 수 약 50만개
=> 50만개의 차원이 필요

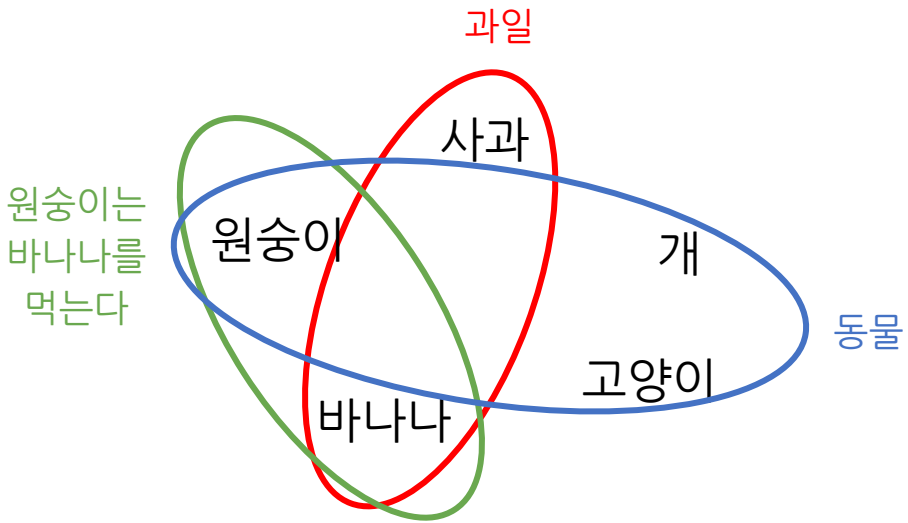
원숭이 = [1, 0, 0, 0, 0, 0, 0, 0, 0, ... 0, 0, 0, 0, 0]

50만 차원 벡터

Insight campus

원핫-인코딩(One-Hot-Encoding) 한계점

의미를 담지 못하는 문제

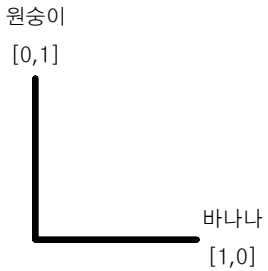


Insight campus

원핫-인코딩(One-Hot-Encoding) 한계점

의미를 담지 못하는 문제

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



$$\text{similarity} = \frac{(1 \times 0) + (0 \times 1)}{(1^2 + 0^2) \times (0^2 + 1^2)} = 0$$

Insight campus

원핫-인코딩(One-Hot-Encoding) 한계점

의미를 담지 못하는 문제

원숭이, 바나나, 사과, 개, 고양이를 표현할 때

원숭이 = [1, 0, 0, 0, 0]
바나나 = [0, 1, 0, 0, 0]
사과 = [0, 0, 1, 0, 0]
개 = [0, 0, 0, 1, 0]
고양이 = [0, 0, 0, 0, 1]

- “원숭이, 사과” 코사인 유사도 : 0
- “원숭이, 바나나" 코사인 유사도 : 0
- “개, 고양이" 코사인 유사도 : 0

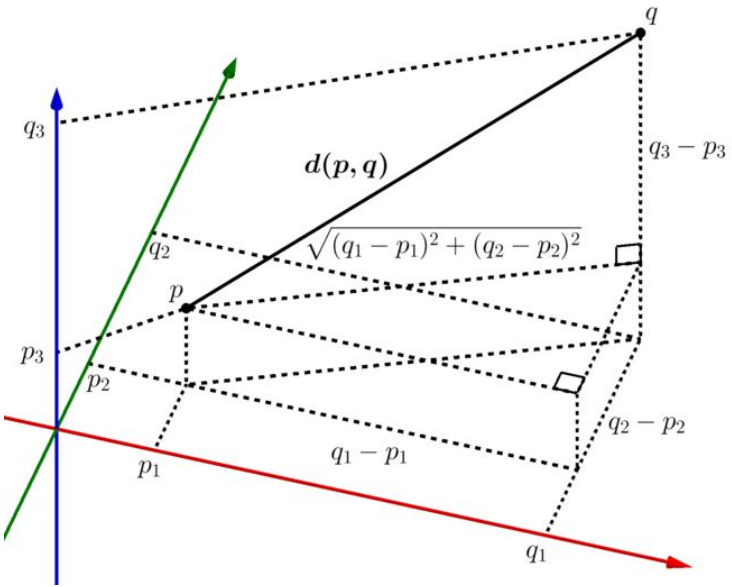
=> 원핫 벡터간 코사인 유사도는 모두 0

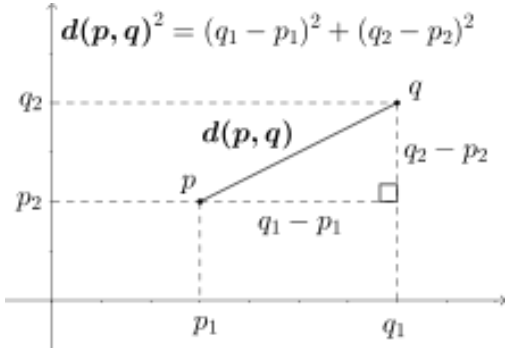
=> 따라서 의미를 분간 하기 어려움

2 유사도 계산(Similarity)

Insight campus

유클리디언 거리(Euclidean distance)





$$d(\mathbf{p}, \mathbf{q}) = d(\mathbf{q}, \mathbf{p}) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2}$$

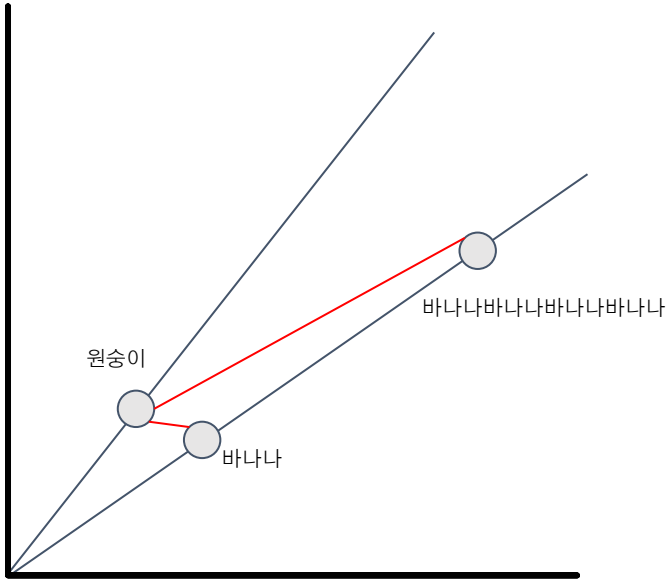
$$= \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}.$$

https://en.wikipedia.org/wiki/Euclidean_distance

벡터간 직진거리 계산

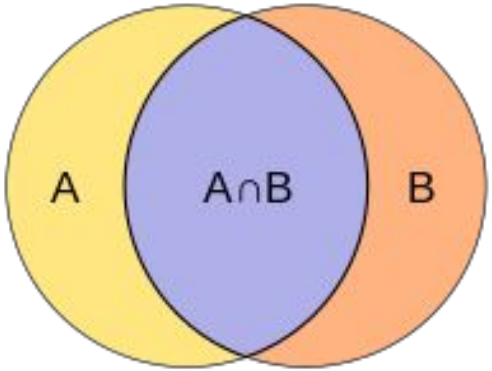
Insight campus

유클리디안 거리의 한계점



Insight campus

자카드 유사도(Jaccard index)



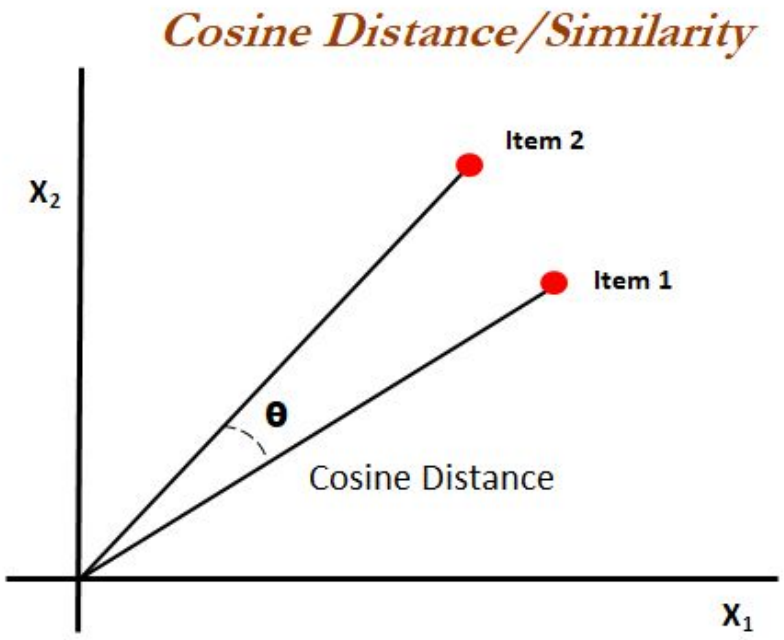
$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}$$

https://en.wikipedia.org/wiki/Jaccard_index

문서 혹은 문장간 유사도 측정 (겹치는 토큰의 비율)

Insight campus

코사인 유사도(Cosine Similarity)



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

- 두 벡터간의 유사도를 측정하는 방법 중 하나
- 두 벡터 사이의 코사인을 측정
- 0도 = 1, 90도 = 0, 180도 = -1
  => 1에 가까울수록 유사도가 높음
  => 유사도가 높다는 것은 유사한 의미를 가짐을 의미

https://en.wikipedia.org/wiki/Cosine_similarity

두 벡터간 각(코사인 유사도)을 이용한 유사도 측정

3 단어 임베딩 (Word Embedding)

Insight campus

원핫-인코딩(One-Hot-Encoding) 한계점

벡터로 표현한 단어 차원이 너무 큼 → 연산이 낭비되어 모델 학습에 불리하게 적용

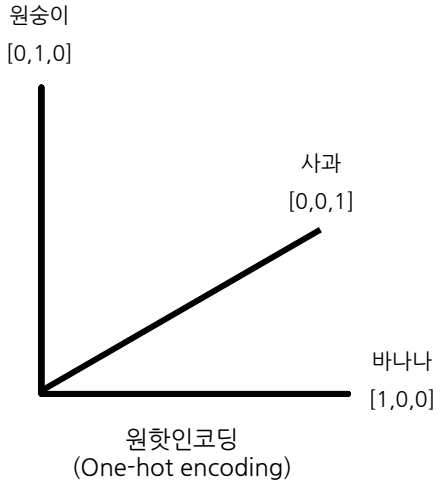
단어 의미를 담지 못함 → 분석을 효과적으로 수행할 수 없음

Insight campus

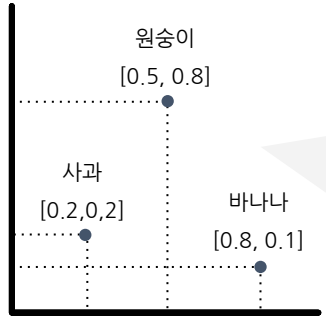
단어 임베딩(Word embedding)

단어 임베딩은 단어의 의미를 간직하는 밀집 벡터(Dense Vector)로 표현하는 방법

원숭이, 바나나, 사과를 표현할 때



원핫인코딩
(One-hot encoding)



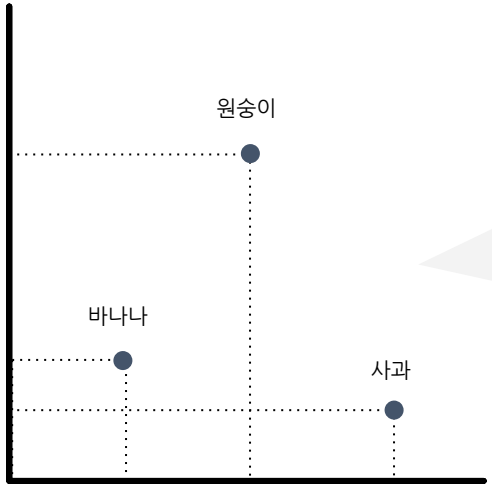
밀집 벡터
(Dense Vector)

- 벡터가 공간에 꽉차 있음
- 새로운 단어 추가시 차원을 추가할 필요가 없음
  => 차원을 줄일 수 있음
  => 추후 분류나 예측 모델을 학습할 때 연산을 줄일 수 있는 이점을 갖음

Insight campus

단어 임베딩 (Word Embedding)의 한계

벡터로 표현한 단어 차원이 너무 큼 → 밀집 벡터(Dense vector)로 해결



사과 벡터는 어디에
표현되는 것이 맞을까요?

=> 단어를 벡터로
표현하는 명확한 방법이
존재하지 않음

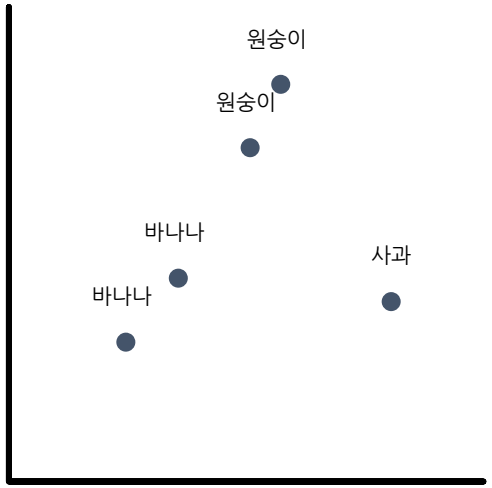
단어 의미를 담지 못함 → ?

Insight campus

밀집 벡터를 만드는 방법

분포 가설이란,

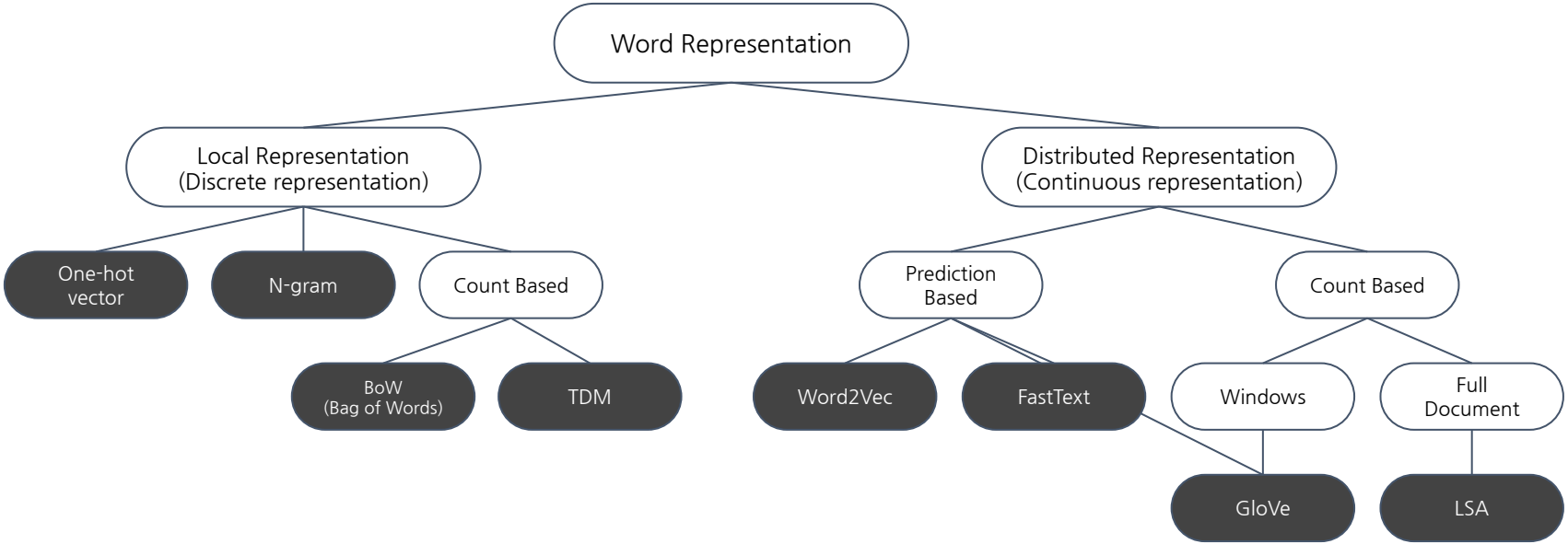
'같은 문맥에서 등장하는 단어는 유사한 의미를 지닌다'



1) 임의의 위치에 벡터 생성

2) 같은 문맥이 등장하는 단어를 더 가까이 표현

Word Representation



- Local representation (Discrete representation) : 해당 단어 그 자체만 보고 값을 매핑하여 표현
- Distributed representation (Continuous representation) : 단어를 표현하기 위해 주변을 참고

4

TF-IDF (단어빈도-역문서빈도)

Term Frequency-Inverse Document Frequency

Insight campus

TF-IDF (Term Frequency-Inverse Document Frequency)

- 단어 빈도 - 역문서 빈도
- TDM 내 각 단어의 중요성을 가중치로 표현
- TDM을 사용하는 것보다 더 정확하게 문서비교가 가능

$$\text{tfidf}(t, d, D) = \text{tf}(t, d) \cdot \text{idf}(t, D)$$

tf(d,t)	특정 문서 d에서의 특정 단어 t의 등장 횟수
df(t)	특정 단어 t가 등장한 문서의 수
idf(d, t)	df(t)의 역수

Insight campus

TF-IDF

$$\text{tfidf}(t, d, D) = \text{tf}(t, d) \cdot \text{idf}(t, D)$$

tf(d,t)	특정 문서 d에서의 특정 단어 t의 등장 횟수
df(t)	특정 단어 t가 등장한 문서의 수
idf(d, t)	df(t)의 역수

TF	IDF	TF-IDF	설명
높	높	높	특정 문서에 많이 등장하고 타 문서에 많이 등장하지 않는 단어 (중요 키워드)
높	낮	-	특정 문서에도 많이 등장하고 타 문서에도 많이 등장하는 단어
낮	높	-	특정 문서에는 많이 등장하지 않고 타 문서에만 많이 등장하는 단어
낮	낮	낮	특정 문서에 많이 등장하지 않고 타 문서에만 많이 등장하는 단어

Insight campus

TF-IDF 가중치 계산

Variants of term frequency (tf) weight

weighting scheme	tf weight
binary	$0, 1$
raw count	$f_{t,d}$
term frequency	$f_{t,d} \Big/ \sum_{t' \in d} f_{t',d}$
log normalization	$\log(1 + f_{t,d})$
double normalization 0.5	$0.5 + 0.5 \cdot \frac{f_{t,d}}{\max_{\{t' \in d\}} f_{t',d}}$
double normalization K	$K + (1 - K)\frac{f_{t,d}}{\max_{\{t' \in d\}} f_{t',d}}$

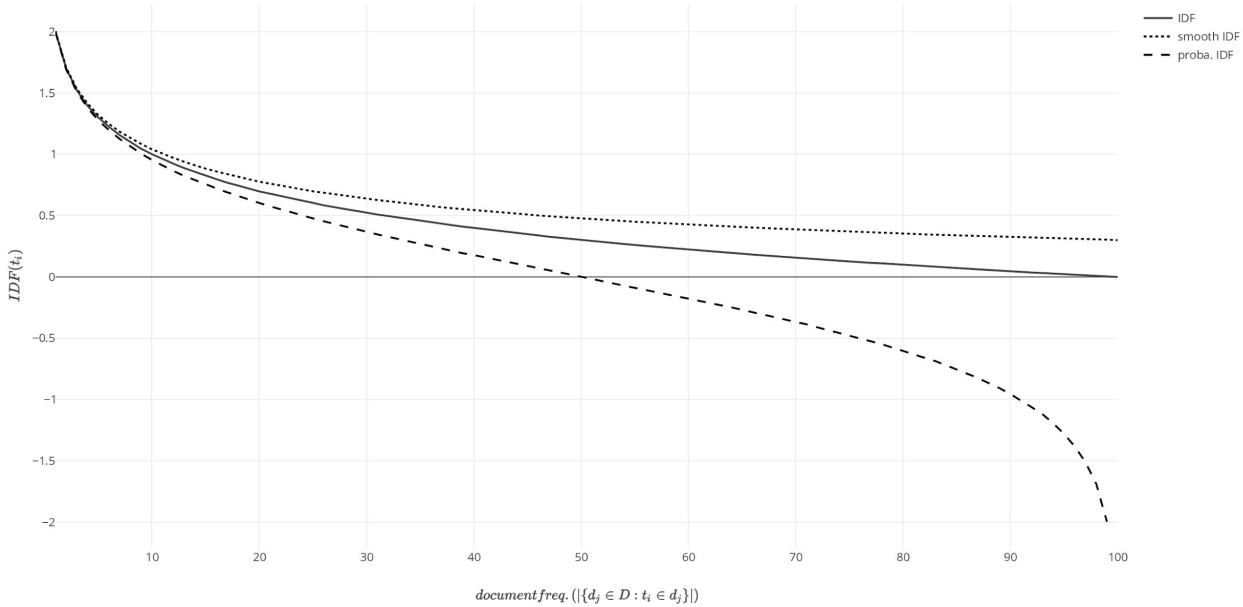
Variants of inverse document frequency (idf) weight

weighting scheme	idf weight ($n_t = \left\|\{d \in D : t \in d\}\right\|$)
unary	1
inverse document frequency	$\log \frac{N}{n_t} = -\log \frac{n_t}{N}$
inverse document frequency smooth	$\log\left(\frac{N}{1 + n_t}\right)$
inverse document frequency max	$\log\left(\frac{\max_{\{t' \in d\}} n_{t'}}{1 + n_t}\right)$
probabilistic inverse document frequency	$\log \frac{N - n_t}{n_t}$

출처 : https://en.wikipedia.org/wiki/Tf%E2%80%93idf

Insight campus

IDF에 로그를 사용하는 이유

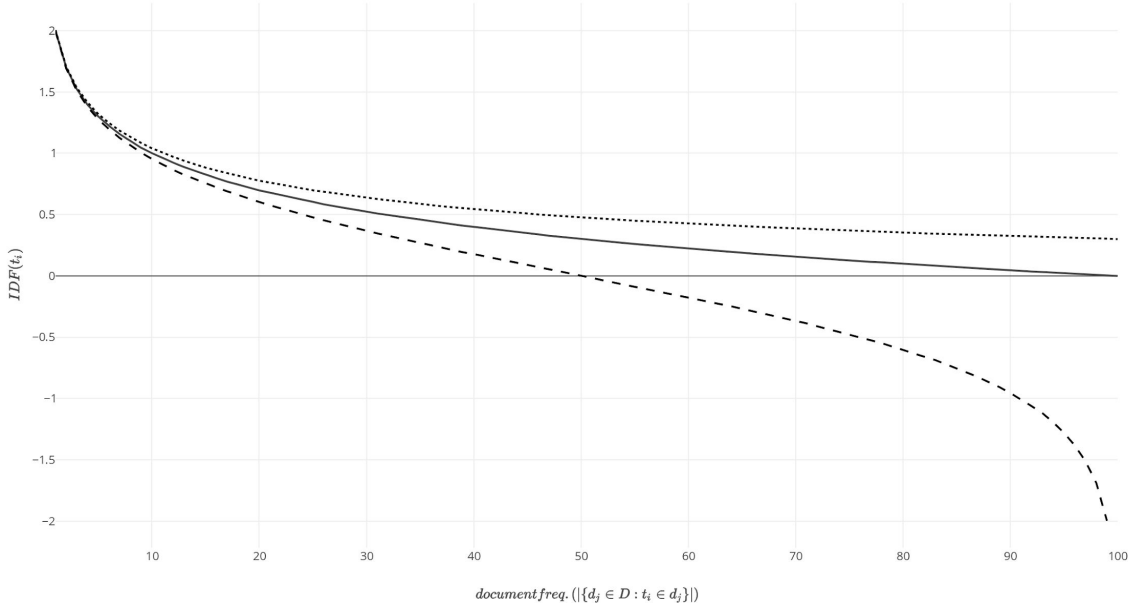


총문서수	10,000	
	단어A	단어B
Freq	8	9
DF	0.0008	0.0009
IDF	7.1309	7.0131
변화율	1.65%	
	단어C	단어D
Freq	8000	8001
DF	0.8	0.8001
IDF	0.2231	0.2230
변화율	0.06%	

출처 : https://en.wikipedia.org/wiki/Tf%E2%80%93idf

Insight campus

IDF에 로그를 사용하는 이유



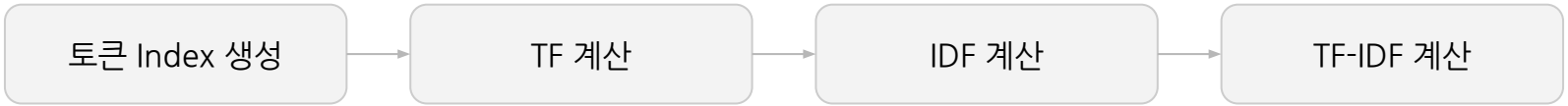
$$\text{idf} = \log\left(\frac{N}{n_t}\right) = -\log\left(\frac{n_t}{N}\right)$$

$$\text{smooth idf} = \log\left(\frac{N}{1+n_t}\right) + 1$$

$$\text{probabilistic idf} = \log\left(\frac{N-n_t}{n_t}\right)$$

출처 : https://en.wikipedia.org/wiki/Tf%E2%80%93idf

TF-IDF 계산절차



각 토큰(단어)을 그대로 사용할 수 없기 때문에 토큰에 Index를 부여

각 토큰(단어)의 등장 빈도를 계산

각 토큰(단어)의 문서 등장 빈도를 계산하여 역수를 취함

계산한 TF와 IDF를 곱하여 TF-IDF를 계산

예제 1 : 토큰 Index 생성

문서1 : d1 = "The cat sat on my face I hate a cat"
문서2 : d2 = "The dog sat on my bed I love a dog"

	Index
The	0
cat	1
sat	2
on	3
my	4
face	5
I	6
hate	7
a	8
dog	9
bed	10
lov	11

Insight campus

예제 : TF 계산

문서1 : d1 = "The cat sat on my face I hate a cat"
문서2 : d2 = "The dog sat on my bed I love a dog"

$$f_{t,d} \Big/ \sum_{t' \in d} f_{t',d}$$

$f_{t,d}$ = 문서내 토큰 빈도
SUM($f_{t,d}$)= 문서내 전체 토큰빈도

문서1

	문서내 토큰 빈도	문서내 전체 토큰빈도	TF
The	1	10	0.1
cat	2	10	0.2
sat	1	10	0.1
on	1	10	0.1
my	1	10	0.1
face	1	10	0.1
I	1	10	0.1
hate	1	10	0.1
a	1	10	0.1
dog	0	10	0
bed	0	10	0
lov	0	10	0

문서2

	문서내 토큰 빈도	문서내 전체 토큰빈도	TF
The	1	10	0.1
cat	0	10	0
sat	1	10	0.1
on	1	10	0.1
my	1	10	0.1
face	0	10	0
I	1	10	0.1
hate	0	10	0
a	1	10	0.1
dog	2	10	0.2
bed	1	10	0.1
love	1	10	0.1

Insight campus

예제 : IDF 계산

문서1 : d1 = "The cat sat on my face I hate a cat"
문서2 : d2 = "The dog sat on my bed I love a dog"

$$\log \frac{N}{n_t} = -\log \frac{n_t}{N}$$

N = 문서수

n_t = 토큰이 등장한 문서수

	문서수	토큰이 등장한 문서수	IDF
The	2	2	0
cat	2	1	0.301
sat	2	2	0
on	2	2	0
my	2	2	0
face	2	1	0.301
I	2	2	0
hate	2	1	0.301
a	2	2	0
dog	2	1	0.301
bed	2	1	0.301
love	2	1	0.301

Insight campus

예제 : TF-IDF 계산

문서1 : d1 = "The cat sat on my face I hate a cat"
문서2 : d2 = "The dog sat on my bed I love a dog"

문서1

	TF	IDF	TF-IDF
The	0.1	0	0
cat	0.2	0.301	0.060
sat	0.1	0	0
on	0.1	0	0
my	0.1	0	0
face	0.1	0.301	0.301
I	0.1	0	0
hate	0.1	0.301	0.301
a	0.1	0	0
dog	0	0.301	0.301
bed	0	0.301	0.301
lov	0	0.301	0.301

문서2

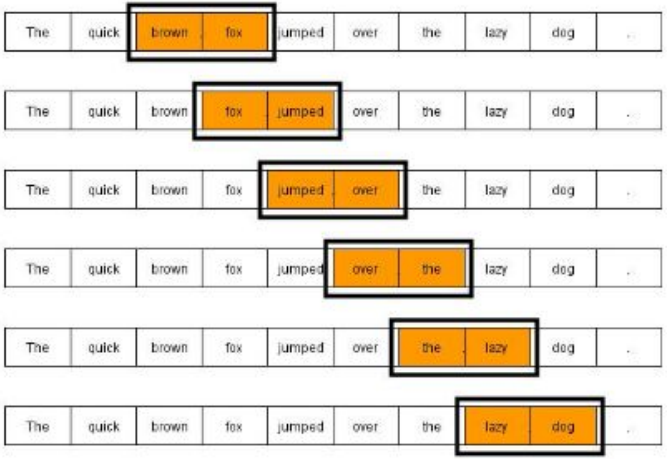
	TF	IDF	TF-IDF
The	0.1	0	0
cat	0	0.301	0
sat	0.1	0	0
on	0.1	0	0
my	0.1	0	0
face	0	0.301	0.301
I	0.1	0	0
hate	0	0.301	0.301
a	0.1	0	0
dog	0.2	0.301	0.601
bed	0.1	0.301	0.301
love	0.1	0.301	0.301

5 n-Gram

Insight campus

n-Gram 이란?

- 복수개(n개) 단어를 보는냐에 따라 unigram, bigram, trigram 등 으로 구분
- 제한적으로 문맥을 표현할 수 있음



n-Gram 이란?

an adorable little boy is spreading smile

- unigrams : an, adorable, little, boy, is, spreading, smiles
- bigrams : an adorable, adorable little, little boy, boy is, is spreading, spreading smiles
- trigrams : an adorable little, adorable little boy, little boy is, boy is spreading, is spreading smiles
- 4-grams : an adorable little boy, adorable little boy is, little boy is spreading, boy is spreading smiles

n-Gram 한계

- n의 크기는 trade-off 문제
  - 1보다는 2를 선택하는 것이 대부분 언어 모델 성능을 높일 수 있음
  - n을 너무 크게 선택하면 n-gram 이 unique 할 확률이 높아 등장수가 낮을 확률이 높음. (OOV, Out of Vocabulary 문제가 발생할 수 있음)
  - n을 너무 작게 하면 카운트는 잘되지만 정확도가 떨어질 수 있음. n은 최대 5를 넘지 않도록 권장

-	Unigram	Bigram	Trigram
Perplexity	962	170	109

스탠포드에 3,800만개 단어 토큰을 n-Gram으로 학습한 결과

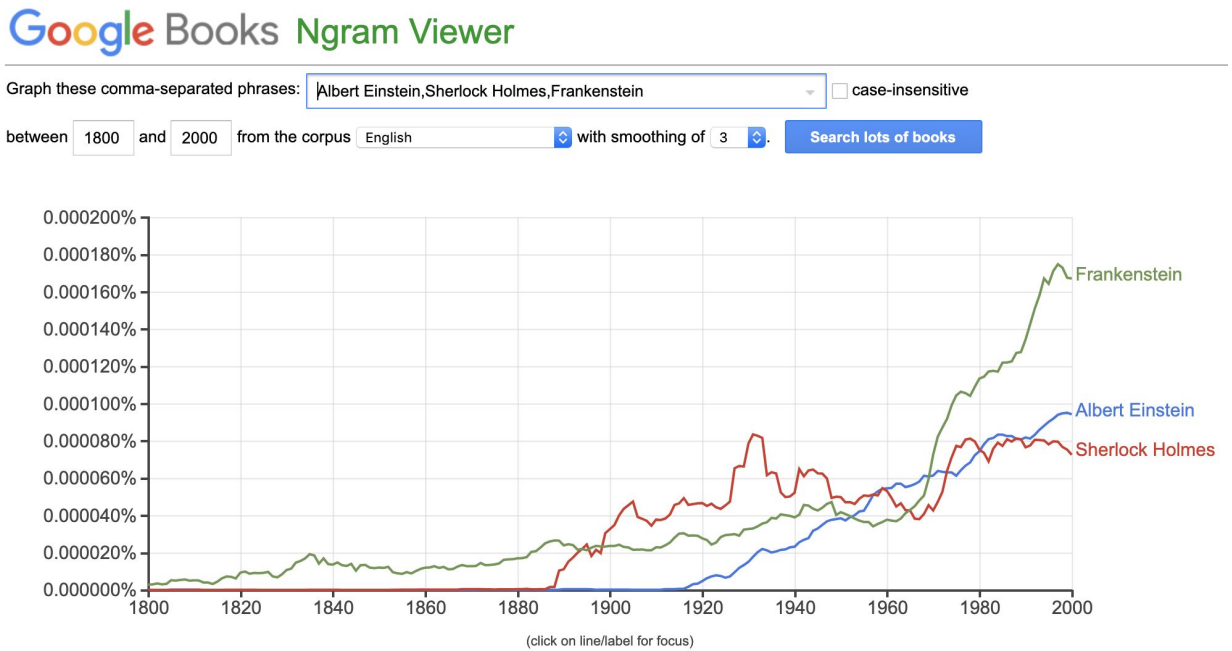
- n-Gram 카운트가 0인 경우
  - n-Gram 이 모든 단어를 커버 할 수 없기 때문에 Out of Vocabulary 문제가 발생할수 있음

적용 분야(Domain)에 맞는 코퍼스의 수집

- 분야(Domain)에 따라 단어들의 확률 분포는 다름
(금융 분야는 금융 관련 용어가 많이 등장하고, 마케팅은 관련 용어가 많이 등장할 것임)

- 분야에 적합한 코퍼스를 사용하면 언어 모델의 성능이 높아질 수 있음
(훈련에 사용되는 코퍼스에 따라 언어 모델의 성능이 달라짐 이는 언어 모델의 약점으로 분류되기도함)

Google Books Ngram Viewer



https://books.google.com

1

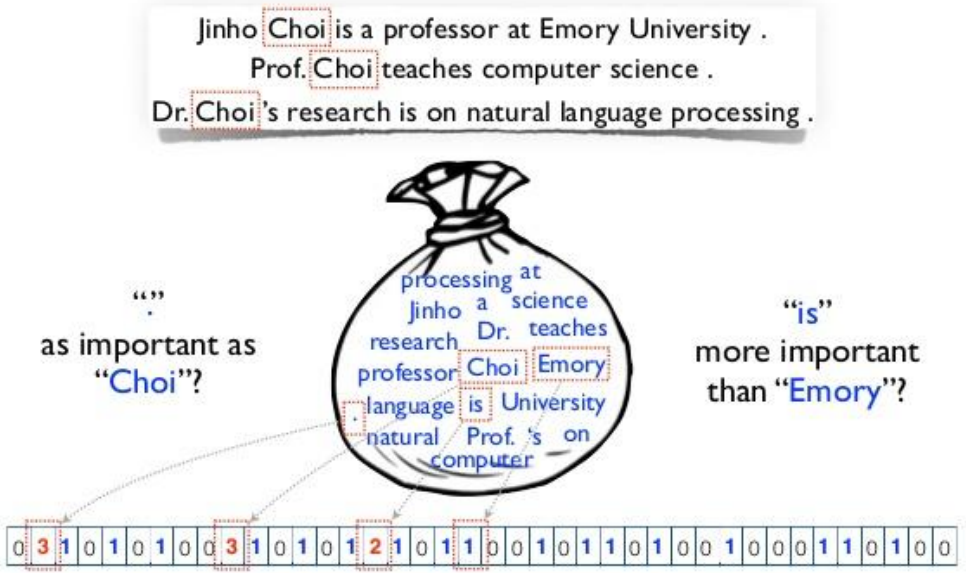
BoW (단어 주머니)

Bag of Words

Insight campus

BoW (Bag of Words)

BoW(Bag of Words) : 문서 내 단어 출현 순서는 무시. 빈도수만 기반으로 문서를 표현하는 방법



BoW 생성 방법

문서1: 오늘 동물원에서 코끼리를 봤어
문서2:오늘 동물원에서 원숭이에게 사과를 줬어

Step1. 각 토큰에 고유 인덱스 부여

오늘	0
동물원에서	1
코끼리를	2
봤어	3
원숭이에게	4
사과를	5
줬어	6

Step2. 각 인덱스 위치에 토큰 등장 횟수를 기록

	오늘	동물원에서	코끼리를	봤어	원숭이에게	사과를	줬어
문서1	1	1	1	1	0	0	0

	오늘	동물원에서	코끼리를	봤어	원숭이에게	사과를	줬어
문서2	1	1	0	0	1	1	1

Insight campus

한계

- 단어의 **순서를 고려 하지 않음**
- **BoW 는 Spare** 함. 벡터 공간의 낭비, 연산 비효율성 초래
- **단어 빈도수가 중요도를 바로 의미 하지 않음.** 단어가 자주 등장한다고 중요한 단어는 아님.
- 전처리가 매우 중요함. 같은 의미의 다른 단어 표현이 있을 경우 다른것으로 인식될 수 있음.
  (뉴스와 같이 정제된 어휘를 사용하는 매체는 좋으나, 소셜에서는 활용하기 어려움)

2

TDM (단어-문서 행렬)

Term-Document Matrix

TDM (Term-Document Matrix)

BoW(Bag of Words) 중 하나
문서에 등장하는 각 단어 빈도를 행렬로 표현한 것

문서1: 동물원 코끼리
문서2: 동물원 원숭이 바나나
문서3: 엄마 코끼리 아기 코끼리
문서4: 원숭이 바나나 코끼리 바나나

	동물원	코끼리	원숭이	바나나	엄마	아기
문서1	1	1	0	0	0	0
문서2	1	0	1	1	0	0
문서3	0	2	0	0	1	1
문서4	0	1	1	2	0	0

Insight campus

TDM vs DTM

	Tweet 1	Tweet 2	Tweet 3	…	Tweet N
Term 1	0	0	0	0	0
Term 2	1	1	0	0	0
Term 3	1	0	0	0	0
…	0	0	3	1	1
Term M	0	0	0	1	0

Term Document Matrix (TDM)

	Term 1	Term 2	Term 3	…	Term M
Tweet 1	0	1	1	0	0
Tweet 2	0	1	0	0	0
Tweet 3	0	0	0	3	0
…	0	0	0	1	1
Tweet N	0	0	0	1	0

Document Term Matrix (DTM)

TDM (Term-Document Matrix) 의 한계

- 단어의 **순서를 고려 하지 않음**
- **TDM 는 Spare** 함. 벡터 공간의 낭비, 연산 비효율성 초래
- **단어 빈도수가 중요도를 바로 의미 하지 않음.** the와 같은 단어는 빈번하게 등장하고 TDM에서 중요한 단어로 판단 될 수 있음
  => 이를 보완 하기 위하여 TF - IDF 를 사용

감사합니다.

Insight campus

daniel@fins.ai