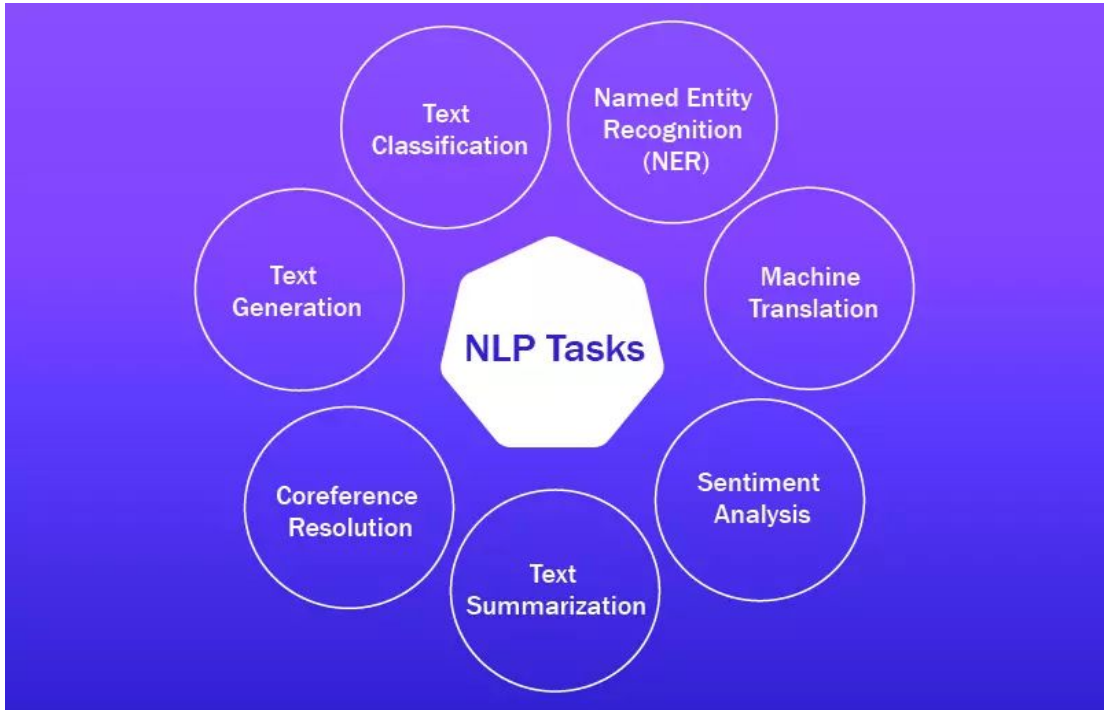
NLP paper study week-01
NLP - Tutorial 

Cheonghae Kim

NLP (Natural Language Processing)

인간의 언어(자연어)"를 컴퓨터가 이해하고 처리하게 만드는 기술



Tokenization (토큰화)

텍스트를 “의미 있는 단위(토큰)”로 나누는 작업

ex) Time is an illusion. Lunchtime double so! -> "Time", "is", "an", "illustion", "Lunchtime", "double", "so"

한국어는 띄워쓰기가 아닌 형태소 단위로 토큰화를 진행한다.

형태소? 뜻을 가진 가장 작은 말의 단위.

ex) 에디가 책을 읽었다 -> ['에디가', '책을', '읽었다']

Tokenization (토큰화)

- 띄어쓰기 기준: whitespace tokenization
- 단어 단위: Word tokenization (NLTK, spaCy 등)
- 형태소 단위: Mecab, Okt, Khaiii (한국어)
- 문자 단위: Character tokenization
- Subword 단위: BPE, WordPiece (BERT)

Vectorization (벡터화)

토큰을 수치 벡터로 변환하는 작업

벡터화 방법 분류 (통계 기반, 신경망 기반)

1. 통계 기반

+ Bag-of-words (BoW)

+ TF-IDF

+ LSA (Latent Semantic Analysis)

+ LDA (Latent Dirichlet Allocation)

2. 신경망 기반

+Word2Vec(CBOW,Skip-gram)

+ Glove

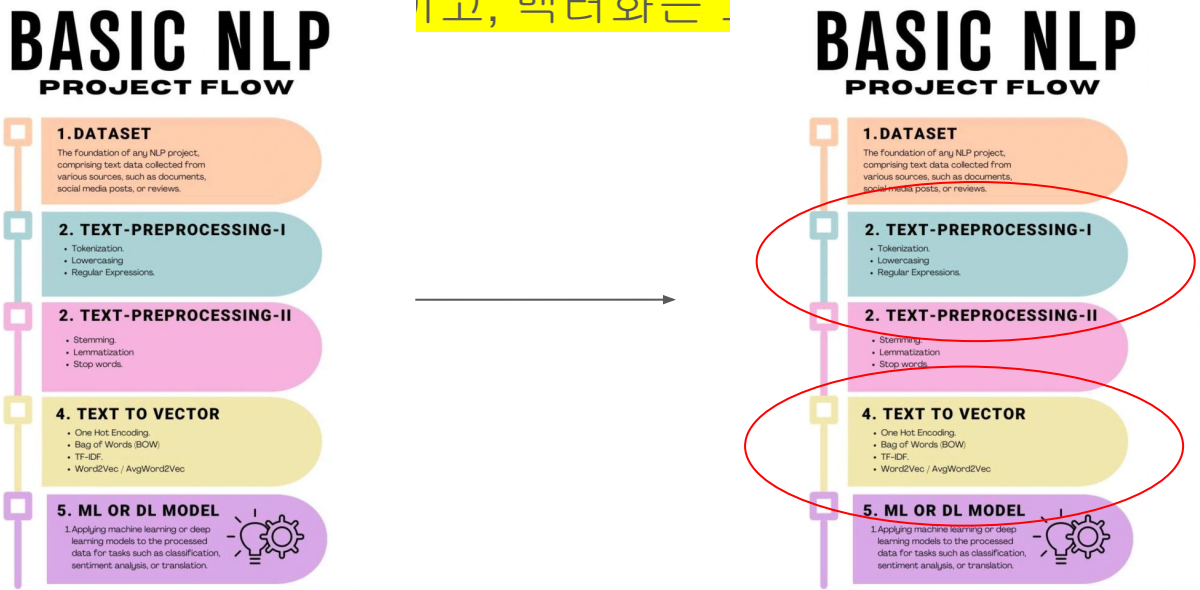
+ FastText

+ Transformer 기반: BERT,GPT

Tokenization -> Vectorization

+ 토큰화를 한 뒤 벡터화를 진행하여 모델이 학습할 수 있는 형태의 데이터로 정제한다.

토큰화는 텍스트를 자르는 작업이고, 벡터화는 [illegible] 작업이다.



출처:
https://www.linkedin.com/in/md-tahseen-equbal/

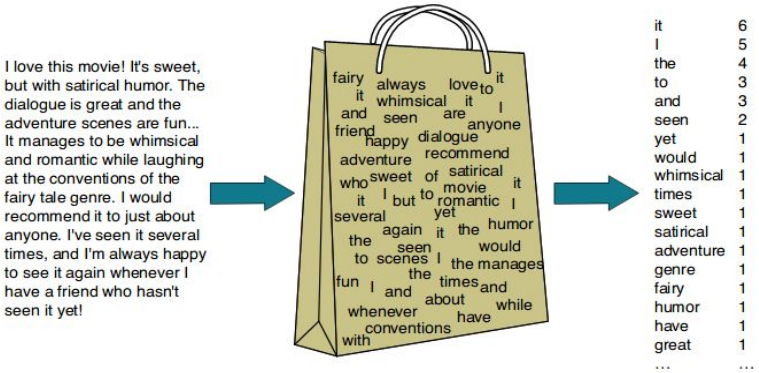
Bag of Words (BoW)

NLP에서 가장 기본적인 텍스트 벡터화 방법 중 하나

문서 안의 단어들이 어떤 단어들이 얼마나 나왔는지만 세고,

단어의 순서는 무시하는 방식의 벡터화 방법입니다.

→ 수많은 단어들을 전부 가방에 집어넣고 마구 흔들어서 Bag of Words



ex) **doc1 = 'John likes to watch movies. Mary likes movies too.'**

BoW1 = {"John":1, "likes":2, "to":1, "watch":1, "movies":2, "Mary":1, "too":1}

Document-Term Matrix (DTM)

문서(Document)들과 단어(term)들 간의 관계를 행렬(Matrix) 형태로 표현한 것

문서1: "나는 밥을 먹었다"

문서2: "너는 밥을 안 먹었다"

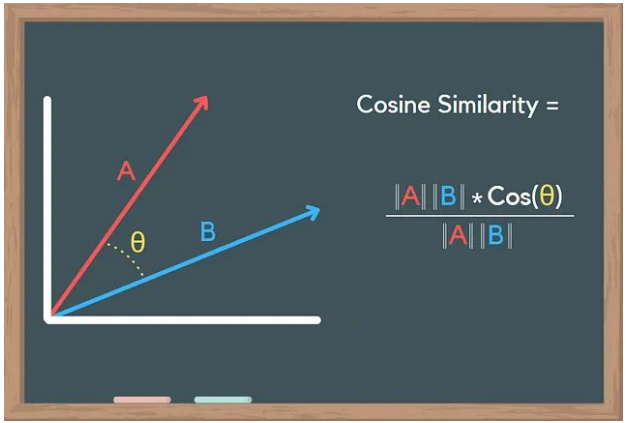
문서3: "나는 운동을 했다"

문서\단어	나는	너는	밥을	먹었다	안	운동을	했다
문서1	1	0	1	1	0	0	0
문서2	0	1	1	1	1	0	0
문서3	1	0	0	0	0	1	1

코사인 유사도 (Cosine Similarity)

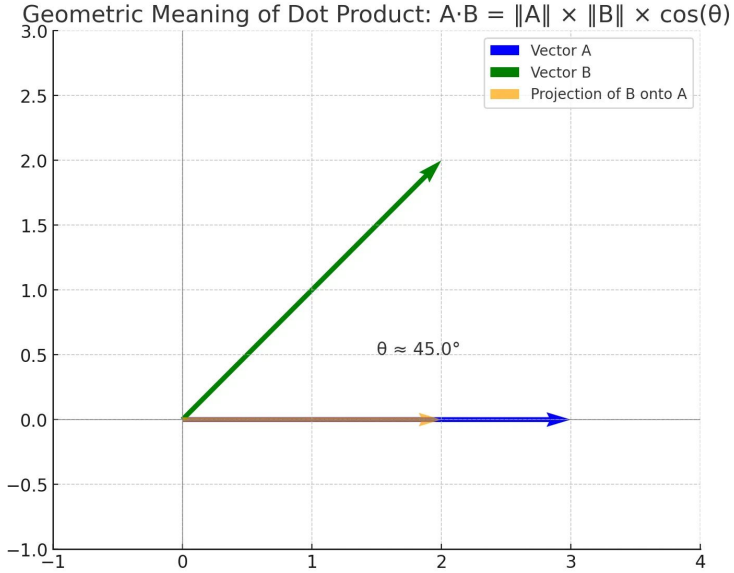
두 벡터 간의 코사인 각도를 이용하여

구할 수 있는 두 벡터의 유사도



$$\|A\| \times \|B\| \times \cos(\theta)$$

벡터 내적의 기하학적 해석



- $\mathbf{A} = [1, 0, 1, 1, 0, 0, 0]$
- $\mathbf{B} = [0, 1, 1, 1, 1, 0, 0]$

$$\mathbf{A} \cdot \mathbf{B} = (1 \times 0) + (0 \times 1) + (1 \times 1) + (1 \times 1) + (0 \times 1) = 2$$

$$\|\mathbf{A}\| = \sqrt{1^2 + 0^2 + 1^2 + 1^2} = \sqrt{3} \quad \|\mathbf{B}\| = \sqrt{1^2 + 1^2 + 1^2 + 1^2} = \sqrt{4} = 2$$

$$\text{유사도} = \frac{2}{\sqrt{3} \cdot 2} = \frac{1}{\sqrt{3}} \approx 0.577$$

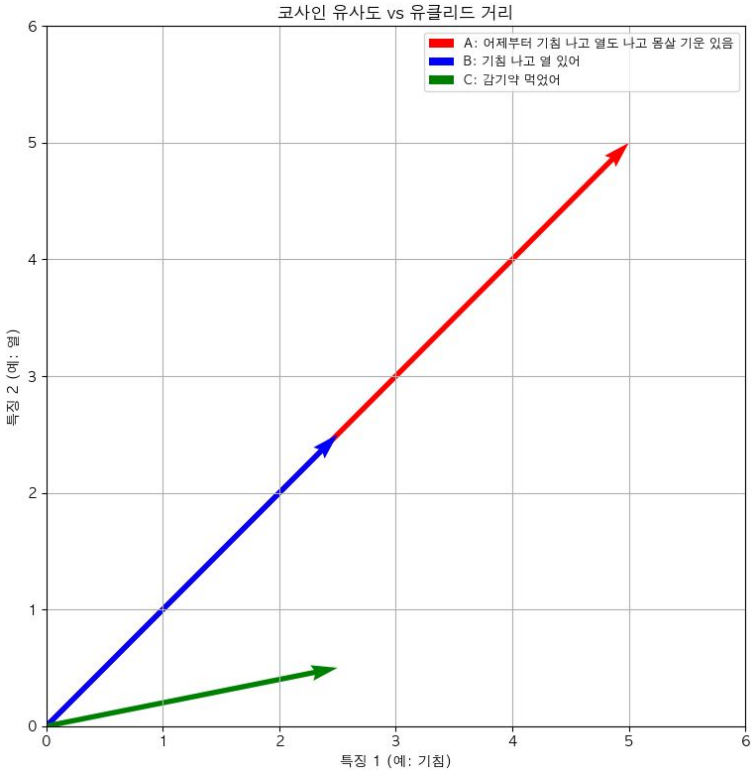
문서\단어	나는	너는	밥을	먹었다	안	운동을	했다
문서1	1	0	1	1	0	0	0
문서2	0	1	1	1	1	0	0
문서3	1	0	0	0	0	1	1

왜 유클리드 거리가 아닌 코사인 유사도를 사용할까?

A: 어제부터 기침 나고 열도 나고 몸살 기운 있음

B: 기침 나고 열 있어

C: 감기약 먹었어



자연어처리에서는 문서의 길이나 벡터의 크기가 의미 유사성 측정에 방해가 되므로, 벡터를 정규화하여 방향만을 고려하는 코사인 유사도를 사용한다.

코사인 곡선에 따라 각도가 낮을수록 값이 커진다.

따라서 0도이면 1, 90도이면 0

코사인 함수에 특성에 따라 각도가 작을수록 코사인 값이 커진다.

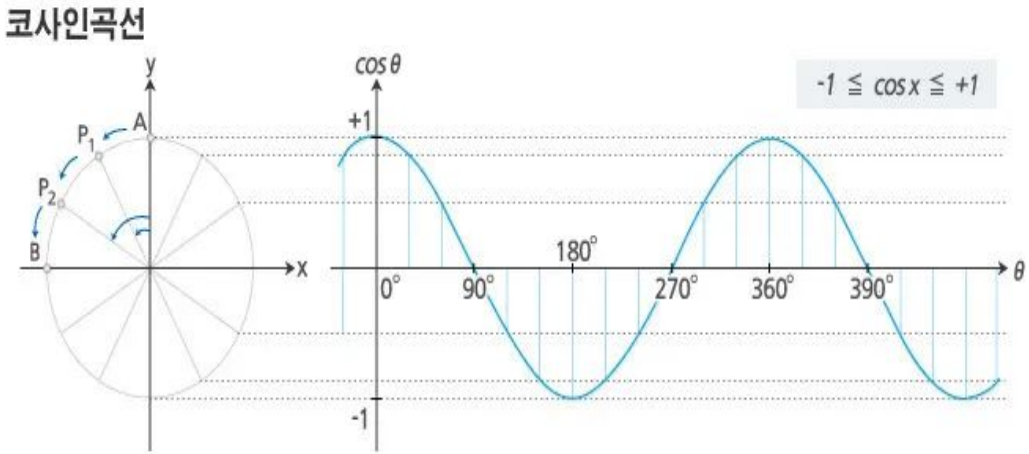
- **0도**: cos(0°) = 1 (최대값)
- **90도**: cos(90°) = 0 (직교 상태)
- **180도**: cos(180°) = -1 (완전히 반대 방향

자연어처리에서는 각 단어나 특징이

벡터 공간의 축(axis)이 되며, 문서 벡터들

대부분 양의 값을 가지므로 벡터 간 각도

0°~90° 범위에 분포한다.



DTM의 한계

- 의미(Semantic)를 반영하지 않음
- 희소 행렬(Sparse Matrix)
- 단어의 순서 정보 손실

문서\단어	나는	너는	밥을	먹었다	안	운동을	했다
문서1	1	0	1	1	0	0	0
문서2	0	1	1	1	1	0	0
문서3	1	0	0	0	0	1	1

TF-IDF (Term Frequency-inverse Document Frequency)

TF : Term Frequency - 한 문서내에서 단어가 얼마나 자주 등장하는가

IDF : Inverse Document Frequency - 그 단어가 전체 문성 중 몇 개 문서에 등장하는가

N: 전체 문서 수

df(t): 단어 t가 등장한 문서 수

많이 등장하는 단어일수록 중요도가 낮아지도록 조정

+1은 분모가 0이 되는 걸 방지하는 smoothing

$$IDF(t) = \log\left(\frac{N}{1+df(t)}\right)$$

TF-IDF - log

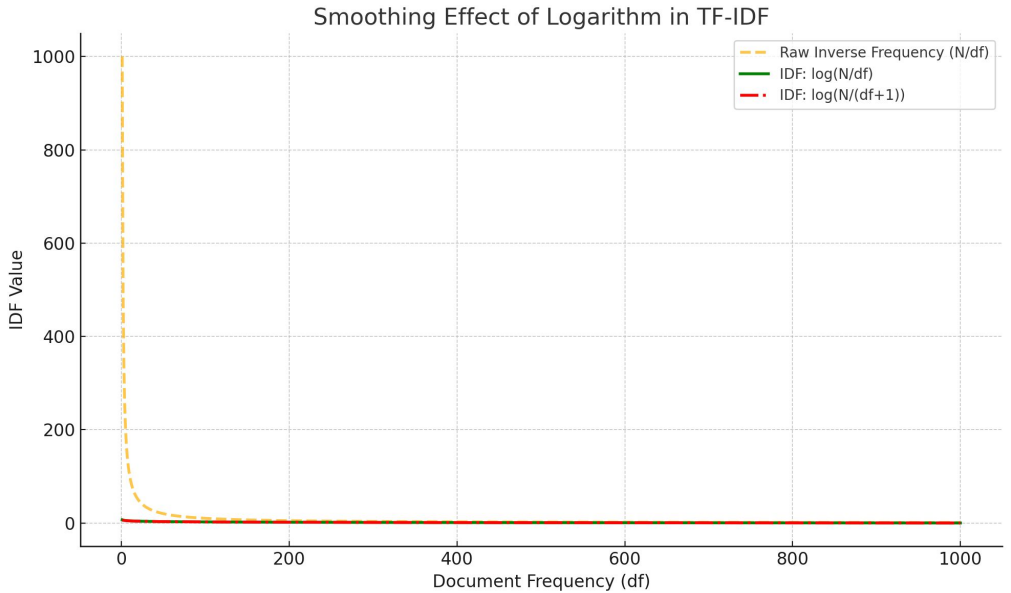
TF-IDF 식의 핵심은 log

log를 사용함으로써,

문서 빈도가 높은 단어에는 완만하게 패널티를

문서 전체에서 드물게 등장하는 단어에는

더 큰 가중치를 부여



문서 1: "나는 이번 여름에 제주도를 여행했다. 바다도 예쁘고, 한라산도 멋졌다."

문서 2: "제주도는 정말 아름답다. 제주도 여행은 언제나 최고다."

문서 3: "파리는 유럽에서 가장 매력적인 도시 중 하나다. 파리 여행은 환상적이다."

문서 4: "서울은 밤에 더 아름답다. 서울 야경을 꼭 봐야 한다."

KEY: 제주도 | DF = 2 ⟶ $IDF(\text{"제주도"}) = \ln\left(\frac{N}{DF}\right) = \ln\left(\frac{4}{2}\right) = \ln(2) \approx 0.6931$

문서1 TF : 1 ⟶ $TF\text{-}IDF = 1 \times 0.6931 = 0.6931$

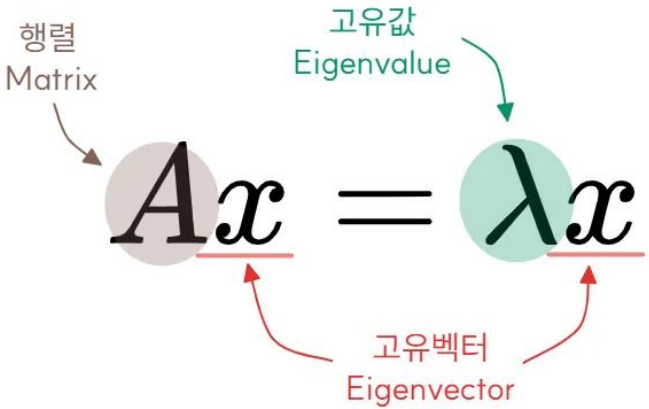
문서2 TF : 2 ⟶ $TF\text{-}IDF = 2 \times 0.6931 = 1.3862$

LSA (Latent Semantic Analysis) - 잠재 의미 분석

텍스트 데이터에서 단어와 문서 간의 의미적 관계를 파악하기 위해 사용하는 차원 축소 기법

대표적인 기법으로 SVD가 있다.

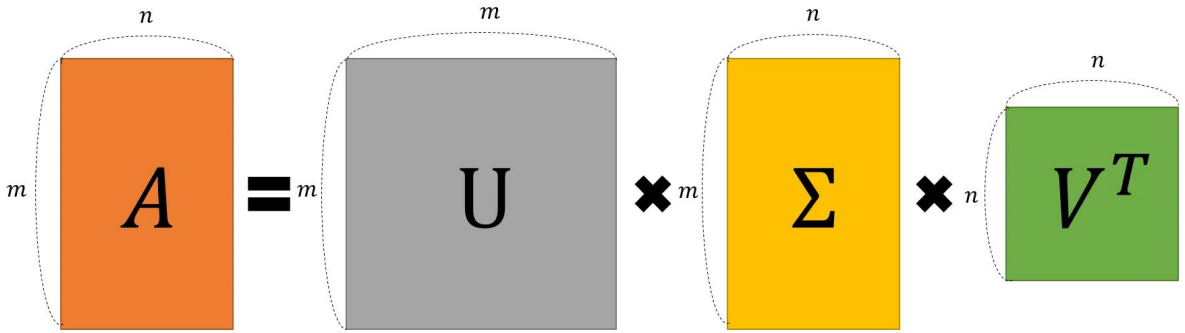
고유벡터와 고유값 (Eigenvector & Eigenvalue) : 고유벡터는 선형 변환이 적용될 때 방향이 변하지 않는 벡터(0이 아니다. 그 외 스칼라 배수만큼 변한다.



SVD (Singular Value Decomposition) - 특이값 분해

SVD의 목적

- 행렬의 크기를 감소시킨다.
- 정방행렬이 아닌 행렬의 해를 구할 수 있게 해준다 $A = U\Sigma V^T$
- 데이터 측면에서 데이터의 크기를 줄여준다.



reference

위키독스 딥 러닝을 이용한 자연어처리 입문: https://wikidocs.net/book/2155

An introduction to Bag of Words: https://medium.com/@vamshiprakash001/an-introduction-to-bag-of-words-bow-c32a65293ccc

Consine Similarity in Graph Similarity:

https://medium.com/@rohan10dalvi/cosine-similarity-in-graph-similarity-3c260e7efd3f

모두의 연구소 - 고유 벡터:

https://modulabs.co.kr/blog/eigenvalue-and-eigenvector

공돌이의 수학정리노트(SVD):

https://angeloyeo.github.io/2019/08/01/SVD.html

Fynd Academy:

https://www.fynd.academy/blog/nlp-in-data-science