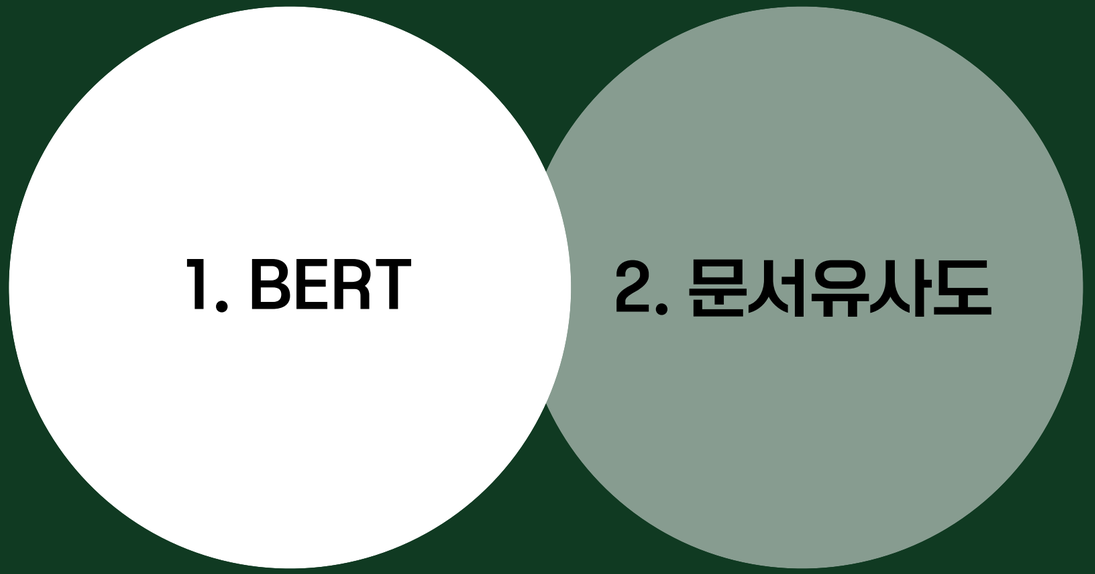
SKKU SCHOLARSHIP

| TECHNICAL DETAIL

TEAM 스꿀라, SKKULAR

강병준 | 김규진 | 박진아 | 장이준 | 주재현

CONTENTS



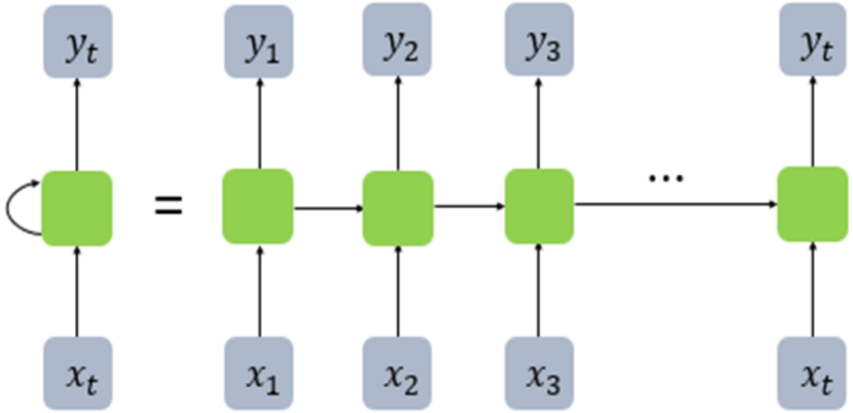
TEAM. 스꿀라
SKKULAR

1. BERT

BERT 모델 사전 배경: RNN & LSTM

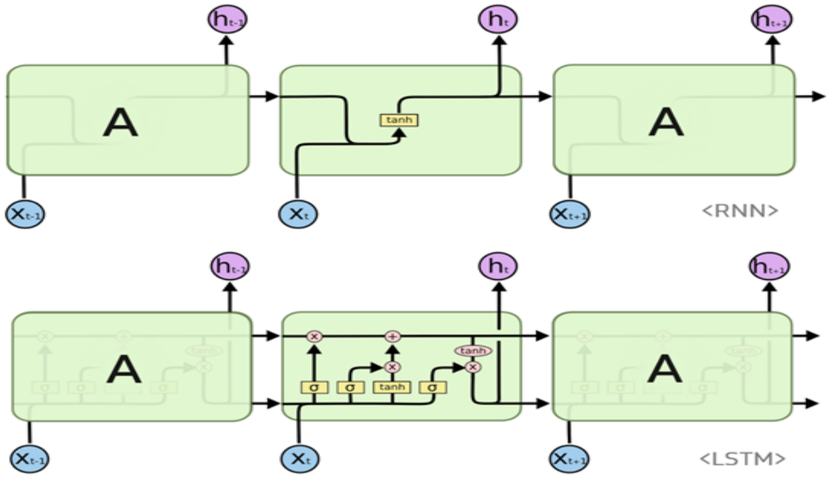
1) RNN(Recurrent Neural Network)

- 입력과 출력을 시퀀스 단위로 처리하는 시퀀스 모델
- Hidden Layer에서 계산된 값을 Output Layer로 보내는 동시에 그 값을 다음 Hidden Layer의 노드로 보냄

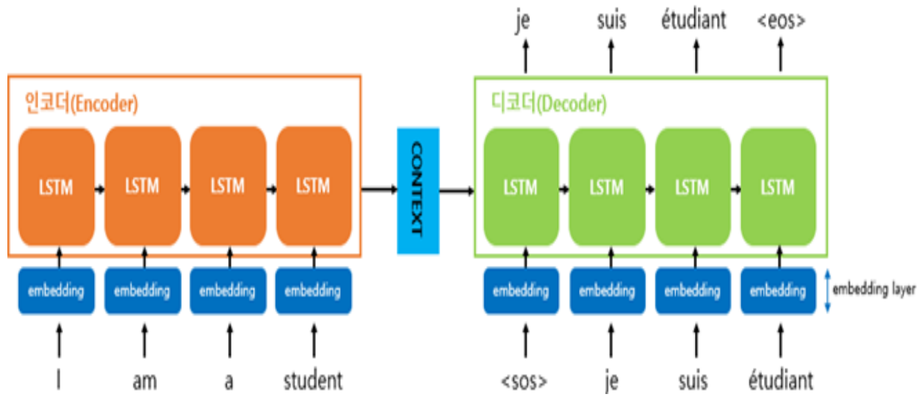


2) LSTM

-RNN의 장기 의존성 문제 해결을 위해 고안된 모델.
-hidden state h이외에도 장기 기억을 담당할 새로운 변수 Cell State c를 추가하고, Cell 안에 Input Gate, Forget Gate, Output Gate라는 새로운 구조를 추가.



BERT 모델 사전 배경: seq2seq



다음 등장한 모델인 seq2seq도 여전히

1. 장기의존성 문제: Input에 먼저 들어간 데이터 일수록 그 정보가 더 많이 손실됨

2. 병렬 처리 불가능: RNN의 모델 구조로 인해 모든 계산이 순서대로, 즉 직렬로 진행됨.

이 문제를 해결해줄 “Transformer” 등장

BERT 모델 사전 배경: Transformer

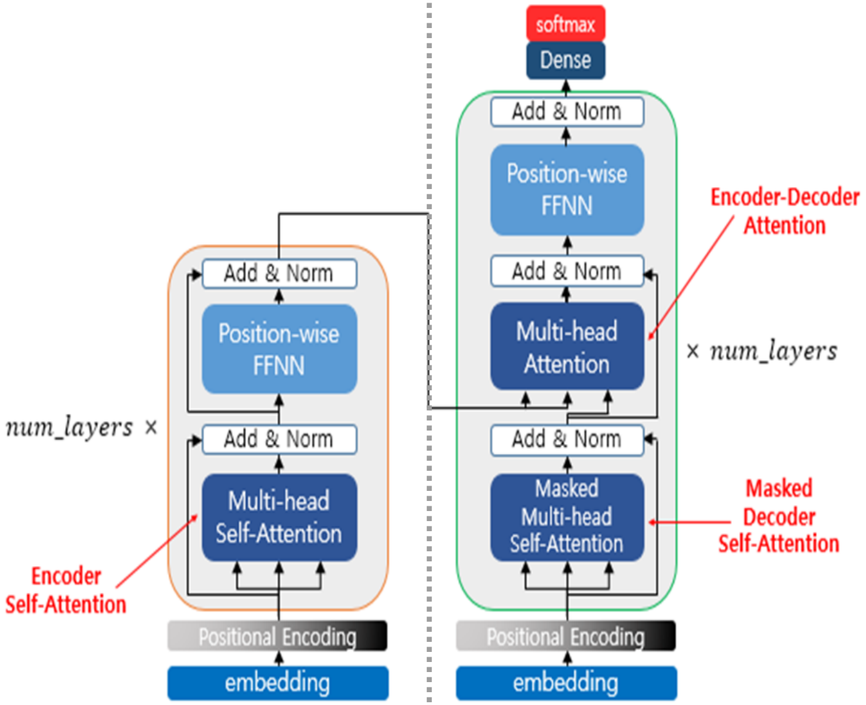
1) Encoder - Decoder 구조

Encoder: 'I am a student'와 같은 문장에 있는 단어들을 하나의 context vector로 압축.
Decoder: Encoder에서 압축한 Context Vector를 기반으로 정답 값을 재현.

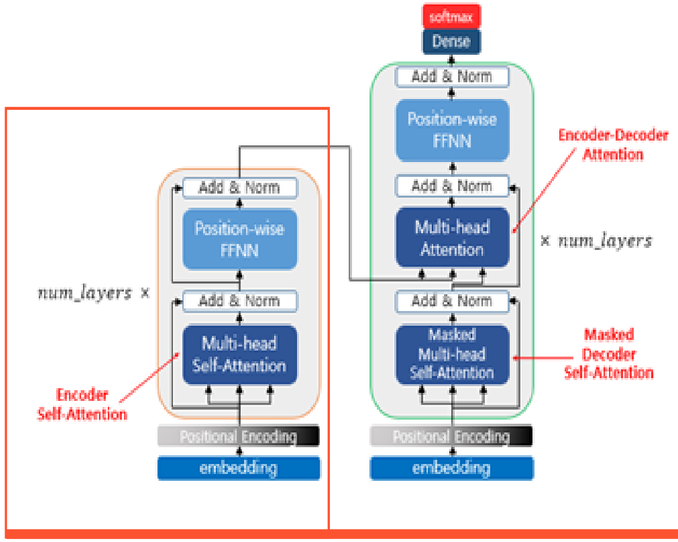
2) RNN과 같이 데이터를 각 데이터에 Sequential하게 입력해주는 것이 아닌, 한번에 데이터를 입력

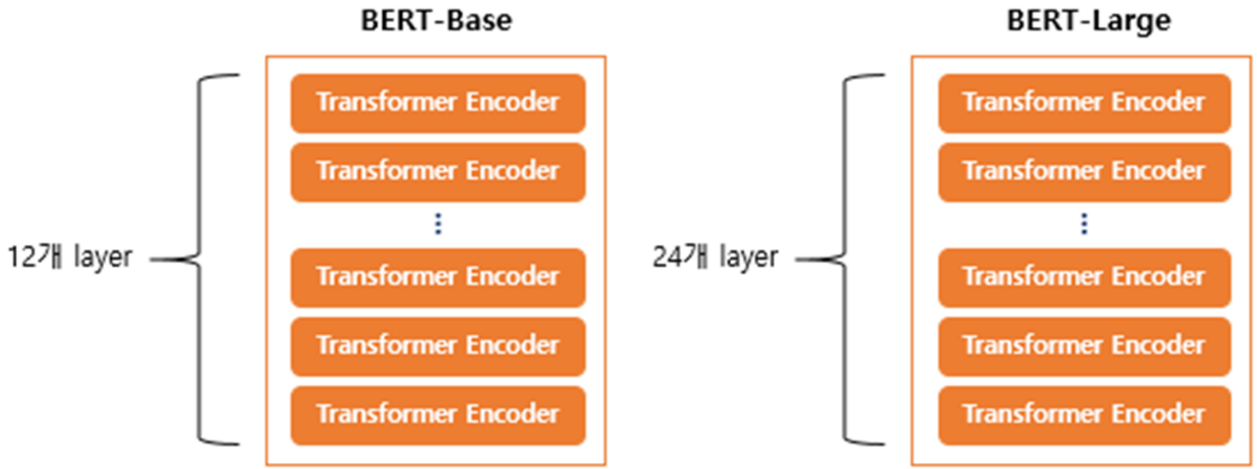
3) Attention method를 사용하여 각 시점마다 서로 다른 부분에 집중할 수 있음.

*논문 "Attention is all you need" 참고.



BERT





BERT는 transfomer의 Encoder만을 사용하는 대표적인 모델 중 하나.
위 구조에서 주로 Encoder 파트를 사용한다고 보면 되고, Base 버전에서는
총 12개를 쌓았으며, Large 버전에서는 총 24개의 인코더를 쌓아올림.

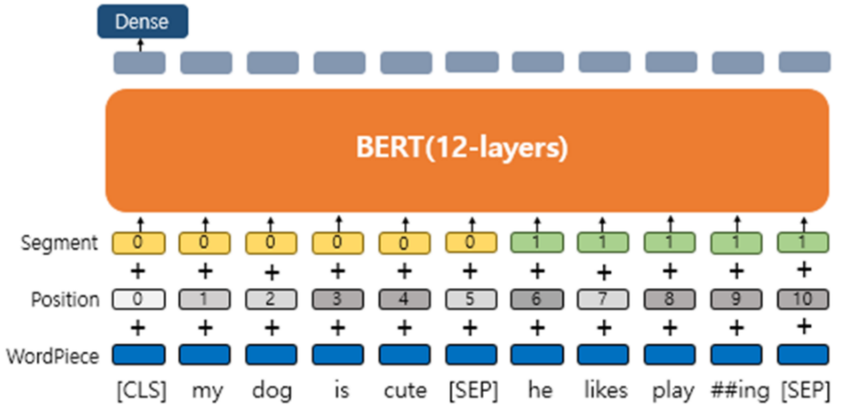
BERT: Embedding 및 Tokenizing

BERT는 총 3가지 embedding을 거쳐 input값을 생성한다.

1) WordPiece Embedding
: 실질적인 입력이 되는 워드 임베딩.

2) Position Embedding
: 위치 정보를 학습하기 위한 임베딩.

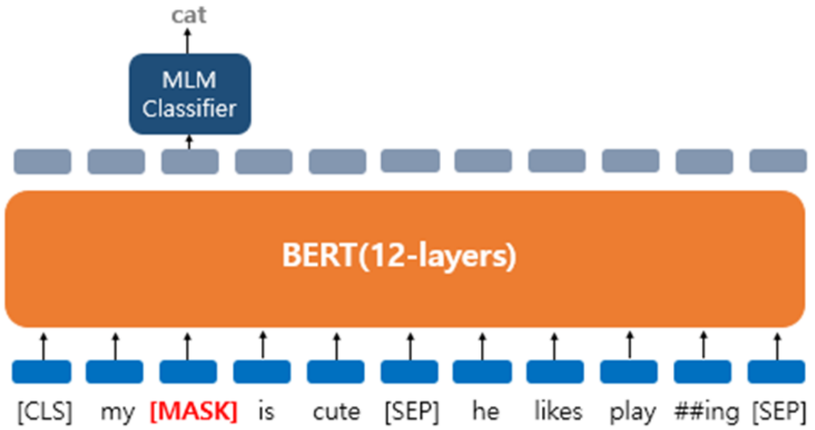
3) Segment Embedding
: 두 개의 문장을 구분하기 위한 임베딩.



기본적으로는 wordpiece tokenizer를 거친 후 embedding을 하지만, 어떤 tokenizer를 사용하냐에 따라 성능이 달라지기도.

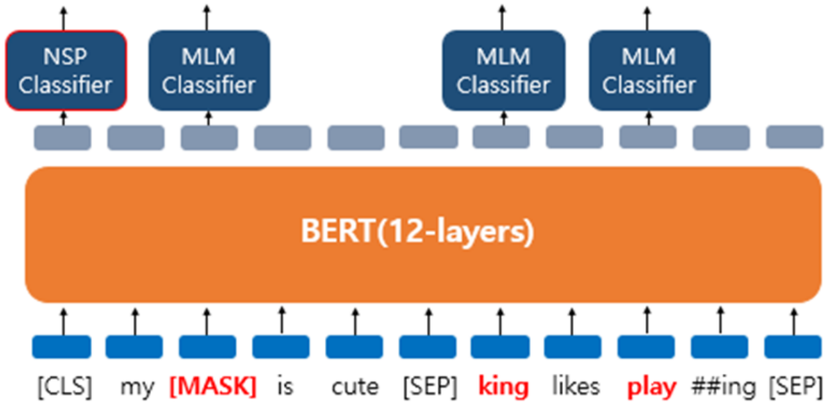
BERT: 사전 학습 방법

2-1) Masked Language Model: 마스크드 언어 모델



입력 텍스트의 단어를 랜덤으로 마스킹(Masking).
그리고 MASK된 단어에 어떤 단어가 들어갈 것인지를 예측하는 방식으로 훈련. * MASK 된 단어가 무엇인지를 단어로부터 양쪽 정보를 모두 이용하기 때문에 “양방향” 학습 모델이라 불림.

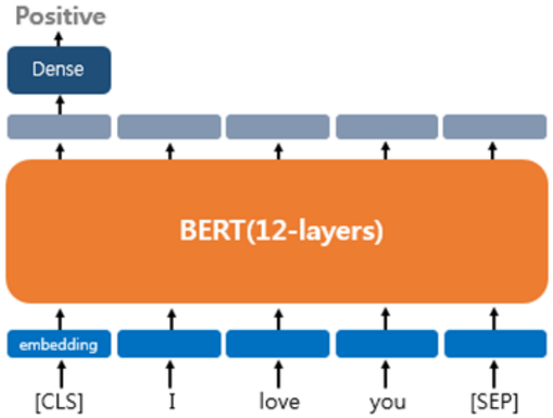
2-2) Next Sentence Prediction: 다음 문장 예측



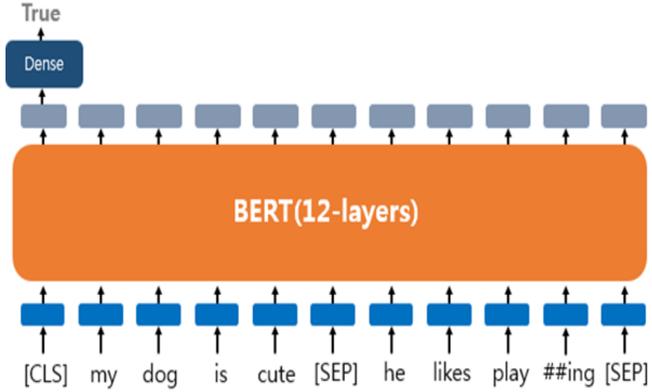
BERT는 두 개의 문장을 준 후에
이 문장이 이어지는 문장인지 아닌지를 맞추는 방식으로 훈련

BERT: Finetuning 및 해결 가능한 task

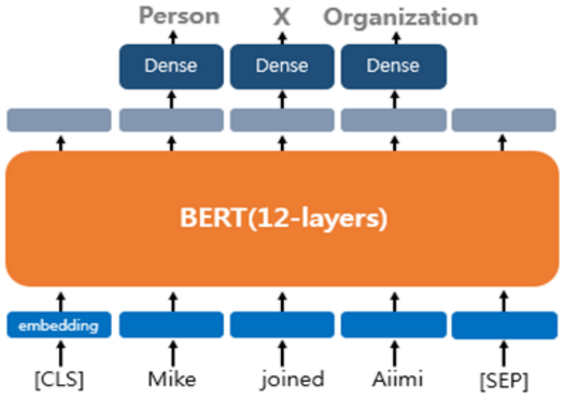
BERT는 fine tuning을 통해 여러가지 task를 수행할 수 있게 된다.



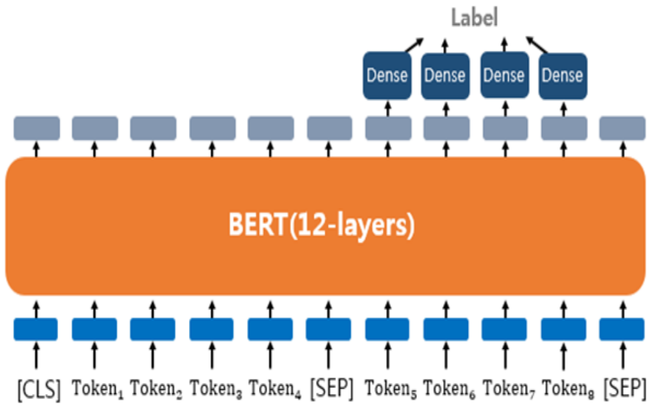
1) 하나의 텍스트에 대한 텍스트 분류 유형



3) 텍스트의 쌍에 대한 분류 또는 회귀 문제



2) 하나의 텍스트에 대한 태깅 작업 -〉 NER



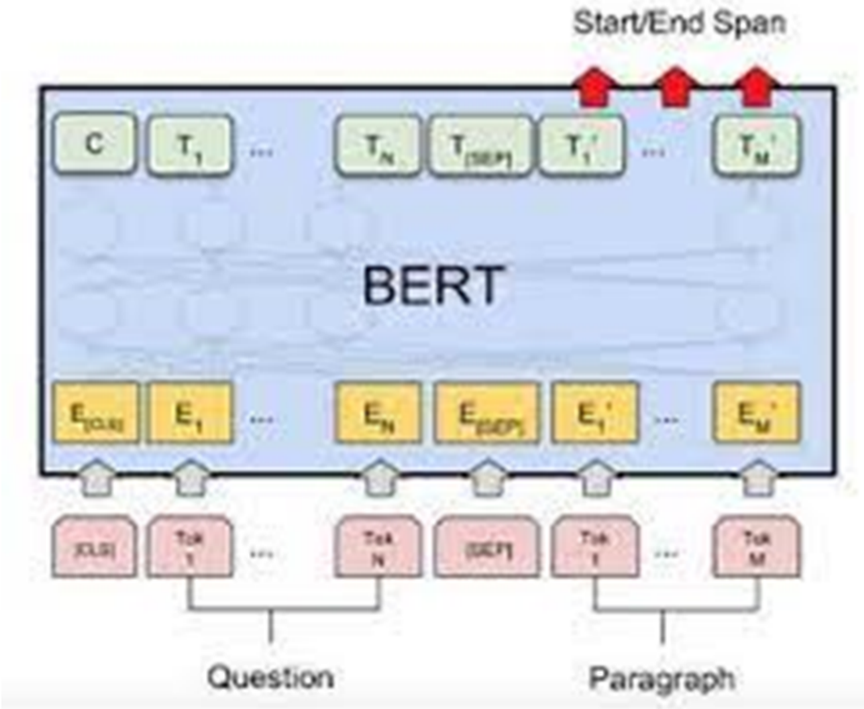
4) 질의 응답 (Question Answering)

BERT: 기계독해(=질의응답) 모델

1. 데이터셋
-기존 기계 독해 모델의 데이터셋: KorQUAD
-Input으로 context(단락), question(단락에 대한 질문), output은 answer(질문에 대한 답변)으로 구성됨.

2. Pretrained bert 모델 종류
- ETRI 한국어 bert 언어모델
[공공 인공지능 오픈 API·DATA 서비스 포털 (etri.re.kr)]
- SKTbrain koBART
[GitHub - SKT-AI/KoBART: Korean BART]

〈정답파트〉



〈질문〉 〈단락〉

BERT Challenges

https://scienceon.kisti.re.kr/srch/selectPORSrchArticle.do?cn=JAKO202032860595394&dbt=NART

1. 데이터 증식
- Back Translation / 개체명 인식을 통해 유사어를 치환하는 방법 / 노이즈를 사용하여 유사 데이터를 생성하는 Adversarial 증식

2. Hyper Parameter fine-tuning : Epoch, Batch size, Random seed, Learning Rate, Max Length 등을 조절

3.Network 설계

4. Tokenizer에 따른 성능 차이
[2010.02534] An Empirical Study of Tokenization Strategies for Various Korean NLP Tasks (arxiv.org)

5. 특정 도메인 관련 데이터 생성

2. 문서유사도

문서 내 단어 행렬

1. DTM

다수의 문서에서 등장하는 각 단어들의 빈도를 행렬로 표현

-	cat	dog	I	like
문서1	0	1	1	1
문서2	1	0	1	1
문서3	2	0	2	2

각 단어집합에서 단어의
빈도수를 표현

```python
from sklearn.feature_extraction.text import CountVectorizer

data = ["aa bb cc", "cc dd ee"]
count_vectorizer = CountVectorizer(binary='true')
data = count_vectorizer.fit_transform(data)

# Check if your vocabulary is being built perfectly
print count_vectorizer.vocabulary_
```

Sklearn countvectorizer 이용

문서 내 단어 행렬

2. TF-IDF

문서 내 각 단어의 중요도 계산

중요도 : 모든 문서에서 자주 등장하는 단어는 중요도가 낮고 특정문서에서만 자주 등장하는 단어는 중요도가 높다고 판단.

$$w_{x,y} = tf_{x,y} \times \log\left(\frac{N}{df_x}\right)$$

TF-IDF

Term x within document y

$tf_{x,y}$ = frequency of x in y

df_x = number of documents containing x

N = total number of documents

W(x,y) : x문서 내에서 y단어의 중요도

tf (x,y) : 특정 문서 x에서의 특정 단어 y의 등장횟수

idf (x,y) : 특정 단어 t가 등장한 문서의 수의 역수

문서 내 단어 행렬

2. TF-IDF

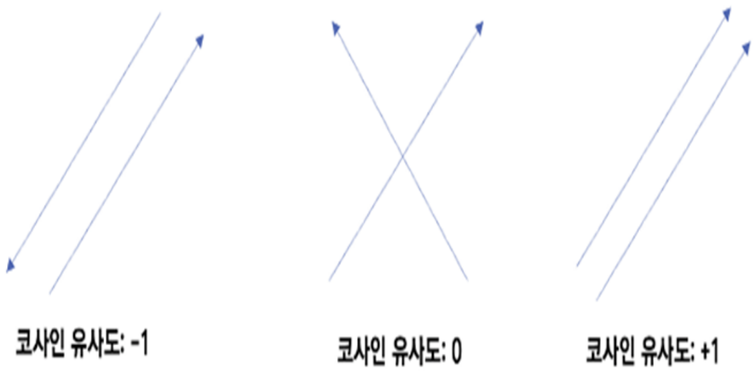
word	TF				IDF	TF * IDF			
	d1	d2	d3	d4		d1	d2	d3	d4
Italian	1/8	0/6	0/6	0/6	log(4/1)=0.6	0.075	0	0	0
Restaurant	1/8	1/6	1/6	1/6	log(4/4)=0	0	0	0	0
enjoy	1/8	1/6	1/6	0/6	log(4/3)=0.13	0.016	0.02	0.02	0
the	2/8	1/6	1/6	2/6	log(4/4)=0	0	0	0	0
best	2/8	1/6	1/6	2/6	log(4/4)=0	0	0	0	0
pasta	1/8	0/6	0/6	0/6	log(4/1)=0.6	0.075	0	0	0
American	0/8	1/6	0/6	1/6	log(4/2)=0.3	0	0.05	0	0.05
hamburger	0/8	1/6	0/6	0/6	log(4/1)=0.6	0	0.1	0	0
Korean	0/8	0/6	1/6	0/6	log(4/1)=0.6	0	0	0.1	0
bibimbap	0/8	0/6	1/6	0/6	log(4/1)=0.6	0	0	0.1	0

문서 d(n) 내에서 tf-idf를 구해서 각 문서에서 벡터로 구함

유사도 계산

1.코사인 유사도

두 벡터 사이 각도를 계산하여 두 벡터가 얼마나 유사한지 측정



1과 가까울수록 유사도가 높다고 판단

Document	TF-IDF Bag of Words	Cosine similarity with d4
The best Italian restaurant enjoy the best pasta	[0.075, 0, 0.016, 0, 0, 0.075, 0, 0, 0, 0]	0
American restaurant enjoy the best hamburger	[0, 0, 0.02, 0, 0, 0, 0.05, 0.1, 0, 0]	0.5
Korean restaurant enjoy the best bibimbap	[0, 0, 0.02, 0, 0, 0, 0, 0, 0.1, 0.1]	0
The best the best American restaurant	[0, 0, 0, 0, 0, 0, 0.05, 0, 0, 0]	1

문서의 벡터를 코사인으로 계산하여
문서 사이의 유사도 구하기

유사도 계산

2. 유클리드, 자카드

유클리드 유사도

문서 벡터 사이의 유클리드 거리 계산

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

계산한 벡터의 유클리드 거리가 짧을수록
문서의 유사도가 높음

문서의 길이가 유사하면 상관이 없어도
문서가 유사하다고 판단하는 한계가 있음

자카드 유사도

$$J(doc_1, doc_2)=\frac{doc_1 \cap doc_2}{doc_1 \cup doc_2}$$

문서 내에서 교집합 단어의 개수를
합집합 단어의 개수로 나눔

```
print('자카드 유사도 :',len(intersection)/len(union))
```

합집합 : union 함수 사용
교집합 : intersect 함수 사용

야, 너두 장학금 받을 수 있어

"스꿀라로 나한테 딱 맞는 장학금 확인하자!"

Q&A

지금까지 TEAM. SKKULAR 였습니다.
감사합니다.