

데이터
문화가
되다

“데이터, 문화가 되다 : League1”

AI야, 진짜 뉴스를 찾아줘!

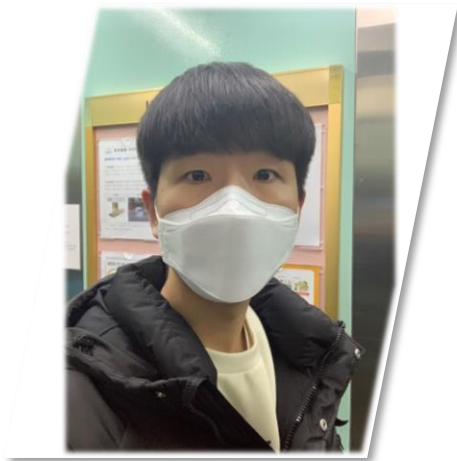
<https://www.youtube.com/watch?v=vFUmwJJHSCU>





팀장 김도연

광운대학교 정보융합학부 4학년
데이터 분석 및 알고리즘 설계 담당



팀원 안희승

광운대학교 정보융합학부 3학년
데이터 분석 및 알고리즘 설계 담당



팀원 최수지

광운대학교 정보융합학부 4학년
프로그램 개발, 디자인 담당



목차

1	문제 정의	p05
2	탐색적 데이터 분석(EDA)	p08
3	데이터 전처리	p16
4	모델 선택	p19
5	모델 구조(KoBERT)	p21
6	모델 활용 방안	p30



문제 정의

문제 정의

저자나 신문사가 불분명한 ‘가짜’일 확률이 높은 뉴스..
자극성을 위해 진위관계 파악 안된 내용까지도



코로나19 관련 사례별 팩트 체크

■ 사례 1

[스미싱 문자 내용]

오늘 코로나 피싱 당했다네요. 문자로 대구코로나 확진 내용이 와서 클릭했는데 바로 은행계좌에서 통장잔액이 인출되었다고 합니다. 신원은 명제화되었고요. 오늘 대구 북부경찰서에 신고하니 오늘 대구 북부경찰서에서 한 장수전계 58290과 함께, 신원은 명제에서 기압은 명제로 넘어갔네요. 문자나 SNS상에서 링크 절대 클릭하지 마세요.

- 인터넷 및 SNS 등을 통해 전파되고 있는 코로나19관련 스미싱 문자에 대해 확인 결과, **거짓**입니다.
- 스미싱 문자 내용과 관련하여 대구경찰에서 확인한 결과, 북부경찰서에 위와 같은 내용으로 신고 접수된 사건은 없는 것으로 확인되었습니다.
- 위와 같이 허위사실 유포 및 생산 행위는 **형법상 공무집행방해죄**로 처벌될 수 있습니다.

※ 형법 제137조(허위 제의)에 의한 공무집행방해죄 : 5년 이하의 징역, 1천만원 이하 벌금

■ 사례 2

[유포된 문자 내용]

[9시 30분] 현재 31번 확진자 확인 후 집에서 자가격리하겠다고 발병통지고 병원 온 나서라, 계단하러던 간호사 등 마스크 벗기고 뽀개고 뽀개고 뽀개고 시도 [10시 30분] 현재 간호사 다수 병원 잠진 진행 중 발병이 계속 상태 환자 가족 및 신진지 신도를 다수 병원으로 물려와 병원 업무 방해 중

- 인터넷 및 SNS 등을 통해 전파되고 있는 31번째 확진자가 간호사와 몸싸움 및 신도를 방문하여 병원 업무 방해 문자내용은 **거짓**입니다.
- 위와 같이 허위사실 유포한 경우 **형법상 공무집행방해죄** 및 해당 병원에

■ 사례 3

[유포된 문자 내용]

<47번째 확진자 동선(32일 날짜)> 14시 신계역역화점 동대구 08시 현대백화점 수성 16시 현대백화점 대구 10시 동아마트 수성 18시 아마도 인근 11시 동성로 일식집 19시 경산 모 웰스장 12시 홈플러스 동촌 21시 성서 쇼핑월드 솔림(물놀이)

- 인터넷 및 SNS 등을 통해 전파되고 있는 47번째 확진자 동선이 포함된 문자 내용은 **거짓**입니다.
- 오늘 질병관리본부 및 대구시 발표에 따라 47번째 확진자가 발생하지 않았음에도 47번째 확진자의 동선이 포함된 내용은 명백히 허위사실입니다.
- 위와 같이 허위사실 반박적으로 유포할 경우 **정보통신망법 위반**으로 처벌될 수 있으며, 추가적으로 해당 7명제에 대한 **업무방해죄**로 처벌될 수 있습니다.

※ 정보통신망법 제44조(7 제1항 제3호(명예훼손 조항) : 5년 이하의 징역, 1천만원 이하 벌금
형법 제145조(허위 제의)에 의한 공무집행방해죄 : 5년 이하의 징역, 1천만원 이하 벌금

현재 대구 전역에 퍼지고 있는 위와 같은 문자 내용에 대해 방송통신심의위원회 통해 **삭제 · 차단 요청** 하였습니다.

아울러, 코로나19 관련하여 국민 불안과 사회 혼란을 불러 일으키는 온라인상 허위조작정보 · 개인정보 유포행위 및 스미싱 문자 등 범죄행위에 대해 경찰은 철저히 수사하여 엄정대응할 방침입니다.

최근 한국에서도 코로나19 관련 가짜뉴스가 기승해..
실제 피해 사례가 속출하기도

문제 정의



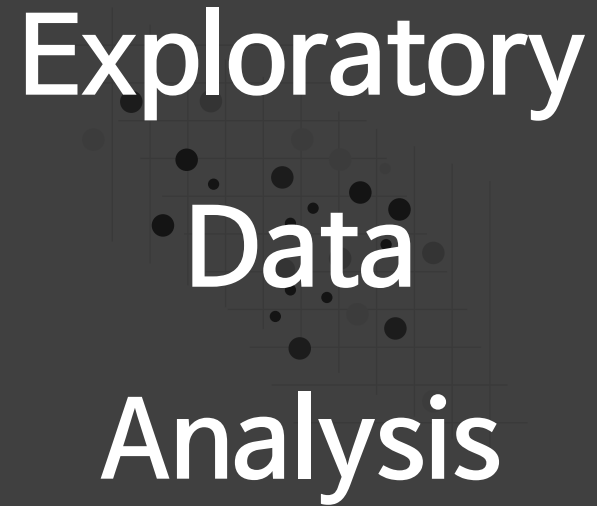
2016년 11월 미국 대통령 선거 당시 ‘가짜뉴스’가 보여준 파급력은 전 세계를 놀라게 했다.

당시 페이스북에 가장 많이 공유된 기사
5개 중 4개가 가짜뉴스였다.
‘프란체스코 교황이 트럼프를 지지한다’(1위)거나
‘힐러리가 테러단체 이슬람국가(IS)에
무기를 팔았다’(3위) 등은 삼시간에 전 세계로 퍼졌다.

진실은 중요하지 않았다.
페이스북 이용자들은 자극적인 내용을 접하자마자 ‘공유(share)’
버튼을 눌렀다. 이 가짜뉴스에 대한
공유나 댓글 건수는 각각 96만건, 79만건에 달했다.

출처 :

지식백과 <https://terms.naver.com/entry.nhn?docId=3559797&cid=42107&categoryId=42107>

A graphic with a yellow square border containing the text 'Exploratory Data Analysis' in white. In the background, there is a faint grid and a cluster of black dots of varying sizes, resembling a scatter plot.

Exploratory Data Analysis

데이터셋 탐색

데이터 개수 118,745개
결측치 0개

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 118745 entries, 0 to 118744  
Data columns (total 6 columns):  
#   Column      Non-Null Count  Dtype  
---  ---  
0   n_id        118745 non-null object  
1   date        118745 non-null int64  
2   title       118745 non-null object  
3   content     118745 non-null object  
4   ord         118745 non-null int64  
5   info        118745 non-null int64  
dtypes: int64(3), object(3)  
memory usage: 5.4+ MB
```

News_Train.csv

- n_id, date, title, content, ord, info 총 5개의 컬럼으로 구성
- 중복 데이터 다수 존재
- Train 데이터는 약 12만개, Test데이터는 14만개로 구성(훈련데이터 < Test 데이터)

데이터셋 탐색

News_Train.csv

	A	B	C	D	E	F
1	n_id	date	title	content	ord	info
2	NEWS0258	20200605	[마감]코스	[이데일리	1	0
3	NEWS0258	20200605	[마감]코스	"실적기반"	2	1

진짜

품목별로 보면 실리콘 매출이 50%, 쿼츠가 34%를 차지한다

나머진 알루미늄 등으로 15%다

지난 2009년 미국의 실리콘 잉곳 및 쿼츠 전문회사인 WCQ(West Coast Quartz Corporation)을 인수했다

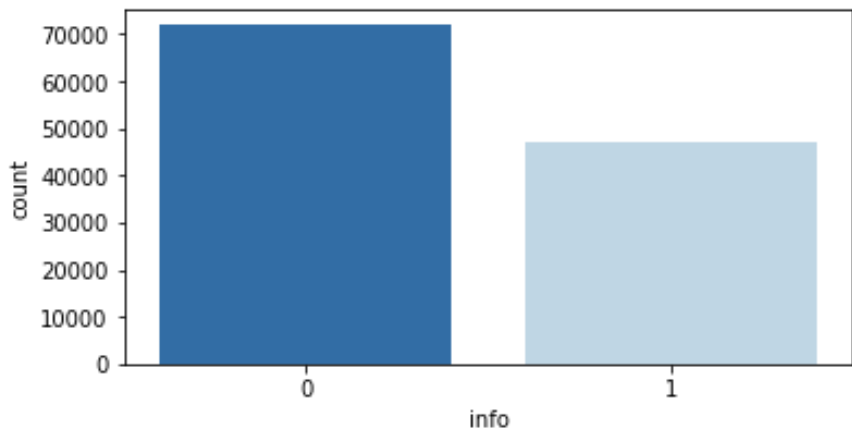
가짜

소름 돋는 적중률. 정통사주·토정비결·이름풀이 무료

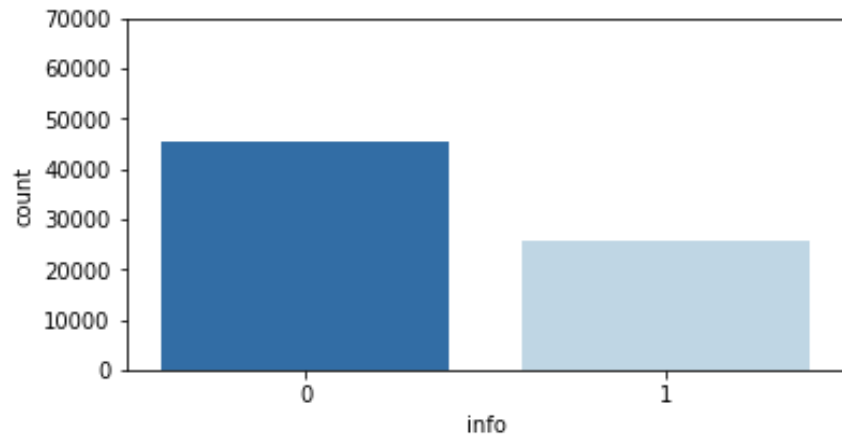
임상3상완료 - 셀트리온 넘을 800% 초대형 바이오株.

소름 주의. 올해 대박 나는 기회가 찾아올까.

탐색적 데이터 분석(EDA)



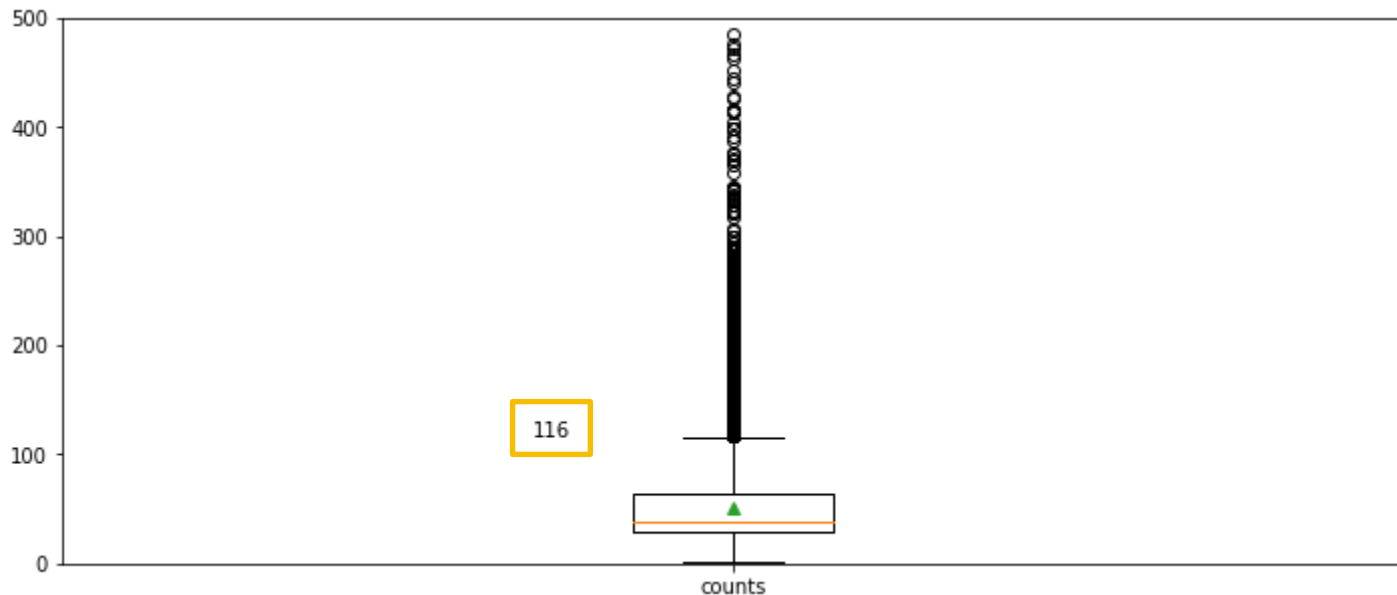
중복 데이터
제거 전



중복 데이터
제거 후

0 : 진짜
1 : 가짜

탐색적 데이터 분석(EDA)



Train set의 Content 길이 분포

탐색적 데이터 분석 (EDA)

워드 클라우드로 살펴보는 news_train.csv에서
많이 사용된 단어 300개

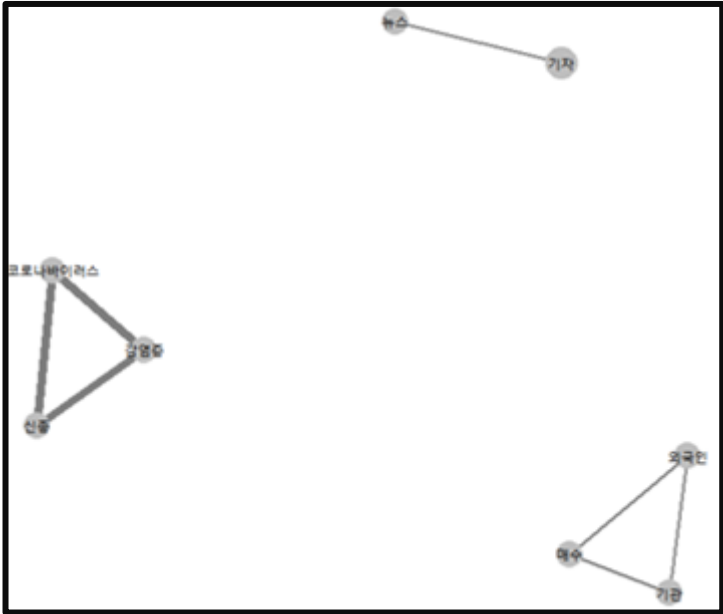


출현 빈도가 가장 높은 단어는 '코로나'

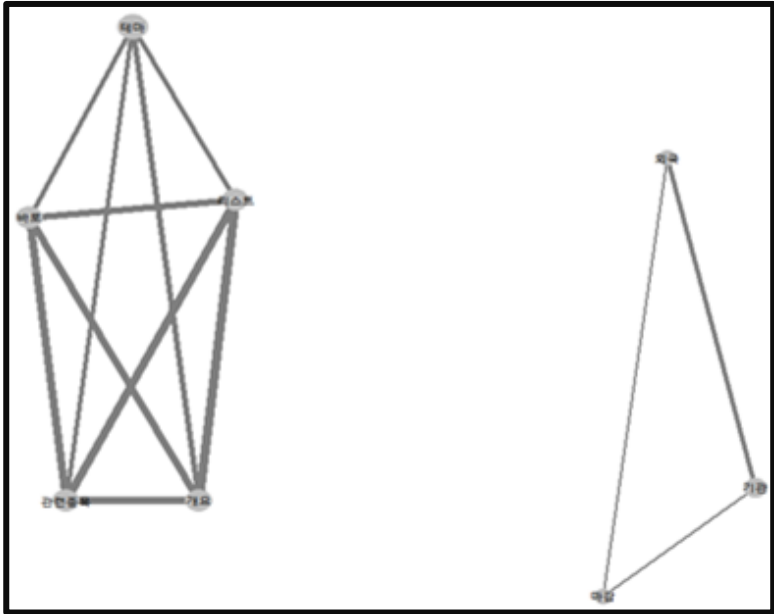
그 다음으로 '기자', '밝혔', '사업',
'투자', '서울' 등
단어 빈도수가 높았다.

탐색적 데이터 분석(EDA)

빈출 단어의 네트워크 구조 도식화



REAL



FAKE

탐색적 데이터 분석(EDA)

진짜 뉴스

[신종, 코로나바이러스, 감염증]
[외국인, 기관, 매수]
[뉴스, 기자]

끼리 상관성을 보인다.

가짜 뉴스

[테마, 바로, 리스트, 관련종목, 개요]
[외국인, 기관, 마감]

끼리 상관성을 보인다.

네트워크 맵에서 알 수 있는 단어들의 상관관계

데이터 전처리

데이터 전처리

01 Title 열과 Content 열 합체

한 개의 Title 당 평균 32개의 content가 있습니다.

둘을 합할 경우 데이터 크기 문제가 생겨 기존의 모델에서는 Content 열 만 뽑아서 분석하였습니다.

하지만 koBERT 모델의 경우 Title과 Content 간의 유의미한 연관성이 있어 합쳐서 사용하였습니다.

```
고유 title 개수 : 3688  
한 title 당 content 최소 개수 : 1  
한 title 당 content 최대 개수 : 396  
한 title 당 content 평균 개수 : 32.2
```

데이터 전처리

02 중복 데이터 제거

Train데이터의 경우 행의 개수가 약 12만개에 달할 정도로 굉장히 데이터의 크기가 큼니다.

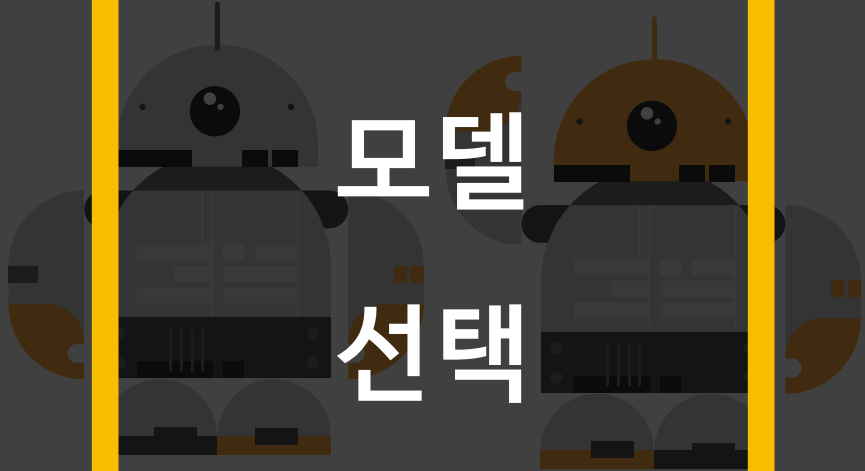
따라서 중복을 제거할 수 있다면 데이터를 전처리 하거나 모델에 대한 학습을 진행할 때 요구되는 시간이 줄일 수 있습니다.

또한 Kobert 모델의 경우 이미 한글데이터를 통해 사전학습이 진행된 모델이기 때문에,

중복제거를 했을 때 나타나는 레이블 값 비율의 차이가 큰 영향을 가지지 못할 것이라 판단하였습니다.

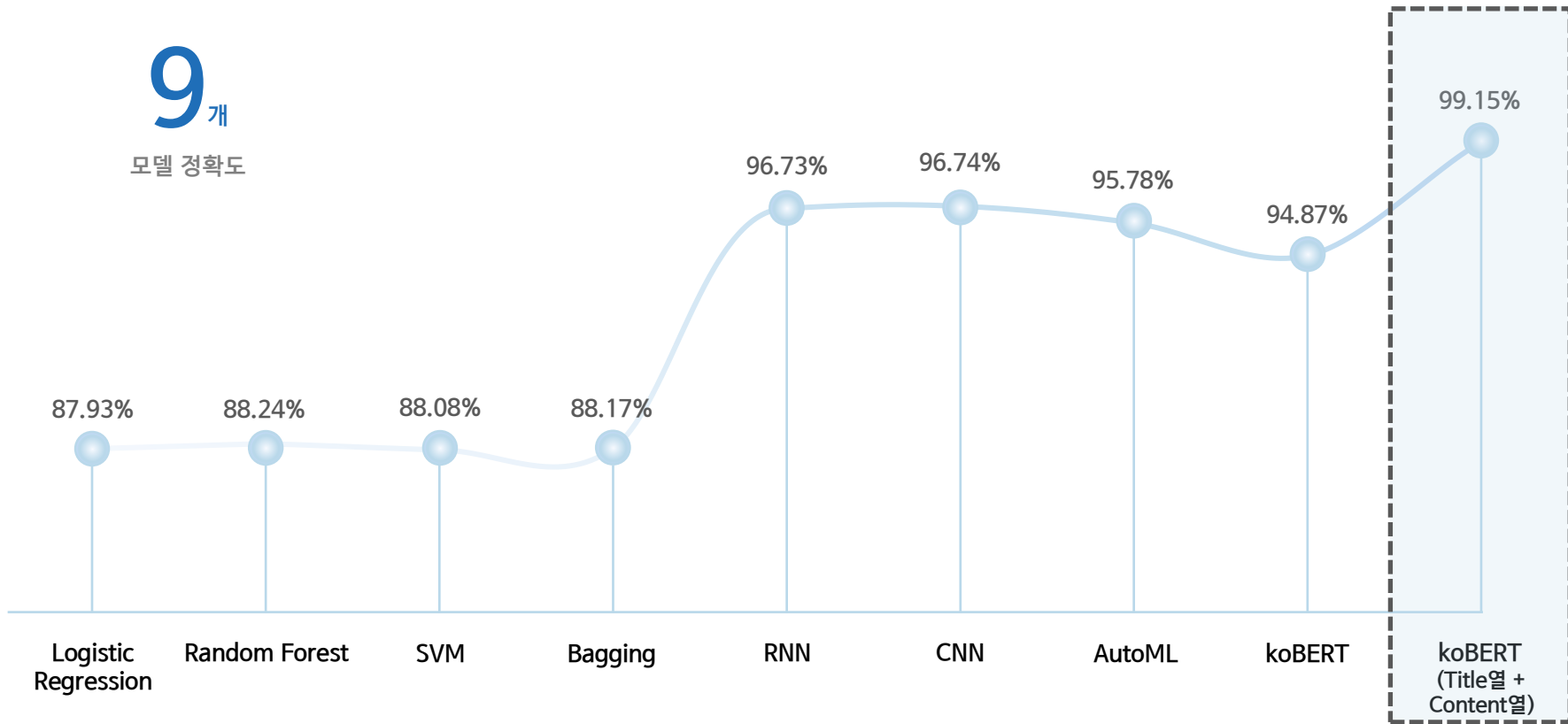
중복 제거 전 개수:	118745
중복 제거 후 개수:	70982

모델 선택



9개

모델 정확도



* 소수점 세 자리 이하 반올림
* 기본 전처리(중복 제거) 기준



Transformer (모델링 기법)

“Attention is all you need“ (Google, 2017) 논문에서 공개된 모델입니다.
SeqtoSeq의 Encoder/Decoder 구조를 가지며,
RNN과는 다르게 Attention이라는 구조만으로 전체 모델이 만들어집니다.

Transformer은 Encoder에 입력한 문장에 대한 정보와,
Decoder에 입력한 문장 정보를 조합하여
Decoder의 문장 다음에 나올 단어를 생성하는 방법입니다.
다만, 순환 신경망을 활용하지 않고
Attention 기법과 순방향 신경망을 사용하였습니다.

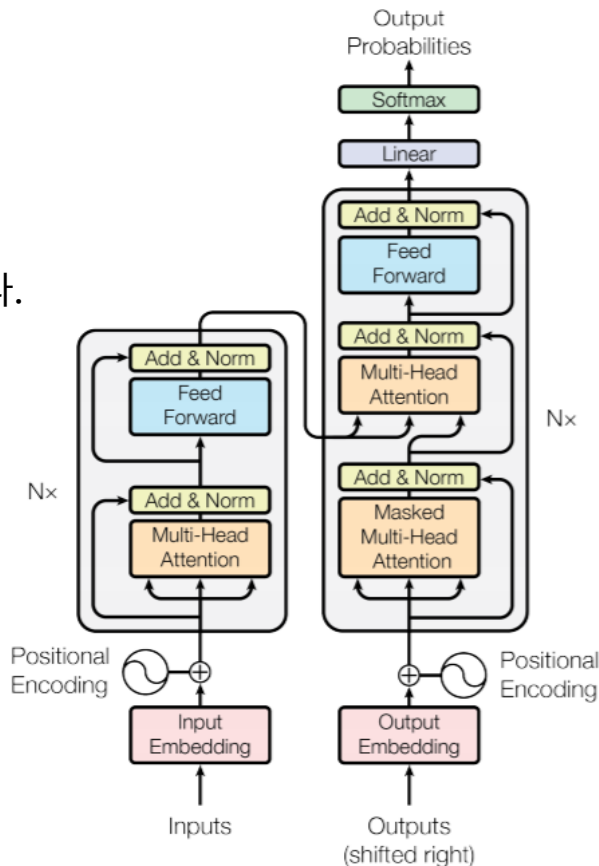


Figure 1: The Transformer - model architecture.

Attention Layer

이 계층에서는 Transformer의 핵심이기도 한 Attention이 일어납니다.

Attention이란, 문장에서 각 단어끼리 얼마나 관계가 있는지 계산하여 반영하는 방법입니다.

Attention을 하기 위하여 우선 Attention Score을 구합니다.

Attention Score는 각 단어를 기준으로 다른 단어들과의 관계를 값으로 계산하여 얻은 값입니다.

이 값이 높을 수록 서로 관계가 높은 단어입니다.

이렇게 구한 Attention Score을 하나의 테이블로 만들면, Attention Map을 구할 수 있습니다.

Attention Map을 활용하면 한 문장을 단어마다 서로의 관계를 반영한 값으로 바꿀 수 있습니다.

[BERT]

BERT Model

Pre-Trained Model

Transformer Architecture의 Encoder를 차용합니다.

Transformer과의 차이점으로는,

Feed-Forward Neural Network에서 활성화 함수로

ReLU대신 GeLU를 사용하였습니다.

(GeLU : 0값 주위에서 부드럽게 변화하는 함수입니다.)

다른 사전 학습 모델(GPT, ELMo)와 달리 양방향성을 이용합니다.

Attention 계층과 Feed-Forward 신경망 계층이 총 12번 반복되는 구조를 가집니다.

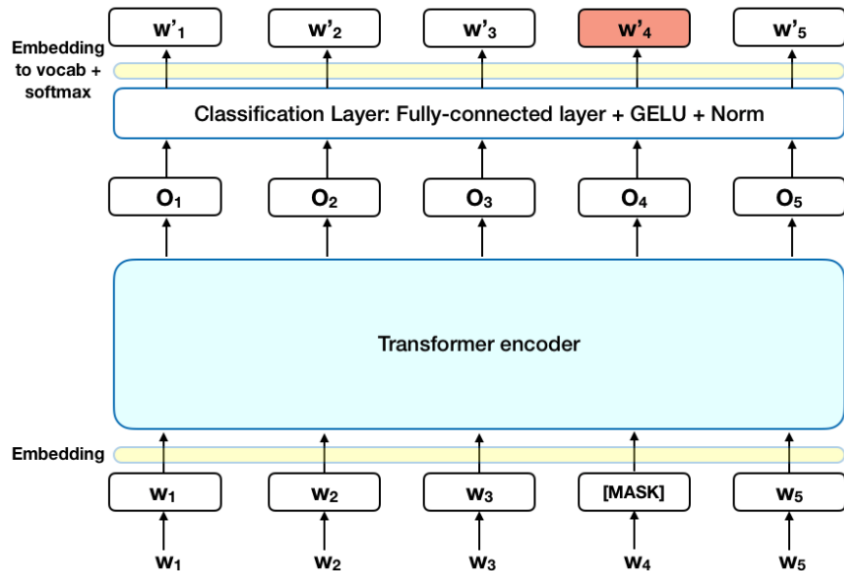


그림 : BERT 구조

Language Model

Masked Language Model은 입력값으로 문장을 넣고 마스킹된 단어를 예측하는 학습
BERT의 양방향성으로 마스킹된 단어를 예측하는 방향이 문장의 순서와 무관하게 앞 뒤 모두 진행하여
이러한 방법을 사용한다.

ex) 매일 [MASK] 해는 [MASK] 에서 뜨지 -> 매일 아침 해는 동쪽에서 뜨지

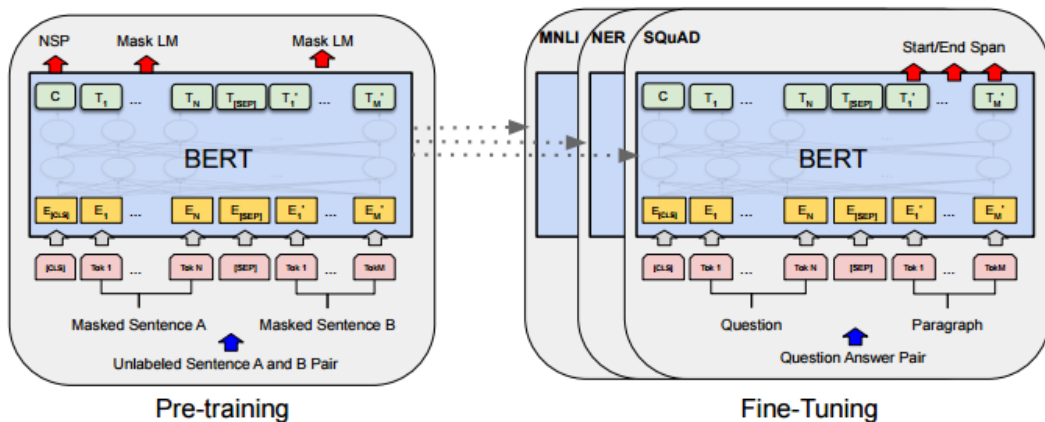
NSP : 입력으로 주어진 두 문장이 이어진 문장인지를 예측한다.

위의 비지도 사전학습을 진행하며 BERT Model이 해당 언어에 대한 전반적인 이해를 가능하게 한다.

Pre-trained Model

버트 모델의 경우 Language Model 형태로 해당 데이터에 대한 사전학습을 진행한 후, 가장 적합한 가중치 파라미터를 학습한다.

학습된 모델은 fine-tuning을 통해 자연어 처리에 관련된 각종 Task에 적용 가능하다.

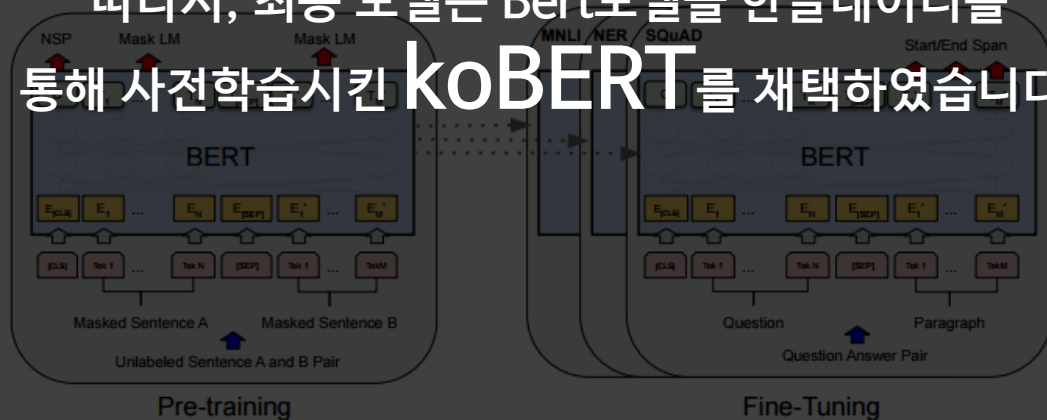


Pre-trained Model

버트 모델의 경우 Language Model 형태로 해당 데이터에 대한 사전학습을 진행한 후, 가장 적합한 가중치 파라미터를 학습한다.

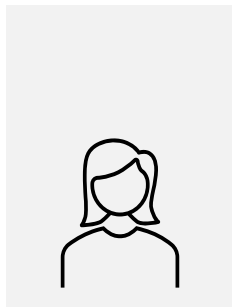
학습된 모델은 fine-tuning을 통해 자연어 처리에 관련된 각종 Task에 적용 가능하다.

따라서, 최종 모델은 Bert모델을 한글데이터를
통해 사전학습시킨 **koBERT**를 채택하였습니다.





고객 조사 (Persona)



박OO (24, 여)
대학교 재학중

“200만원 저축한
돈으로 주식을
시작해보려고 해요.”

“주식 종목 트렌드를
알고 싶은데 어떤
정보를 믿어야 할지
모르겠어요.”

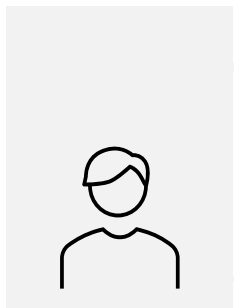
고객 프로필

현재 재학중인 대학생. 주말마다 아르바이트를 함.
대학 동기들이 주식을 시작하는 것을 보고 본인도 주식을 시작하고자 함.
주식 용어를 모르며, 어떤 주식에 투자해야 하는지 아무 정보가 없음.

고객 요구사항

아르바이트로 모은 200만원으로 시작하여 조금씩 수익을 원한다.
주식에 대한 공부 보다는 우선 먼저 투자를 하길 원한다.
상승이나 하강이 발생하면 그 이유에 대한 정확한 정보를 빠르게 얻길 원한다.

고객 조사 (Persona)



김OO (34, 남)
회사원

“ 은행에 저축만
해서는 돈이 모이지
않아 주식에 좀 자금을
배분해 보려고 해요.”

“ 자산을 지키면서
안전한 투자를 하고
싶은데 생각보다 어려워서
모르겠어요.”

고객 프로필

사기업을 다님. 은행에 예금통장과 적금통장을 동시에 가지고 있음.
현재 오피스텔에서 거주하며 여자친구와 결혼 계획을 가지고 있음.

고객 요구사항

통장보다는 높은 금리이면서도 안전 투자를 하길 원한다.
큰 수익률보다는 현 자산을 지키면서도 투자로 접근하기를 원한다.
일정 수익이 발생하면 통장에서 주식으로 자산을 이동하길 원한다.

아이디어 제안

이를 바탕으로, 아이디어를 제안합니다.

나무 증권 어플리케이션의 챗봇 서비스

가짜 뉴스 필터링 / 기사 추천 / 통계

뉴스 탐지를 통한 트레이딩 봇 개발

진짜 뉴스를 바탕으로 주식 트렌드 파악
거래량, 상승폭/하락폭 등 고려
여러 가지 성향의 트레이딩 봇을 개발하여 사용자가 선택

서비스 개요

개요

트레이딩 챗봇

정보의 정확성 여부 필터링을 한 뉴스 데이터를
사용자의 요구 사항에 맞춰
대답하는 챗봇을 제공한다.

또한 챗봇 서비스를 제공하며 얻은 정보를 기반으로 한 트레이딩 기능을 제공한다.

서비스 개요

특징 및 장점

사용자가 챗봇에게 현재 주식 시장 동향을 물어본다.

챗봇은 뉴스 데이터에서 가짜 정보를 필터링한다.

이렇게 필터링을 거쳐 **신뢰 할 수 있는 정보**를 중요도 우선 순위별로 추려 **제공**한다.

주식 초보자가 자신에게 어려운 주식 용어 등을 챗봇에게 질문하면 이해하기 쉬운 답변을 준다.

이를 통하여 **초보자도 쉽게 주식에 접근**할 수 있도록 도와준다.

주식 투자가 처음이라 생소한 사용자에게 **봇을 이용한 트레이딩 기능**을 제공하여

봇에게 돈을 맡기기만 하면 **수익률을 낼 수 있도록 편의**를 제공한다.

▶ 트레이딩 챗봇 쿠나봇입니다.
무엇을 도와드릴까요?



오늘 트렌드 뉴스 알려줘



오늘 트렌드 뉴스입니다.
꼭 챙겨 보셔야 할 뉴스도 있어요!

트렌딩 토픽

종합 경제 기술 정치 사회 문화 세

01 삼성전자 특별배당의 의미

02 IMF 한국 공매도 재개 가능 자영업자 지원

03 반격 나선 철강업계 포스코 2023년 102兆 목표

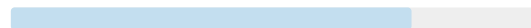
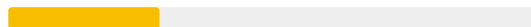


쿠나봇 (Qu&a)

나의 투자형은?



‘나무’님은
시장위기에 강한
안정적인 투자형



쿠나
권리 공매도가 뭐야??
너무 어려워..



권리 공매도란?
쉽게 말하자면, **가지고 있지 않은 주식의
팔 권리를 빌려서 매도를 하는 행위**를 말합니다.

권리 공매도를 원하신다면,
[홈] 버튼을 누르시고
[종목검색] - [국내주식] - [권리 공매도]
들어가시면 권리 공매도 주문이 가능하세요!

국내/해외 주식	금융상품	디지털 자산관리	계좌/이체 청약/대출	안내/문의 등록/설정
HOME/관심	주식 현재가			
종목검색	주식 주문			
국내주식	국내주식 예약주문			
해외주식	시세포착주문			
선물옵션	연금저축 ETF주문			
기타시장	권리공매도			

쿠나
OO전자 주식 동향은 어때?



제가 보기에요!

OO전자는 현재 하락장이라
신중하게 투자 하셔야 할 것 같아요.



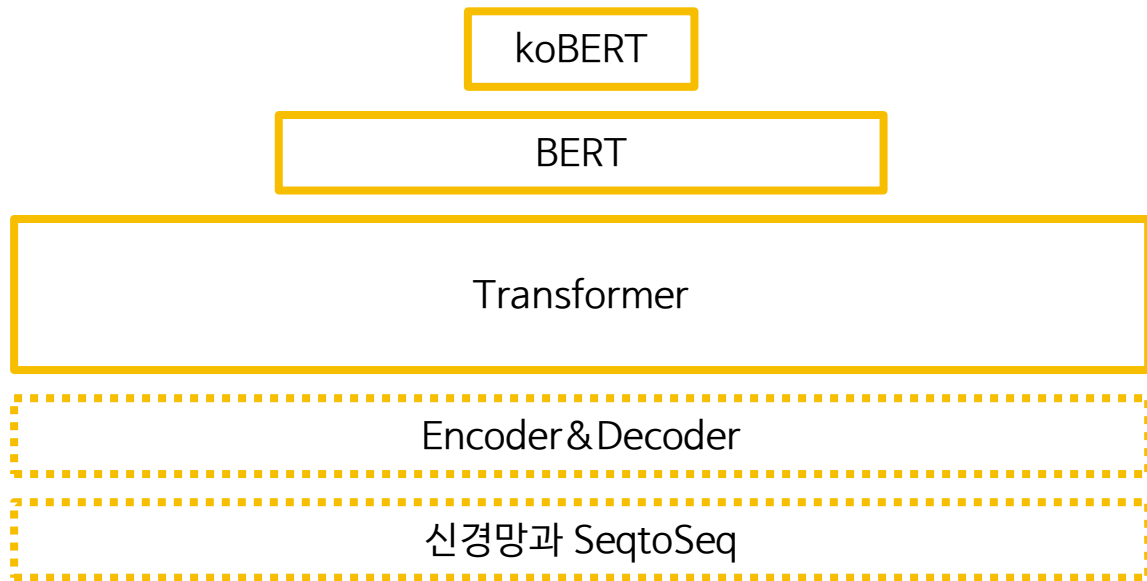
지금까지 League1_[정윙팡팡] 입니다.

감사합니다.

(별첨)

koBERT모델의 기반 및 핵심개념 설명 순서

koBERT모델의 기반을 설명하기 위하여, 단계적으로 설명 드리고자 합니다.



SeqtoSeq

SeqtoSeq란, Sequence를 입력으로 넣어서 Sequence를 출력합니다.

Ex) Input : 지금 어디 가?

Ex) Output : 나는 집에 가고 있어

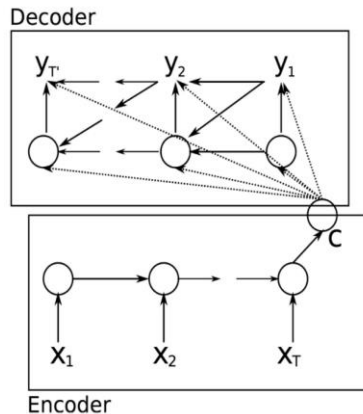
특징으로는, 매 단계마다 재귀 순환 신경망을 적용합니다.

패딩 처리와 각종 특징 토큰들이 필요하며, 인코더와 디코더 모델을 사용합니다.

Why?) 인코더에서 재귀 순환 신경망을 통하여

문장의 전체 정보가 요약된 벡터 C를 구합니다.

이 벡터 C를 이용하여 디코더의 타임 스텝마다 출력을 구합니다.

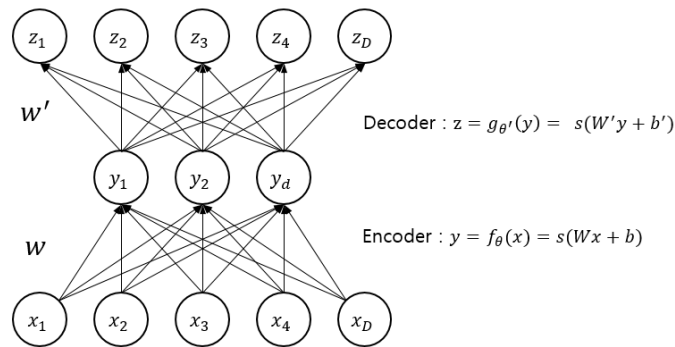


AutoEncoder (신경망)

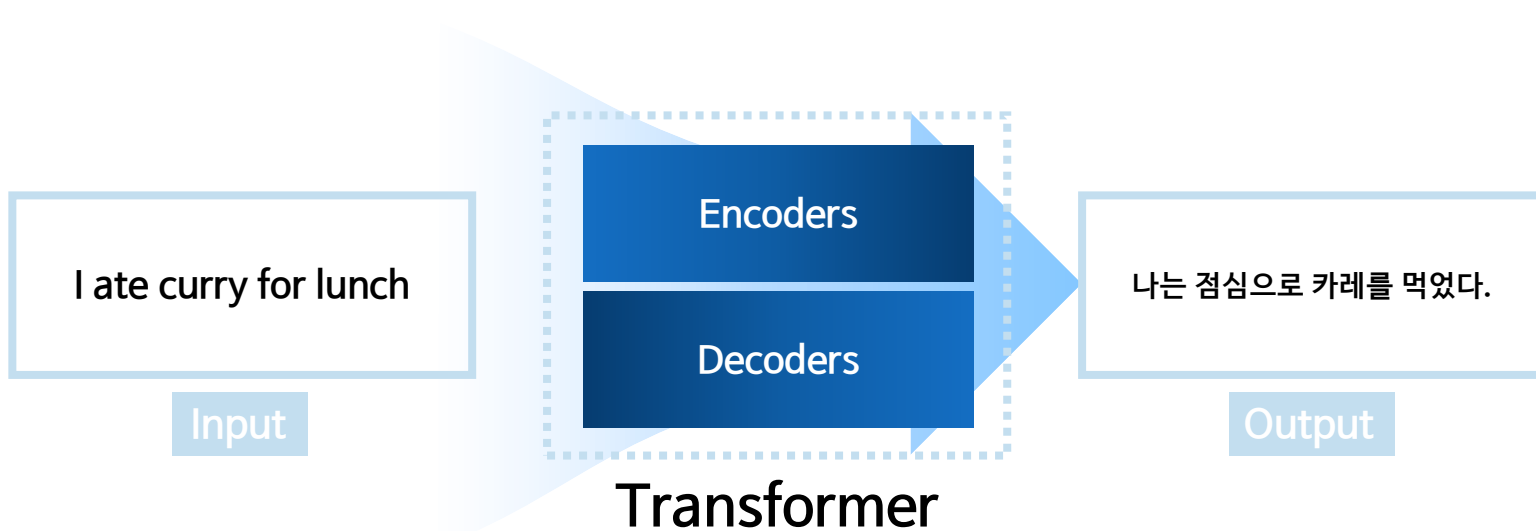
AutoEncoder는 비지도 데이터, 즉 라벨이 없는 데이터에 대하여
입력데이터의 밀집 표현을 학습할 수 있는 인공 신경망 모델입니다.

밀집 표현이란, 입력 벡터를 밀집 벡터로 바꾸는 것입니다.
밀집 벡터는 희소 벡터의 반대의 의미를 가지고 있습니다.

AutoEncoder는 이름은 Encoder이지만 인코더와 디코더 부분으로 나뉘어 있습니다.



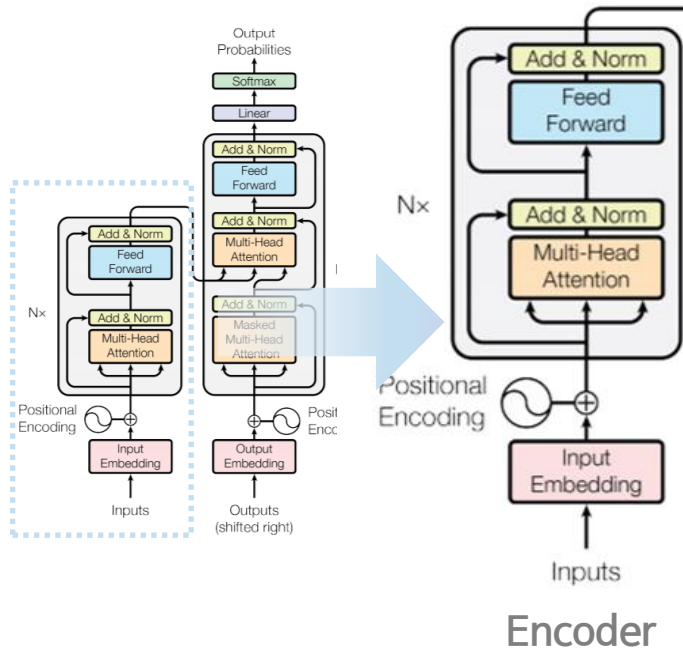
Transformer (모델링 기법)



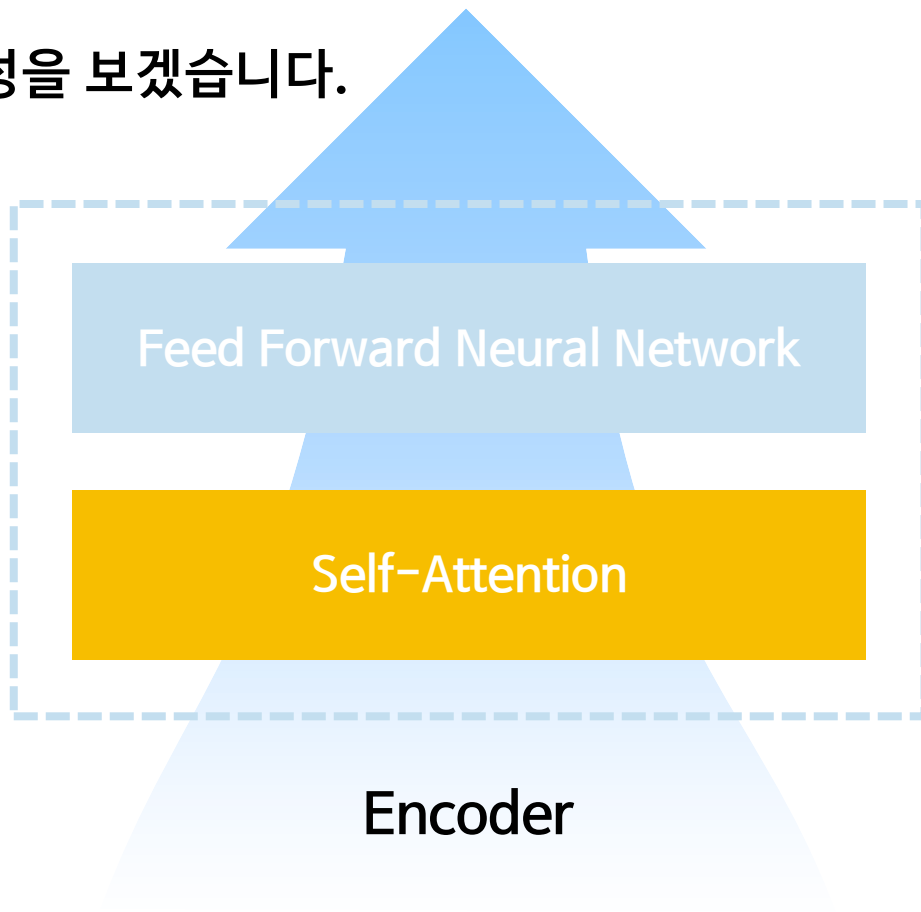
BERT 모델 소개

이전 슬라이드에서 언급한 Transformer에서
Encoders 부분만 사용한 모델이 BERT입니다.

BERT는 다국어를 지원하지만, 한국어의 특성 상 미흡한 부분의 성능을
한국어에 특화하여 향상한 모델이 koBERT입니다.



Encoder 내부 구성을 보겠습니다.



Attention Layer

이 계층에서는 Transformer의 핵심이기도 한 Attention이 일어납니다.

Attention이란, 문장에서 각 단어끼리 얼마나 관계가 있는지 계산하여 반영하는 방법입니다.

Attention을 하기 위하여 우선 Attention Score을 구합니다.

Attention Score는 각 단어를 기준으로 다른 단어들과의 관계를 값으로 계산하여 얻은 값입니다.

이 값이 높을 수록 서로 관계가 높은 단어입니다.

이렇게 구한 Attention Score을 하나의 테이블로 만들면, Attention Map을 구할 수 있습니다.

Attention Map을 활용하면 한 문장을 단어마다 서로의 관계를 반영한 값으로 바꿀 수 있습니다.

Attention 기법을 사용하는 이유

나는 점심으로 카레를 먹었다. 그것은 무척 맛있었다.

[카레를 ↔ 그것은]

이 두 단어가 서로 지칭하는 것이 동일하다는 것을 어떻게 알 것인가?

Attention 개념

Self-Attention + Multi-Head Attention

Self-attention에 multi-headed attention 매커니즘을 더하여 attention layer 성능 향상

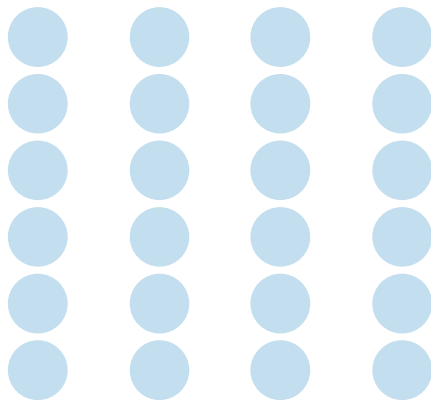
Self-attention의 결과로, 8개의 가중치 행렬을 거치면 각기 다른 행렬들을 가지게 된다.

이 행렬을 바로 feed-forward layer로 보낼 수는 없다.

왜냐하면 feed-forward layer은 한 위치에 대하여 한 행렬만 입력 값으로 받기 때문이다.

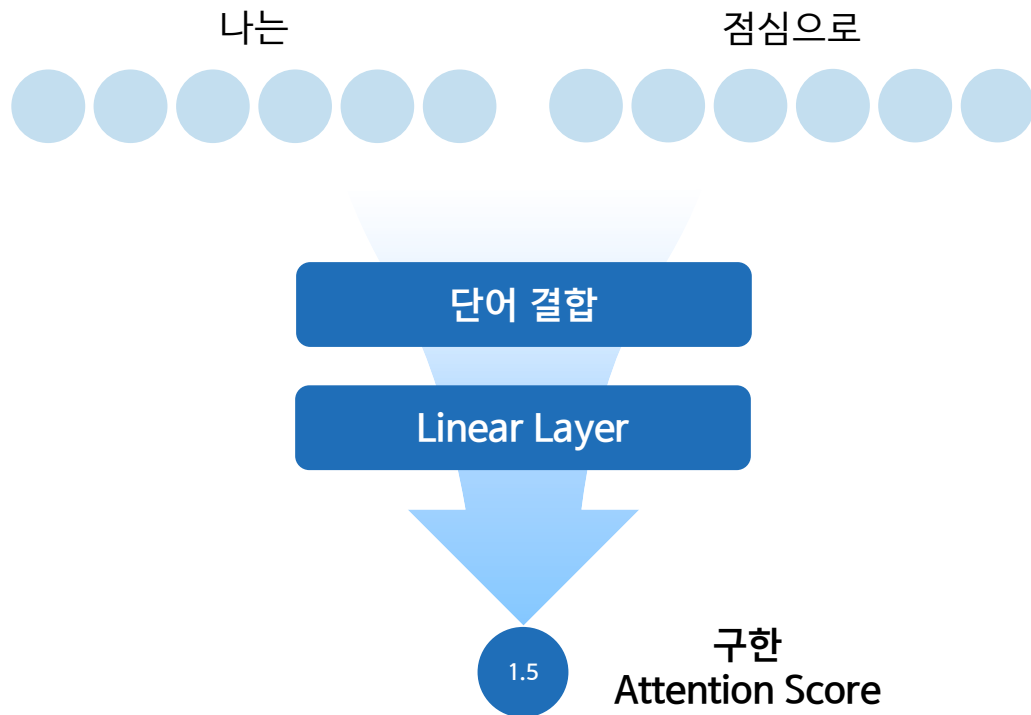
Attention Score을 구하는 방법

나는 점심으로 카레를 먹었다.

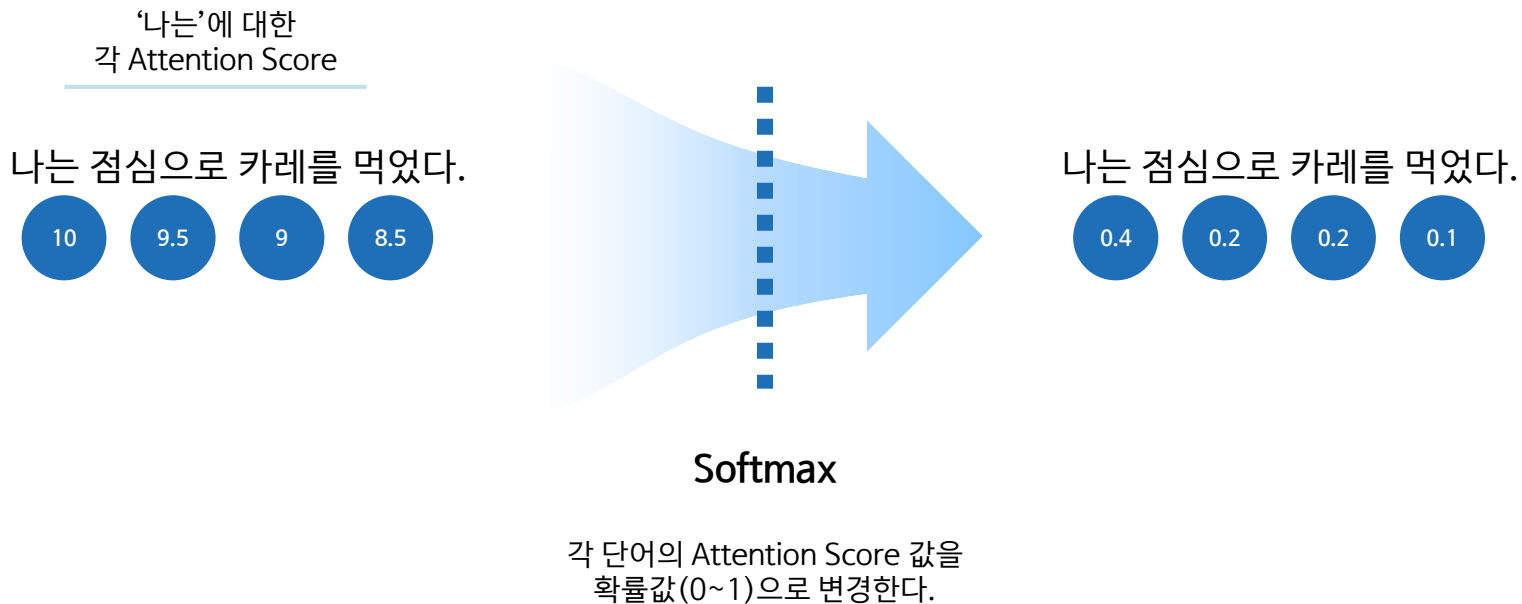


각 단어의 의미를 embedding 알고리즘을 이용하여 벡터 값으로 표현

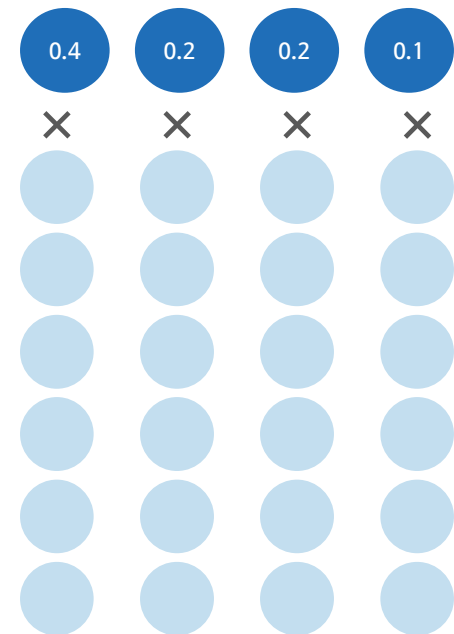
Attention Score을 구하는 방법



Attention Score로 Attention Map 구하기



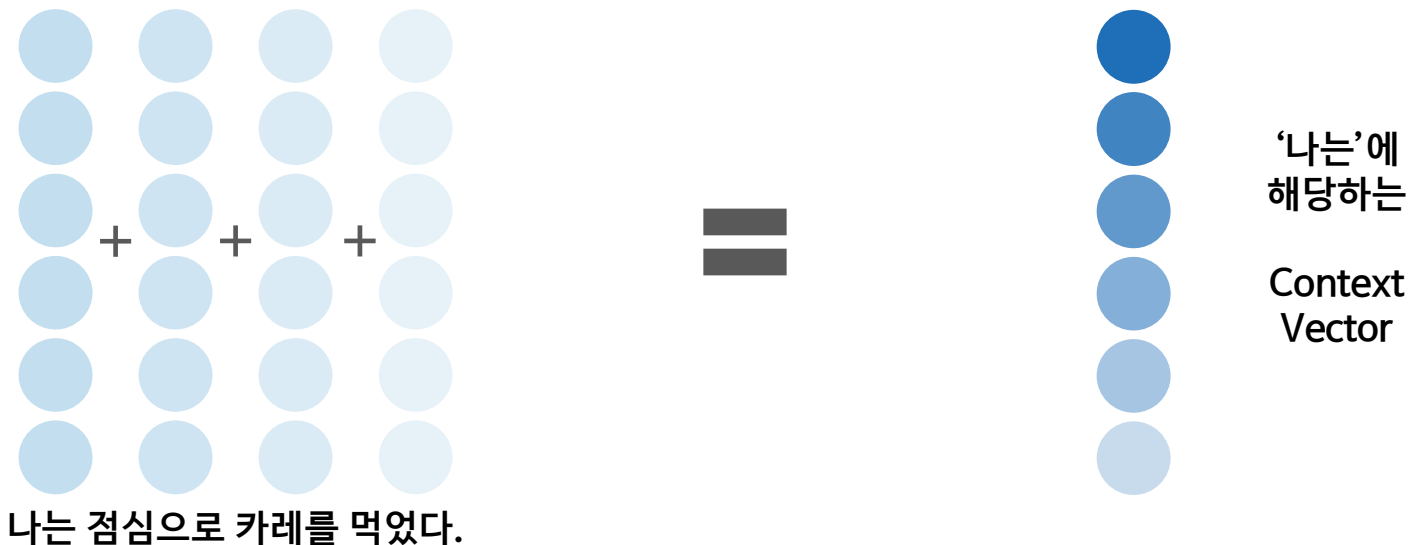
Attention Score로 Attention Map 구하기



각 어텐션 스코어와
각 단어벡터끼리
상수곱 연산

나는 점심으로 카레를 먹었다.

Attention Map으로 Context Vector 구하기



Encoder에서 Attention

Self-Attention

값 입력시 Query(질의 벡터) / Key (키 벡터) / Value (값 벡터) 가 생성됩니다.

Ex)

Query : 사전에서 단어를 찾을 때 찾고자 하는 단어

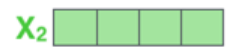
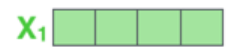
Key : 사전에 등록된 단어

Value : Key에 해당하는 단어의 의미

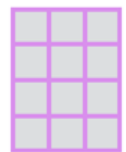
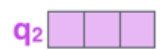
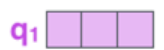
이 세 가지의 벡터는 Attention을 계산하기 위하여 도움을 주는 추상적 개념입니다.

Input 나는 점심으로

Embedding

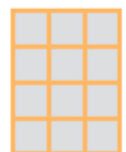
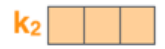
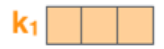


Queries



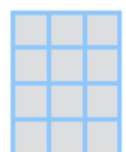
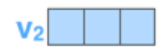
W^Q

Keys

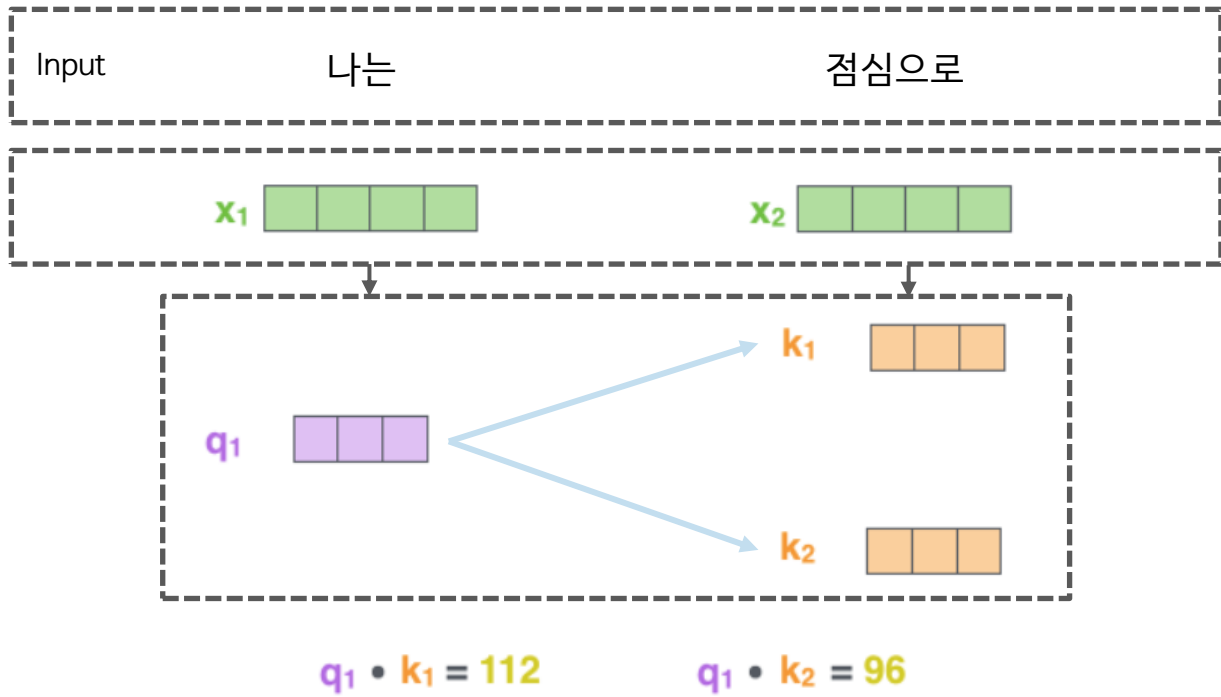


W^K

Values



W^V



$$q_1 \cdot k_1 = 112$$

$$q_1 \cdot k_2 = 96$$

Divide by 8 ($\sqrt{d_k}$)

14

12

Softmax

0.88

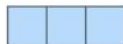
0.12

Softmax

X

Value

V₁



V₂



각 값을 8로 나눈다.

8은 key 벡터 사이즈 64의 제곱근이다.

이 값을 softmax(활성화 함수)를 통과하여 확률값으로 변환한다.
따라서 이 값들을 모두 합치면 1이 나온다.

$$q_1 \cdot k_1 = 112$$

$$q_1 \cdot k_2 = 96$$

Divide by 8 ($\sqrt{d_k}$)

14

12

Softmax

0.88

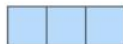
0.12

Softmax

X

Value

V₁



V₂

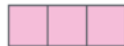


이렇게 나온 값은 현재 ‘점심으로’ 라는 단어의 encoding에서
각 단어들의 표현이 들어갔는지를 알 수 있다.

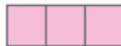
이 말은, ‘점심으로 ’ 라는 값이 가장 높은 점수를 가지며,
그 다음으로 연관성 있는 단어 순대로 높은 점수를 가지게 된다는 뜻이다.

Sum

z_1



z_2



이렇게 얻게 된 ‘점심으로’ 에 대한 각 단어들의 값에 value 벡터를 곱한다.

Value 벡터를 곱하면 중요한 값을 가진 단어들은 남게되고, 상대적으로 작은 값을 가지고 있던 단어들은 더 영향을 낮출 수 있다.

이렇게 구한 것이 attention map (weighted value 벡터들의 조합)이며, 이 구성원들을 모두 합한다.

Feed-Forward Neural Network에 벡터 넣기

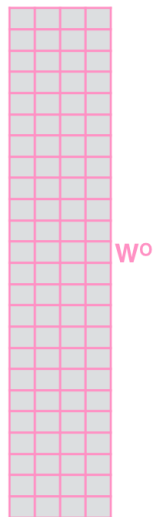
따라서 이 8개의 행렬을 하나의 행렬로 합치려면
일단 모두 이어 붙여 하나의 행렬로 만들어버리고, 다른 가중치 행렬을 곱한다.

1) Concatenate all the attention heads

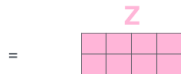


2) Multiply with a weight matrix W^O that was trained jointly with the model

X

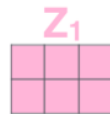
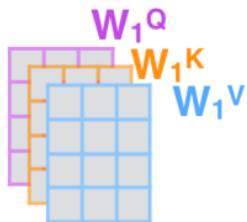
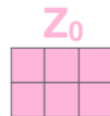
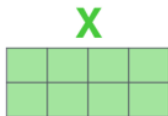


3) The result would be the Z matrix that captures information from all the attention heads. We can send this forward to the FFNN



Encoder의 Attention Layer 내부 과정

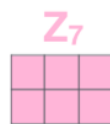
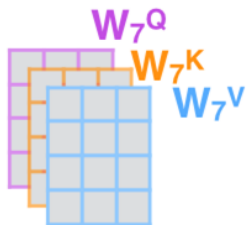
Thinking
Machines



...

...

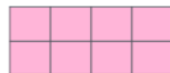
...



W^O



Z



* In all encoders other than #0,
we don't need embedding.
We start directly with the output
of the encoder right below this one

Code Review

전처리

```
#title과 content를 구분자 공백으로 합친 뒤 content열에 삽입
df["content"] = df["title"] + " " + df["content"]
#사용하지 않는 열 n_id / title / date / ord 삭제
df = df.drop(['n_id', 'title', 'date', 'ord'], axis=1)
#중복 행 제거
df = df.drop_duplicates()
#TSVDataset 적용시 공백으로 인한 에러 제거
df["content"] = df["content"].apply(lambda x: ' '.join(x.strip().split()))
#train : test = 90 : 10 비율으로 분할
trainset, testset = train_test_split(df, test_size=0.1, random_state=20)
```

1단계 :

Title과 Content를 구분자와 함께 합친 뒤 Content열에 삽입합니다.

2단계 :

이후 둘을 합쳐도 중복되는 행을 삭제합니다.

데이터 형태 변환

```
trainset.to_csv(path_or_buf='/content/drive/My Drive/trainset.csv',sep='\t',line_terminator='\r', encoding = 'utf-8-sig')
#trainset.head()
```

```
testset.to_csv(path_or_buf='/content/drive/My Drive/testset.csv',sep='\t',line_terminator='\r', encoding = 'utf-8-sig')
#testset.head()
```

#TSVDataset 함수에는 csv파일이 아닌 txt파일이 적용 가능하므로 파일 형식 변경부분

#TSVDataset에 적용할 txt파일의 경로 지정

```
trainset_txt = '/content/drive/My Drive/news_train.txt'
```

```
testset_txt = '/content/drive/My Drive/news_test.txt'
```

#csv파일을 txt파일로 변경

```
with open(trainset_txt, "w+") as my_output_file:
    with open('/content/drive/My Drive/trainset.csv', "r") as my_input_file:
        [my_output_file.write(" ".join(row)+'\n') for row in csv.reader(my_input_file)]
    my_output_file.close()
with open(testset_txt, "w+") as my_output_file:
    with open('/content/drive/My Drive/testset.csv', "r") as my_input_file:
        [my_output_file.write(" ".join(row)+'\n') for row in csv.reader(my_input_file)]
    my_output_file.close()
```

1단계 :

Train을 90%, Test를 10%로 나눈 뒤
csv파일로 저장합니다.

2단계 :

TSVDataset 함수 적용을 위하여 csv파일을
다시 txt파일로 변환합니다.

```
#TSVDataset
dataset_train = nlp.data.TSVDataset(trainset_txt, field_indices=[1,2], num_discard_samples=1)
dataset_test = nlp.data.TSVDataset(testset_txt, field_indices=[1,2], num_discard_samples=1)
```

Tokenizer

```
tokenizer = get_tokenizer()  
tok = nlp.data.BERTSPTokenizer(tokenizer, vocab, lower=False)
```

nlp.data.BERTSPTokenizer을 이용하여 어절단위로 데이터를 분리합니다.

Dataset

```
class BERTDataset(Dataset):
    def __init__(self, dataset, sent_idx, label_idx, bert_tokenizer, max_len,
                 pad, pair):
        transform = nlp.data.BERTSentenceTransform(
            bert_tokenizer, max_seq_length=max_len, pad=pad, pair=pair)

        self.sentences = [transform([i[sent_idx]]) for i in dataset]
        self.labels = [np.int32(i[label_idx]) for i in dataset]

    def __getitem__(self, i):
        return (self.sentences[i] + (self.labels[i], ))

    def __len__(self):
        return (len(self.labels))
```

Classifier에 들어갈 입력 값의 크기는 고정되어야 합니다.

따라서 BERTDataset class에서 transform 함수를 이용하여 padding 처리합니다.

BERT모델 입력에 포함되어야 할 attention_mask, token_ids등을 이 class에 정의하여 이용합니다.

Dataloader

```
train_dataloader = torch.utils.data.DataLoader(data_train, batch_size=batch_size, num_workers=5)
test_dataloader = torch.utils.data.DataLoader(data_test, batch_size=batch_size, num_workers=5)
```

Pytorch의 DataLoader을 이용하여 batch 단위로 나누어 데이터를 처리합니다.

Fine-Tuning

```
class BERTClassifier(nn.Module):
    def __init__(self,
                  bert,
                  hidden_size = 768,
                  num_classes=2,
                  dr_rate=None,
                  params=None):
        super(BERTClassifier, self).__init__()
        self.bert = bert
        self.dr_rate = dr_rate

        self.classifier = nn.Linear(hidden_size , num_classes)
        if dr_rate:
            self.dropout = nn.Dropout(p=dr_rate)

    def gen_attention_mask(self, token_ids, valid_length):
        attention_mask = torch.zeros_like(token_ids)
        for i, v in enumerate(valid_length):
            attention_mask[i][:v] = 1
        return attention_mask.float()

    def forward(self, token_ids, valid_length, segment_ids):
        attention_mask = self.gen_attention_mask(token_ids, valid_length)

        _, pooler = self.bert(input_ids = token_ids, token_type_ids = segment_ids.long(), attention_mask = attention_mask.float().to(token_ids.device))
        if self.dr_rate:
            out = self.dropout(pooler)
        return self.classifier(out)
```

분류 Task 적용을 위하여
BERT 마지막에 Linear layer을 이용한
Classifier을 만들었다.

Parameter

`max_len = 128`

`batch_size = 64`

`warmup_ratio = 0.01`

`num_epochs = 5`

`max_grad_norm = 1`

`log_interval = 200`

`learning_rate = 5e-5`

- padding을 위한 length설정
- 한 학습 시 64개씩 쪼개어 학습
- 과적합 방지를 위해 워밍업 단계를 설정 ,
warmup_ratio는 이 학습에서 이용될 learning_rate
- 총 5번의 BertModel학습
- 기울기를 1로 규제
- 한번의 epoch 당 200 batch id마다 train accuracy출력
- 학습률

Learning

```
for e in range(num_epochs):
    train_acc = 0.0
    test_acc = 0.0
    model.train()
    for batch_id, (token_ids, valid_length, segment_ids, label) in enumerate(tqdm_notebook(train_dataloader)):
        optimizer.zero_grad()
        token_ids = token_ids.long().to(device)
        segment_ids = segment_ids.long().to(device)
        valid_length = valid_length
        label = label.long().to(device)
        out = model(token_ids, valid_length, segment_ids)
        loss = loss_fn(out, label)
        loss.backward()
        torch.nn.utils.clip_grad_norm_(model.parameters(), max_grad_norm)
        optimizer.step()
        scheduler.step() # Update learning rate schedule
        train_acc += calc_accuracy(out, label)
        if batch_id % log_interval == 0:
            print("epoch {} batch id {} loss {} train acc {}".format(e+1, batch_id+1, loss.data.cpu().numpy(), train_acc / (batch_id+1)))
    print("epoch {} train acc {}".format(e+1, train_acc / (batch_id+1)))
    model.eval()
    for batch_id, (token_ids, valid_length, segment_ids, label) in enumerate(tqdm_notebook(test_dataloader)):
        token_ids = token_ids.long().to(device)
        segment_ids = segment_ids.long().to(device)
        valid_length = valid_length
        label = label.long().to(device)
        out = model(token_ids, valid_length, segment_ids)
        test_acc += calc_accuracy(out, label)
    print("epoch {} test acc {}".format(e+1, test_acc / (batch_id+1)))
```

1단계 : Learning step(warmup)

초기 과적합 방지를 위하여 매우 작은 학습률($5e-5$)부터 시작합니다.

이후 가중치가 안정화 될 수 있도록 설정해 놓은 학습률(0.01)까지 도달하도록 학습합니다.

2단계 :

학습된 모델에 대한 Test(news_train 데이터셋의 10%) Accuracy를 확인합니다.

Prediction

```
model.eval()
def model_pred(arg_list):
    for batch_id, (token_ids, valid_length, segment_ids, label) in enumerate(tqdm_notebook(test_dataloader)):
        token_ids = token_ids.long().to(device)
        valid_length = valid_length
        segment_ids = segment_ids.long().to(device)
        label = label.long().to(device)
        pred_var = model(token_ids, valid_length, segment_ids)
        _, predict = torch.max(pred_var, 1)
        arg_list.extend(predict.tolist())
```

1단계 :

토큰, 세그먼트, 라벨 이 세 개의 인자 값을 모델에 넣고 예측한다.

2단계 :

예측 값을 리스트에 넣어 확장한다.

참고자료

인터넷 링크

(<http://jalammar.github.io/illustrated-transformer/>) p42~p49 자료 이미지 스크랩

(<https://nlpinkorean.github.io/illustrated-transformer/>)

(<https://velog.io/@jinml/BERT>)

(<https://medium.com/platfarm/%EC%96%B4%ED%85%90%EC%85%98-%EB%A9%94%EC%BB>)

(<https://medium.com/platfarm/%EC%96%B4%ED%85%90%EC%85%98-%EB%A9%94%EC%BB-%A4%EB%8B%88%EC%A6%98%EA%B3%BC-transformer-self-attention-842498fd3225>)

도서

(텐서플로 2와 머신러닝으로 시작하는) 자연어 처리 : 로지스틱 회귀부터 BERT와 GPT2까지 / 저자 전창욱

저작권

팀 로고 : 미리캔버스 (<https://www.miricanvas.com/>) _ By zhezu

[SKTBrain koBERT \(https://github.com/SKTBrain/KoBERT\)](https://github.com/SKTBrain/KoBERT)

[\(http://jalammar.github.io/illustrated-transformer/\)](http://jalammar.github.io/illustrated-transformer/) _ p42~p49 자료 이미지 스크랩

NH투자증권 로고, 나무 어플리케이션의 아이콘은 모두 NH투자증권 (<https://www.nhqv.com/>)에서 가져왔습니다.

쿠나봇의 디자인은 Lovepik (<https://kr.lovepik.com/image-611549998/c4d-anthropomorphic-christmas-tree.html>)을 참고하였으며, 원본 제작자는 [自渡](#), 2차 창작자는 [정윙팡팡]입니다.

따로 언급한 바가 없는 이미지 / 아이콘은 Microsoft 365의 Power Point에서 기본으로 제공되는 것들을 사용하였습니다.