Text Analysis

김 화 종 (강원대학교)

hjkim3@gmail.com

참고자료

- 참고 도서
  - 파이썬, 딥러닝, 파이토치/이경택,방성수,안상준 (2020)
  - BERT와 GPT로 배우는 자연어처리/이기창 (2021)
  - 구글 BERT의 정석/수다르산 라비찬디란 (2021)

- 블로그
  - 이기창 (도서 예제)
    - https://ratsgo.github.io/nlpbook/
  - 딥 러닝을 이용한 자연어 처리 입문 (유원준, 안상준)
    - https://wikidocs.net/book/2155

내용

- 자연어 처리
- 전통적 방법
- 토픽 모델링
- 임베딩
- 트랜스포머
- BERT/GPT
- BERT 실습

자연어 처리

NLP 예

- 문서 분류
  - 스팸 필터링
  - 감성 분석
  - 이메일, 트위터, 블로그 등 문서의 타입을 분류
  - 작문 (에세이)늘 보고 잘 쓴 글인지 평가
  - 유사한 글이나 저자를 찾는 작업
  - 유사 논문이나 특허 찾기
  - 문서간의 연계성
  - 클러스터링
- 주요 정보 추출
  - 회사의 주가변동, 지명, 고유명사 등을 추출
  - 문서 요약 - 키워드 추출, 주제 요약

NLP 예

- 챗봇 (대화)
  - 사람과 대화하듯 동작
  - 룰 기반 방식과 신경망을 사용하여 자유롭게 대화하는 방식
- QA 시스템
  - 질문에 대답하는 서비스 (대한민국의 수도는? 오늘 날씨는? 등의 질문 적절한 답을 찾기
- 문장 완성
  - 가장 자연스러운 다음 문장 완성
  - 맞춤법 검사, 동의어 제안, 검색
- 기계 번역
  - 한-영, 한-일 등 임의의 언어간 번역

텍스트 소스

- 디지털 문서 – 위키피디아, 책, 논문, 보고서, 이메일
- 웹 데이터 - 크롤링
- SNS 데이터 – 페이스북, 트위터, 블로그
- 고객 데이터 – 사용자 데이터, 클레임
- 내부 데이터 – 운영, 유지보수 로그 데이터

문서 분류

- 문서의 긍정, 중립, 부정 등을 분류 (확률을 리턴)
- 문서의 카테고리를 구분
- 프리트레인을 마친 마스크 언어모델 (노란색)을 이용하여 분류를 수행 (초록색)

자연어 추론

- 문장 2개를 입력받아 두 문장 사이의 관계가 참(entailment), 거짓(contradiction), 중립(neutral) 등인지를 추론
- Pair classification

개체명 인식

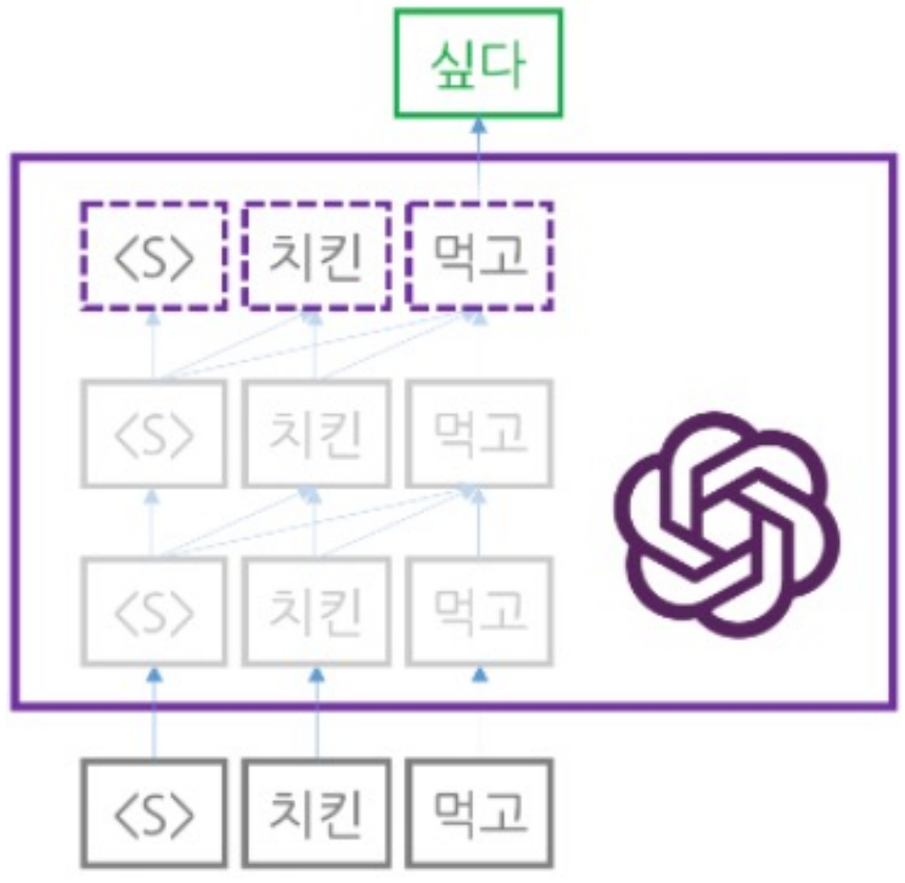
- 단어별로 기관명, 인명, 지명 등 어떤 개체명 범주에 속하는지를 분류
- Named Entity Recognition (NER)

질의응답 모델

- (질문+지문)을 입력받아 지문에서 정답의 범위를 찾는다
- Question Answering (QA)

문장 생성

- 자연어를 입력받고 다음에 나올 가장 확률이 높은 단어를 찾는다
- 문장 생성 모델은 GPT 계열이 BERT 보다 잘 동작한다
  - BERT는 문장의 의미를 추출하는 데 강점을 갖는다



NLP 주요 용어

- 코퍼스
  - 분석에 사용할 전체 문서 집합을 코퍼스 (말뭉치, corpus)
- 문서(document)
  - 한 단위의 텍스트를 말한다. 예를 들어 분석할 대상이 블로그 1천개면 1천개 블로그 집합은 코퍼스이고 각 블로그는 문서이다
- 토큰
  - 문장을 최소한의 의미 단위로 나눈 것 (단어, 띄어쓰기, subword, 음절 등)
- 어휘 집합
  - 토큰들의 집합 (사전)

NLP 발전

- 1990년대까지는 사람이 특성을 추출
  - 문법적인 해석 중심
  - 사람의 지식/문법/ 룰/ 통계적 판단을 프로그래밍으로 구현
- 2000년대 이후는 딥러닝 모델을 도입 (AI)
  - (지식이 아닌) 데이터 기반
  - 엔드투엔드 모델 (사람의 개입을 최소화)
  - 프리트레인과 파인튜닝 방식 도입
  - 멀티 태스크 모델 (한번 학습한 모델을 여러가지 목적에 사용)

→ 컴퓨터는 사람처럼 이해하는 것이 아니라 확률을 계산한다

전통적 방법

텍스트 분석 절차

- 데이터 확보
- 데이터 전처리 – 특성 추출, 표현법 선택
  - HTML 처리, 그림제거, 메타 데이터 활용, 사진 활용
  - 문장 구분(sentence splitiing) EOS: ‘ . ’,‘ ? ’, …
  - 토큰화- 의미를 갖는 최소정보 단위로 나누기
  - 불용어(Stop words) 처리, 어근/어간 추출
- 머신러닝 모델 구현
  - 목적 (분류, 클러스터링, 연관분석,추천)

전통적인 방법

- 음운론 (phonology)
  - 음소(phoneme) 단위를 추출하여 인코딩
- 형태론 (morphology)
  - 단어의 어형 변화를 연구
  - 형태소 분석(morphological analysis) (한글은 어렵다)
- 토큰화 (tokenizing)
  - 단어/서브워드/음절 등 단위로 구분
- 통사론(syntax)
  - 문법기반의 문장 구조 분석
  - 언어의 모호성으로 어렵다 (ex:Time flies like an arrow)
- 의미론(semantics)
  - 문장의 의미를 파악

문장 분석 Lexical Analysis

- 형태소(Morpheme) 단위의 분석
- 품사 (POS) 구분
- 개체명 인식, NER (Named Entity Recognition) : 룰/사전기반
- 지시대명사 인식
- 명사구 인식
- 문장경계 구분
- 토큰화

Stemming, Lemmatization

- Stemming은 공통 부분을 찾는 것
- Lemmatization는 품사를 구분하는 것 (사전에 나오는 표현)

Word	Stemming	Lemmatization
Love	Lov	Love
Loves	Lov	Love
Loved	Lov	Love
Loving	Lov	Love
Innovation	Innovat	Innovation
Innovations	Innovat	Innovation
Innovate	Innovat	Innovate
Innovates	Innovat	Innovate
Innovative	Innovat	Innovative

토큰화

- 텍스트를 토큰화 하는 방법
  - 단어(word) 단위
  - 서브워드(subword) 단위
  - 음절(syllable) 단위
  - 글자 (character) 단위
  - n-gram 단위
    - n-gram이란 n개의 연속된 토큰들을 묶어 하나의 토큰으로 취급하는 방법

텍스트 표현법

- 컴퓨터가 인식할 수 있도록 토큰을 숫자로 바꾸는 방법
  - BoW (Bag of word)
    - 문장내에 등장한 단어 빈도를 표현
    - 원핫 인코딩 사용 (모든 단어간 내적이 0이 된다)
    - Stop words 처리 (영어는 500여개, 한글은 677개를 정의)
  - TF-IDF
    - Term Frequency – Inverse Document Frequency
  - 워드 벡터 (임베딩)
    - 분산 표현
    - 벡터를 사용하여 토큰의 위치와 방향성을 제공한다 (물리적인 의미를 포함)
    - Glove, fastText
  - 사전학습 언어모델 – ELMO, BERT, GPT

토픽 모델링

토픽 모델링

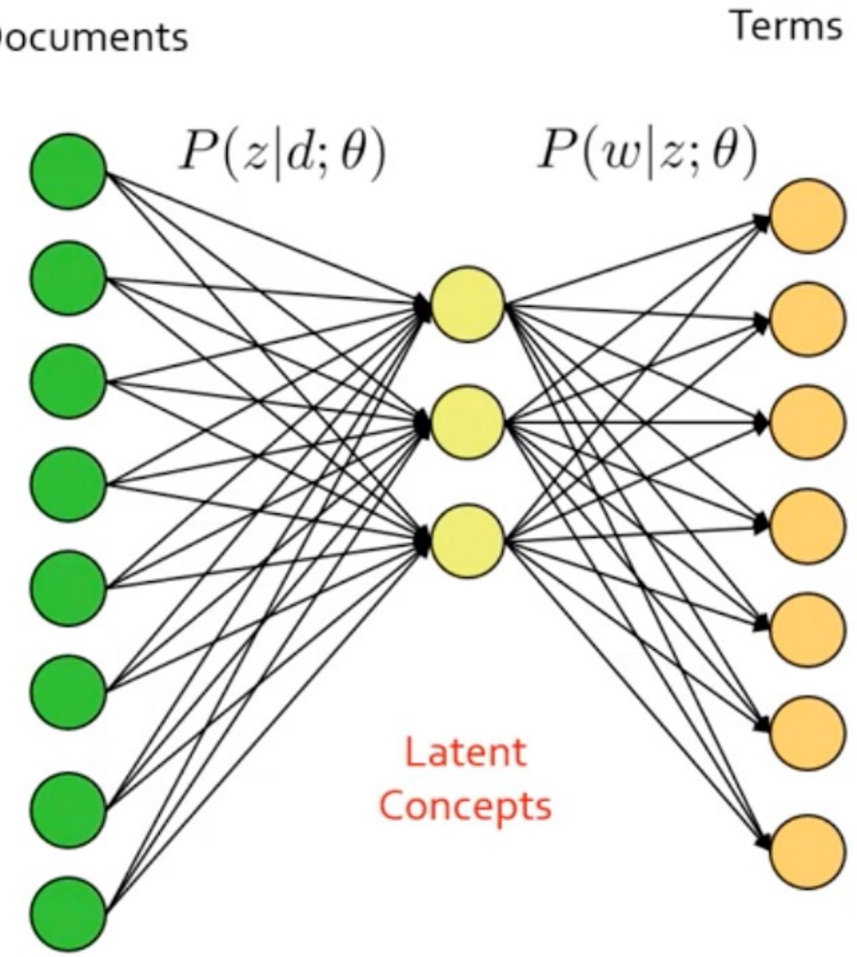
- 토픽 모델링은 주어진 문서의 주제(카테고리)늘 구분하는 것
- 잠재 디리클레 할당, LDA(Latent Dirichlet Allocation) 방법을 주로 사용
  - 관련성이 높은 단어들이 한 문서에서 자주 발생하면 같은 토픽으로 분류
  - 한 문서에는 여러 토픽이 복합적으로 존재할 수 있으며 각 토픽의 비중은 다를 수 있다.
  - 문서의 각 토픽들이 디리클레 분포를 따른다고 가정하고 각 문서를 각 토픽에 "할당"하는 방식으로 동작한다.
- 문서마다 토픽이 어떻게 분포되어 있는지, 그리고 토픽마다 단어의 분포가 어떤지 파악.
  - 토픽에 따라 단어의 분포를 결정하고 그중 가장 높은 확률의 단어를 선택한다.

토픽 모델링

- LDA 분석으로 얻은 결과는 비지도 학습이기 때문에 완벽한 정답은 아니다.
- 문서에 할당된 주제를 확인하고 검증하는 과정은 사람이 해주어야 한다.

토픽 모델링

- 토픽을 벡터로 표현하고 토픽간의 관계도 파악이 가능하다
- 유사한 저널 찾기
- SVD 방식 사용 (LSA: latent semantic analysis)
- DTM (Document-Term Matrix)에서 출발하므로 문장 구조 정보는 사라진다



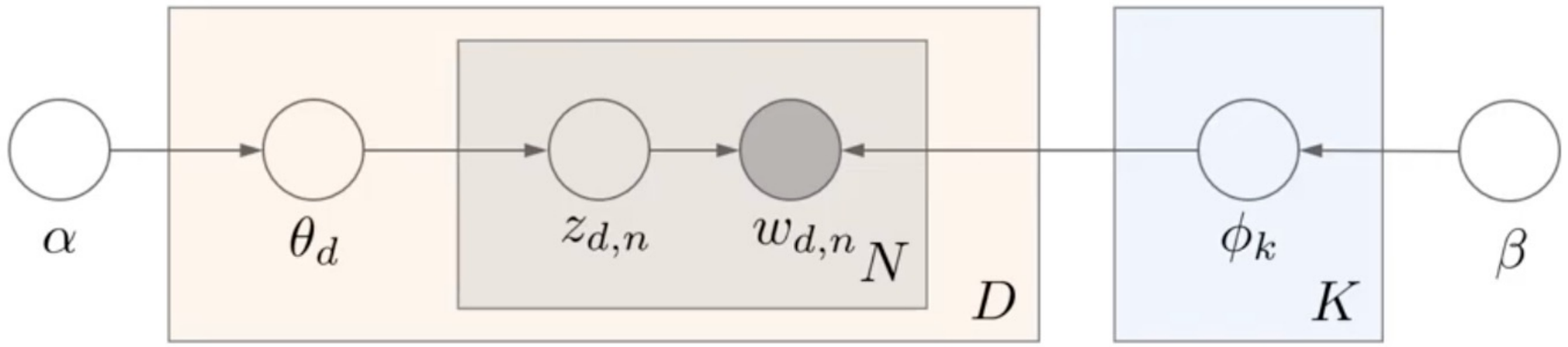
LDA

- Document –topic – word
  - 각 문서는 여러 토픽으로 구성된다
  - 각 토픽은 관련 단어들의 분포로 표현된다
  - 각 단어는 토픽으로부터 생성된다
- 관찰할 수 있는 것은 문서뿐이며 위의 "숨은" 관계를 찾는 작업
  - 토픽의 구성 비중을 찾는 작업
  - 각 토픽을 나타내는 단어들을 찾는 작업

(참고) LDA

- Dirichlet 분포 가정, 문서수 D, 토픽수 K, 단어수 N

• Document generation process

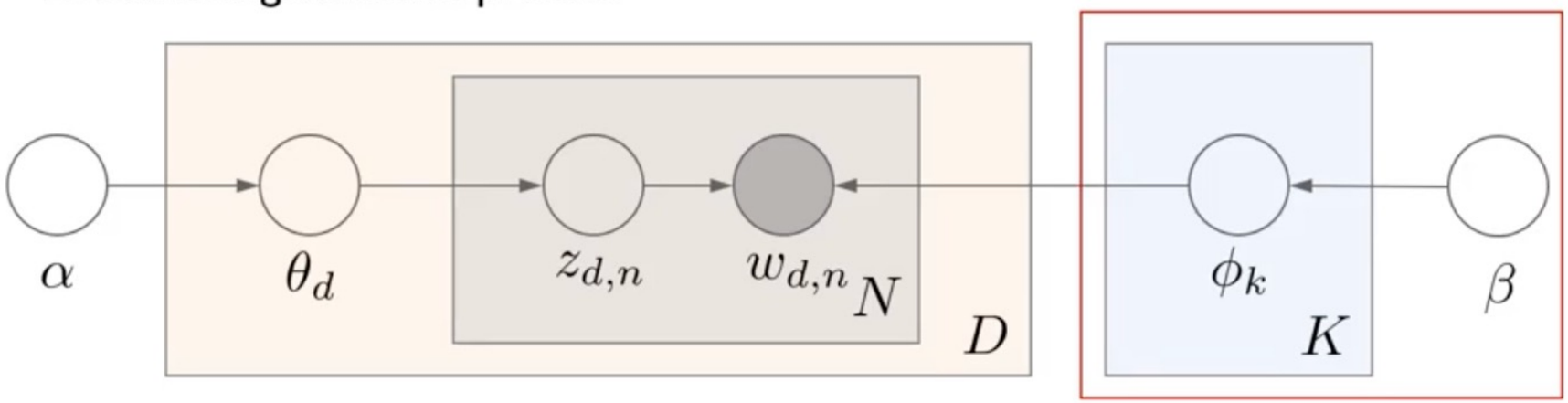


✓ Draw each topic $\phi_k \sim Dir(\beta)$ for $i \in \{1, ..., K\}$

✓ For each document

- Draw topic proportions $\theta_d \sim Dir(\alpha)$
- For each word
  - Draw $z_{d,n} \sim Multi(\theta_d)$
  - Draw $w_{d,n} \sim Multi(\phi_{z_{d,n},n})$

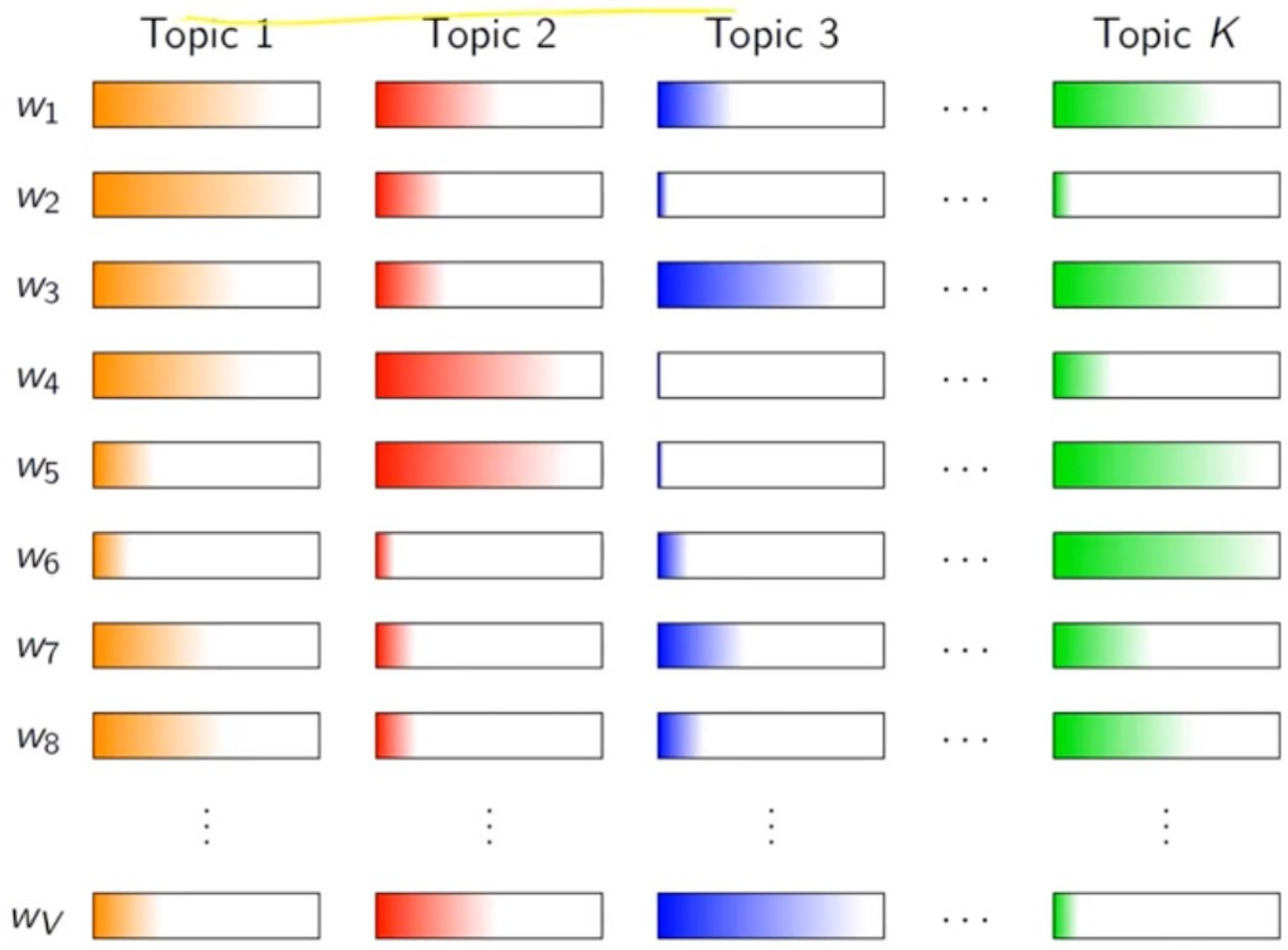
- Document generation process



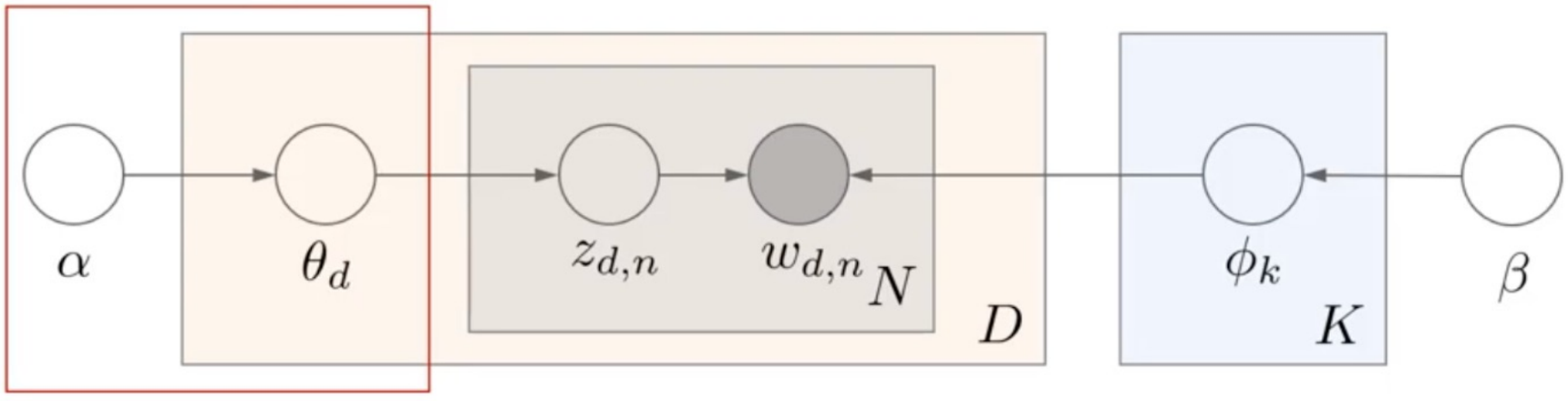
✓ Term distribution per topic

	Term 1	Term 2	Term 3	Term 4	Term 5=V
Topic 1 $\phi_{k=1}$	0.1	0.1	0	0.7	0.1
Topic 2 $\phi_{k=2}$	0.2	0.1	0.2	0.2	0.3
Topic 3 $\phi_{k=3}$	0.01	0.2	0.39	0.3	0.1
Topic 4 $\phi_{k=4=K}$	0.0	0.0	0.5	0.3	0.2

✓ Term distribution per topic



- Document generation process



✓ Topic distribution per document

		Topic 1	Topic 2	Topic 3	Topic 4
Document 1	$\theta_{d=1}$	0.5	0.1	0.3	0.1
Document 2	$\theta_{d=2}$	0.0	0.9	0.1	0.0
Document 3	$\theta_{d=3=D}$	0.02	0.48	0.25	0.25

차원 축소

- Feature selection
  - 빈도수, 정보이득, 크로스엔트로피 등을 사용
- Feature extraction
  - 차원수가 줄어야 한다, 매트릭스 축소, SVD, PCA
  - LDA: 문서의토픽을 구분하는데 어떤 단어들이 사용되었는지를 파악

(a) Per-document topic proportions (θ_d)

	Topic 1	Topic 2	Topic 3	...	Topic K	Sum
Doc 1	0.20	0.50	0.10	...	0.10	1
Doc 2	0.50	0.02	0.01	...	0.40	1
Doc 3	0.05	0.12	0.48	...	0.15	1
...	...	...	...	...	...	1
Doc N	0.14	0.25	0.33	...	0.14	1

(b) Per-topic word distributions (ϕ_k)

	Topic 1	Topic 2	Topic 3	...	Topic K
word 1	0.01	0.05	0.05	...	0.10
word 2	0.02	0.02	0.01	...	0.03
word 3	0.05	0.12	0.08	...	0.02
...	...	...	...	...	...
word V	0.04	0.01	0.03	...	0.07
Sum	1	1	1	1	1

단어 임베딩

언어 표현 방법

- 컴퓨터가 언어를 처리하도록 하려면 언어를 적절한 수치로 표현해야만 한다.
- 언어를 수치로 표현하는 방법으로 두 가지가 있다.
  - 단어 인덱싱 방식 (원핫인코딩 표현)
  - 단어 임베딩 방식 (벡터 표현)

원핫(one-hot) 인코딩

- 토큰에 고유 번호를 배정하여 사전을 만들고 모든 토큰의 고유번호 위치의 컬럼만 1로 표시하고, 나머지 컬럼은 모두 0인 벡터로 표시하는 방법이다.

- 텍스트:“어제 러시아에 갔다가 러시아 월드컵을 관람했다”
- 토큰 사전: {“어제”:0,“러시아”:1,“갔다”:2,“월드컵”:3,“관람”:4}
- 원핫 엔코딩:

어제 = [1, 0, 0, 0, 0]
러시아 = [0, 1, 0, 0, 0]
갔다 = [0, 0, 1, 0, 0]
월드컵 = [0, 0, 0, 1, 0]
관람 = [0, 0, 0, 0, 1]

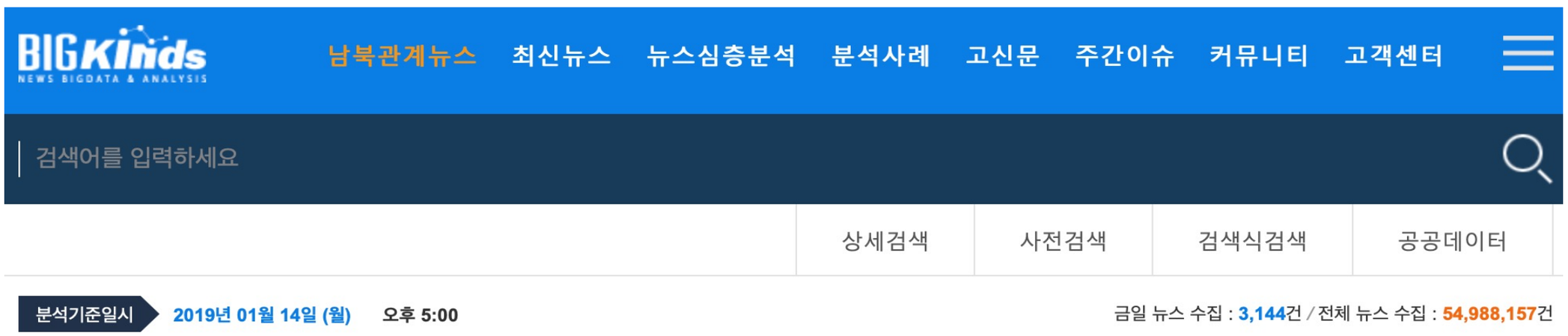
BOW(Bag of Word)

- "문장"을 벡터로 표현하는 방법
  - 아래는 임의의 6개의 문장을 BOW로 표현한 예이다. (5000 단어)
  - DTM (Document-Term Matrix)

토큰번호	0	1	2	3	4	5	6	7	8	9	10	...	4998	4999
Text_1	1	0	0	1	1	0	1	0	1	0	0	0	0	0
Text_2	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Text_3	0	0	0	1	0	0	0	0	0	0	0	0	0	0
Text_4	0	0	0	0	0	0	0	1	0	0	0	0	1	0
...														
Text_6	0	0	0	0	0	0	1	0	0	0	0	0	0	0

뉴스기사 분석

- 문서-단어 행렬의 예시를 위해 인터넷 뉴스기사를 한다
- 한국언론진흥재단에서 운영하는 빅카인즈(https://www.bigkinds.or.kr)



- "뉴스분석하기"늘 선택하면 카테고리 선택화면이 나타난다. 여기서 원하는 언론사, 주제, 사건 등을 선택하면 해당하는 뉴스기사 목록을 리스트 형태로 제공한다."다운로드" 버튼으로 현재 검색된 결과 뉴스 전체를 다운로드 할 수 있다.

단어 인덱싱

- 원핫 인코딩, BOW(단어모음), 문서-단어 행렬(DTM) 등에서는 단어마다 고유번호를 배정하여 사용
- 그러나 이 고유 번호 숫자에는 아무런 의미가 들어 있지 못하며 단지 인덱스(구분)의 성격만 갖는다.
- Dense Representation 라고도 한다

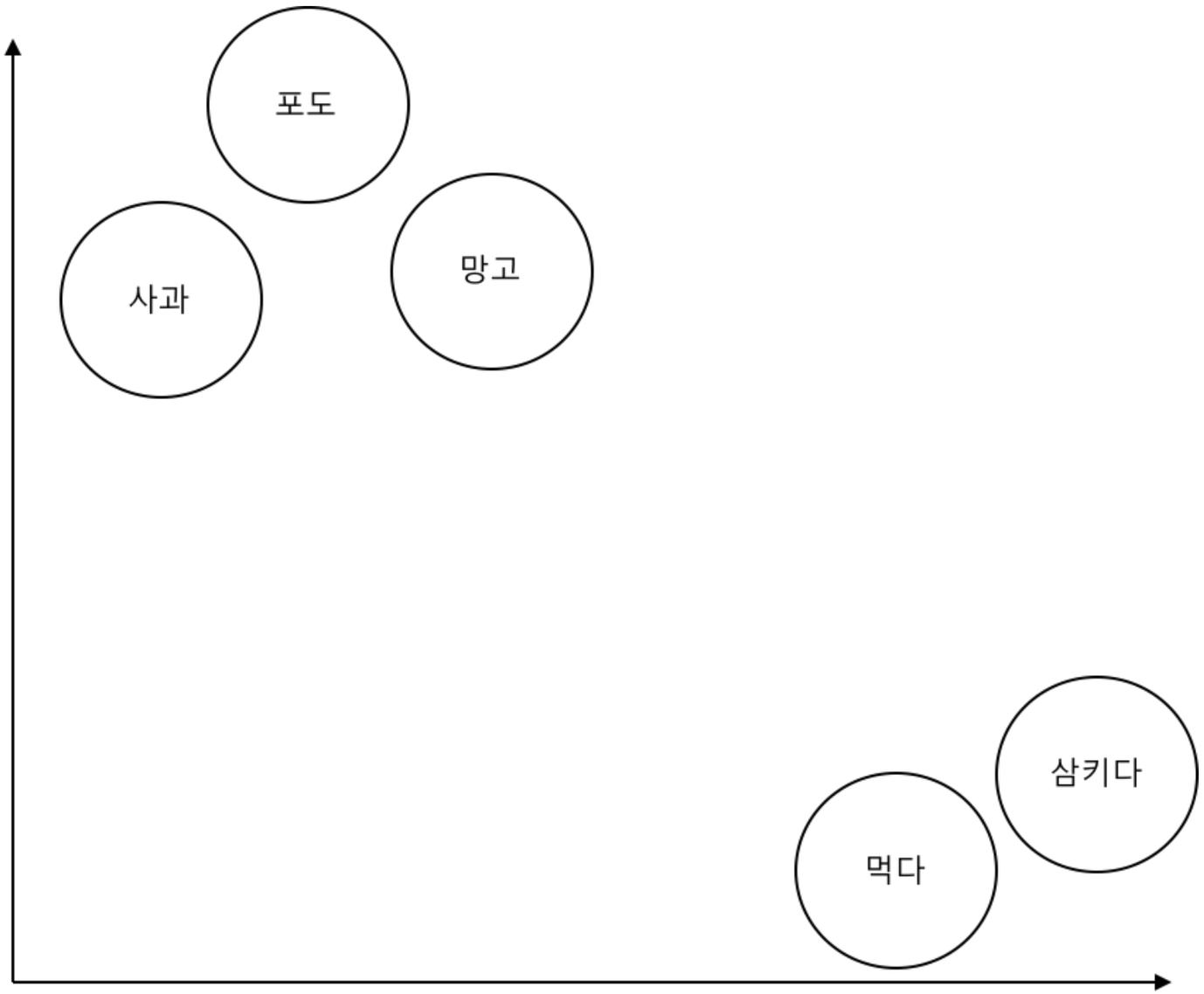
단어 임베딩

- 단어를 고차원 공간 상의 벡터로 표현함으로써 단어간 거리(유사도)와 방향성(벡터)늘 계산할 수 있게 되었다.
  - 코사인 유사도를 사용한다
- 차원이 높을수록 정교한 의미 구분이 가능하며 50~300개 정도의 차원을 사용한다.
  - 예를 들어 학교와 바다라는 단어를 단어 벡터로 표시하면 아래와 같이 보인다(아래 수치는 임의의 예시임)

    - 학교 = [0.23, 0.58, 0.97, … , 0.87, 0.95]
    - 바다 = [0.45, 0.37, 0.81, … , 0.22, 0.64]
- 단어 임베딩을 딥러닝의 입력으로 사용하여 성능 개선
  - 다른 곳에서 학습된 (만들어진) 임베딩을 활용할 수 있다
  - 전이학습을 사용

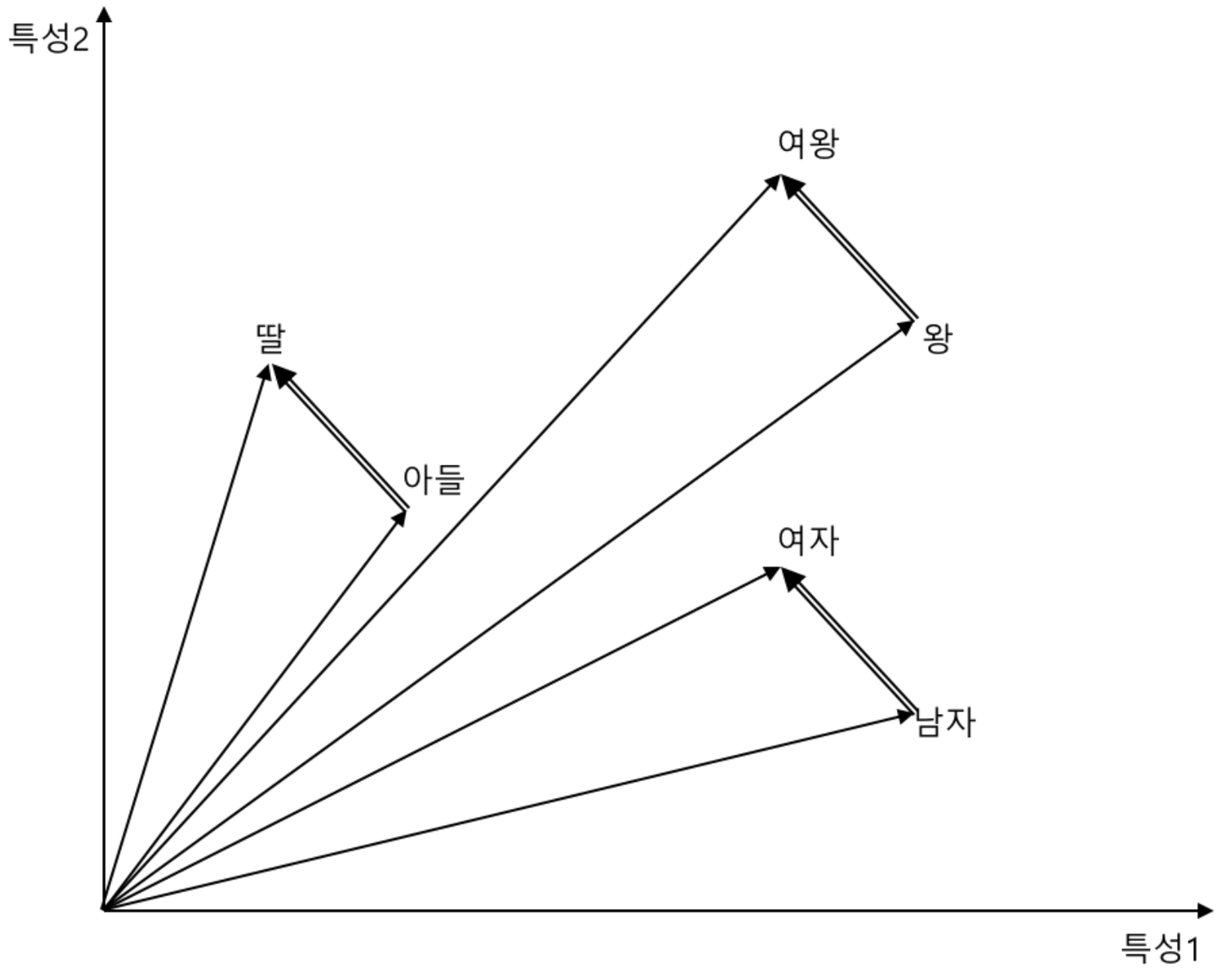
단어 임베딩 (벡터)

- 단어간의 거리와 방향을 계산할 수 있다



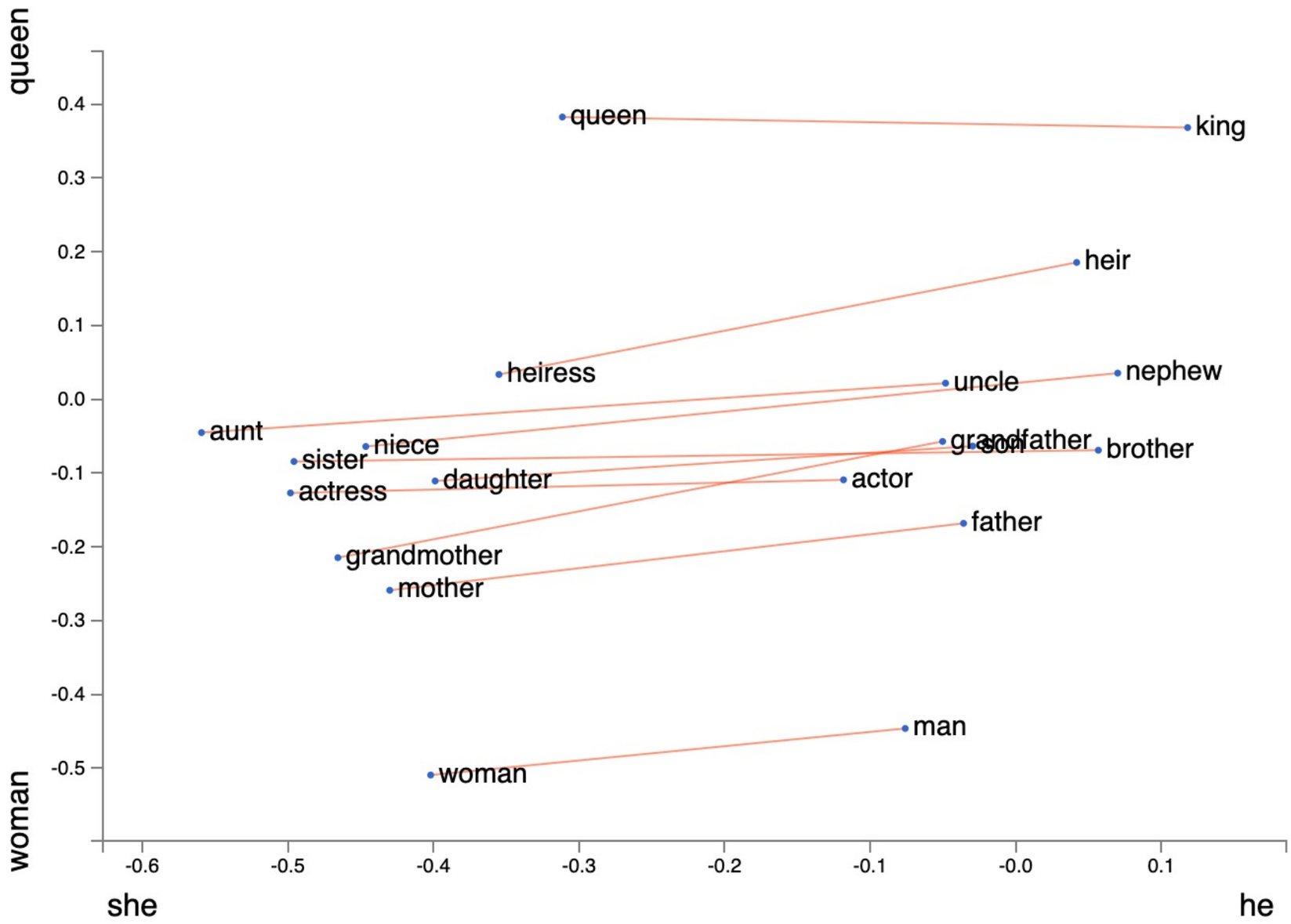
단어 임베딩 (벡터)

- 예를 들어 왕:여왕 = 아들:?에서 ? 부분이 딸이 될 것이다.



word2viz

- https://lamyiowce.github.io/word2viz/



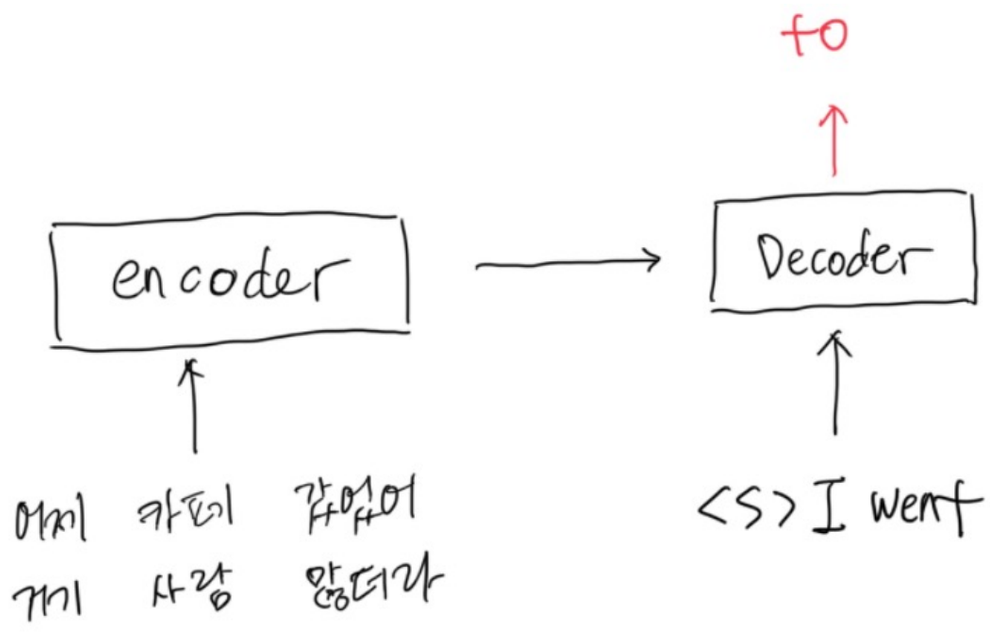
단어 임베딩/문장 임베딩

- 단어 수준 임베딩
  - Word2Vec, Glove, FastRwxr, Swivel 등
  - 주변의 단어와의 관계에서 임베딩 벡터를 생성
  - 동음 이의어를 분간하지 못하는 문제
  - 문맥 정보를 반영하지 못하는 한계가 있다
- 문장수준
  - ELMo (Embedding from Language Models)
  - Bert (Bidirectional Encoder Representation from Transformer)
  - GPT (Generative Pre-Training)

Transformer

트랜스포머

- 시퀀스-투-시퀀스(sequence-to-sequence) 모델 기반으로 동작
  - 2017년 구글이 제안
  - 대표적인 언어 모델에서 사용되고 있음 (BERT, GPT)
  - 인코더와 디코더로 구성되며 기계번역 등 다양한 목적으로 사용



언어 모델

- 언어 모델(Language Model)
  - 단어들이 주어졌을 때 다음 단어가 나타날 확률을 계산하는 모델
  - 문장이 그럴 듯 한지를 확률로 나타내는 것
  - 기계 번역, 맞춤법 교정, 문장 생성, 저자 스타일 구분 등에 사용
- 초기에는 통계적 기반, HMM을 사용했다
- 신경망기반의 언어모델 제안
  - 2003년 벤지오 교수가 처음 제안
  - 2017년 이후 트랜스포머가 언어 모델에 널리 사용되고 있음

언어 모델

- 순방향(forward) 언어 모델
  - GPT, ELMo 등

어제

어제 카페

어제 카페 갔었어

어제 카페 갔었어 거기

어제 카페 갔었어 거기 사람

어제 카페 갔었어 거기 사람 많더라

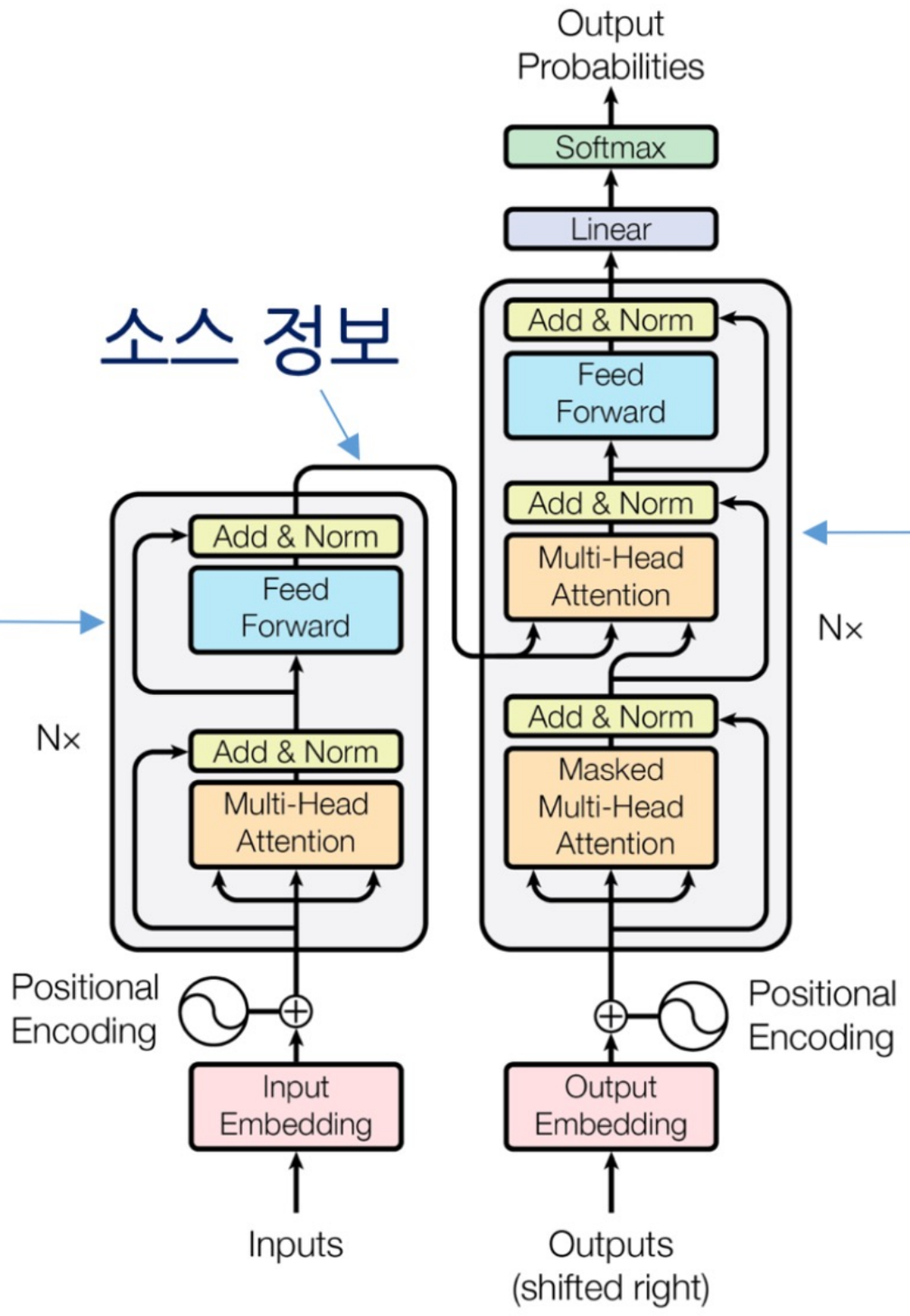
언어 모델

- 마스크 언어 모델(Maksed Language Model)
  - 문장 빈칸에 올 단어로 적절한 단어가 무엇일지 예측하며 학습 (예: BERT)
  - 양방향(bidirectional) 성질이 있다

어제	카페	갔었어	거기	사람	많더라
어제	**카페**	갔었어	거기	사람	많더라
어제	카페	**갔었어**	거기	사람	많더라
어제	카페	갔었어	**거기**	사람	많더라
어제	카페	갔었어	거기	**사람**	많더라
어제	카페	갔었어	거기	사람	**많더라**

트랜스포머

Encoder
소스 시퀀스 압축



Decoder
타겟 시퀀스 생성

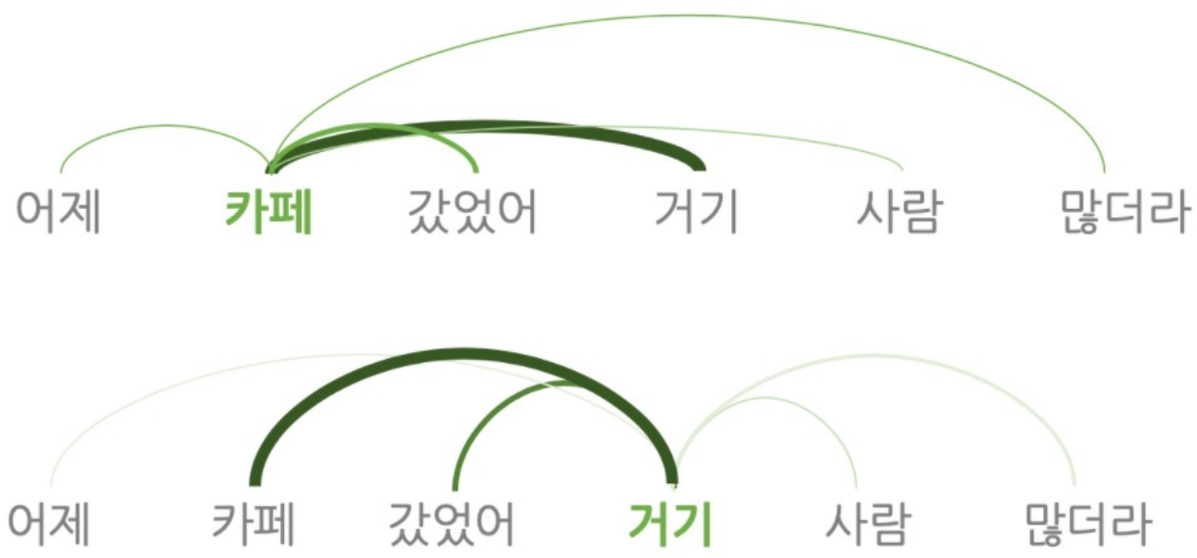
셀프 어텐션

- 멀티 헤드 어텐션은 셀프 어텐션(self attention)으로 구현된다
- 셀프 어텐션
  - 시퀀스 요소들 가운데 태스크 수행에 중요한 요소에 집중하고 그렇지 않은 요소는 무시해 태스크 수행 성능을 올리는 개념
  - 기계 번역에서 처음 도입
- 다른 딥러닝 모델과 비교
  - CNN은 합성곱 필터 크기를 넘어서는 문맥은 읽어내기 어렵다는 단점이 있다
  - RNN은 시퀀스 길이가 길어질수록 정보 압축에 문제가 발생한다

셀프 어텐션

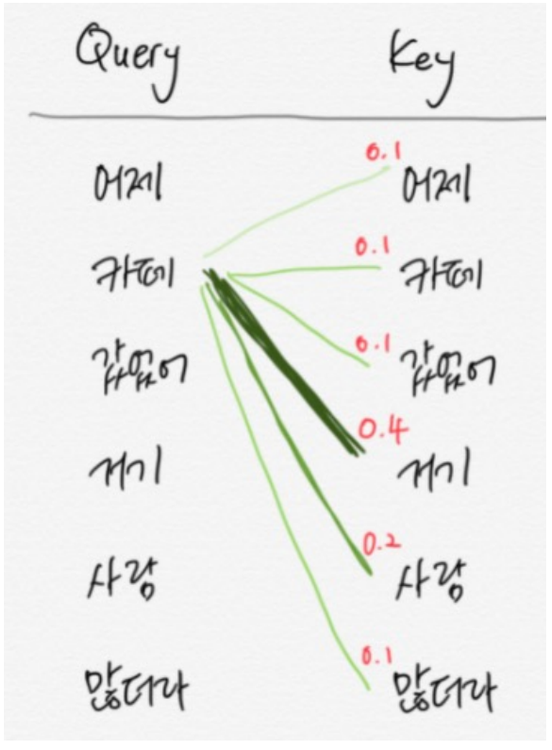
- (참고) 초기 어텐션: café를 디코딩할 때 주의해서 볼 내용은?
  - RNN 구조에서 동작 seq2seq

- 셀프 어텐션: 입력 자신 전체에 대해 수행하는 어텐션
  - CNN과 RNN의 단점을 개선 (RNN 없이 동작한다)



셀프 어텐션 계산

- 쿼리(query), 키(key), 밸류(value) 세 가지 요소를 사용
  - 쿼리 각 단어에 대해 모든 키 단어와 얼마나 유기적인 관계를 맺고 있는지 확률값으로 나타낸다
  - 셀프 어텐션은 이 결과에 밸류 벡터들을 가중합(weighted sum)하는 방식으로 계산한다 (아래에서 W가 학습대상임)



$$\mathbf{Q} = \mathbf{X} \times \mathbf{W}_\mathrm{Q}$$
$$\mathbf{K} = \mathbf{X} \times \mathbf{W}_\mathrm{K}$$
$$\mathbf{V} = \mathbf{X} \times \mathbf{W}_\mathrm{V}$$

$$\mathrm{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathrm{softmax}(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_\mathrm{K}}})\mathbf{V}$$

$$\mathbf{Z}_{\text{카페}} = 0.1 \times \mathbf{V}_{\text{어제}} + 0.1 \times \mathbf{V}_{\text{카페}} + 0.1 \times \mathbf{V}_{\text{갔었어}} + 0.4 \times \mathbf{V}_{\text{거기}} + 0.2 \times \mathbf{V}_{\text{사람}} + 0.1 \times \mathbf{V}_{\text{많더라}}$$

Q, K, V, Z

1) This is our input sentence*

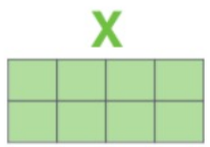
2) We embed each word*

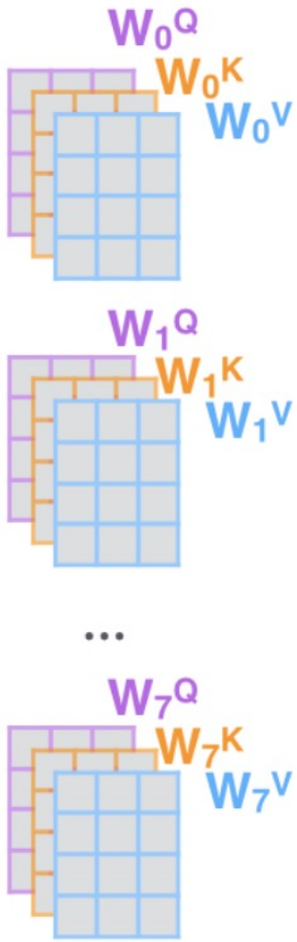
3) Split into 8 heads. We multiply X or R with weight matrices

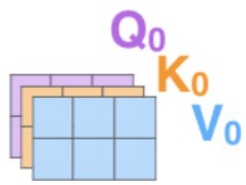
4) Calculate attention using the resulting Q/K/V matrices

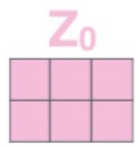
5) Concatenate the resulting Z matrices, then multiply with weight matrix W^O to produce the output of the layer

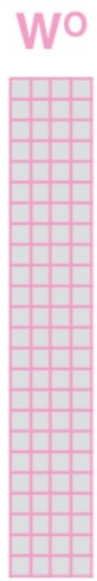
Thinking Machines

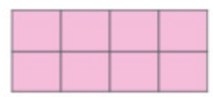












* In all encoders other than #0, we don't need embedding. We start directly with the output of the encoder right below this one

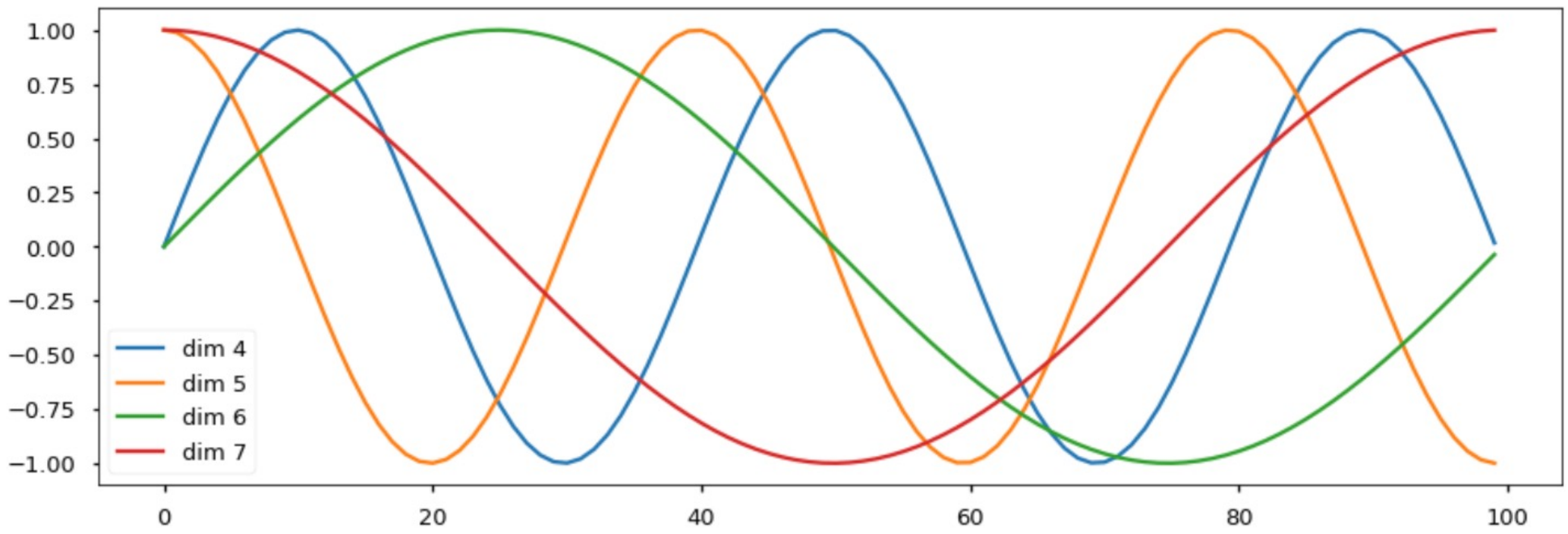
트랜스포머

- 셀프 어텐션
  - 벡터의 내적은 두 벡터가 얼마나 유사한지를 나타낸다
  - QK는 문장의 각 단어가 다른 단어들과 얼마나 유사한지를 파악
- 소프트맥스로 정규화를 수행한다
  - 확률로 표현된 스코어 행렬을 얻는다
- Z, 어텐션 행렬은 문장의 각 단어의 벡터값을 얻는다
- 멀티헤드 어텐션
  - 오분류가 일어날 확률을 줄인다
  - 멀티헤드 어텐션에 새로운 가중치 행렬을 곱해서 최종값을 얻는다
- 위치 인코딩
  - 병렬 입력을 가능하게 하여 학습 시간을 줄이고 RNN의 장기 의존성 문제를 해결한다

(참고) Positional Encoding

Positional Encoding

- Norm = 1
- Distance value between two points is proportional to real positional distance
- Why sine and cosine waves with different frequences?



Positional Encoding

	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10
X1	0.000	1.275	2.167	2.823	3.361	3.508	3.392	3.440	3.417	3.266
X2	1.275	0.000	1.104	2.195	3.135	3.511	3.452	3.442	3.387	3.308
X3	2.167	1.104	0.000	1.296	2.468	3.067	3.256	3.464	3.498	3.371
X4	2.823	2.195	1.296	0.000	1.275	2.110	2.746	3.399	3.624	3.399
X5	3.361	3.135	2.468	1.275	0.000	1.057	2.176	3.242	3.659	3.434
X6	3.508	3.511	3.067	2.110	1.057	0.000	1.333	2.601	3.169	3.118
X7	3.392	3.452	3.256	2.746	2.176	1.333	0.000	1.338	2.063	2.429
X8	3.440	3.442	3.464	3.399	3.242	2.601	1.338	0.000	0.912	1.891
X9	3.417	3.387	3.498	3.624	3.659	3.169	2.063	0.912	0.000	1.277
X10	3.266	3.308	3.371	3.399	3.434	3.118	2.429	1.891	1.277	0.000

BERT/GPT

BERT와 GPT

- GPT(Generative Pre-trained Transformer)
  - 언어모델(Language Model) 개념을 구현
  - 문장 시작부터 순차적으로 계산하는일방향(unidirectional)
- BERT(Bidirectional Encoder Representations from Transformers)
  - 마스크 언어모델(Masked Language Model)
  - 양방향(bidirectional) 성격을 가진다
- GPT는 문장 생성에, BERT는 문장의 의미를 추출하는 데 강점을 지닌다

BERT

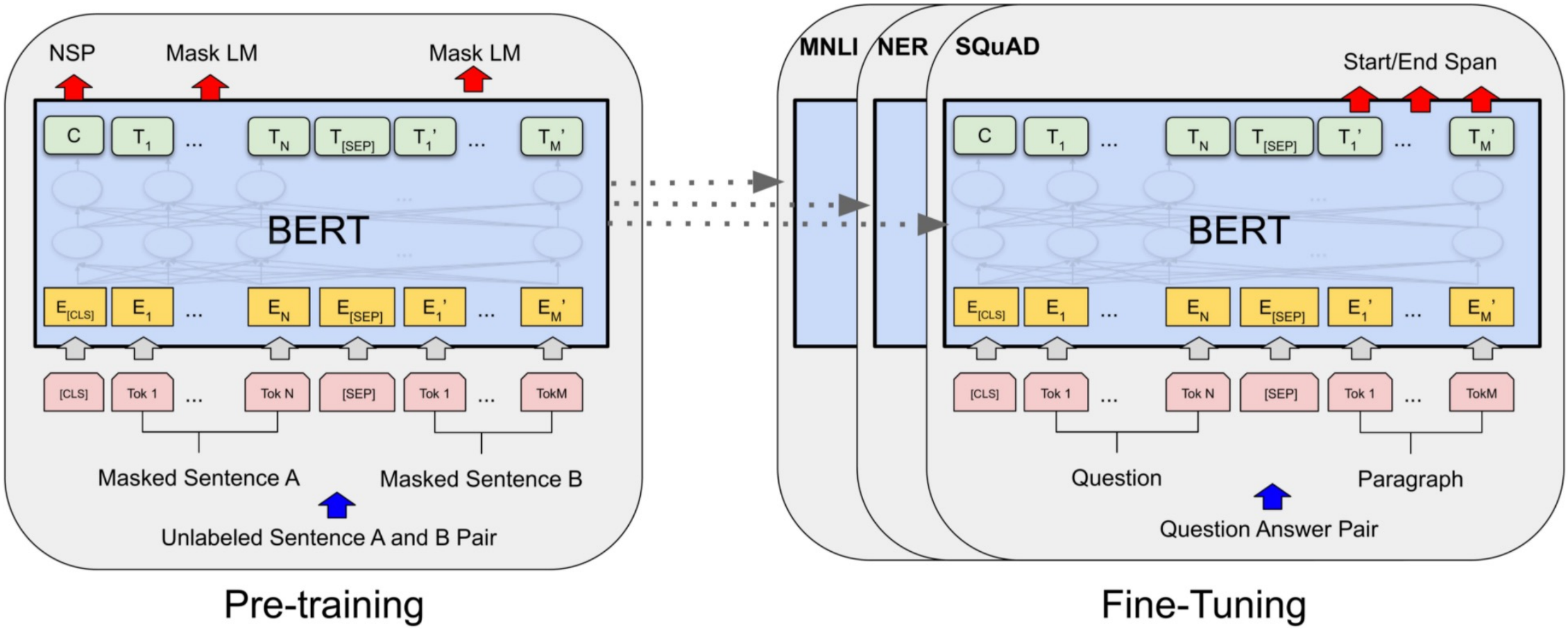
- BERT의 입력 표현
  - 토큰 임베딩
  - 세그먼트 임베딩
  - 위치 임베딩
- 활성화 함수로 GELU를 사용한다

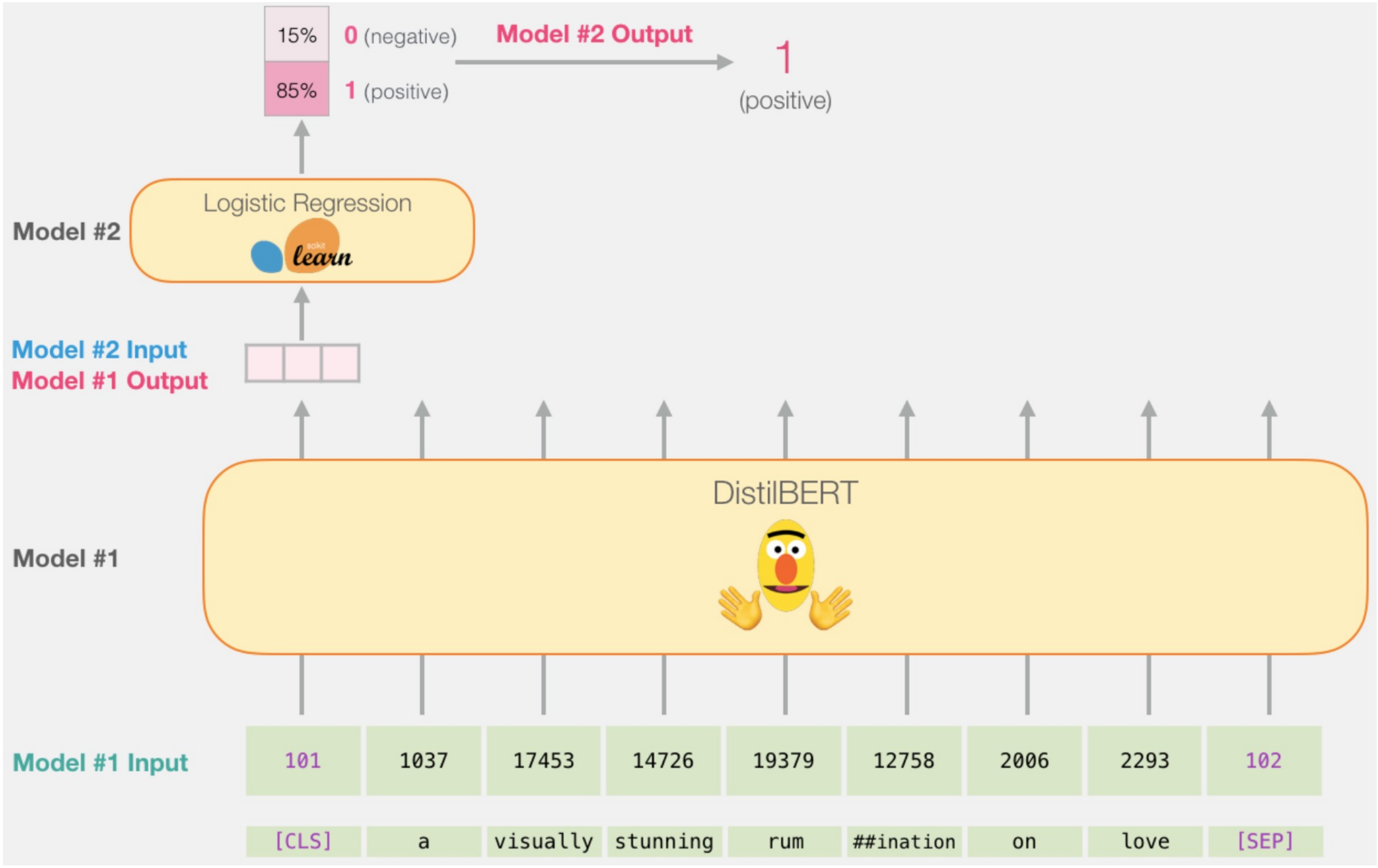
사전학습

- 두가지 학습
  - 마스크 언어 모델링 (MLM)
  - 다음 문장 예측 (NSP)
- 언어 모델링
  - 양방향인 자동 인코딩 언어 모델링 (BERT에서 사용) – 15%
    - 파인 튜닝에서는 [MASK] 가 없으므로 80:10:10 룰을 사용한다
    - 전체단어 마스킹 – 하위 단어와 관련된 모든 단어를 같이 마스킹한다
  - 단방향인 자동 회귀 언어 모델링

BERT

- Bidirectional Encoder Representations from Transformer (2018)
- Masked Language Model (MLM) – 15% (80%, 10%, 10% rule)
- Next Sentence Prediction (NSP) – 문장의 연속관계를 학습 (50%)




15%
0 (negative)
85%
1 (positive)
Model #2 Output
1
(positive)
Logistic Regression
Model #2
Model #2 Input
Model #1 Output
DistilBERT
Model #1
Model #1 Input
101
1037
17453
14726
19379
12758
2006
2293
102
[CLS]
a
visually
stunning
rum
##ination
on
love
[SEP]

BERT 파생모델

- ALBERT
  - Light version of BERT
  - 크로스 레이어 변수 공유 (예: 첫번째 레이어의 변수 사용)
  - 팩토라이즈 임베딩 레이어 변수화 (임베딩 행렬을 작게 분해)
    - VH 대신 VE – EH로 분해 (V:30000, H;768, E:128)
  - NSP 대신 문장 순서 예측을 사용한다
- RoBERTa
  - Robust optimized BERT approach
  - MLM에서 정적 매스킹이 아닌 동적 마스킹을 사용 (epoch 마다 달리)
  - NSP는 사용하지 않음
  - 배치 크기를 증가
  - BBPE 토크나이저 사용 (50000 토큰)

BERT 파생모델

- ELECTRA
  - 생성기와 판별기를 사용
  - 매스킹된 토큰을 교체한 후 이를 예측하는 방식으로 학습 (replaced token detection)
- SpanBERT
  - 질문 응답, 관계 추출과 같은 작업에 널리 사용된다
  - 마스킹할 토큰을 연속된 범위로 선택한다
  - 스팬 경계에 있는 토큰의 표현만 사용한다 (위치 임베딩 값을 사용)

DistilBERT

- 대형 모델의 동작을 재현하는 소형 모델을 학습시키는 모델 압축 기술 (60% 빠르고 40% 작다)
  - Teacher-student learning 방식
  - Dark knowledge를 학습에 사용하도록 소프트맥스에 temperature를 사용한다 (확률 분포를 평활화)
  - Distillation loss: 소프트 타깃과 소프트 예측 간의 교차 엔트로피 (T=5)
    - 소프트 타깃: 교사 네트워크의 출력
    - 소프트 예측: 학생 네트워크의 예측
- 학생 손실: 하드 타깃(실제값)과 하드 예측 (T=1) 간의 교차 엔트로피
  - 하드 타깃: 실제 레이블
  - 하드 예측: 학생 네트워크의 예측 (T=1)
- 손실함수
  - 증류 손실과 학생 손실의 가중합

BERT 파생모델

- TinyBERT
  - 교사의 출력 계층 뿐 아니라 임베딩 및 여러 인코더 레이어에서 지식을 전달한다 (은닉 상태 및 어텐션 행렬 사용)
  - 태스크 특화 증류
    - 파인 튜닝 단계에서도 증류를 적용한다
    - 파인 튜닝 단계에서 많은 데이터, 즉, 데이터 증식이 필요하다
      - 유사한 확률이 높은 후보 단어로 대체한다
      - 형태소 기반 단어 대체: 같은 품사로 대체한다
- BERTSUM
  - 텍스트 요약에 맞춰 파인 튜닝된 모델
    - 추출요약 – 문장의 표현을 사용한다
    - 생성요약 (의역) – 인코더-디코더를 사용한다
    - 성능평가: ROGUE-N : 실제요약과 예측간의 n-gram 재현율

BERT 파생모델

- M-BERT
  - Multilingual BERT
  - 104개 언어 위키피이아로 학습, 11만개 워드 피스 사용
- Sentence-BERT
  - 추론 시간을 줄이면서 문장쌍 분류, 문장 유사도 등에 널리 사용된다
- Clinical-BERT
  - 임상 말뭉치로 학습, 재입원 예측, 체류기간, 사망위험, 진단 예측 등
- Bio-BERT
  - PubMED 데이터로 학습
- Video-BERT
  - 영상과 언어의 표현을 동시에 배운다
  - 이미지 캡션, 비디오 캡션, 비디오 프레임 예측 등
  - 언어 토큰과 시각 토큰을 동시에 추출 (20 fps, 1.5 초 구간 토큰화)

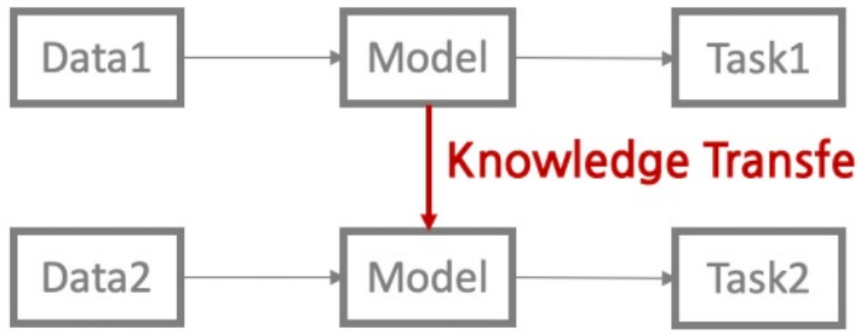
BERT 실습

BERT 기반 NLP

- 자연어 입력이 특정 범주일 확률을 얻는 모델을 BERT로 구현하는 실습
- NLP 범위
  - 문서 분류(document classification)
  - 질의 응답(question answering)
  - 개체명 인식(named entity recognition)
  - 문장 생성(sentence generation) 등
- 참고
  - BERT와 GPT로 배우는 자연어처리/이기창/이지 퍼브리싱 (2021)
  - (블로그) https://ratsgo.github.io/nlpbook/docs/tutorial_links

전이학습

- 다른 데이터로 학습한 모델을 다른 목적에 사용하는 기술
  - 데이터 부족시 성능을 높일 수 있다
- Task1
  - 업스트림(upstream) 태스크
  - 다음 단어 맞히기, 빈칸 채우기 등
  - 대규모 말뭉치의 문맥을 이해하는 과제
  - Self-supervise learning
  - 언어 모델링 사용
  - 프리트레인
- Task2
  - 다운스트림(downstream) 태스크
  - 문서 분류, 개체명 인식 등 자연어 처리의 구체적 문제들
  - 파인 튜닝 (fine tuning) 작업



토큰화

- 문장을 토큰 시퀀스로 나누는 과정
  - 문자, 단어, 서브워드(subword) 등 세 가지 방법이 있다.
- 한국어 토크나이저
  - BERT 이전에 주로 사용하던 은전한닢(mecab), 꼬꼬마(kkma) 등이 있으며 품사 부착(Part-Of-Speech Tagging)도 수행해준다

토크나이저

- 바이트 쌍 인코딩 (1994)
  - Byte pair encoding (BPE) –GPT에서 사용
  - 어휘 사전의 크기를 정해두고 가장 빈도수가 큰 기호 쌍을 식별한다
- 바이트 수준 바이트 쌍 인코딩
  - Byte level byte pair encoding
  - 문자 시퀀스가 아니라 바이트 수준의 시퀀스를 사용한다
  - 다국어 처리에서 유용하며 어휘 사전 이외의 단어 (out of vocabulary) 처리에 효과적이다 (여러 언어로 어휘 사전을 공유하기 좋다)
- 워드피스 토크나이저
  - Subword로 분할 – BERT에서 사용
  - BPE의 발생빈도 대신, 병합 가능도가 가장 높은 기호 쌍을 병합한다
  - OOV 단어를 처리하는데 효과적

실습 환경

- Colab 사용
- 하이퍼파라미터(hyperparameter) 설정

```
from ratsnlp.nlpbook.classification import
ClassificationTrainArguments
args = nlpbook.TrainArguments(
    pretrained_model_name="beomi/kcbert-base",
    downstream_corpus_name="nsmc",
    downstream_corpus_root_dir="/content/Korpora",
    downstream_model_dir="/gdrive/My
Drive/nlpbook/checkpoint-cls",
    learning_rate=5e-5,
    batch_size=32,
)
```

데이터 받기

- https://github.com/ko-nlp/korpora (한국어 말뭉치 사용)
- 네이버 영화평 말뭉치 (박은정)
  - NAVER Sentiment Movie Corpus(NSMC)
  - downstream_corpus_name(nsmc)에 해당하는 말뭉치를 코랩의 downstream_corpus_root_dir(/content/Korpora) 에 다운로드

```
from Korpora import Korpora
Korpora.fetch(
    corpus_name=args.downstream_corpus_name,
    root_dir=args.downstream_corpus_root_dir,
    force_download=True,
)
```

Kc-bert 모델 받기

- 허깅페이스 트랜스포머에 등록한 kcbert-base 모델 받기 (이준범)

```
from transformers import BertConfig,
BertForSequenceClassification
pretrained_model_config = BertConfig.from_pretrained(
    args.pretrained_model_name,
    num_labels=2,
)
model = BertForSequenceClassification.from_pretrained(
        args.pretrained_model_name,
        config=pretrained_model_config,
)
```

토크나이저 선택

- kcbert-base 모델이 사용하는 토크나이저를 선택

```
from transformers import BertTokenizer
tokenizer = BertTokenizer.from_pretrained(
    args.pretrained_model_name,
    do_lower_case=False,
)
```

파이토치 데이터 로더

- 데이터 로더
  - 학습을 하기 위해서 데이터를 배치(batch) 단위로 모델에 보내는 작업을 처리
- 컬레이트(collate)
  - 배치 단위로 맞추어 학습에 사용할 최종 입력을 만들어주는 과정
  - 파이토치에서 인식하는 텐서(tensor)로의 자료형 변환도 포함

태스크 정의

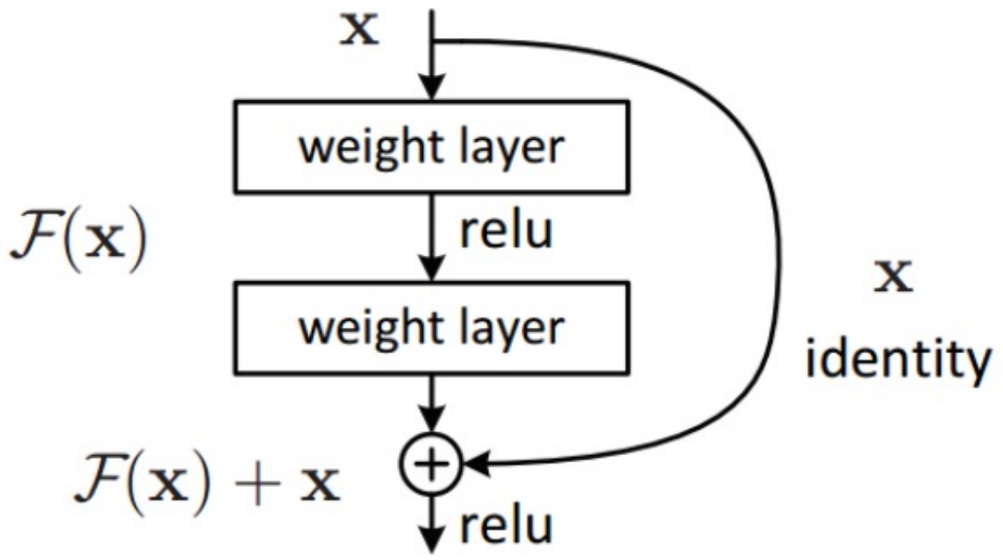
- 모델 학습을 할 때 파이토치 라이트닝(pytorch lightning) 라이브러리를 사용
  - 반복적인 내용을 대신 수행해 사용자가 모델 구축에만 신경쓸 수 있도록 돕는 라이브러리
- 라이트닝(lightning) 모듈을 상속받아 태스크(task)를 정의하며 다음의 기능을 수행한다
  - 모델 설정
  - 최적화 (learning rate 스케쥴링 포함)
  - 학습 과정 정의 (스텝 단위)

바이트 페어 인코딩

- BPE (Byte Pair Encoding)
  - GPT에서 채택한 방법이며 BERT에서는 이와 유사한 워드피스 단위를 사용한다
  - 빈도수가 높은 바이그램 쌍(merges.txt)을 병합하는 방식으로 토큰화를 수행하여 vocab.txt를 생성한다
  - 워드피스 방법은 빈도수가 아니라 우도(likelihood)를 기준으로 병합하며 미리 만든 어휘집합 (vocab.txt)만 가지고 수행한다

Residual Connection

- 잔차 연결은 모델 중간에 블록을 건너뛰는 경로를 설정함으로써 학습을 용이하게 한다

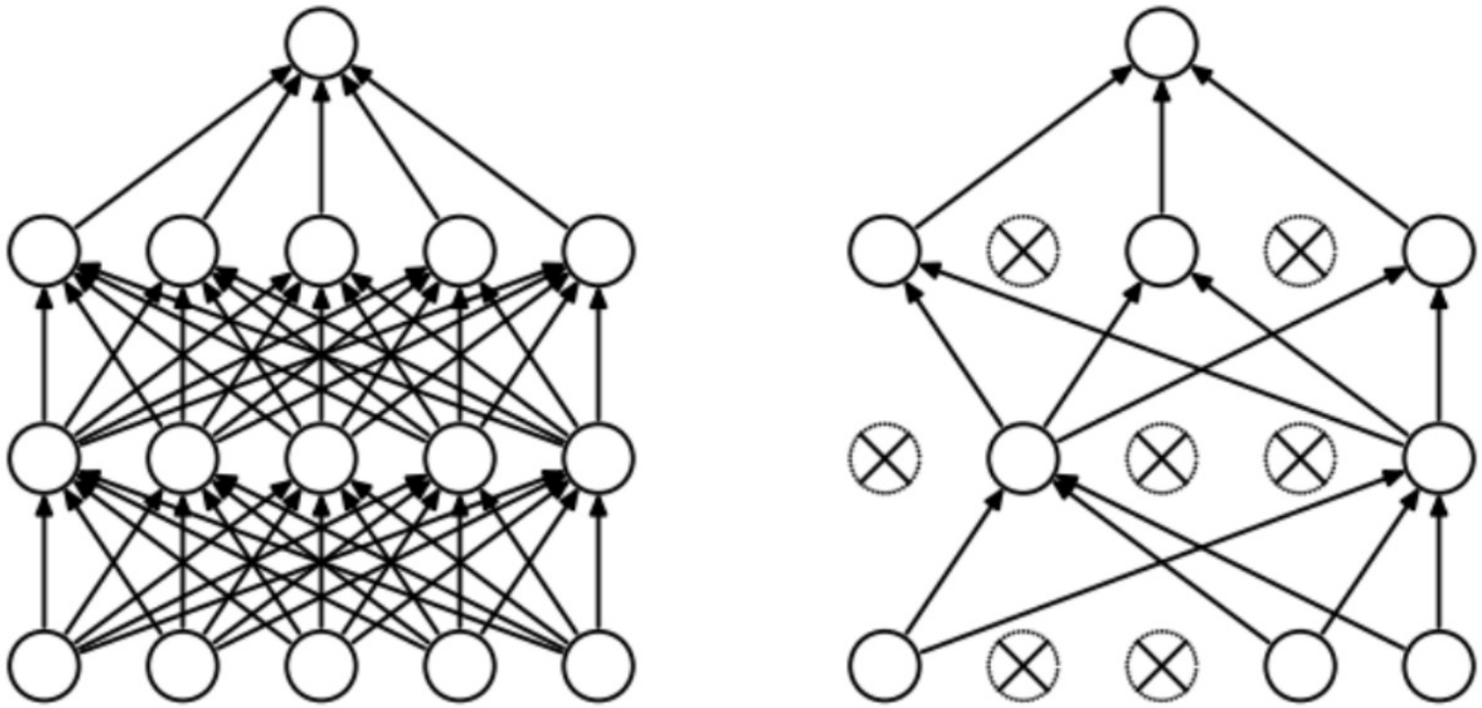


Batch Normalization

- Internal Covariant Shift
  - 각 레이어마다 분포가 달라짐에 따라 학습 속도가 느려지는 현상
  - 레이어 별로 입력의 분포를 정규화 하여 학습 속도를 개선
  - 비선형, 활성화 함수를 사용하는 목적을 달성하기 위해서도 필요 (입력 데이터의 확률 분포가 균등하게 분포되도록 한다)

드롭아웃

- 과대적합을 줄이는 방법으로 레이어 중간에서 임의로 특성값을 선택하여 전달하지 않는다 (0으로 만든다)

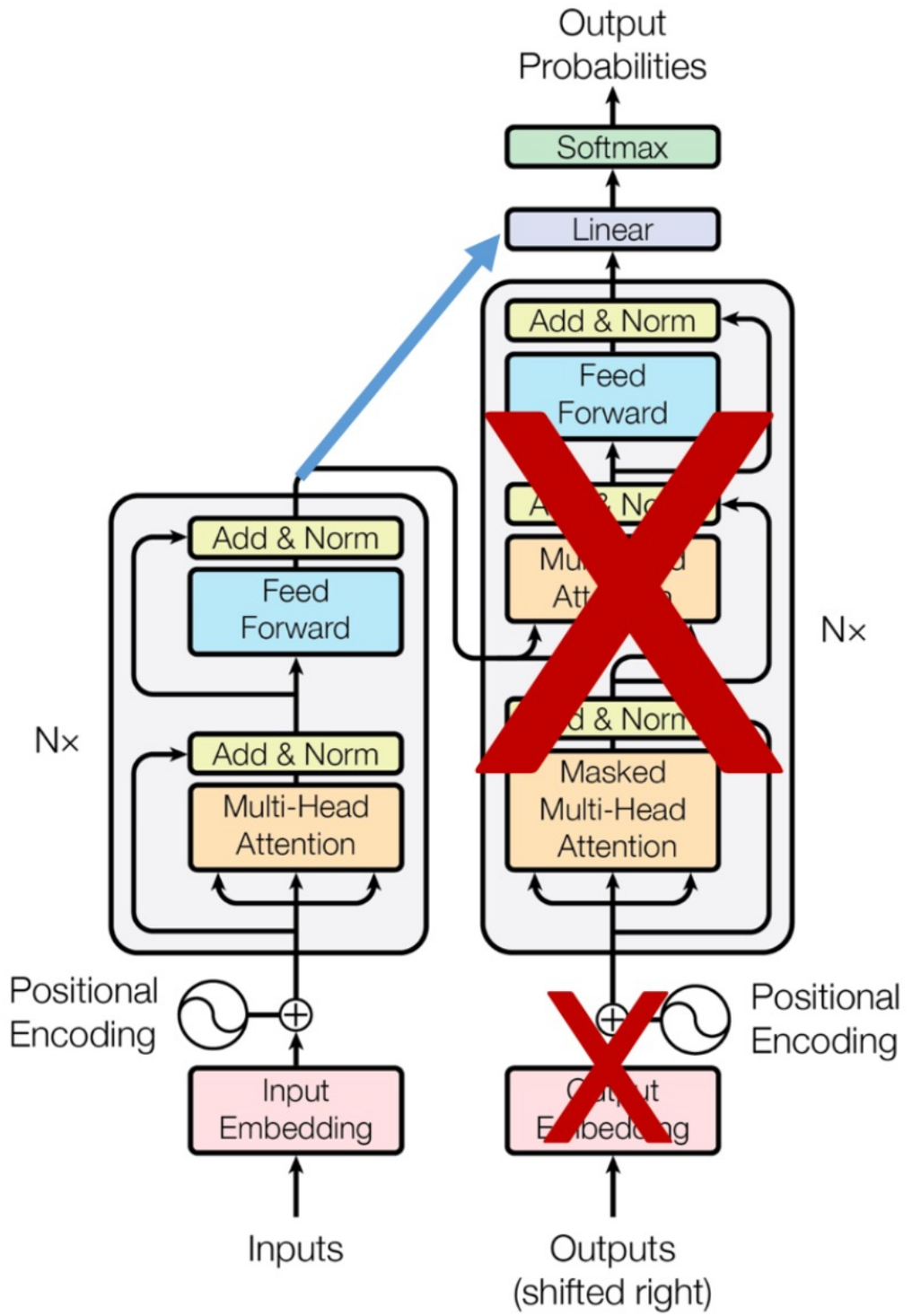


(a) Standard Neural Net

(b) After applying dropout.

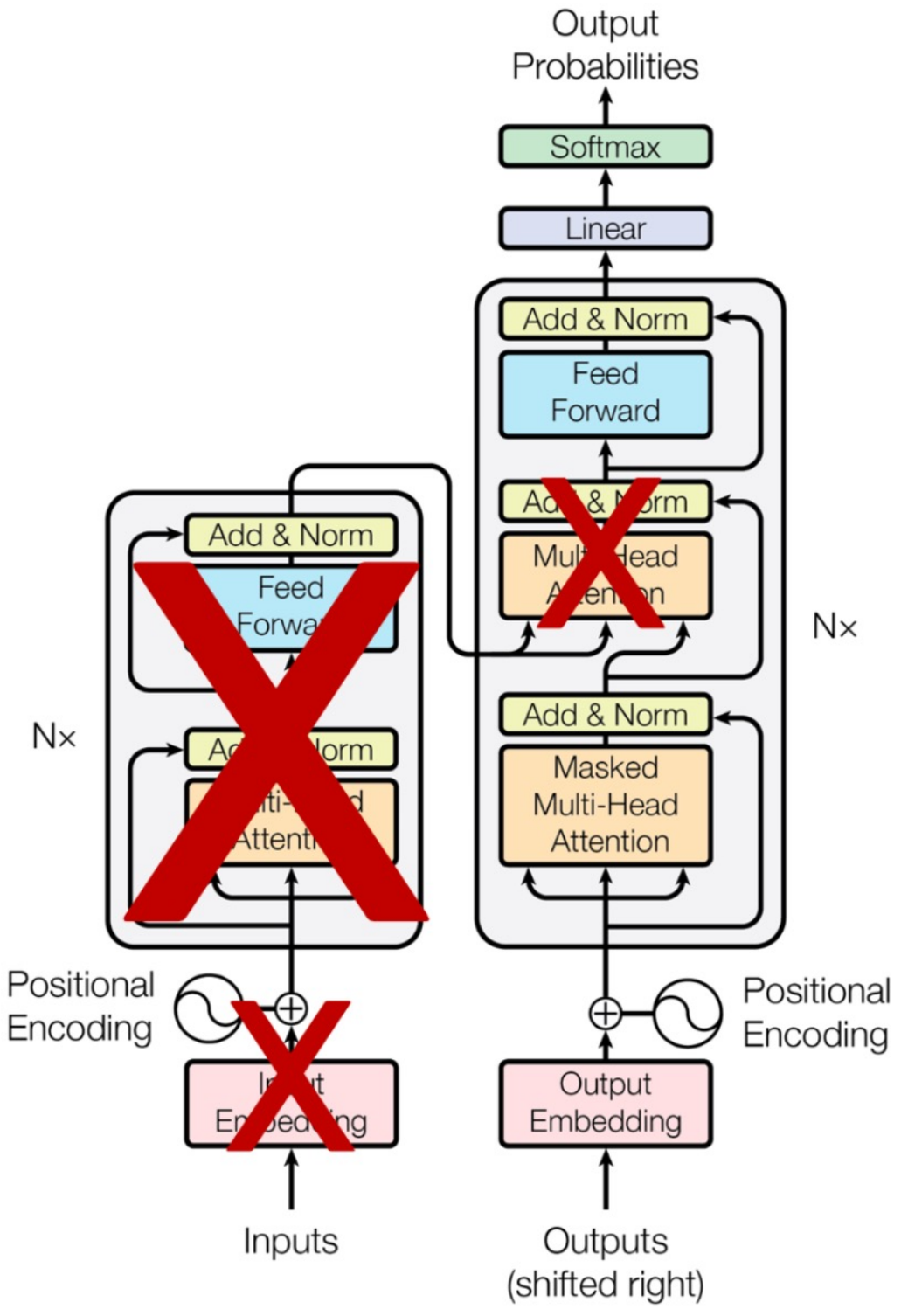
BERT

- 트랜스포머에서 디코더를 제외하고 인코더만 사용



GPT

- 트랜스포머에서 인코더를 제외하고 디코더만 사용

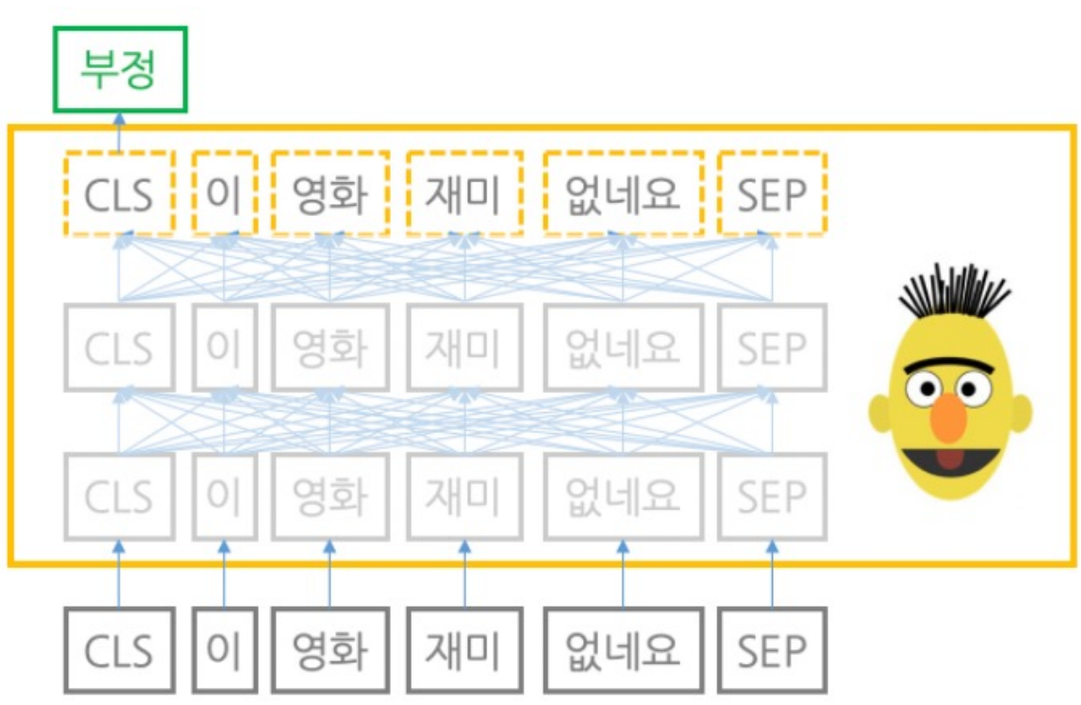


문서 분류

- 뉴스를 보고 범주(정치, 경제, 연예 등)를 맞추거나, 영화 리뷰가 어떤 극성(긍정/부정 등)을 가지는지 분류하는 작업
- 예: 영화 리뷰 (Naver Sentiment Movie Corpus, NSMC) 각 문장이 속한 범주 확률(긍정, 부정)를 출력
  - 진짜 짜증나네요 목소리 → [0.02, 0.98]
  - 너무재밌었다그래서보는것을추천한다 → [0.99, 0.01]

문장 임베딩

- 마지막 레이어의 [CLS] 벡터를 가공하여 확률을 얻는다
  - 문장 임베딩 (768,2) 크기의 벡터에 소프트맥스를 적용 (범주가 2)



CUDA

- CUDA
  - GPU를 각종 프로그래밍 언어에서 사용할 수 있도록 해주는 GPGPU의 일종 (General purpose computing on GPU)
- cuDNN
  - NVIDIA CUDA Deep Neural Network Library
  - 텐서플로우, 파이토치를 지원하며 CUDA와 함께 설치하여 사용
- 파이토치
  - 페이스북이 만든 딥러닝 개발 라이브러리