손에 잡히는 프로그래밍 언어 모델 구현

언어 모델 개발 튜토리얼

2023. 8. 23

류연희, 허기홍

KAIST Programming Systems Laboratory

배경 & 목표

- 프로그래밍**언어** 연구회
- 프로그래밍언어로 글을 읽고 쓰고 이해하는 두 가지 방법
  - 규칙 기반 (전통적): 파싱, 컴파일, 분석 등
  - 확률 기반 (장안의 화제): 언어 모델
- 목표:
  - 두 가지 방법을 적재적소에 활용하여 PL 문제 해결
  - 두 가지 방법의 장점을 결합한 새로운 프로그래밍 시스템 실현
- 진행: 류연희 (KAIST 박사과정, 안전한 프로그래밍 언어 모델 연구)
- 도움: 김태은, 김재호, 박종찬, 이동재 (KAIST)

목차

- 실습 목표: 언어 모델 구현 방법을 알고 깜깜이 API 에서 독립하기
- 자료: https://github.com/prosyslab/sigpl23-tutorial

시간	내용
13:30 – 14:45	• 프로그래밍 언어 모델 소개 • Colab 을 이용한 실습 환경 설정 • Transformers 라이브러리 사용해보기
14:45 – 15:00	• 휴식
15:00 – 16:15	• 오픈소스 거대 언어 모델의 생성 기능 이용해보기 • Transformers 라이브러리를 이용한 코드 수정 Fine-tuning 실습

시작하기 전에

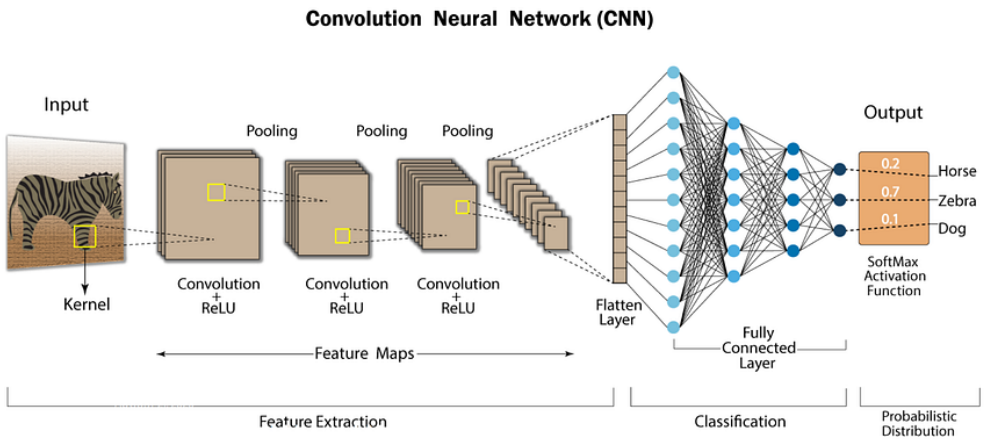
- 구글 Colab: https://colab.research.google.com/
  - Jupyter Notebook 환경
  - 구글이 제공해주는 무료 GPU 클라우드 환경 사용 가능!

프로그래밍 언어 모델

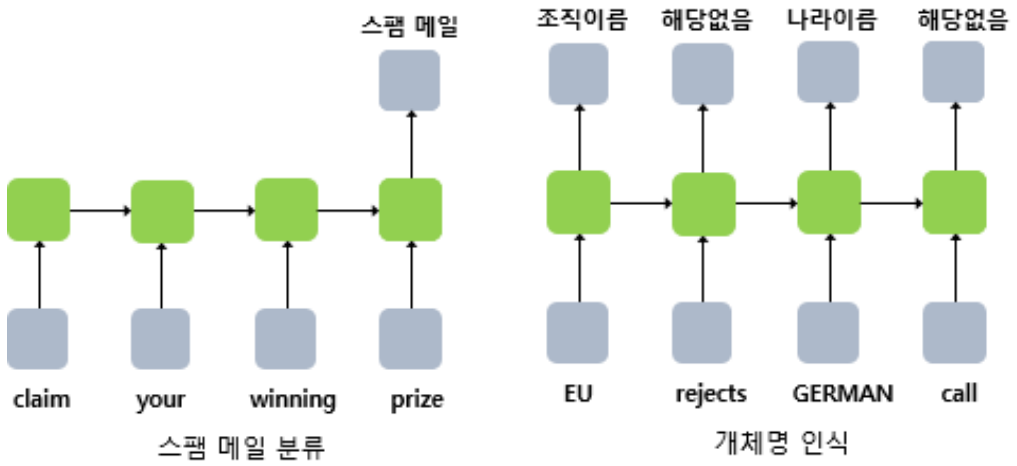
- 언어 모델 기술을 이용하여 프로그램 소스 코드를 학습한 모델
- 전통적인 언어 모델: 주어진 문자열 다음에 올 문자열을 예측하는 방법
  - 예시: N-gram 모델, GPT 모델
  - 예시: "가는 말이 고와야 오는 말도 □"에서 □는 무엇인가?
- 보다 넓은 의미: 문자열의 순서에 기반해서 자연어 문장의 의미를 이해하는 방법
  - 예시: BERT 모델, T5 모델
  - 예시: "가는 말이 고와야 □는 말도 곱다." 에서 □는 무엇인가?
- 최근의 좁은 의미: 트랜스포머 구조를 사용하여 학습된 언어 모델

트랜스포머 구조 (1/2)

- 심층 신경망(DNN) 구조의 일종
  - CNN과 같이 입력의 길이가 고정
  - RNN과 같이 순차적 데이터의 의미를 이해
- 어텐션(attention) 층을 여러 층 적재하여 데이터를 심층적으로 이해
  - "Attention is all you need" (2017, Google)



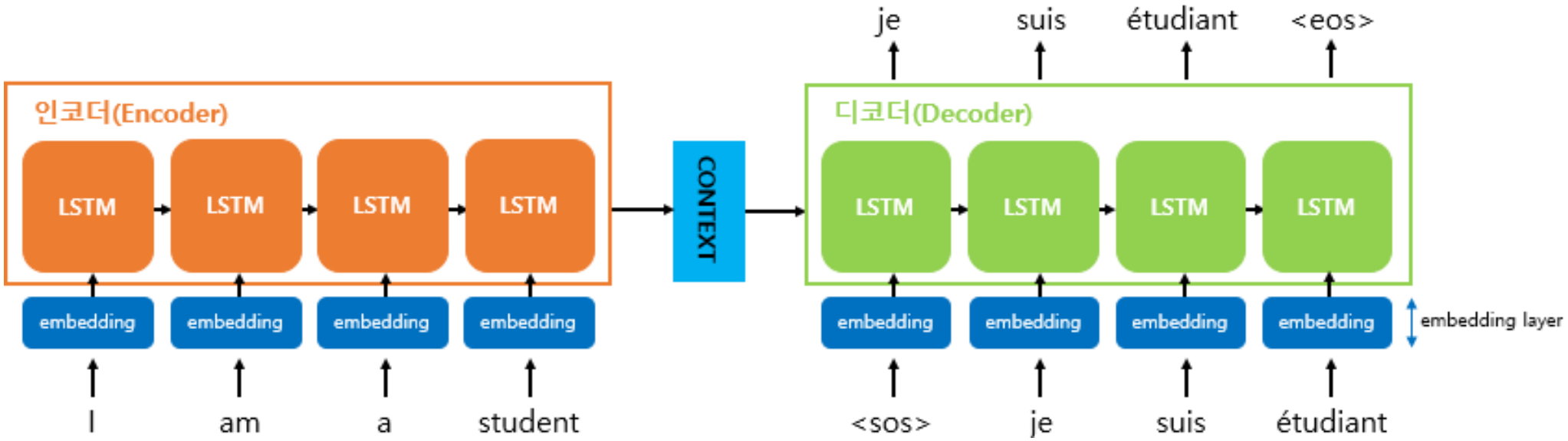
※ 이미지 출처



※ 이미지 출처

트랜스포머 구조 (2/2)

- 인코더-디코더 연결 구조로 구성
  - 인코더: 데이터를 잘 이해하는 구조
  - 디코더: 데이터를 잘 생성하는 구조



※ 이미지 출처

실습 1. 모델 구조 이해하기

mBERT 토크나이저

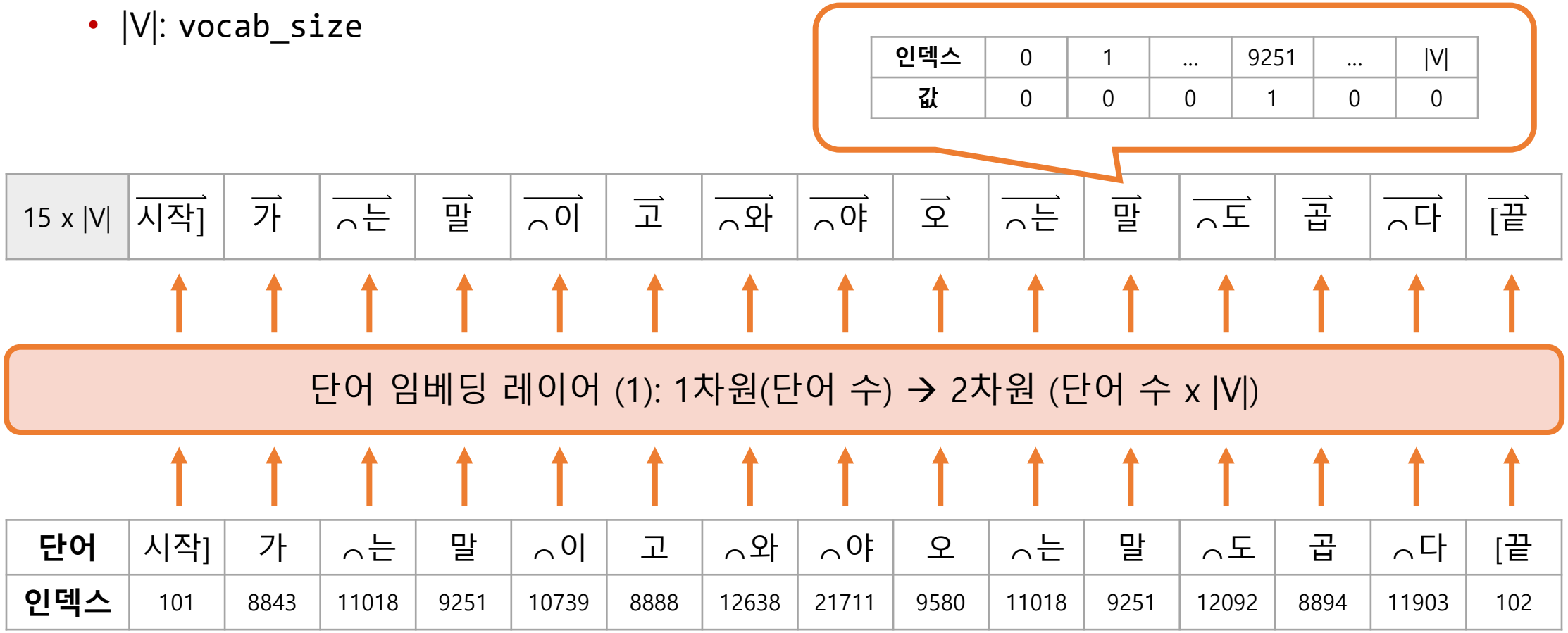
- 토크나이저: 문장 → 문장에 있는 단어의 인덱스 목록 (1차원 배열)
  - 사전(vocab): 모델이 이해할 수 있는 단어 목록
  - 사전 크기(vocab_size): 사전에 있는 단어 개수
- 사전은 학습 데이터에 있는 단어 통계를 학습하여 구축
- 모델의 일부는 아니지만 모델과 항상 세트로 사용

단어	시작]	가	◠는	말	◠이	고	◠와	◠야	오	◠는	말	◠도	곱	◠다	[끝
인덱스	101	8843	11018	9251	10739	8888	12638	21711	9580	11018	9251	12092	8894	11903	102

◠부호: 띄어쓰기 없이 앞 토큰에 붙여쓰기

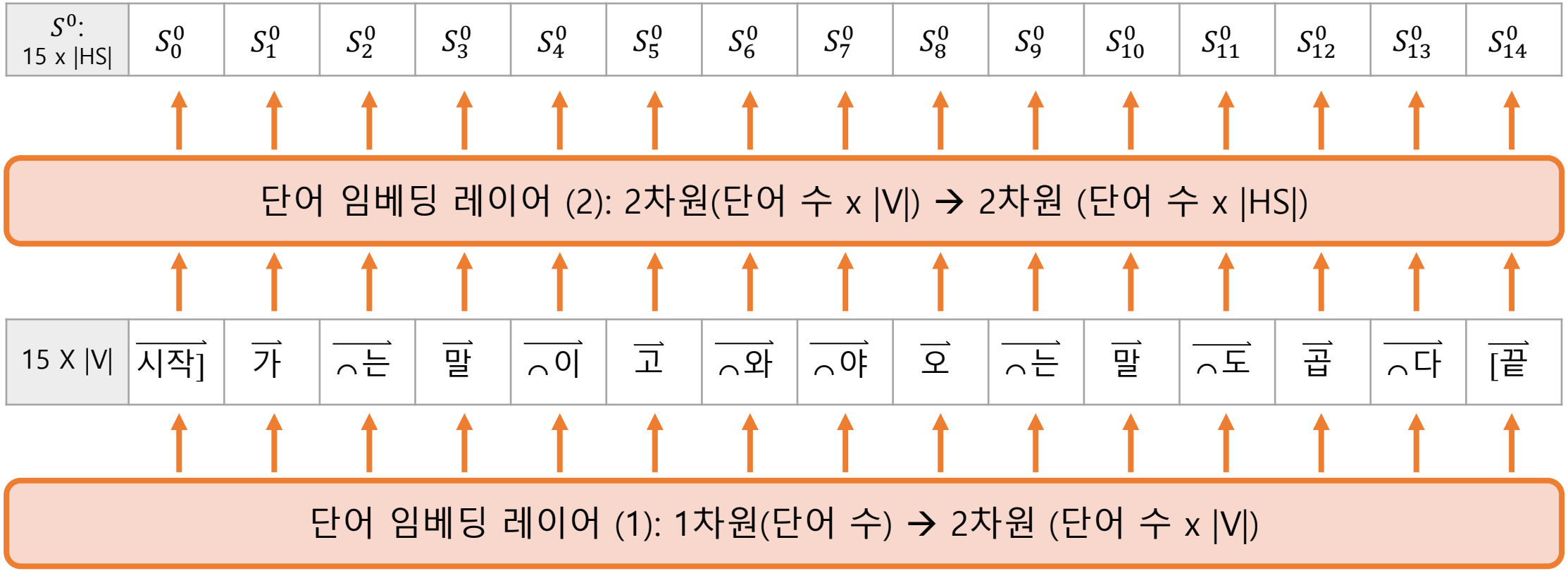
mBERT 모델 구조 – 단어 임베딩 (1/2)

- 단어 임베딩 레이어 (1): 사전 내 단어의 인덱스를 one-hot 인코딩
  - |V|: `vocab_size`



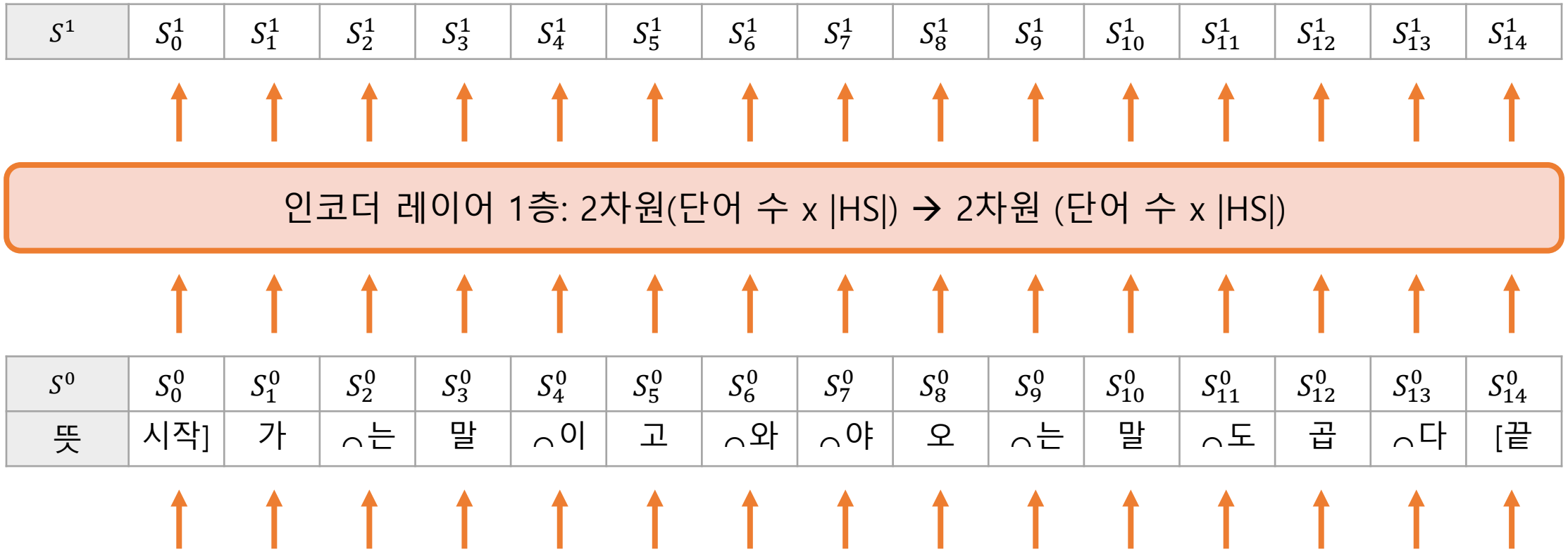
mBERT 모델 구조 – 단어 임베딩 (2/2)

- 단어 임베딩 레이어 (2): one-hot 벡터를 모델의 은닉 상태(Hidden State)로 변환
  - |HS|: `hidden_size`



mBERT 모델 구조 – 인코더 (1/2)

- 인코더 레이어: 어텐션(Attention)을 이용하여 은닉 상태 벡터에 문장의 의미 부여



mBERT 모델 구조 – 어텐션

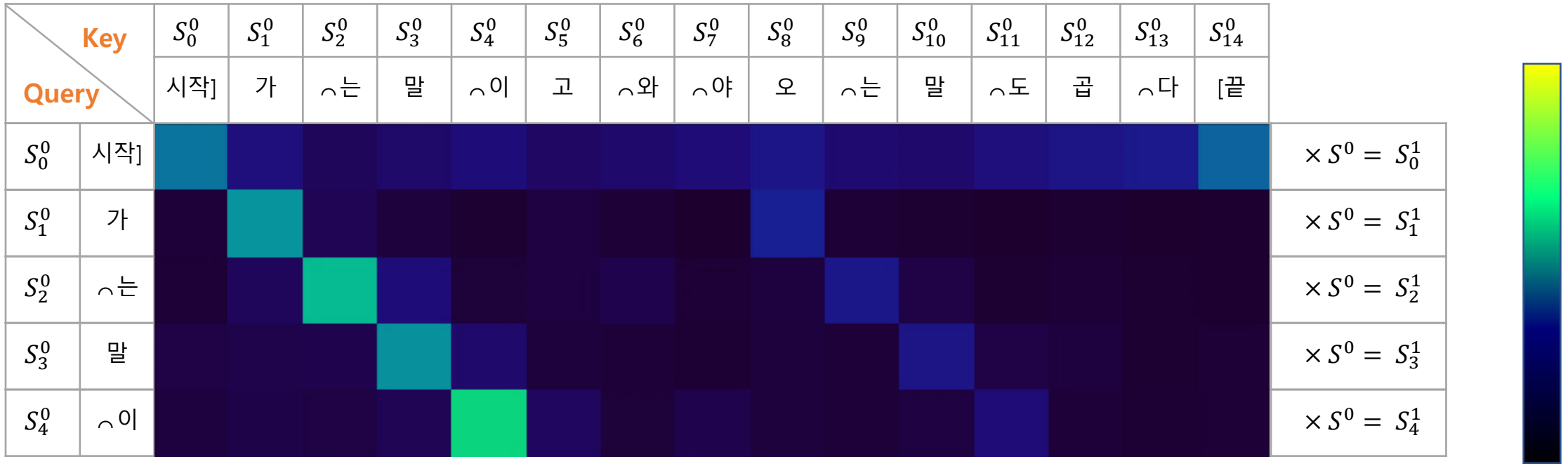
- 현재 문장 내에서 단어와 단어 사이의 관계를 수치화해서 표현
- 0번째 레이어에서의 어텐션 x 0번째 레이어의 은닉 상태= 1번째 레이어의 은닉상태
  - 2차원 (단어 수 x 단어 수) x 2차원 (단어 수 x |HS|) = 2차원 (단어 수 x |HS|)

상태	$\overrightarrow{S^0}$	S_0^0	S_1^0	S_2^0	S_3^0	S_4^0	S_5^0	S_6^0	S_7^0	S_8^0	S_9^0	S_{10}^0	S_{11}^0	S_{12}^0	S_{13}^0	S_{14}^0	
$\overrightarrow{S^0}$	뜻	시작]	가	∩는	말	∩이	고	∩와	∩야	오	∩는	말	∩도	곱	∩다	[끝	
S_0^0	시작]																$\times S^0 = S_0^1$
S_1^0	가																$\times S^0 = S_1^1$
S_2^0	∩는																$\times S^0 = S_2^1$
S_3^0	말																$\times S^0 = S_3^1$
S_4^0	∩이																$\times S^0 = S_4^1$

mBERT 모델 구조 – 어텐션

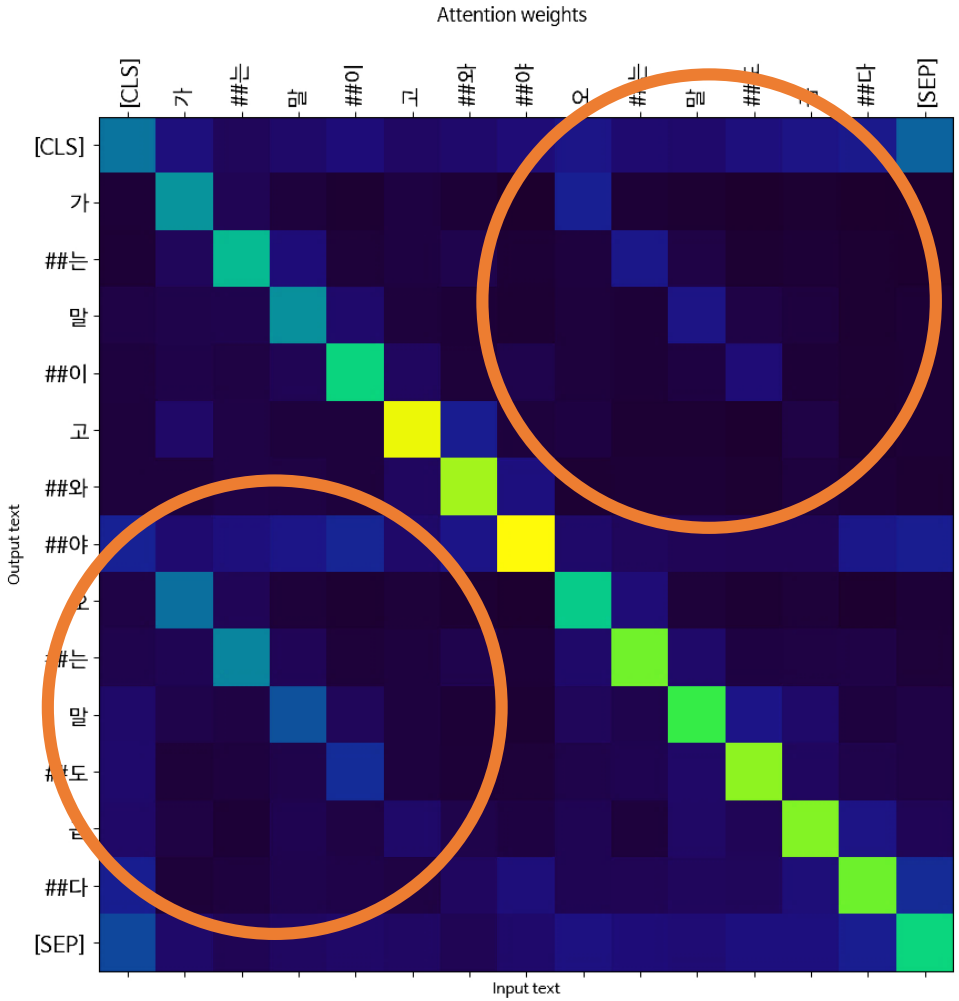
Query · Key　　Value

- 0번째 레이어에서의 어텐션 x 0번째 레이어의 은닉 상태= 1번째 레이어의 은닉상태
  - 2차원 (단어 수 x 단어 수) x 2차원 (단어 수 x |HS|) = 2차원 (단어 수 x |HS|)



어텐션 예시

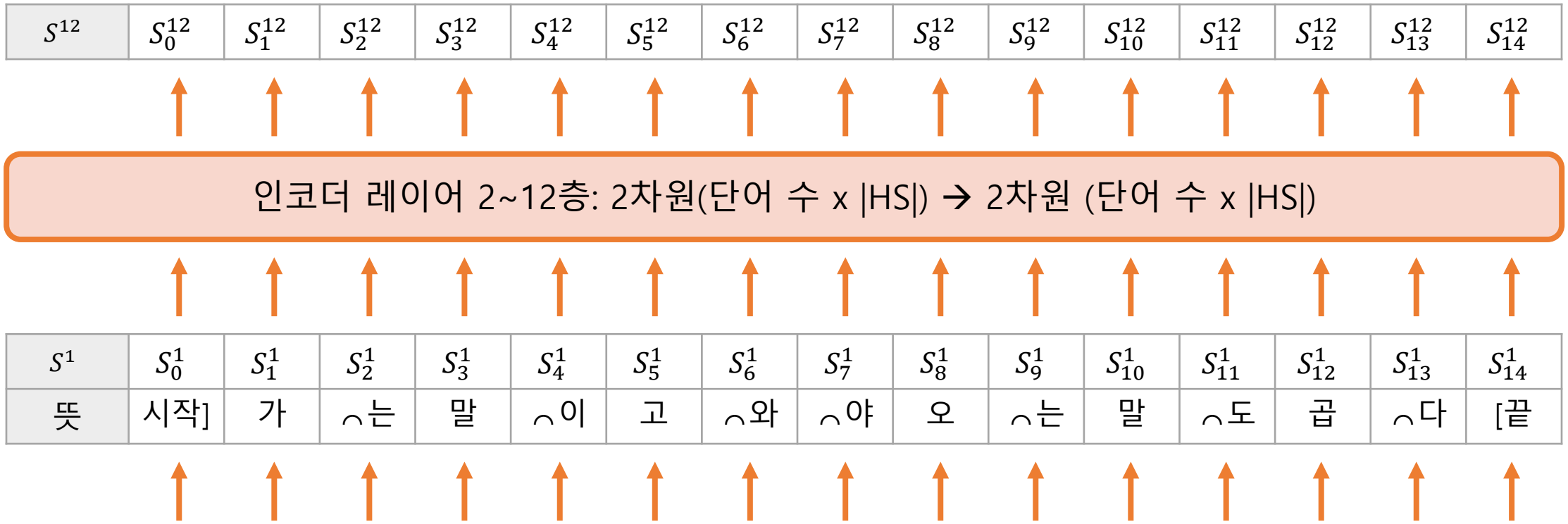
"가는 말이 고" 와 "오는 말도 곱" 사이의 어텐션 점수가 높다



※ 주의: 어텐션 점수에 "사람이 이해할 수 있는 설명 능력이 있는가?"는 아직 연구와 논쟁 진행중

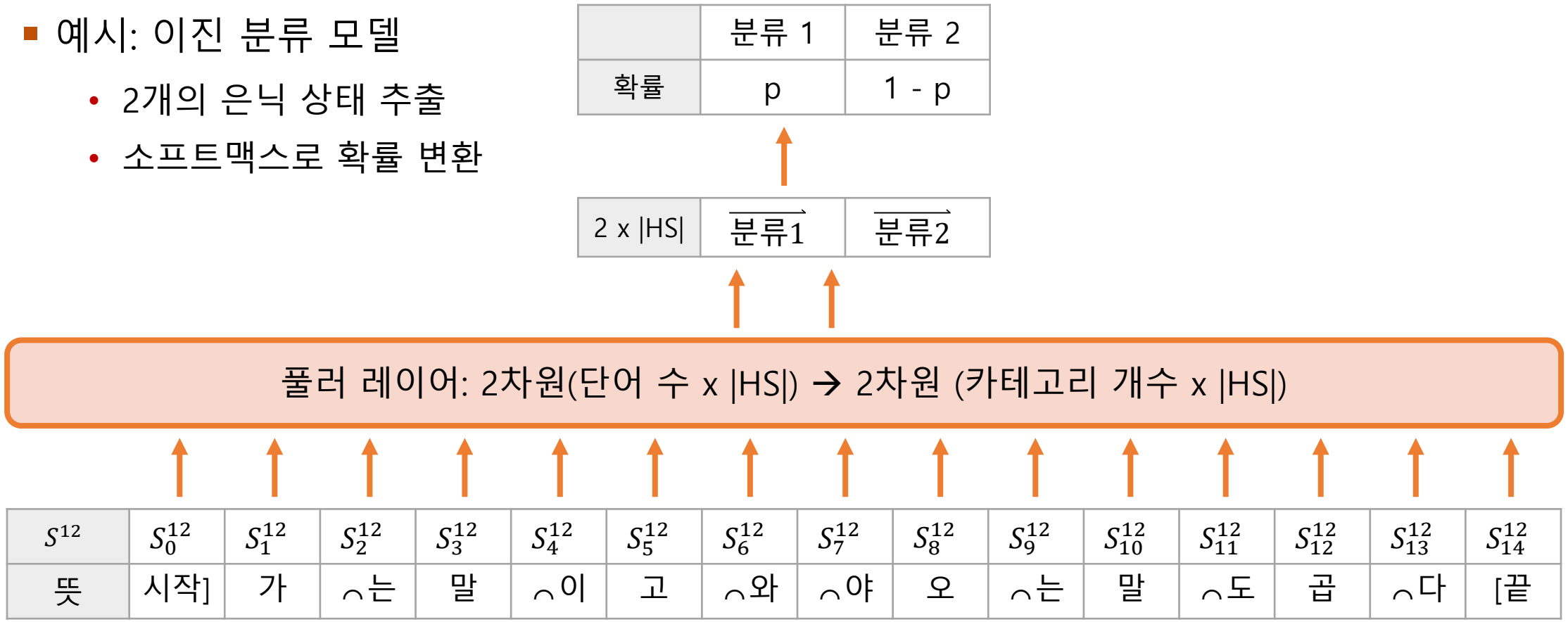
mBERT 모델 구조 – 인코더 (2/2)

- 인코더 레이어: 어텐션(Attention)을 이용하여 은닉 상태 벡터에 문장의 의미 부여
- 심층 신경망 모델(DNN): mBERT 모델은 인코더 레이어를 12층 사용



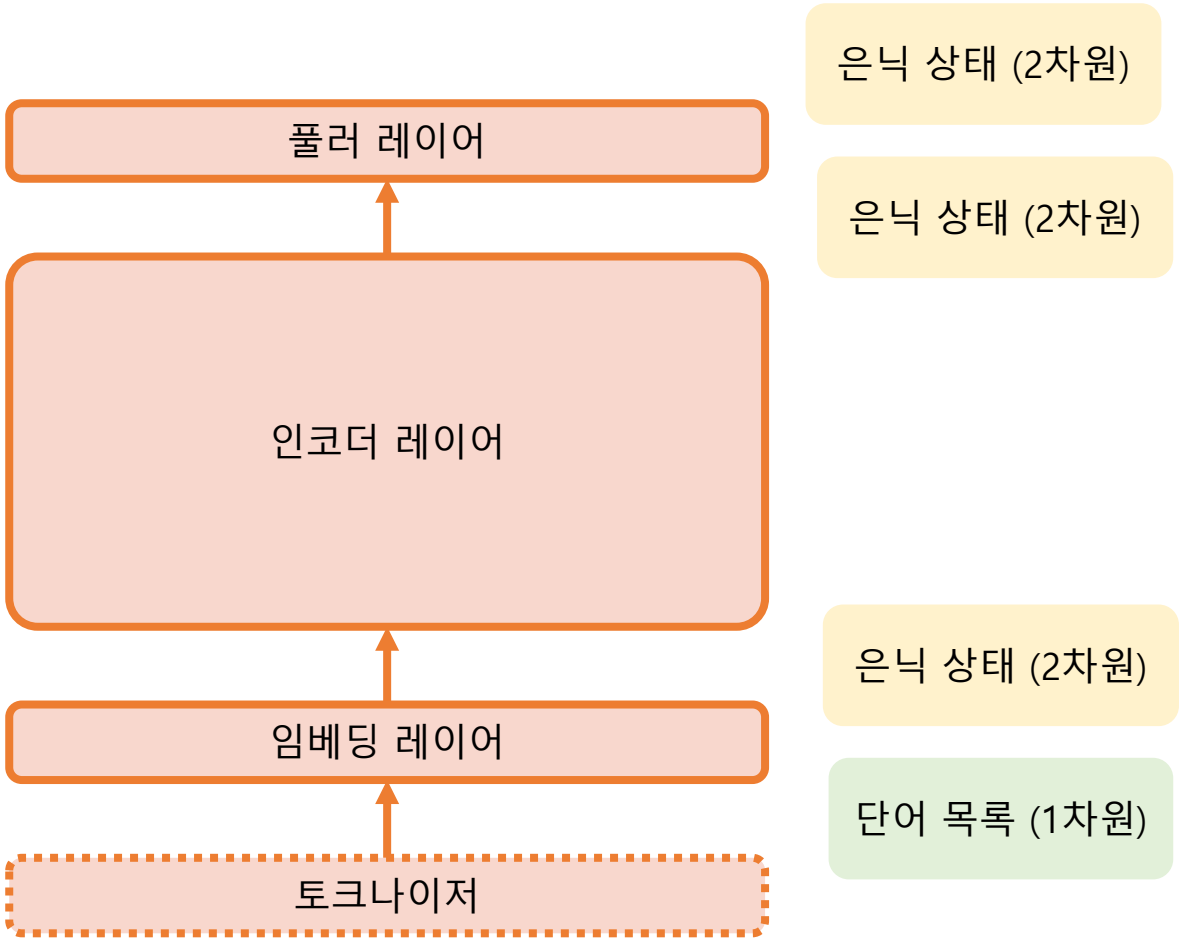
mBERT 모델 구조 – 은닉 상태 해석하기

- 풀러 레이어: 필요한 벡터 크기로 가공
- 예시: 이진 분류 모델
  - 2개의 은닉 상태 추출
  - 소프트맥스로 확률 변환



mBERT 모델 구조 정리

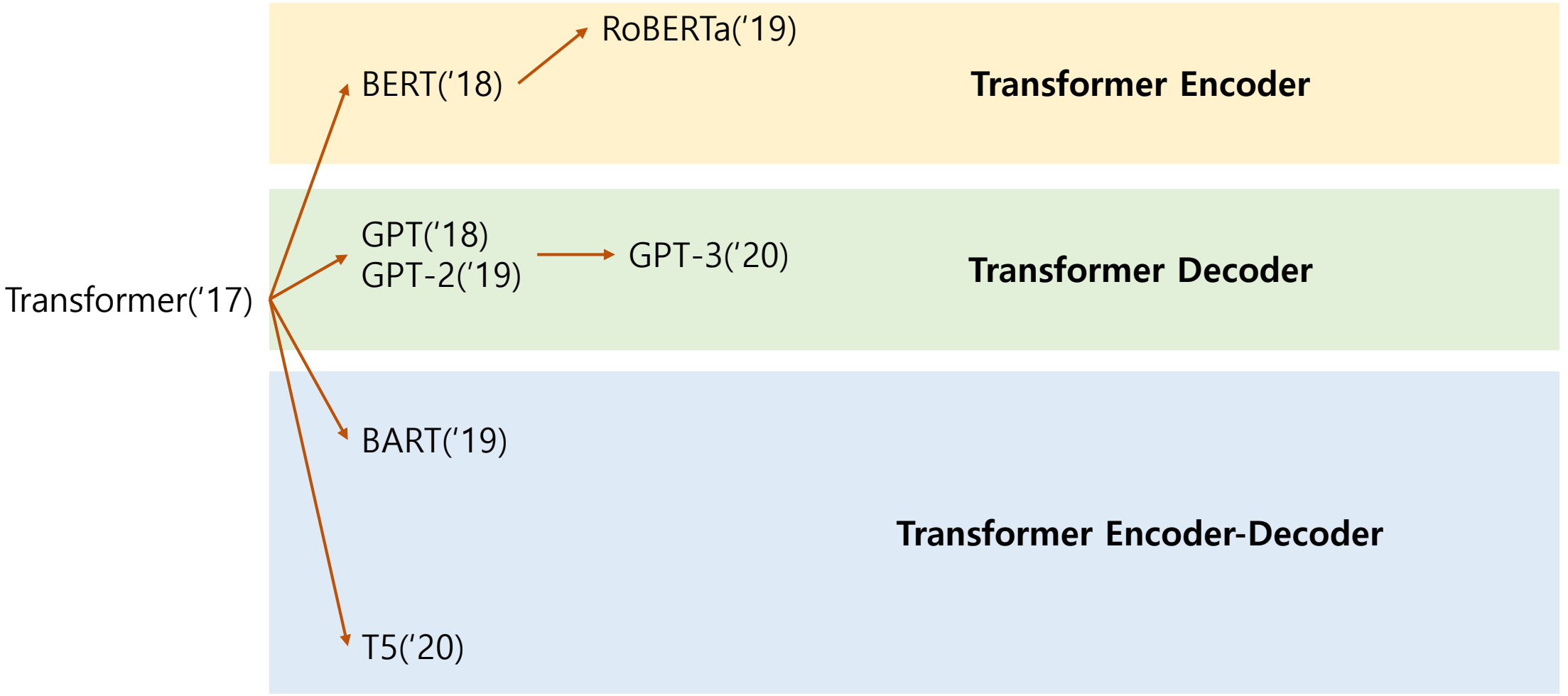
- 토크나이저와 모델은 세트
- 임베딩 이후 일정한 벡터 크기 유지
- 심층 구조로 의미 심화
- 최종 풀러 레이어의 출력 형태를 변경하여 원하는 모델 출력 생성



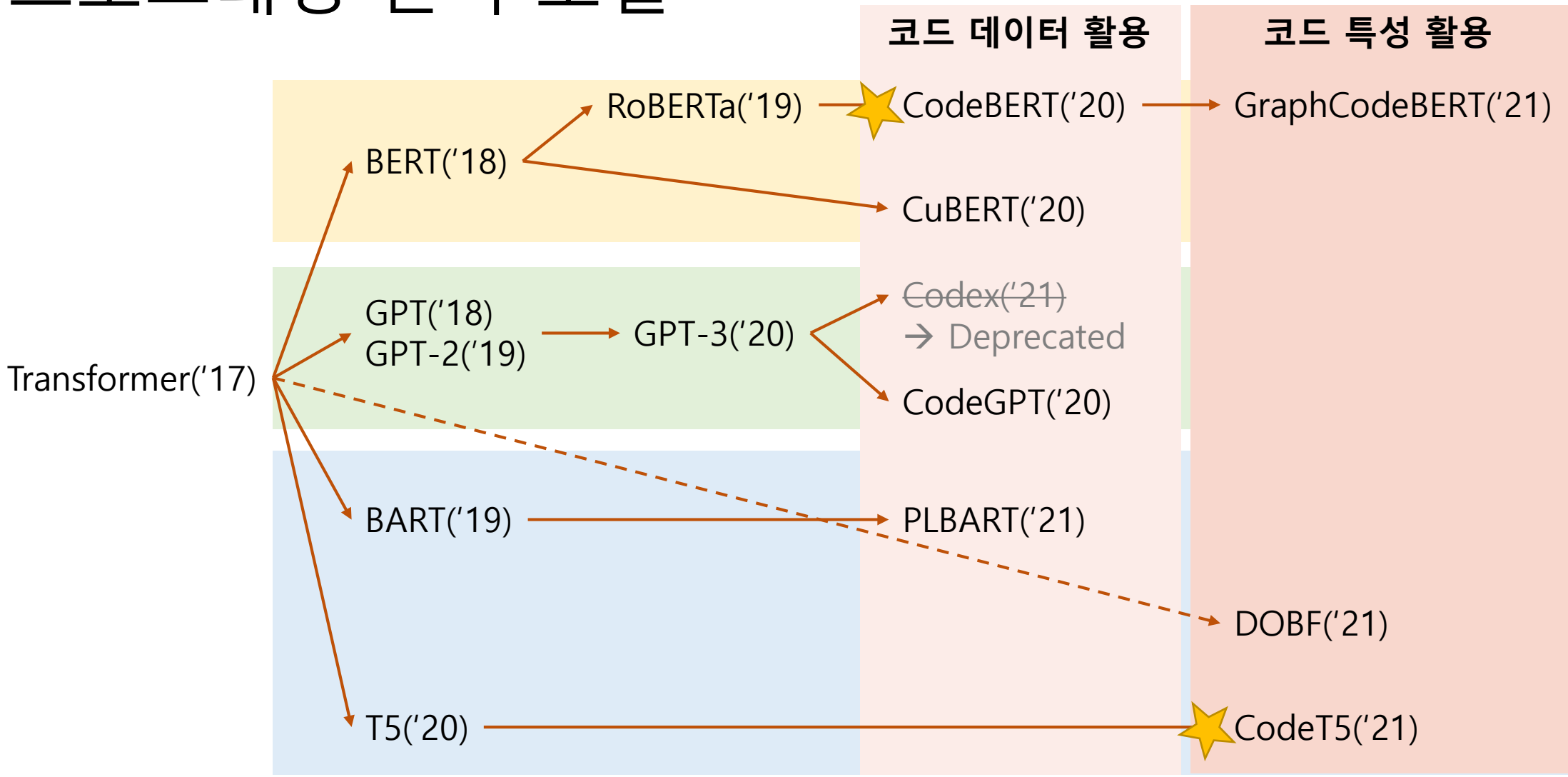
모델 및 데이터 일람

- 오픈 소스 모델과 데이터 목록
- 필요한 GPU 사양 계산하기

자연어 언어 모델


Transformer('17)
BERT('18)
RoBERTa('19)
GPT('18)
GPT-2('19)
GPT-3('20)
BART('19)
T5('20)
Transformer Encoder
Transformer Decoder
Transformer Encoder-Decoder

프로그래밍 언어 모델


코드 데이터 활용
코드 특성 활용
Transformer('17)
BERT('18)
RoBERTa('19)
CodeBERT('20)
GraphCodeBERT('21)
CuBERT('20)
GPT('18)
GPT-2('19)
GPT-3('20)
Codex('21)
→ Deprecated
CodeGPT('20)
BART('19)
PLBART('21)
DOBF('21)
T5('20)
CodeT5('21)

거대 언어 모델

- GPT-3(2020) 이후 거대 언어 모델 (Large Language Model) 널리 사용

그룹	모델 이름	규모	구조	공개 여부	발표 시기	분류
Facebook	InCoder	1B, 6B	Decoder-only	공개	2022.04.	코드 특화
Salesforce	CodeRL	770M+	CodeT5 + 강화학습	공개	2022.07.	코드 특화
BigScience	BLOOM	175B+	Decoder-only	공개	2022.11.	범용
Facebook	LLaMa	7B~65B	Transformer	제한적 공개	2023.02.	범용
OpenAI	GPT-4	unknown	GPT3	유료 API	2023.03.	범용
Salesforce	CodeT5+	110M~770M	T5	공개	2023.05.	코드 특화
Facebook	LLaMa2	7B~70B	Transformer	제한적 공개	2023.07	범용

대화형 언어 모델

- 채팅 형식을 강화학습으로 학습
  - (장점): 강화학습으로 "사람이 더 좋아할만한 출력" 학습
  - (단점): Prefix 제한 불가능

그룹	모델 이름	규모	구조	공개 여부	발표 시기	분류
OpenAI	InstructGPT	Unknown	GPT3	유료 API	2022.01.	범용
Salesforce	CodeGen	350M, 2.7B, 16B	Decoder-only	공개	2022.03.	코드 특화
OpenAI	ChatGPT	Unknown	GPT3	유료 API	2022.12.	범용
Salesforce	InstructCodeT5+	16B	CodeT5	공개	2023.05.	코드 특화
Salesforce	CodeGen2	1B, 7B, 16B	Decoder-only	공개	2023.05.	코드 특화
Facebook	LLaMa2-Chat	7B~70B	Transformer	제한적 공개	2023.07.	범용

코드 데이터셋 – 코드 분류, 요약

그룹	데이터 이름	종류	입력	출력
Microsoft	CodeXGLUE	Clone detection	코드 쌍	이진분류
		Defect detection	코드	이진분류
		Type prediction	코드	타입 분류
		Code summarization	코드	자연어 요약
		Code search	자연어, 코드 쌍	이진 분류
		Text-to-code generation	자연어 요약	코드
Google	CuBERT	Defect detection	코드	이진 분류
		Defect localization	코드	분류

코드 데이터셋 – 코드 생성

그룹	데이터 이름	종류	입력	출력
OpenAI	HumanEval	(original)	자연어/코드 프롬프트	자동 완성 코드
		Infill	자연어/코드 프롬프트	자동 완성+수정 코드
Google	MBPP		자연어 프롬프트	자동 생성 코드
Microsoft	CodeXGLUE	Cloze test	코드	토큰
		Code completion	코드 프롬프트	자동 완성 코드
		Code repair	코드	자동 수정 코드
		Code translation	자바 코드	파이썬 코드
		Text-to-code generation	자연어 요약	코드

모델 사용할 때 필요한 GPU 메모리

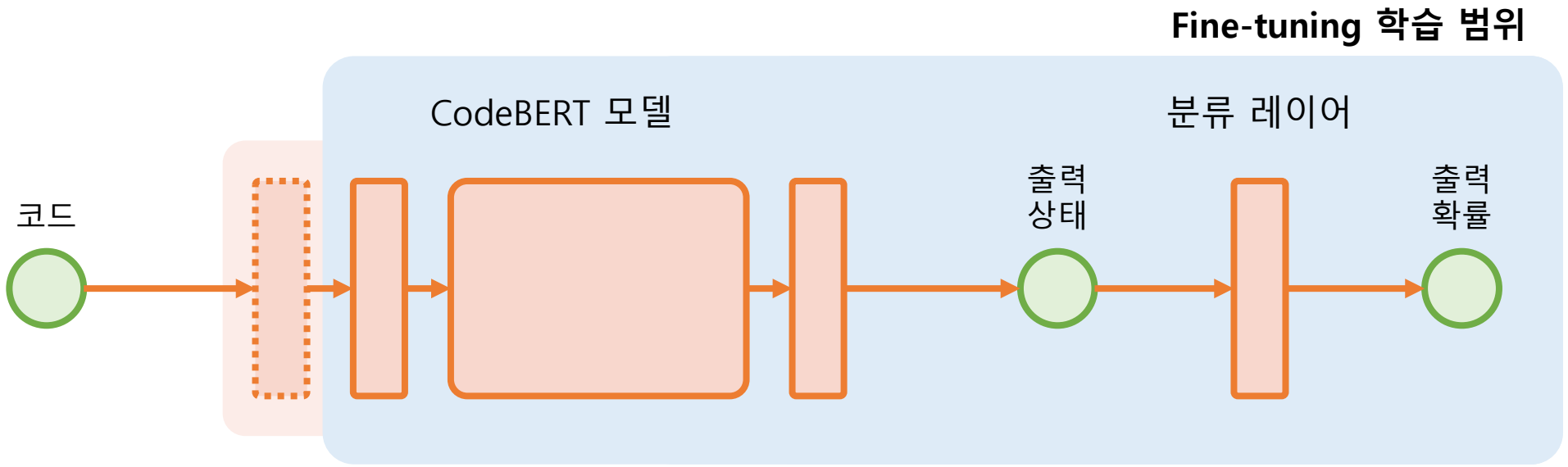
- GPU 메모리 최소 사용량:
  - 파라미터 개수 × 파라미터 크기 = 모델 최소 크기
  - 예시: Facebook Incoder 1B 모델 사용하려면 최소 5 GB (= 1.3B * 4 byte) 메모리 필요
- 모델을 이용하여 추론할 때:
  - 경험적으로 모델 크기의 20% 내외
  - Float32 로 학습된 모델을 Float16으로 사용 가능 → 메모리 사용량 50% 절약, 정확도 손실
- 모델을 학습할 때:
  - 역전파(back propagation)를 위해 중간 상태 변수를 모두 유지하기때문에 메모리 많이 사용
  - 데이터 배치 크기가 작을수록 메모리 덜 사용
  - Fine-tuning 학습의 경우 고정된 레이어가 많을수록 메모리 덜 사용

실습 2. Fine-tuning 학습하기

- 거대 언어 모델을 Fine-tuning 없이 사용하기
- 작은 언어 모델을 Fine-tuning 해서 사용하기

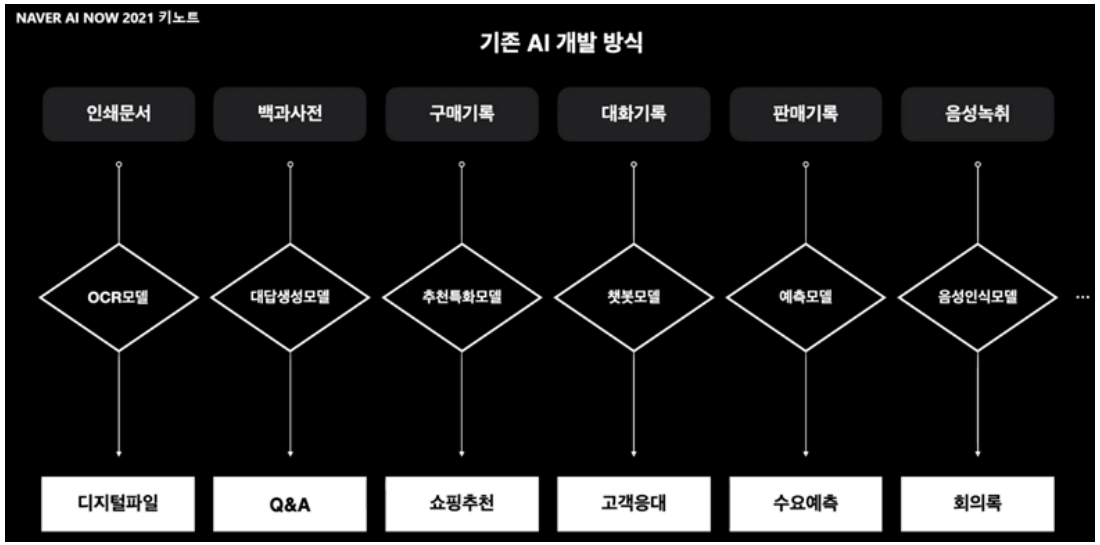
Fine-tuning 학습 (1/2)

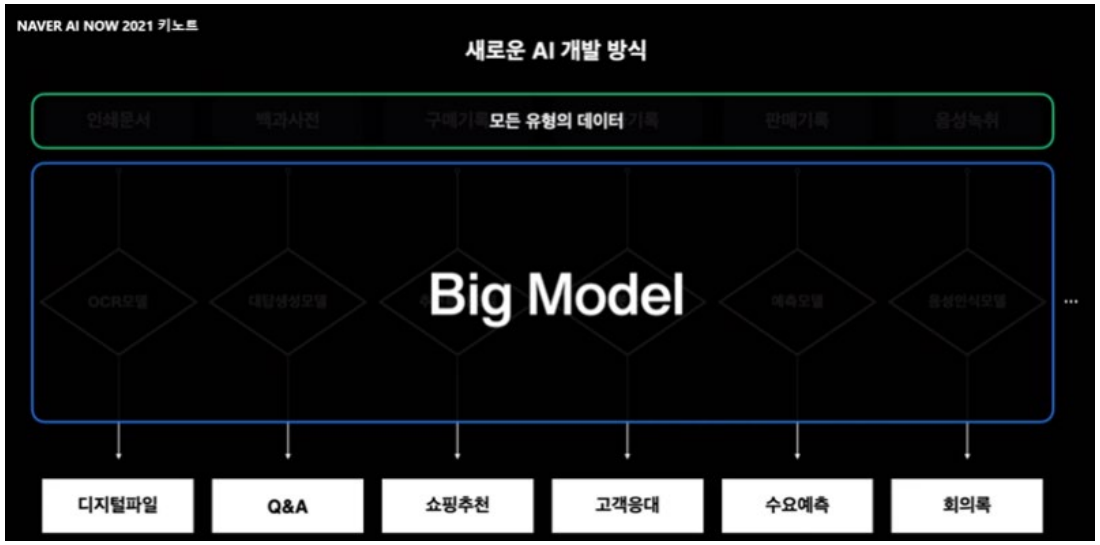
- Fine-tuning
  - 사전 학습된 모델을 적용하고자 하는 과제에 맞게 전이학습 하는 과정
  - 모델의 파라미터 중 일부분을 재학습
  - 추가 학습을 통해 입출력 형태를 조정하거나 레이어를 추가하는 등 모델 변경도 가능
  - 예시: CodeBERT + 분류 레이어 → 입력 코드에 결함이 있는지 분류하는 모델로 재학습



Fine-tuning 학습 (2/2)

- 거대 언어 모델의 Few-shot 혹은 Zero-shot 학습
  - 거대 언어 모델이 이미 다양한 과제와 관련된 데이터를 많이 학습
  - 데이터를 한두개만 더 학습하거나(Few-shot) 학습하지 않아도(Zero-shot) 과제를 잘 수행





※ NAVER AI NOW 2021 Hyper-CLOVA Keynote

거대 언어 모델 vs Fine-tuned 작은 언어 모델

- 학습 및 실행 비용 비교 예시 1. Facebook/incoder-1B vs Microsoft/codebert-base
  - 실습 데이터 (CodeXGLUE Code Completion Java) 를 batch size 1로 학습할 경우 비용 비교
  - 실습 예시 실행 비용 비교

예시 1		거대 언어 모델	작은 언어 모델
모델	이름	Facebook/incoder-1B	Microsoft/codebert-base
	크기	5.9 GB	1.4 GB
Fine-tuning	GPU 메모리	해당 없음	(최소) 3.3 GB
	학습 시간	해당 없음	(최대) 4 시간
자동 완성 실행	GPU 메모리	6.9 GB	1.7 GB
	개별 실행 시간	2.17 sec	1.14 sec

거대 언어 모델 vs Fine-tuned 작은 언어 모델

- 학습 및 실행 비용 비교 예시 2. GPT-3 API vs CodeGPT
  - CodeXGLUE Code Completion Python 데이터를 batch size 1 또는 4 로 학습할 경우 비교
  - 모델이 생성하는 코드에 있는 Python 문법 오류 비교

예시 2		거대 언어 모델	작은 언어 모델	
모델	이름	GPT-3 (text-davinci-003)	CodeGPT-2 small	
	크기	알 수 없음	1.5 GB	
Fine-tuning	GPU 메모리	해당 없음	4.6 GB	9.7 GB
	학습 시간	해당 없음	30 시간	16 시간
자동 완성 실행	GPU 메모리	알 수 없음	1.7 GB	
	개별 API 비용	평균 0.02 $	해당 없음	
	개별 실행 시간	평균 14.45 초	평균 6.26 초	
자동 완성 품질	문법 오류 비율	31.4 %	45.1 %	

API 에서 독립하기

- 거대 언어 모델 API 많이 쓰고 좋지만…
  - 제어가 거의 불가능한 깜깜이 도구
  - 버전마다 출력이 달라지고 과제 수행 성능도 변경
  - API 이용료
- 언어 모델을 보조적인 확률 도구로 쓰기로 결정했다면
  - 작은 모델 Fine-tuning 으로도 좋은 확률 모델 획득 가능
  - 공개된 거대 언어모델 이용해도 GPT-3 수준의 생성 가능
  - 동작 제어 가능: 입출력 형식 제한, 랜덤 시드, 확률 샘플링 파라미터 등
- 참고할 만한 자료
  - Natural Language Processing with Transformers book: homepage
  - Hugging Face 에서 제공하는 Transformers 강좌: homepage, wikidocs.net (우리말 번역)