

[Lecture – 10] Transformers and Pretraining



고려대학교 산업경영공학과

Data Science & Business Analytics Lab

발표자 : 김재희

- 1 Problems of Vocab
- 2 Pretrain
- 3 GPT-1
- 4 BERT
- 5 T5
- 6 **결론**

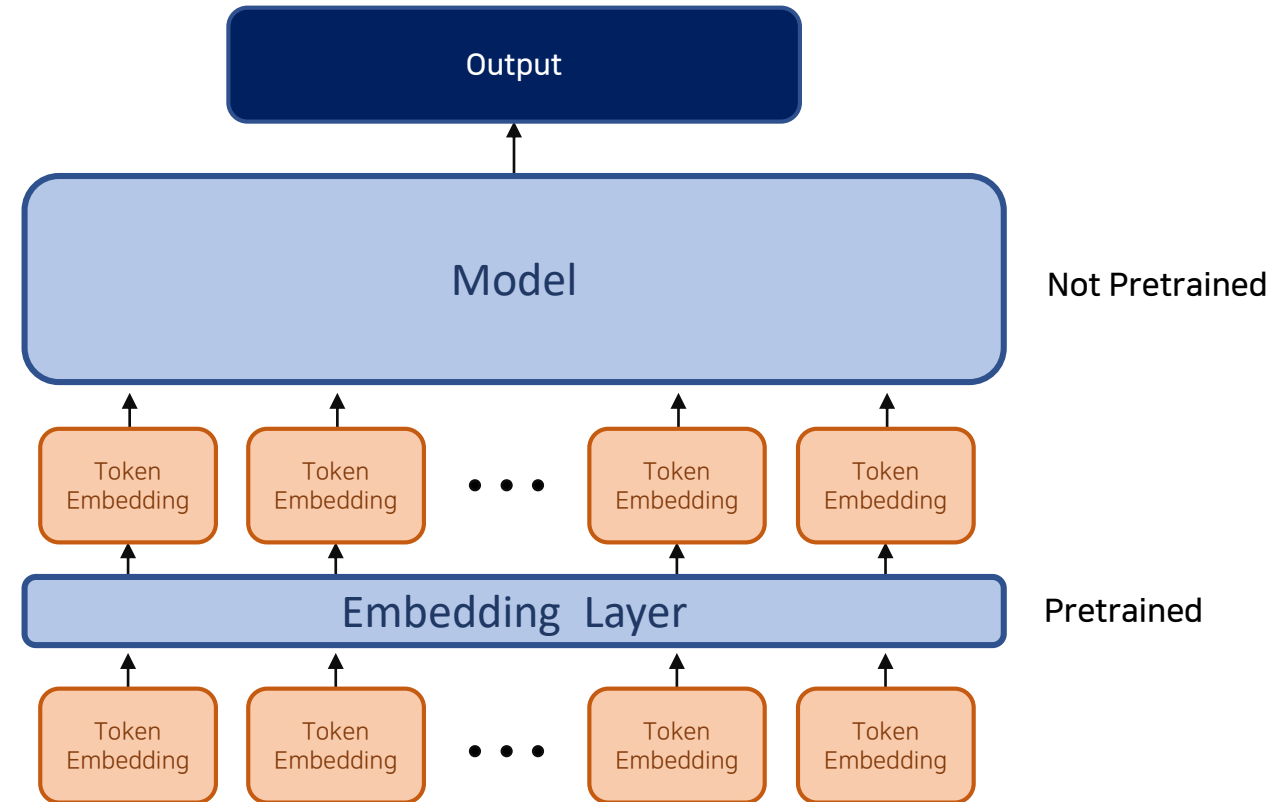
Problems of Vocab

- 현재까지 다룬 방법론에서의 전체 흐름

- 임베딩 레이어 : Pretrained Embedding 사용
- 모델 파라미터 : 초기화



- 학습 데이터 부족
- Out-Of-Vocabulary 문제 발생
- 다른 문장에도 동일한 임베딩 사용

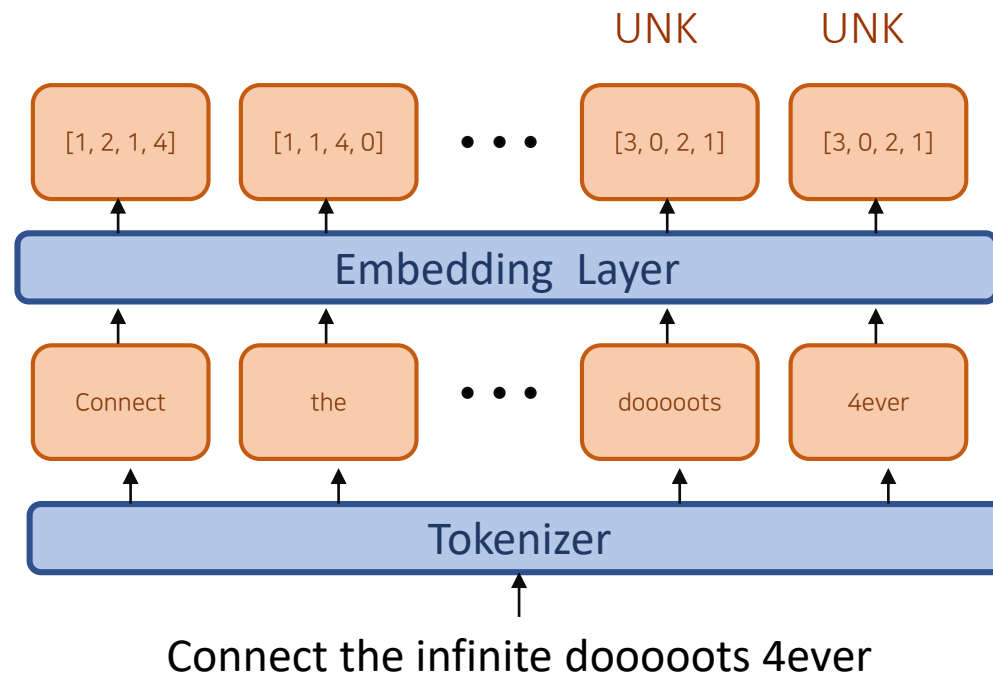


✓ 임베딩 과정

- 문자열 입력
- 학습된 토큰라이저를 통해 토큰화
- 각 토큰별 학습된 임베딩 벡터 변환

✓ 학습 데이터에 존재하지 않는 단어

- 모두 UNK(Unknown) 토큰으로 토큰화
- 다른 토큰이 모두 동일 벡터로 변환



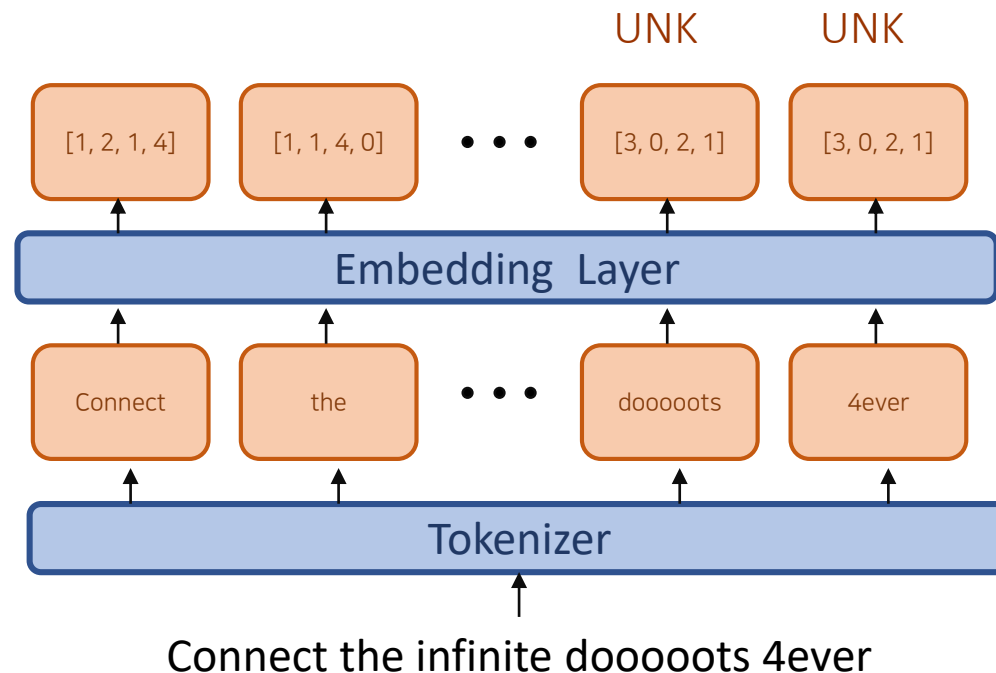
✓ 임베딩 과정

- 문자열 입력
- 학습된 토큰라이저를 통해 토큰화
- 각 토큰별 학습된 임베딩 벡터 변환

✓ 학습 데이터에 존재하지 않는 단어

- 모두 UNK(Unknown) 토큰으로 토큰화
- 다른 토큰이 모두 동일 벡터로 변환

➡ 변형, 오타자, 신조어에 대응 X



✓ Subword Token

- 단어 이하의 단위에서 언어 구조를 추론
- Byte-Pair-Encoding(BPE)
- 문법적 의미 포착

Bidirectional

광광우럭따

✓ Byte-Pair-Encoding

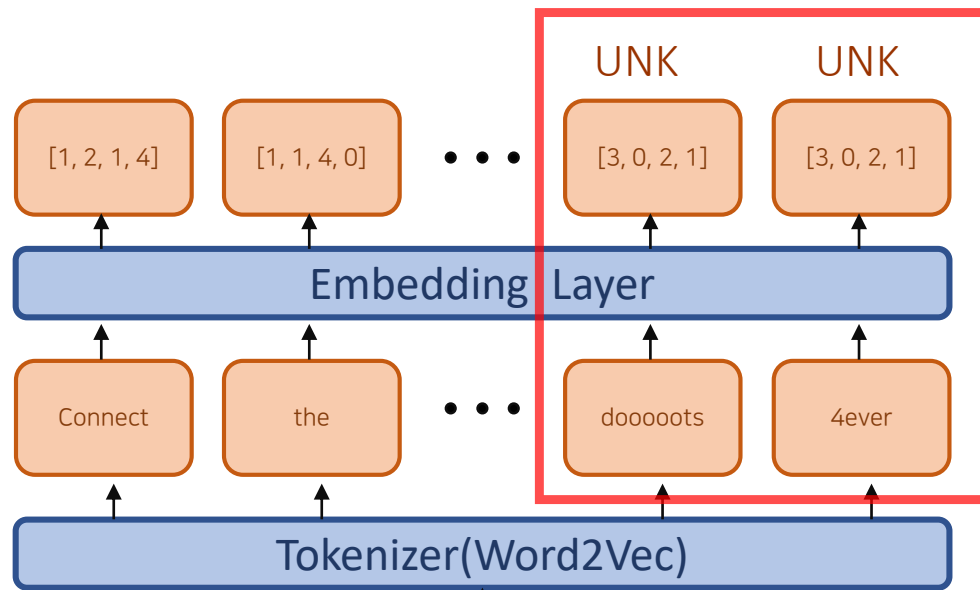
- | | | |
|---|-------|---------------------------|
| 1. 단어를 문자 단위로 분해 | | connect |
| 2. 전체 코퍼스에서 가장 자주 등장하는 연속된 두 문자열 병합 | | c## o## n## n## e## c## t |
| 3. 병합된 토큰을 기반으로 분해 결과 조합 | | n## n## -> nn## |
| 4. 종료 조건(병합 횟수, 토큰 개수 등)을 만족할 때 까지 1 ~ 3 반복 | | c## o## nn## e## c## t |

01

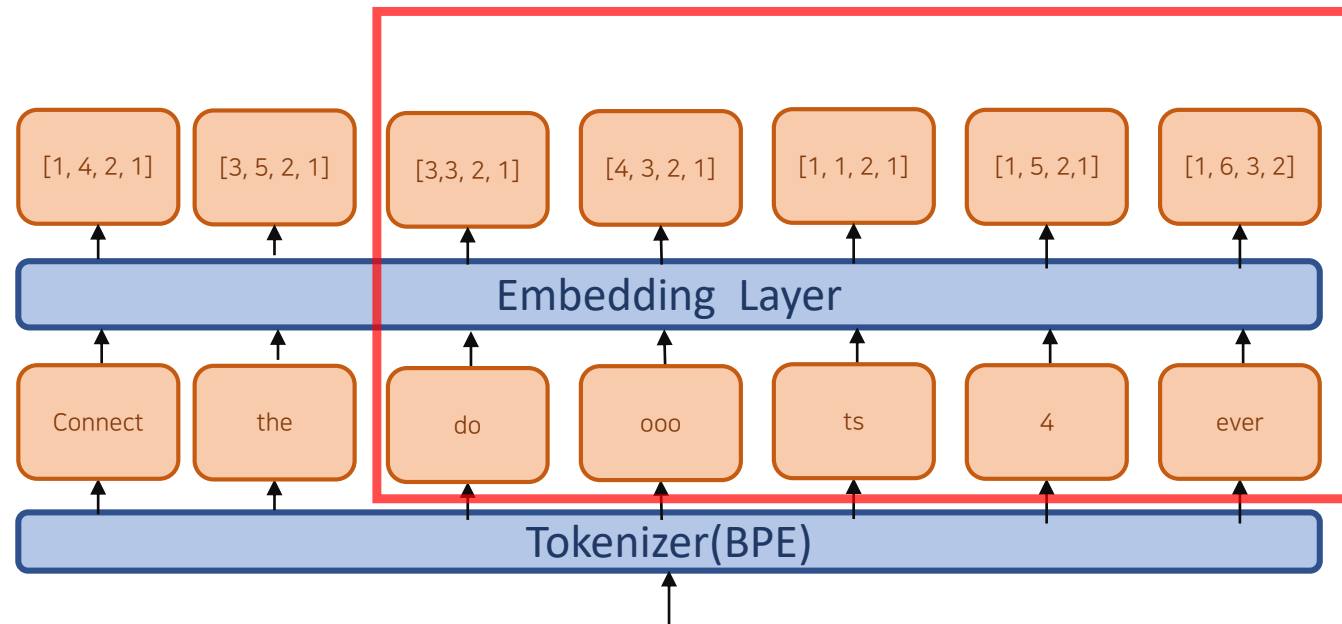
Problems of Vocab

Out of Vocabulary Problem

- BPE와 단어기반 토큰나이징 비교



Connect the infinite doooooots 4ever

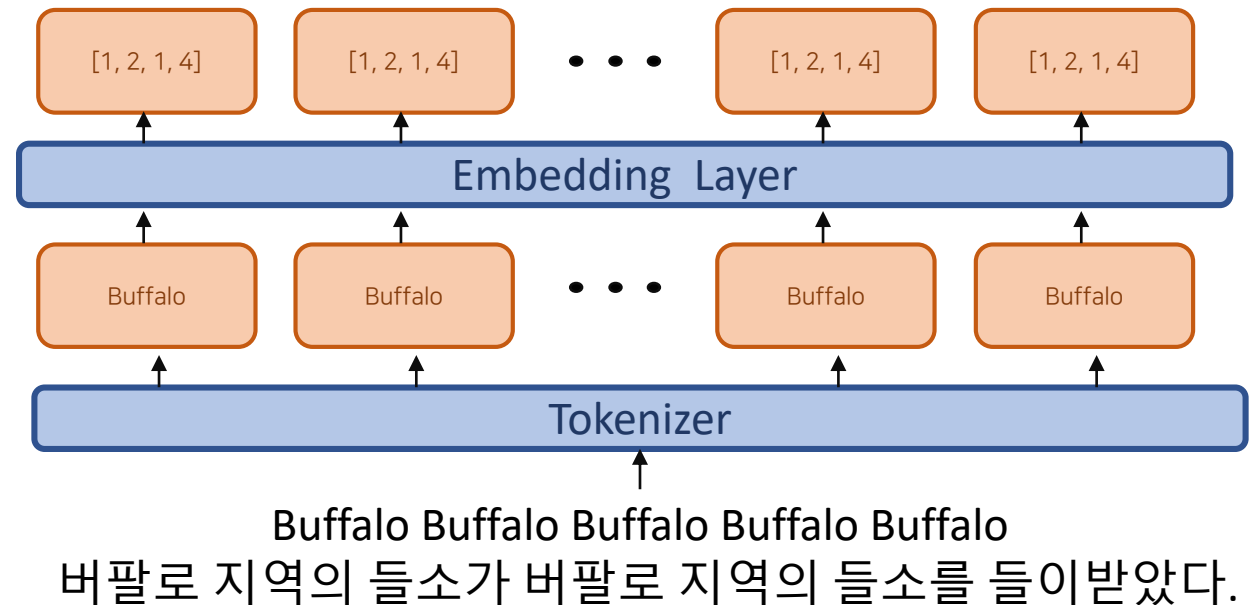


Connect the infinite doooooots 4ever

- ✓ 신조어, 오타에 대응이 가능
- ✓ oov 문제에서 비교적 자유로움

✓ Fixed Word Embedding

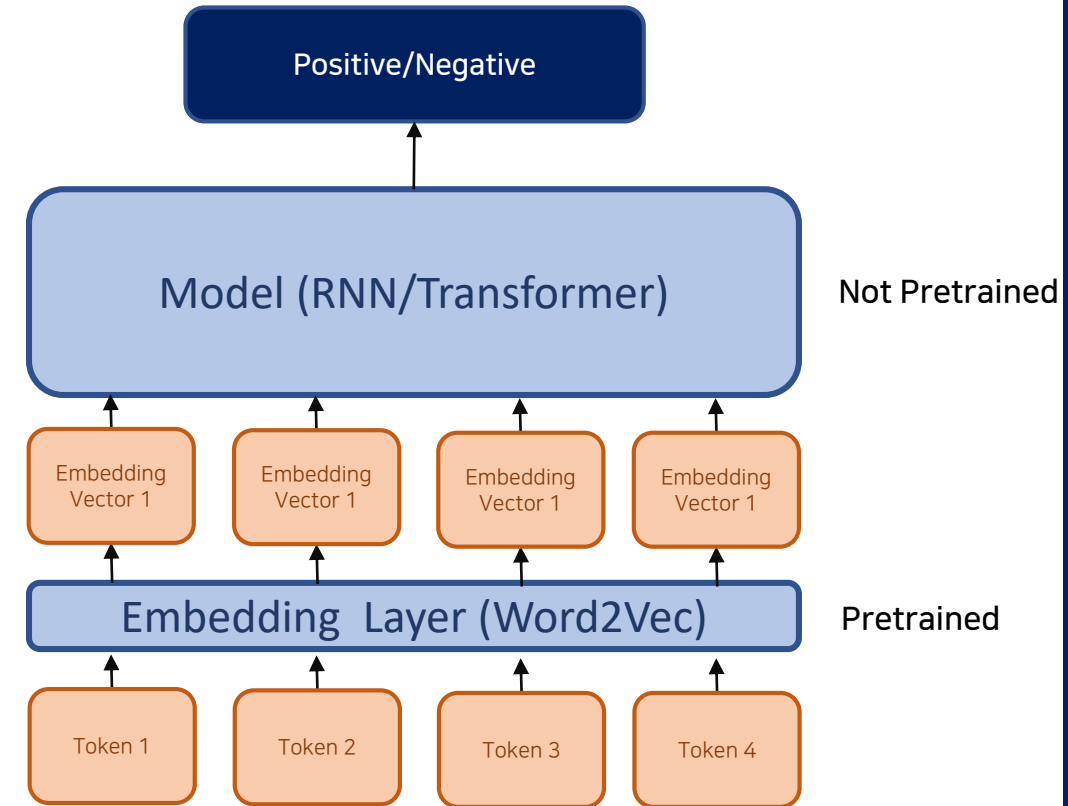
- 각 단어에 대해 사전에 학습된 임베딩 벡터 부여
- 문맥을 고려하지 못한 임베딩 방식
- 다른 문맥임에도 동일 벡터가 맵핑
- 임베딩 레이어로 문맥 반영 불가



✓ Fixed Word Embedding

- 각 단어에 대해 사전에 학습된 임베딩 벡터 부여
- 문맥을 고려하지 못한 임베딩 방식
- 다른 문맥임에도 동일 벡터가 맵핑
- 임베딩 레이어로 문맥 반영 불가

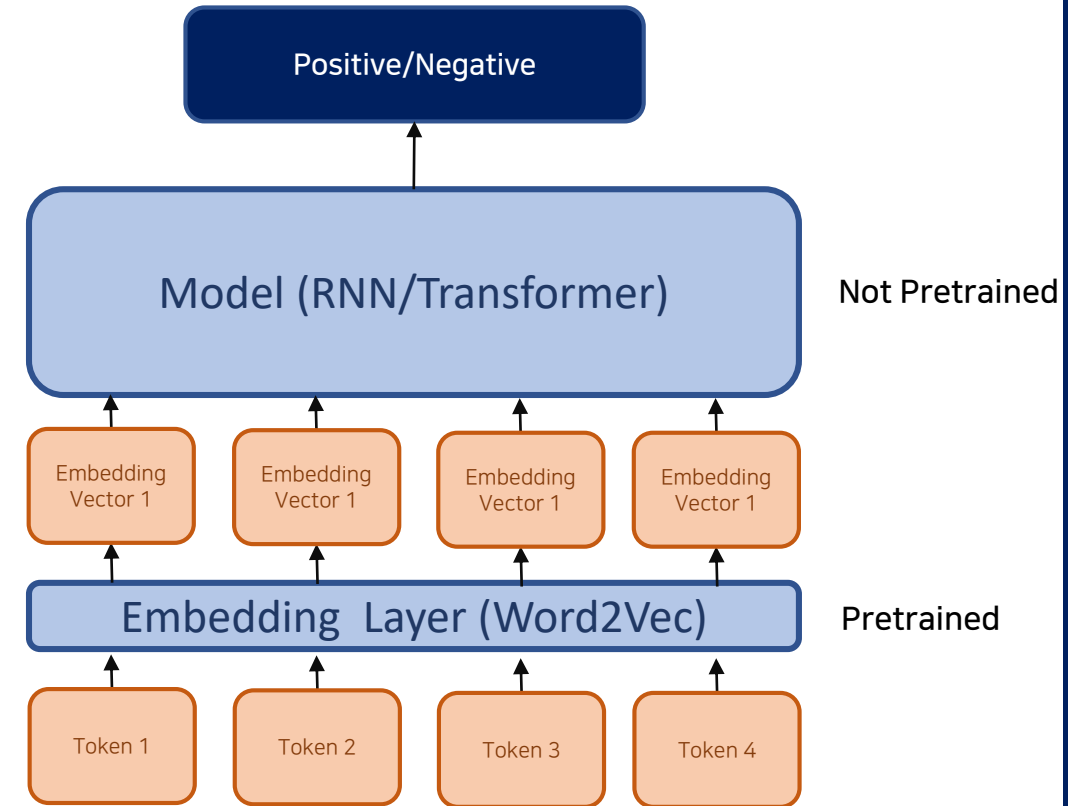
+ 각 태스크마다 대량의 데이터 필요



✓ Fixed Word Embedding

- 각 단어에 대해 사전에 학습된 임베딩 벡터 부여
- 문맥을 고려하지 못한 임베딩 방식
- 다른 문맥임에도 동일 벡터가 맵핑
- 임베딩 레이어로 문맥 반영 불가

+ 각 태스크마다 대량의 데이터 필요

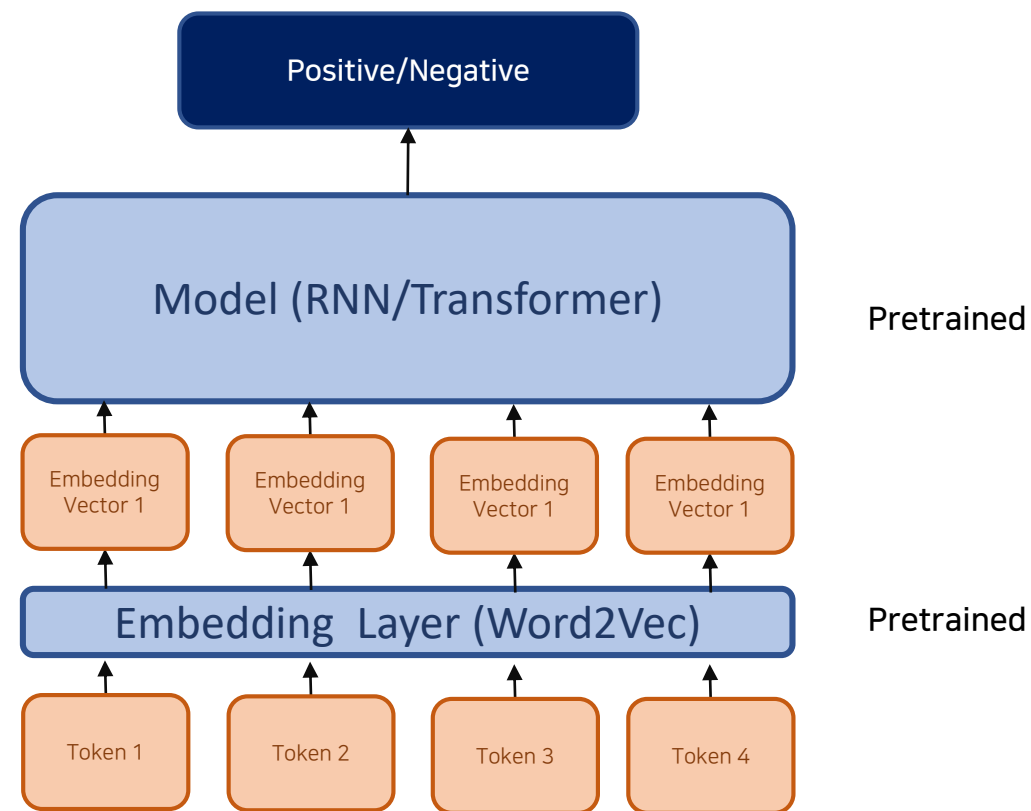


✓ Pretrain(사전학습)

- Input의 일부를 훼손
- 모델이 input을 복원
- 의미적/문법적 구조 학습

✓ Downstream Task

- 실제 수행하고자 하는 태스크
- 사전학습된 모델의 weight를 initial weight로 삼아 학습(fine-tuning) 진행
- QA/Sentiment Analysis/NLI 등등

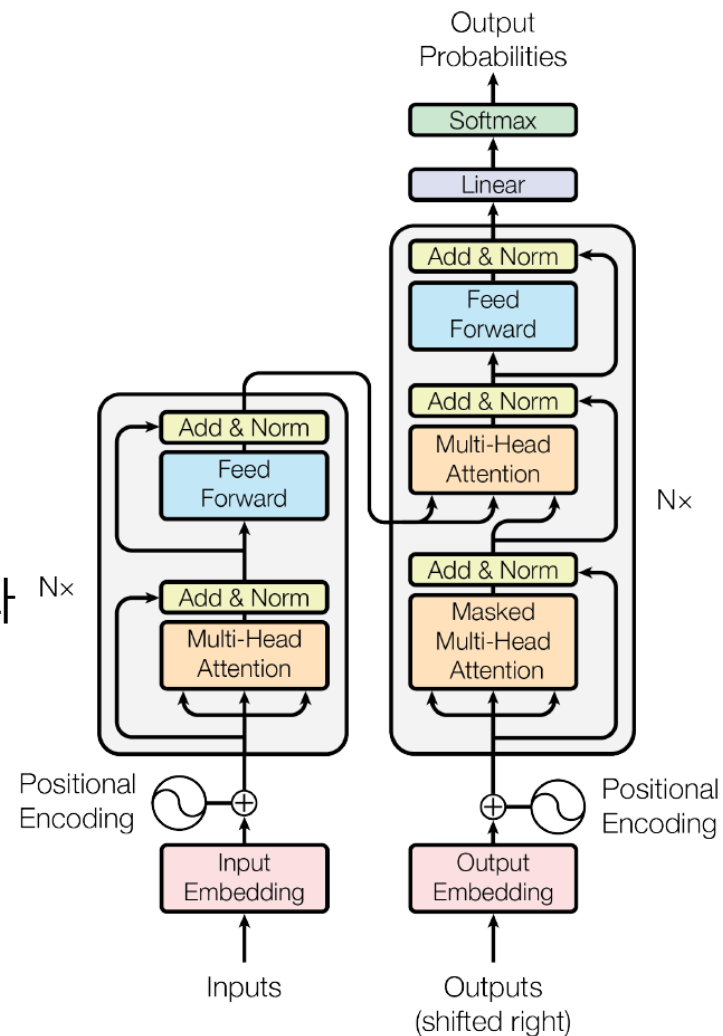


- **Pretrain task** : 문장의 일부를 변형하고, 해당 부분의 단어를 예측

1. 고려대학교는 서울시 성북구 ____에 위치해 있다. [단편적 사실]
2. 나는 어제 저녁으로 돼지고기를 먹__다. [문법 정보]
3. 아이유는 피부가 좋기로 유명하다. ____는 피부 관리를 위해 니베아 로션을 바른다고 한다. [상호참조]
4. 어제 여자 컬링 국가대표팀의 일본전을 봤는데 극적으로 이기는 모습을 보니 ____ [감성]
5. 재희가 커피를 타려고 주방에 갔다. 재희 옆에 서있던 건호는 잠이 와서 ____를 떠났다. [추론]
6. 이 수열 참 이쁘지 않아? 1, 1, 2, 3, 5, ____, 13... [단순 계산]

- ✓ Encoder : input이 연속적으로 레이어 통과
 - Output : Contextual Embedding
(batch, sequence length, embed dim)
 - 입력 문장에서 맥락/의미/문법 구조 등을 파악
- ✓ Decoder : input과 encoder output이 연속적으로 레이어 통과 ^{Nx}
 - Output : 조건부 문장 생성
(batch, sequence length, vocab dim)
 - 입력 문장과 이전 생성 문장을 바탕으로 문장 생성

➡ 인코더와 디코더가 수행하는 역할이 다름

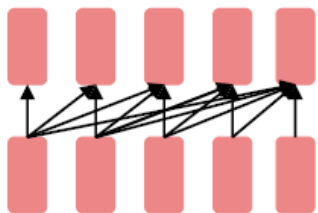


✓ Language Modeling

- $L = \sum \log P(w_t | w_0, \dots, w_{t-1})$
- 별도의 label이 없어도 됨 (다량의 데이터 확보 가능)
- LM은 그 자체로 언어구조/문법 및 의미적 정보 파악을 가능케 함*

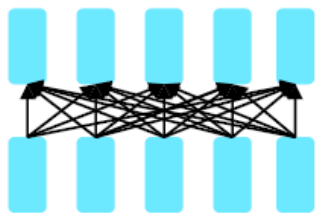
✓ SGD with Pretrain/Finetune

- $\hat{\theta} = \min_{\theta} L_{pretrain}(\theta)$
- $\min_{\theta} L_{finetune}(\theta)$ (*initial weight* = $\hat{\theta}$)
- Random initializing θ 에 비해 $\hat{\theta}$ 가 global optimum에 가까울 수 있다.
- -> 적은 데이터로도 잘 학습, 쉽게 수렴



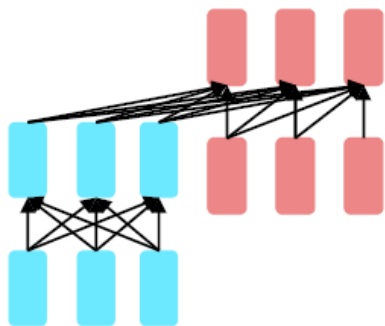
Decoders

- ✓ 일반적인 Language Model
- ✓ 이전 시점의 토큰을 조건으로 하는 생성모델



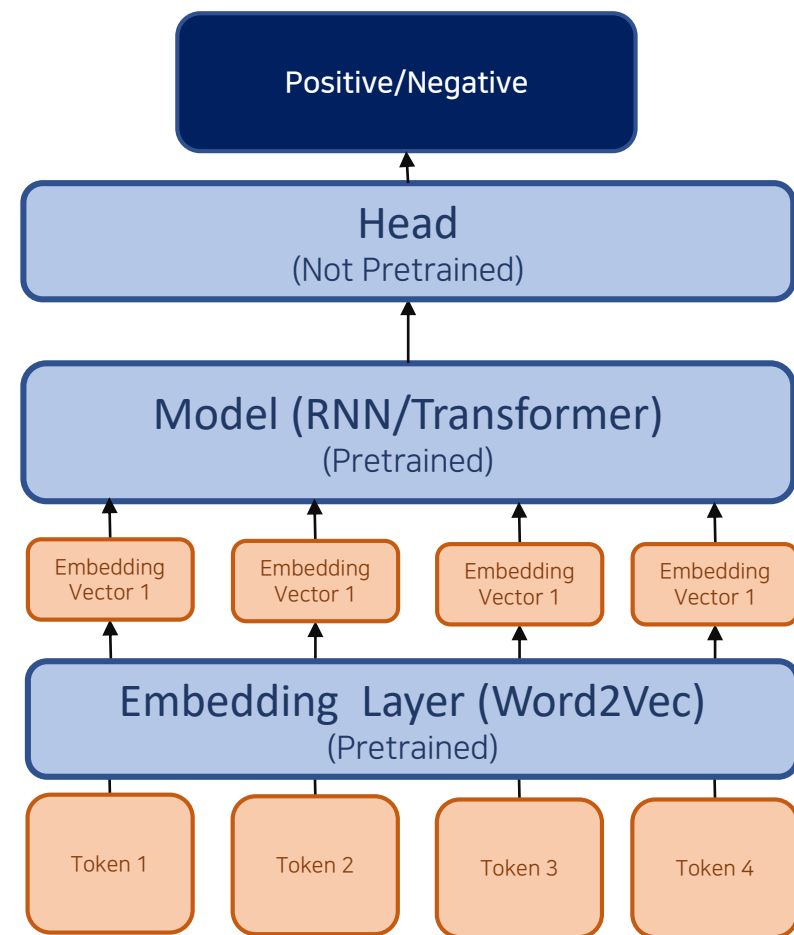
Encoders

- ✓ 양방향 토큰 정보를 활용할 수 있음
- ✓ 이전에 배운 Language Model로 pretrain 불가

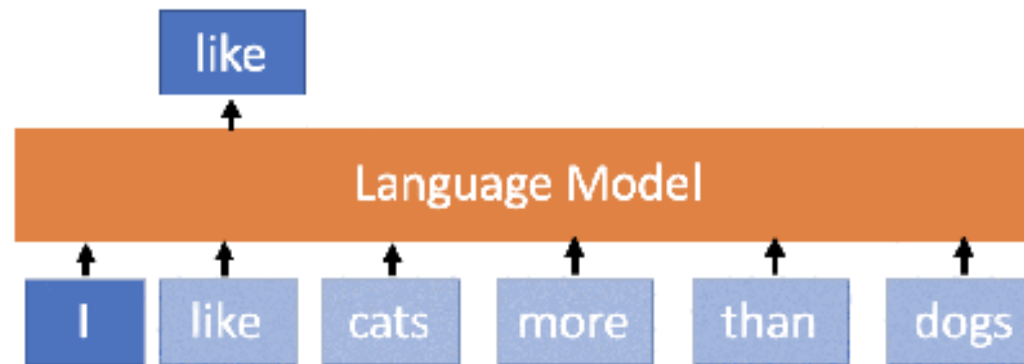
Encoder-
Decoders

- ✓ 인코더가 양방향 정보를 종합
- ✓ 디코더가 생성에 집중
- ✓ 어떤 방법으로 pretrain해야 하나?

- ✓ Pretrain task : Pretraining 시 수행하는 pretrain을 위한 태스크
- ✓ Downstream task : 실제 수행할 구체적인 태스크
- ✓ Head : pretrain된 모델에 붙어 다운스트림 태스크에 맞춘 레이어.
- ✓ Finetuning 이전에 학습이 전혀 되어 있지 않은 상태
- ✓ Pretrained model : pretrain task를 수행한 레이어
- ✓ Finetuning 이전에 이미 어느정도 학습된 상태



- ✓ 트랜스포머 디코더만 사용
- ✓ 단방향 모델 $P(w_i) = P(w_i | w_0, \dots, w_{i-1})$
- ✓ 일반적인 LM을 통해 pretrain
- ✓ Teacher Forcing 이용
- ✓ LM은 레이블이 필요없음
 - 대량의 데이터 확보 가능
 - Train data size : BooksCorpus(800M Words)



$$L = \sum_i \log P(w_i | w_0, \dots, w_{i-1}; \theta)$$

$$h_0 = UW_e + W_p$$

$$h_l = \text{Decoder block}_i(h_{l-1})$$

$$P(w_i) = \text{softmax}(h_n W_e^T)$$

- Transformer와 비교

	Transformer	GPT-1
Model Arch.	Encoder-Decoder	Decoder
model dim	512	768
FFNN dim	2048	3072
Vocab size(BPE)	3.7M	4M
# heads	8	12
# layers	6	12
# params	6.5M	110M

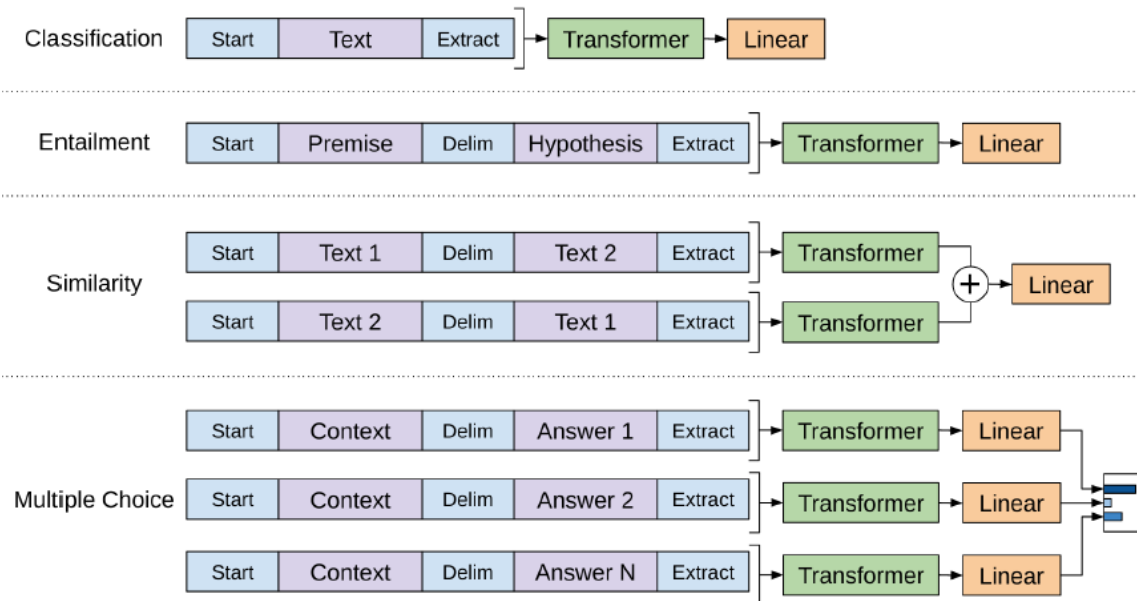


- ✓ 파라미터 수의 증가
- ✓ Pretrain 효과 발생

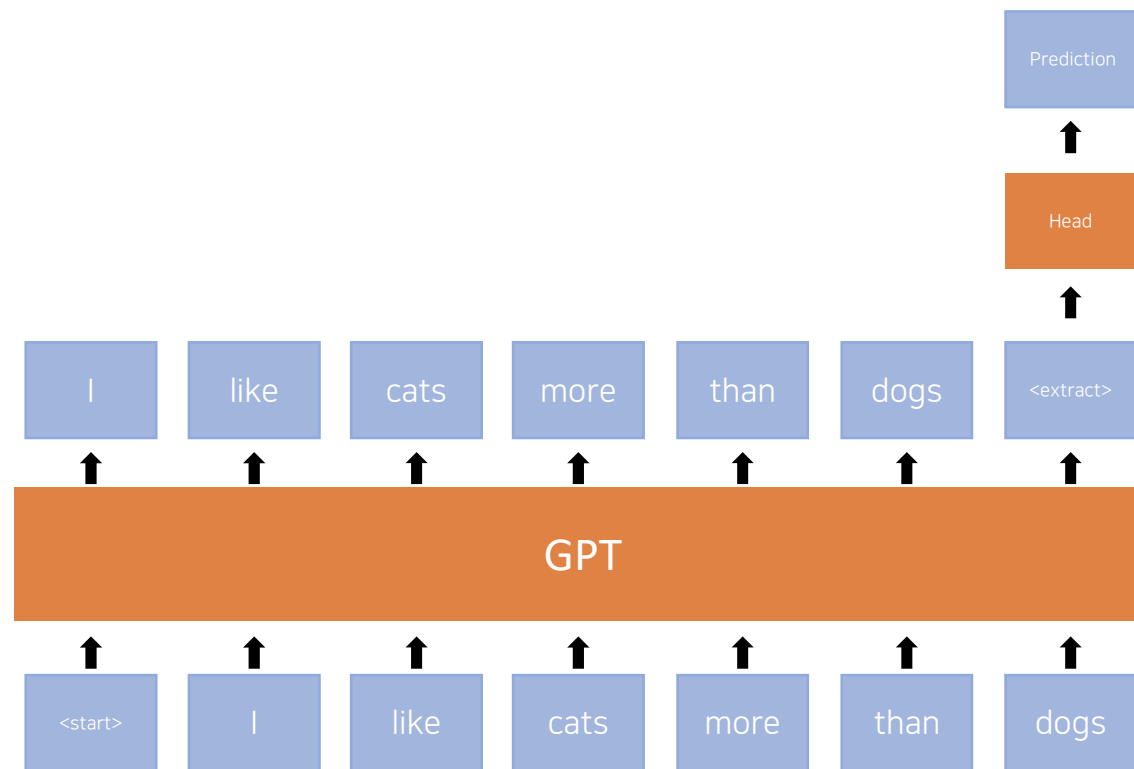
- 총 4가지 finetuning 방법론 실험

- ✓ Classification : 긍/부정, 문법 오류 여부
- ✓ Entailment : 주어진 문장들의 관계 분류
- ✓ Similarity : 두 문장 간 의미적 유사도 파악
- ✓ Multiple Choice :

주어진 문제에 대한 보기 중 정답 고르기



- ✓ 문장 종료 토큰 : <Extract>
- ✓ 문장 마지막에 <Extract>을 생성하도록 함
- ✓ Teacher Forcing 이용하여 학습
- ✓ <Extract> 위치의 출력값을 head에 입력

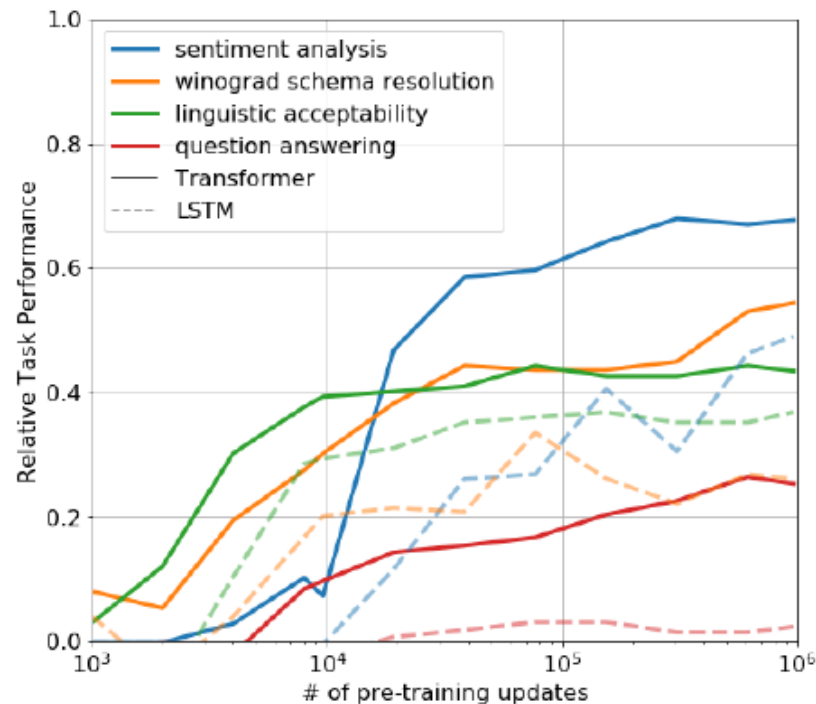
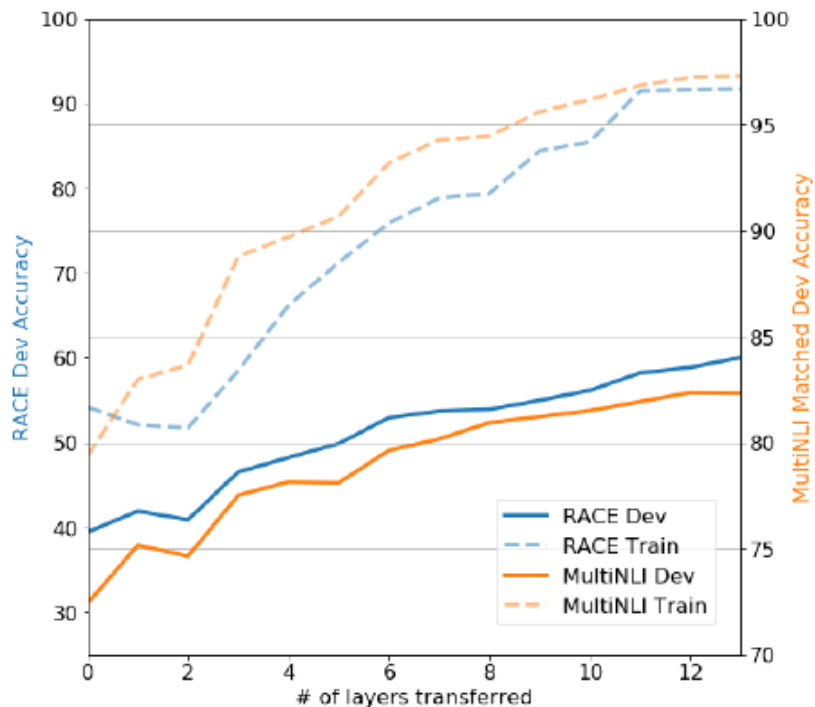


- Natural Language Inference(NLI) Result

Method	MNLI-m	MNLI-mm	SNLI	SciTail	QNLI	RTE
ESIM + ELMo [44] (5x)	-	-	<u>89.3</u>	-	-	-
CAFE [58] (5x)	80.2	79.0	<u>89.3</u>	-	-	-
Stochastic Answer Network [35] (3x)	<u>80.6</u>	<u>80.1</u>	-	-	-	-
CAFE [58]	78.7	77.9	88.5	<u>83.3</u>		
GenSen [64]	71.4	71.3	-	-	<u>82.3</u>	59.2
Multi-task BiLSTM + Attn [64]	72.2	72.1	-	-	<u>82.1</u>	61.7
Finetuned Transformer LM (ours)	82.1	81.4	89.9	88.3	88.1	56.0

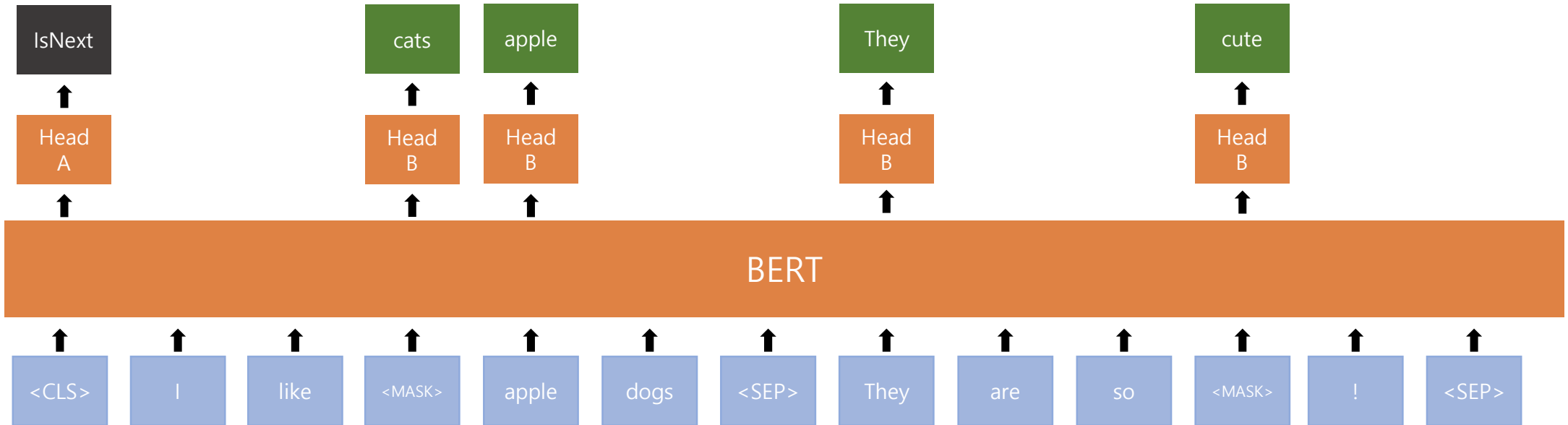
- ✓ 단일 태스크에 학습된 모델보다 좋은 성능 달성
- ✓ 다양한 태스크에 학습된 모델보다 대부분 좋은 성능 달성
- ✓ 비교적 적은 에폭(3)으로도 SOTA를 달성

- Pretrain이 다운스트림 태스크에 효과적임을 보임



- ✓ Pretrain 레이어를 사용할수록 성능이 향상되는 모습
- ✓ Pretrain을 진행한 모델일수록 성능이 향상되는 모습

- BERT의 pretrain task는 MLM과 NSP 두 가지임



- ✓ 트랜스포머 인코더만 사용
- ✓ 양방향 모델
- ✓ MLM과 NSP를 통해 학습

$$h_0 = UW_e + W_p$$

$$h_l = \text{Encoder block}_i(h_{l-1})$$

$$P(w_i) = \text{softmax}(h_n W_e^T)$$

- Masked Language Modeling (MLM)

$$L_{MLM} = -\sum_{i \in M} \log P(w_i | w_0, \dots, w_n; \theta)$$

(M : Masked tokens, n : sequence length)

- ✓ 양방향 모델 - 양방향 토큰의 정보를 활용
 - Language Model 목적함수 사용 불가
- ✓ 입력 토큰의 15%를 변형하고, 이를 예측
 - 80% : [MASK]
 - 10% : 랜덤한 다른 토큰
 - 10% : 원본 토큰



- ✓ Inference 상황과 동일한 환경
- ✓ 모델이 언어구조를 학습

- Next Sentence Prediction (NSP)

$$L_{NSP} = -\log \hat{y}P(IsNext | w_0, \dots, w_n) - (1 - \hat{y})(1 - P(IsNext | w_0, \dots, w_n))$$

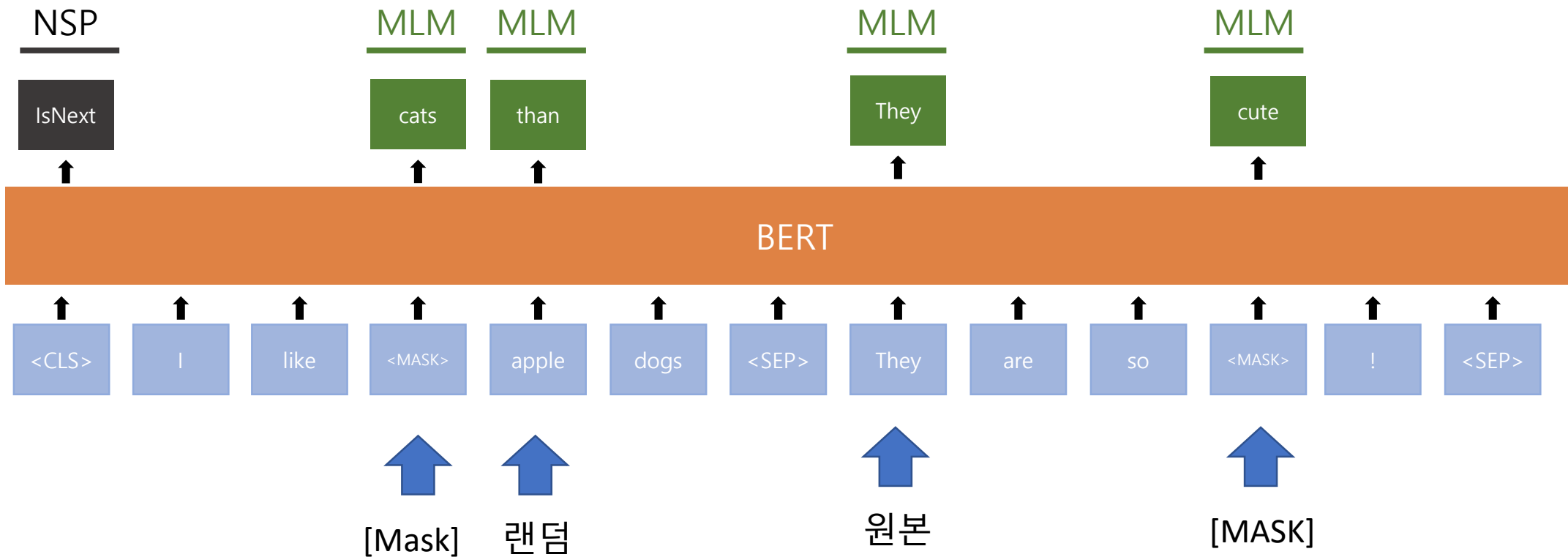
(M : Masked tokens, n : sequence length)

- ✓ 많은 다운스트림 태스크에서 문장 간 관계를 다룸
- ✓ 이를 반영한 pretrain task 필요
- ✓ 두 문장 입력
 - ✓ 50% : 학습 데이터에서 연속된 문장
 - ✓ 50% : 서로 다른 문서에서 사용된 문장
- ✓ 연속된 문장 여부 판단(IsNext)



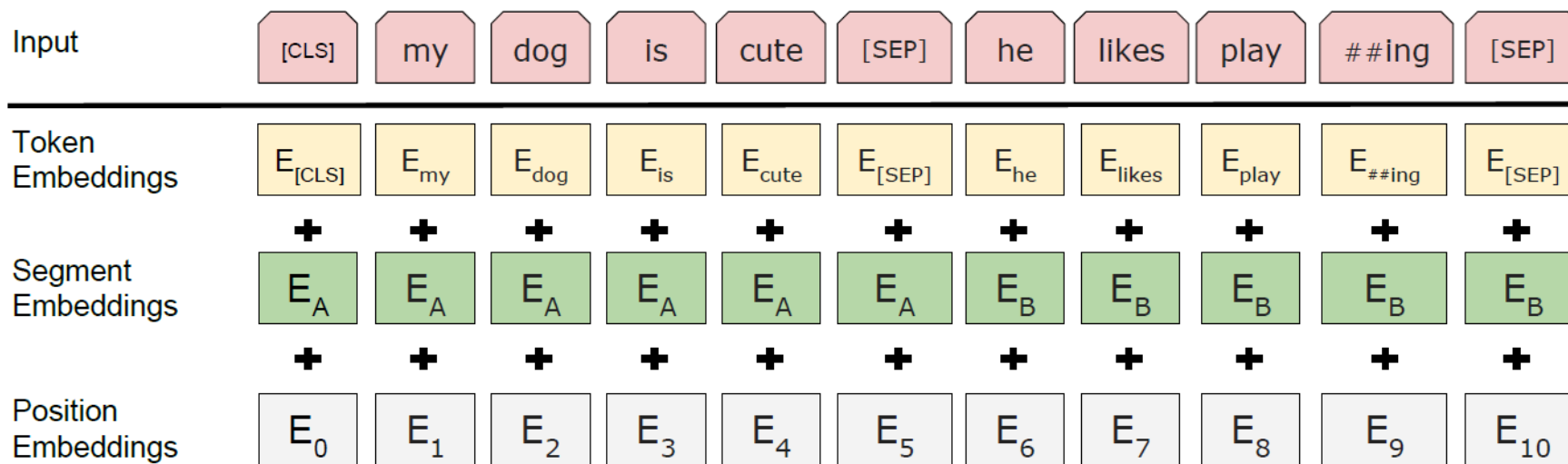
- ✓ 문장 간 관계 학습
- ✓ <CLS> 토큰이 입력 문장의 정보를 담음

- Pretrain Tasks의 모습



=> 이후 RoBerta에서 NSP가 학습에 도움이 되지 않고, Bert가 과소적합 되어 있음을 보임

- BERT는 Token, Segment, Position 3 가지의 임베딩 벡터를 더하여 입력 임베딩을 생성



- Token Embedding : Wordpiece 기반 토큰 단위 임베딩
- Position Embedding : 트랜스포머의 positional encoding
- Segment Embedding : 두 문장을 구분하기 위한 임베딩

- **두가지 버전의 모델 발표**
 - BERT-Base : 12 layer, 768 hidden state dim, 12 attention head, 110M params
 - BERT-Large : 24 layers, 1024 hidden state dim, 16 attention head, 340M params
- **학습 데이터**
 - BooksCorpus(800M words) -> GPT pretrain 데이터
 - English Wikipedia(2500M words)
- Pretraing은 64개의 TPU로 4일에 걸쳐 이루어짐
- Finetuning은 단일 GPU로 이루어짐

- GPT, Transformer와의 비교

	Transformer	GPT-1	BERT
Model Arch.	Encoder-Decoder	Decoder	Encoder
model dim	512	768	768
FFNN dim	2048	3072	3072
Vocab size(BPE)	3.7M	4M	3.7M
# heads	8	12	12
# layers	6	12	12
# params	6.5M	110M	110M



- ✓ GPT-1과 동일한 파라미터 수
- ✓ Model arch만 다름

- 다양한 태스크에 대해 SOTA의 성능 달성

System	MNLI-(m/mm) 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k	Average -
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
BERT _{BASE}	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
BERT _{LARGE}	86.7/85.9	72.1	92.7	94.9	60.5	86.5	89.3	70.1	82.1

- ✓ GPT-1(125M) 보다 적은 파라미터(110M)로 더 좋은 성능을 보임
 - 양방향 모델링에서 오는 이점
- ✓ 모델의 크기를 키움으로서 더 성능을 높일 수 있음을 보임.
- ✓ 하지만 인코더 모델의 한계로 생성 다운스트림 태스크에 파인튜닝이 힘들.

- Bert의 구조를 그대로 가져오되, NSP를 제외하고 추가적인 하이퍼파라미터 튜닝 진행

Model	data	bsz	steps	SQuAD (v1.1/2.0)	MNLI-m	SST-2
RoBERTa						
with BOOKS + WIKI	16GB	8K	100K	93.6/87.3	89.0	95.3
+ additional data (§3.2)	160GB	8K	100K	94.0/87.7	89.3	95.6
+ pretrain longer	160GB	8K	300K	94.4/88.7	90.0	96.1
+ pretrain even longer	160GB	8K	500K	94.6/89.4	90.2	96.4
BERT _{LARGE}						
with BOOKS + WIKI	13GB	256	1M	90.9/81.8	86.6	93.7

- ✓ Bert의 성능을 더 높일 수 있음을 밝힘
- ✓ Pretrain 시 추가적인 데이터셋과 에포크로 더 높은 성능 달성

Original text

Thank you ~~for inviting~~ me to your party ~~last~~ week.

Inputs

Thank you <X> me to your party <Y> week.

Targets

<X> for inviting <Y> last <Z>

- ✓ 트랜스포머 구조를 가져오되 데이터셋과 모델을 키우고 pretrain 과정을 정교화
- ✓ 매우 다양한 다운스트림 태스크에 대한 성능 평가 진행
- ✓ Pretrain task로 span corruption 채택

- T5의 pretrain task

Original text

Thank you ~~for inviting~~ me to your party ~~last~~ week.

Inputs

Thank you <X> me to your party <Y> week.

Targets

<X> for inviting <Y> last <Z>

- ✓ 인코더의 입력으로 MASK 사용
- ✓ 디코더의 출력으로 MASK ID와 해당하는 문장 생성
- ✓ 인코더는 MLM을 디코더는 LM을 수행하는 pretrain task
- ✓ 이를 통해 GPT보다 생성을 잘하고, BERT보다 파악을 잘하는 모델 구성

- BERT MASK와 다른 점

- ✓ 1대1 MASKING vs 1대다 MASKING

Span length	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ Baseline (i.i.d.)	83.28	19.24	80.88	71.36	26.98	39.82	27.65
2	83.54	19.39	82.09	72.20	26.76	39.99	27.63
3	83.49	19.62	81.84	72.53	26.86	39.65	27.62
5	83.40	19.24	82.05	72.23	26.88	39.40	27.53
10	82.85	19.33	81.84	70.44	26.79	39.49	27.69

- ✓ 전체 토큰 중 15%를 25개의 MASK로 MASKING

- ✓ BERT는 모델이 하나의 토큰을 복원하지만, T5는 마스킹된 부분을 생성해야함

- BERT MASK vs T5 MASK

ORIGINAL : I like cats than dogs, cause they are so cute.

BERT : I <MASK> toMato <MASK> dogs, cause they are so cute.

T5 : I <MASK1> <MASK2> dogs, cause <MASK3> cute.

- GPT vs T5

I like cats than → GPT-1 → dogs

I <MASK1> <MASK2> dogs, cause <MASK3> cute.

↓
→ T5 → <MASK1> cats<MASK2>than<MASK3> they are

- 기존의 Question & Answering task : 질문과 정답이 포함된 문장이 입력되면 모델이 정답 생성

✓ Input : 안암에서 제일 맛있는 돈까스 집이 어디야?

✓ Context :

안암에서 제일 맛있는 돈가스집

예로부터 안암 정후는 돈가스집들의경쟁이 되게 치열했었죠 저는 그중에서 지푸라기를 제일 좋아했는데요

군대 복무중에 사라졌더라구요 너무 슬펐는데 동기가 **돈가스냉면**이라구 먹으러 가자고 했습니다.처음엔 무슨 그런 해괴한 음식을 먹냐고 뭐라 했지만 반강제로 따라가 먹은 돈가스는 지푸라기를 잇는 제 최애 돈가스집이 되었습니다

저는 냉면도 좋아하는 데 냉면도 안암에서 제일 맛있다고 느꼈습니다.

더 중요한건 그걸 같이 줘요

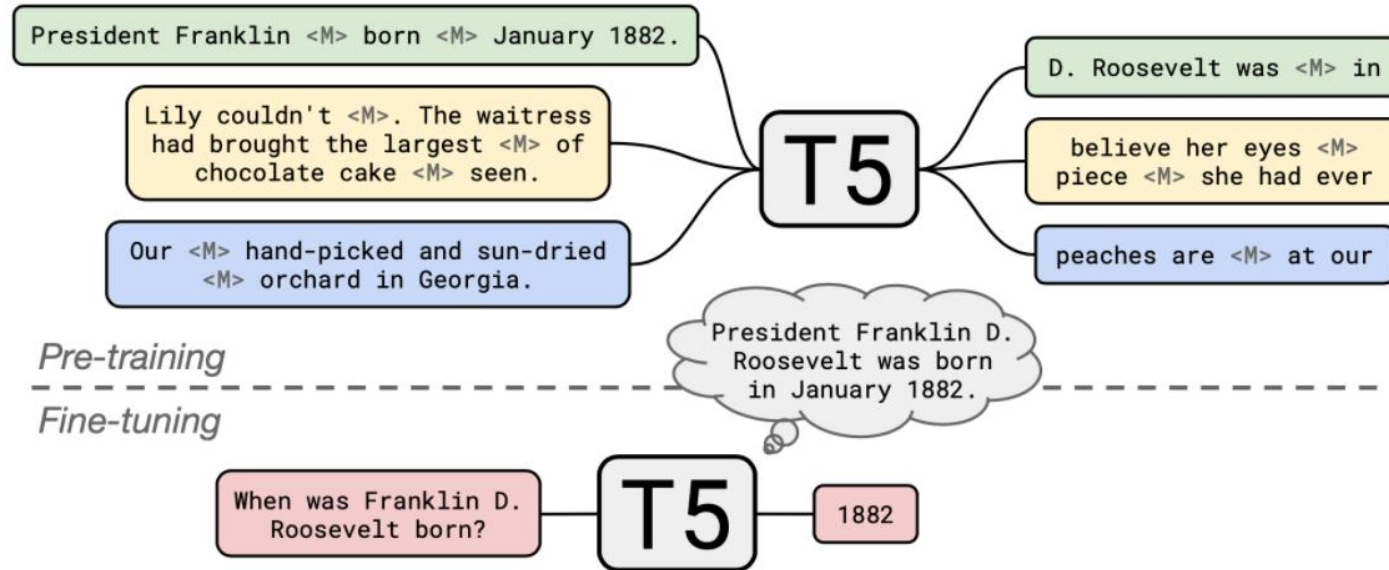
그날 저는 제 오버워치 아이디를 안암동돈가스냉면으로 바꿨습니다

이 집의 단점을 말하자면 돈가스를 너무 많이 줘서 남기게 만듭니다. 전 꾸역꾸역 먹으면 도미노 라지도 혼자서 한판 먹는데 이 집에서는 과식 안하려고 조금 남겨요. 정말 양이 많습시다 많이 못드시는 분들은 두분 가셔서 냉면 하나 돈가스 하나 시키시는걸 추천드려요

위치는 정대후문 더진국 국밥가게 뒤편 예전스파게티 팔던 자리에 있습니다 정확한 상호는 잘 기억 안 나네요 전 돈가스냉면이라구 불러서 이 집 좋아하는데 손님이 생각보다 없어서 추천드립니다 정말 맛있어요

✓ Output : 돈가스냉면

- T5는 질문만 주어져도 어느정도 답변이 가능함

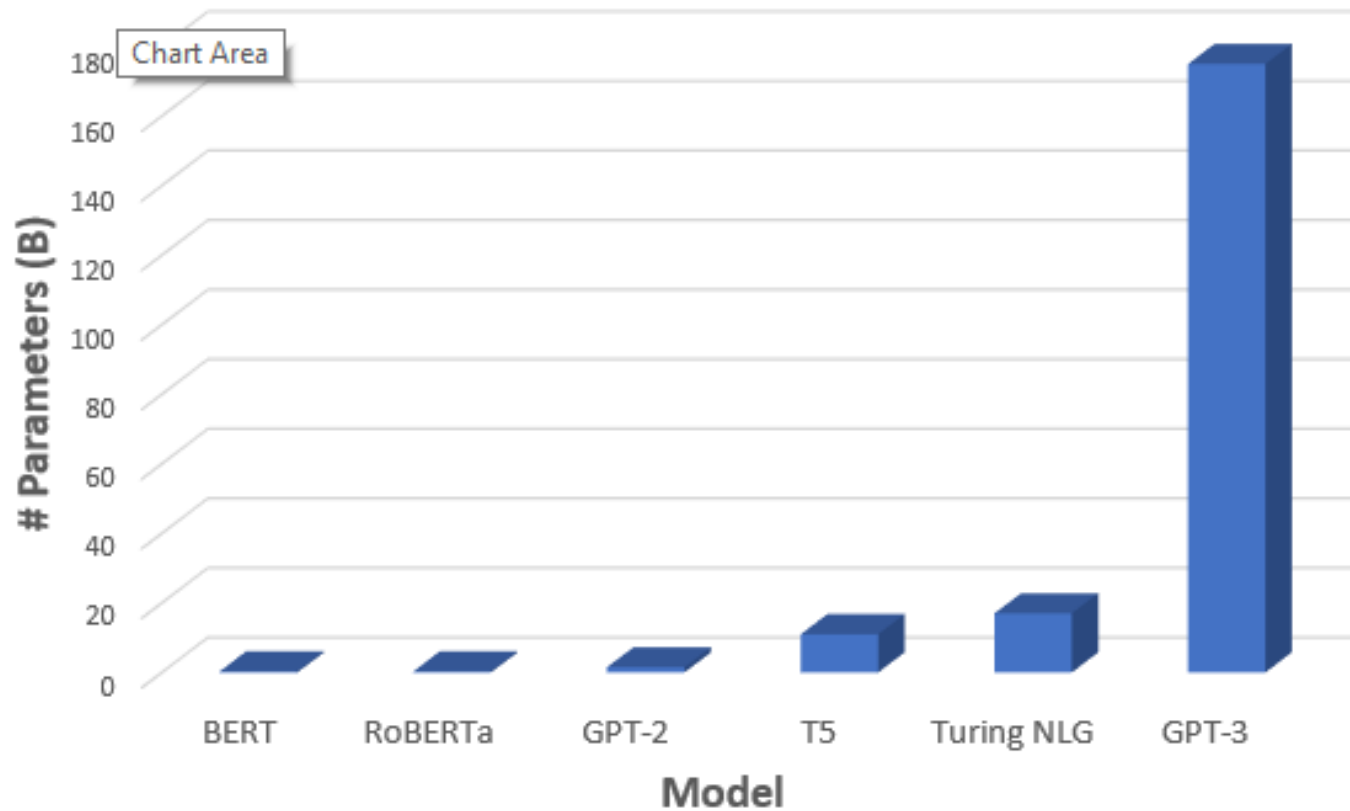


- 이는 정말 대량의 데이터셋(750GB)으로 학습했기 때문
- 모델이 정보를 파라미터화 하여 기억
 - ✓ Input : 안암에서 제일 맛있는 돈까스 집이 어디야?
 - ✓ Output : 돈가스 냉면

- GPT, BERT, Transformer와의 비교 (large 기준)

	Transformer	GPT-1	BERT	BERT
Model Arch.	Encoder-Decoder	Decoder	Encoder	Encoder-Decoder
model dim	512	768	1,024	1,024
FFNN dim	2,048	3,072	3,072	65,536
Vocab size(BPE)	3.7M	4M	3.7M	3.2M
# heads	8	12	16	128
# layers	6	12	24	24
# params	6.5M	110M	340M	11B = 11000M

- 정말 많은 대형 사전학습 모델들



- ✓ 기하급수적으로 늘어나는 훈련 데이터 및 모델 크기
- ✓ 태스크 별 모델 설계에서 사전학습 모델로 패러다임 변화

사전 학습 = 큰 모델 + 대량 데이터 + 적절한 태스크

- 모델의 성능은 파라미터 수와 비례
 - ✓ 과적합을 방지하기 위해서는 대량의 데이터가 필수
 - ✓ 대량의 데이터와 큰 모델은 많은 자원을 소모
- 효과적인 학습을 위해서는 적절한 pretrain task를 구성해야 함
- 적절한 pretrain task, 대량의 데이터, 거대 모델은 매우 효과적인 fine tuning을 보장함

- T5는 최적의 사전학습 방법론을 찾기 위해 정말 많은 태스크에 정말 많은 실험을 수행

Table	Experiment	GLUE												SuperGLUE												WMT																																																																																																																																																																																																																																																																																																																																																																																																																																																															
		Score	CoLA	SST-2	MRPC	MLPC	STSB	GLUE	QQP	QQP	MNLI	MNLI _{mm}	QNLI	RTE	CoLA	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}	CoLA _{mm}

감사합니다