KUBIG 25-W
겨울방학 BASIC STUDY SESSION

NLP SESSION
WEEK5

KUBIG
DATA SCIENCE & AI

CONTENTS

KUBIG
DATA SCIENCE & AI

01 우수 과제 코드 Review, Transformer reminder

KUBIG
DATA SCIENCE & AI

1-1. 우수 과제 코드 Review

KUBIG DATA SCIENCE & AI

장건호 님

4주차
복습과제

Transformer 챗봇 구현

강서연 님

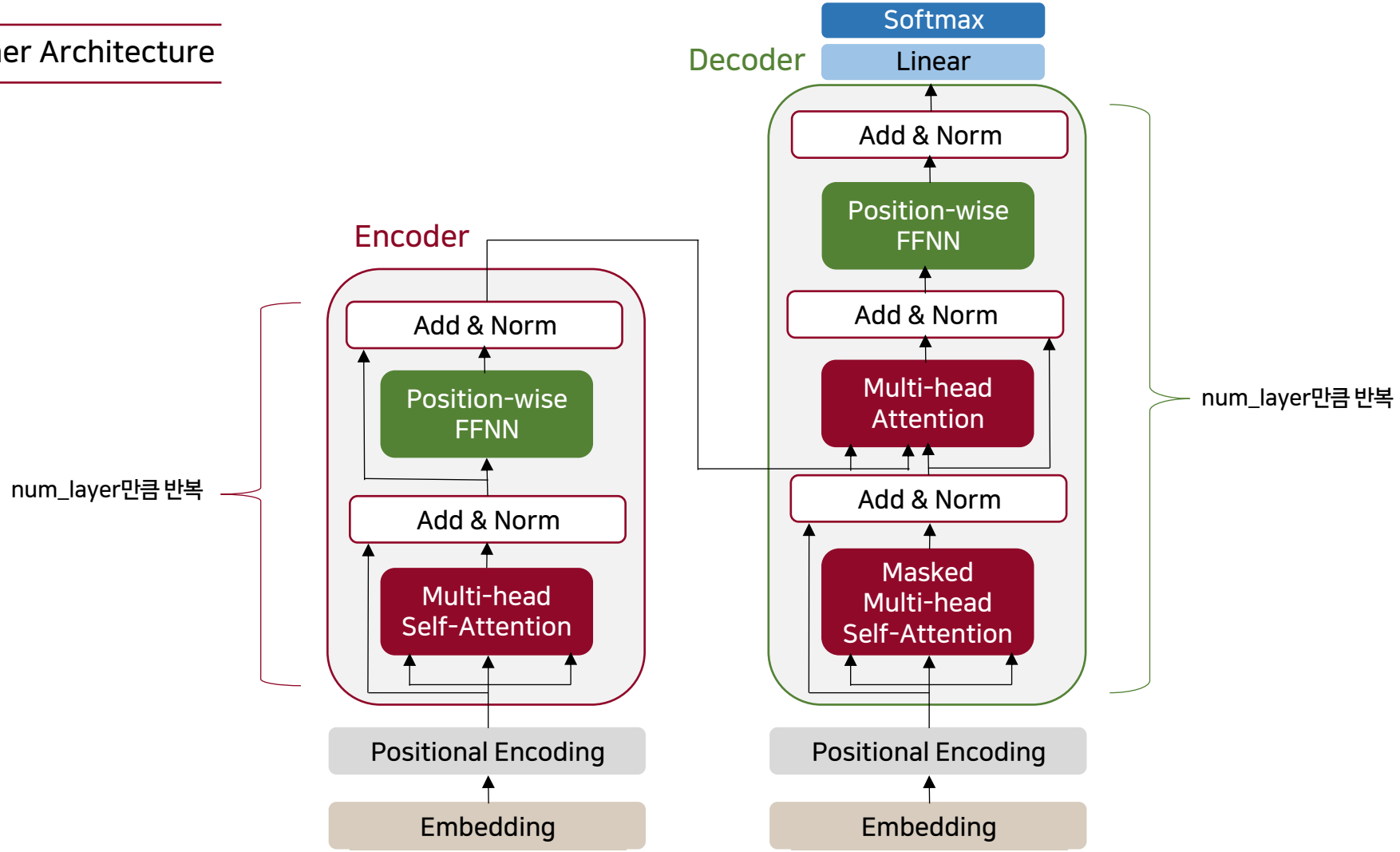
4주차
예습과제

BERT Pytorch
Koalpaca 문장생성

3분 내외로 가볍게 리뷰해주시면 됩니다!

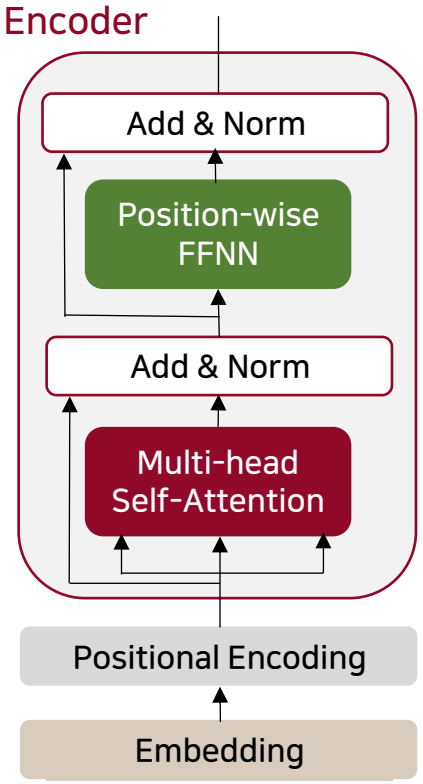
1-2. Transformer reminder

Transformer Architecture



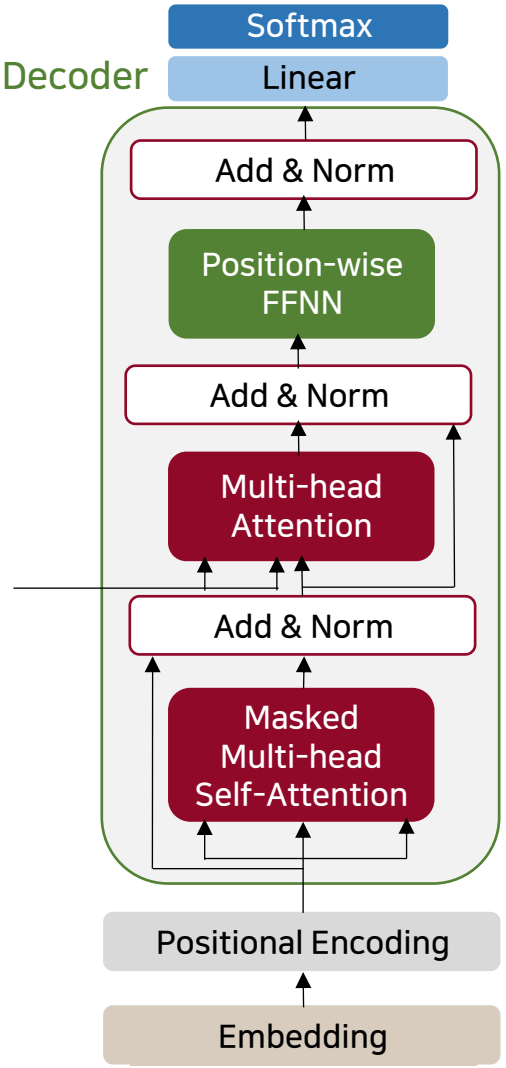
BERT

Bidirectional Encoder
Representations from **Transformers**



GPT

Generative Pre-trained
Transformer



02 Fine-tuning paradigm

Pretrained-LM의 등장

KUBIG
DATA SCIENCE & AI

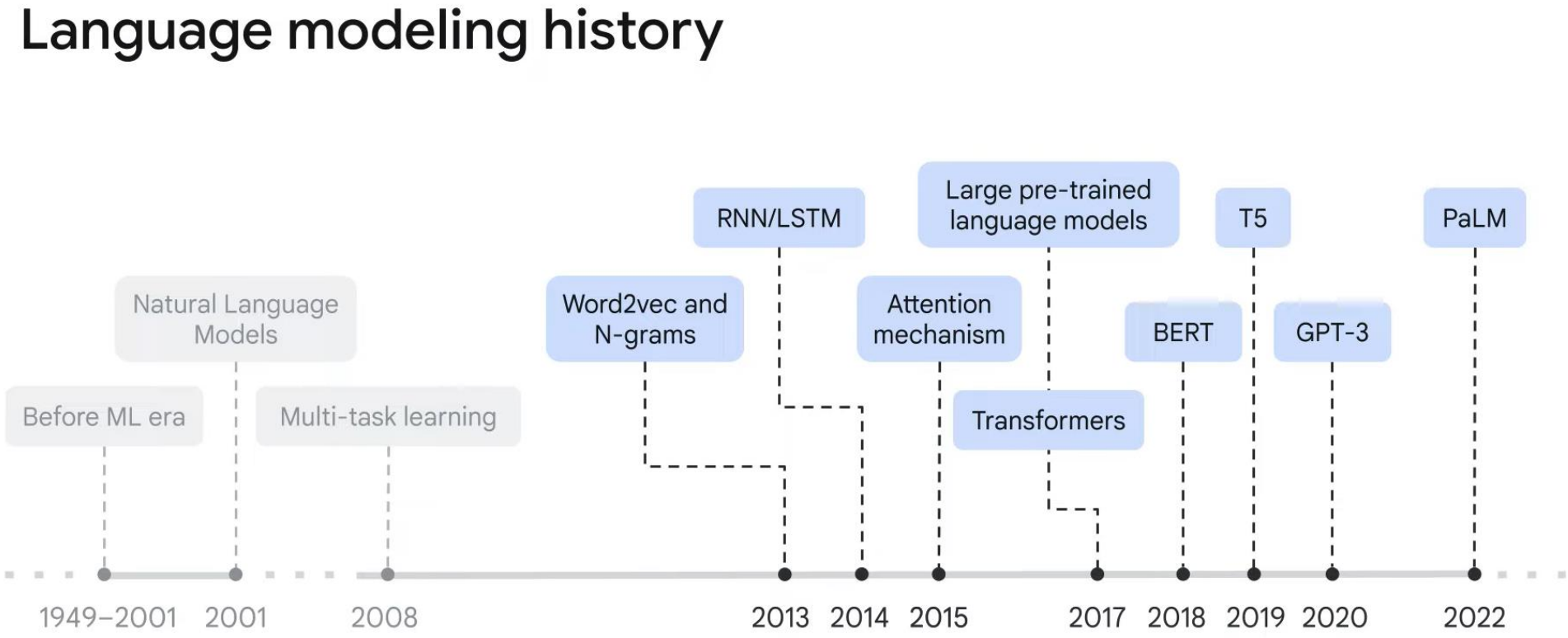
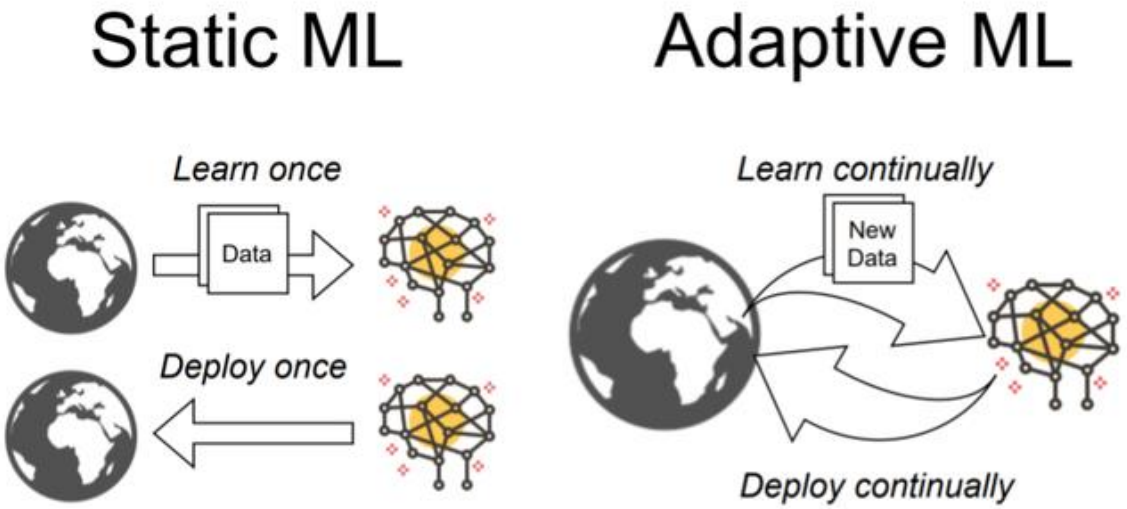


그림: Google Cloud Tech "Transformer models and BERT model: Overview"

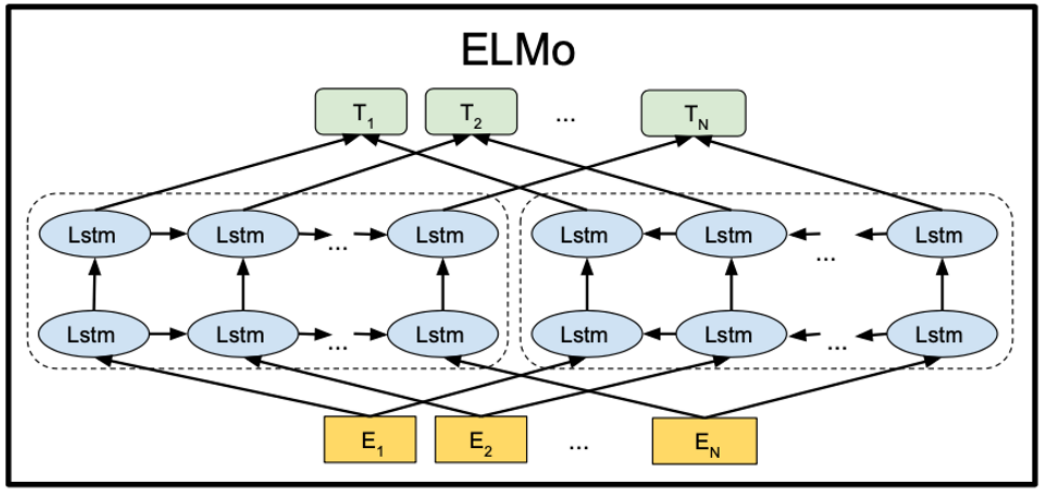


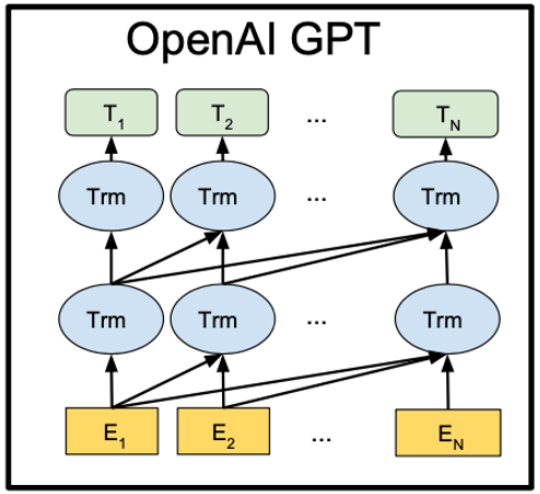
1. Adapter-based approach ex) LoRA
2. Replay-based approach
3. Feature-based approach
4. Representation-based approach ex) fine-tuning
5. Regularization-based approach ex) SPG

...

Continual Learning

이전 데이터에 대한 일반화 기능을 계속 유지하면서
새로운 데이터에 대해서도 잘 일반화할 수 있도록 학습



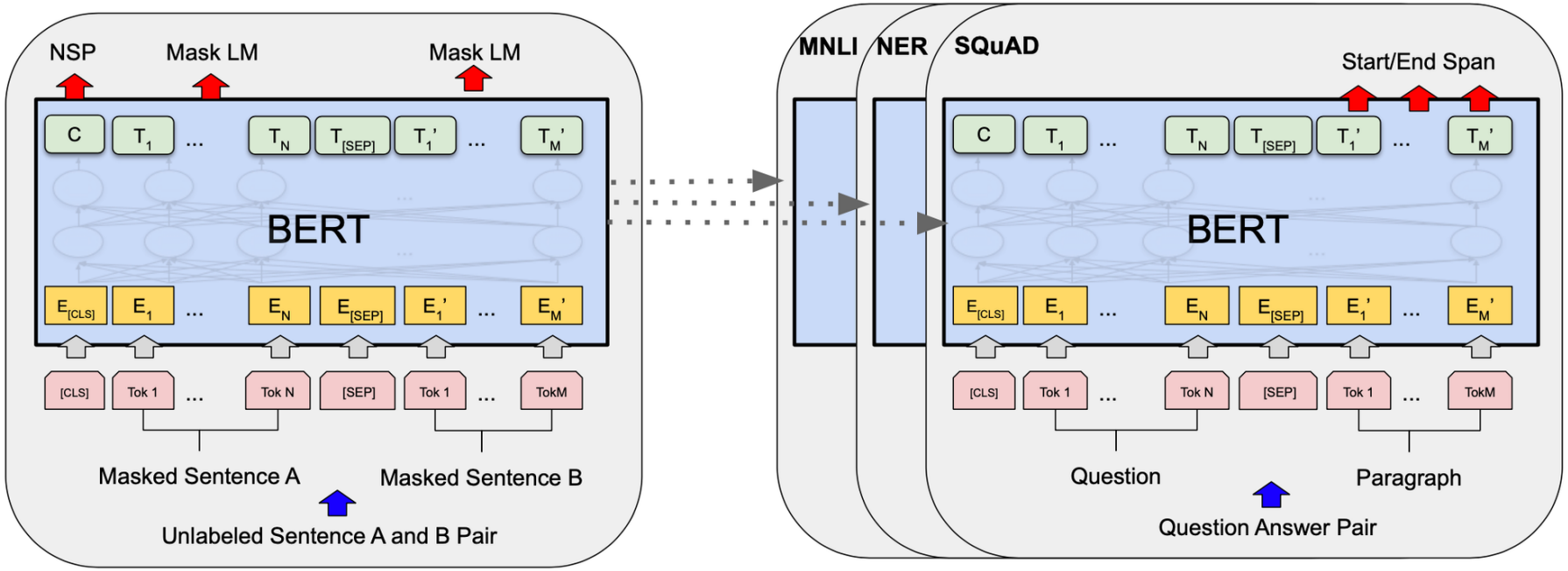


Feature-based (ELMo)

context-sensitive feature를 양방향 LSTM에서 각각 추출,
hidden layers를 concat하는 방식.
특정 단어에 쌓인 embedding들을 선형 결합하여 가중치를 조절.

Fine-tuning

unlabeled text에서 pre-train되고
supervised down-stream task에서 fine-tuning.
모든 parameter를 update.



Pre-training

레이블이 지정되지 않은 텍스트를 사전 학습
(unsupervised learning)

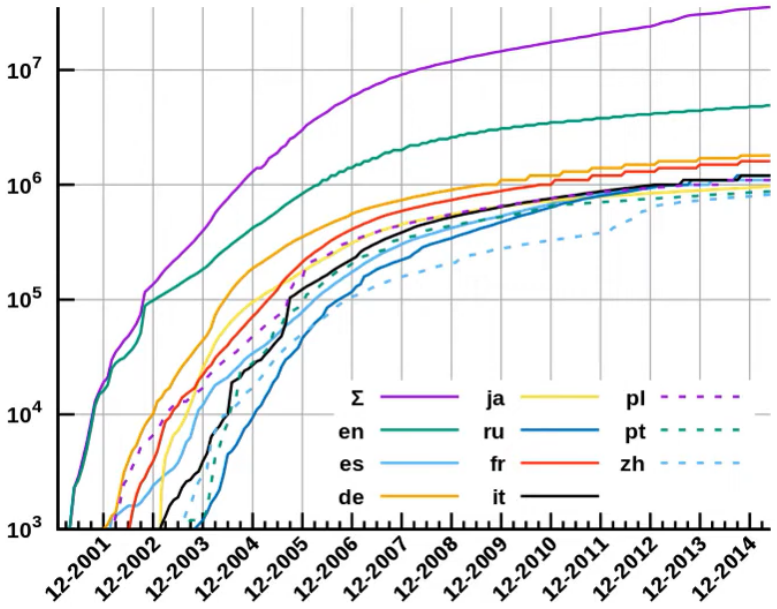
Fine-Tuning

다양한 downstream task 에 대해
레이블이 지정된 데이터로 미세 조정
(supervised learning)

2-3. Fine-tuning paradigm

KUBIG DATA SCIENCE & AI

Unlabeled dataset



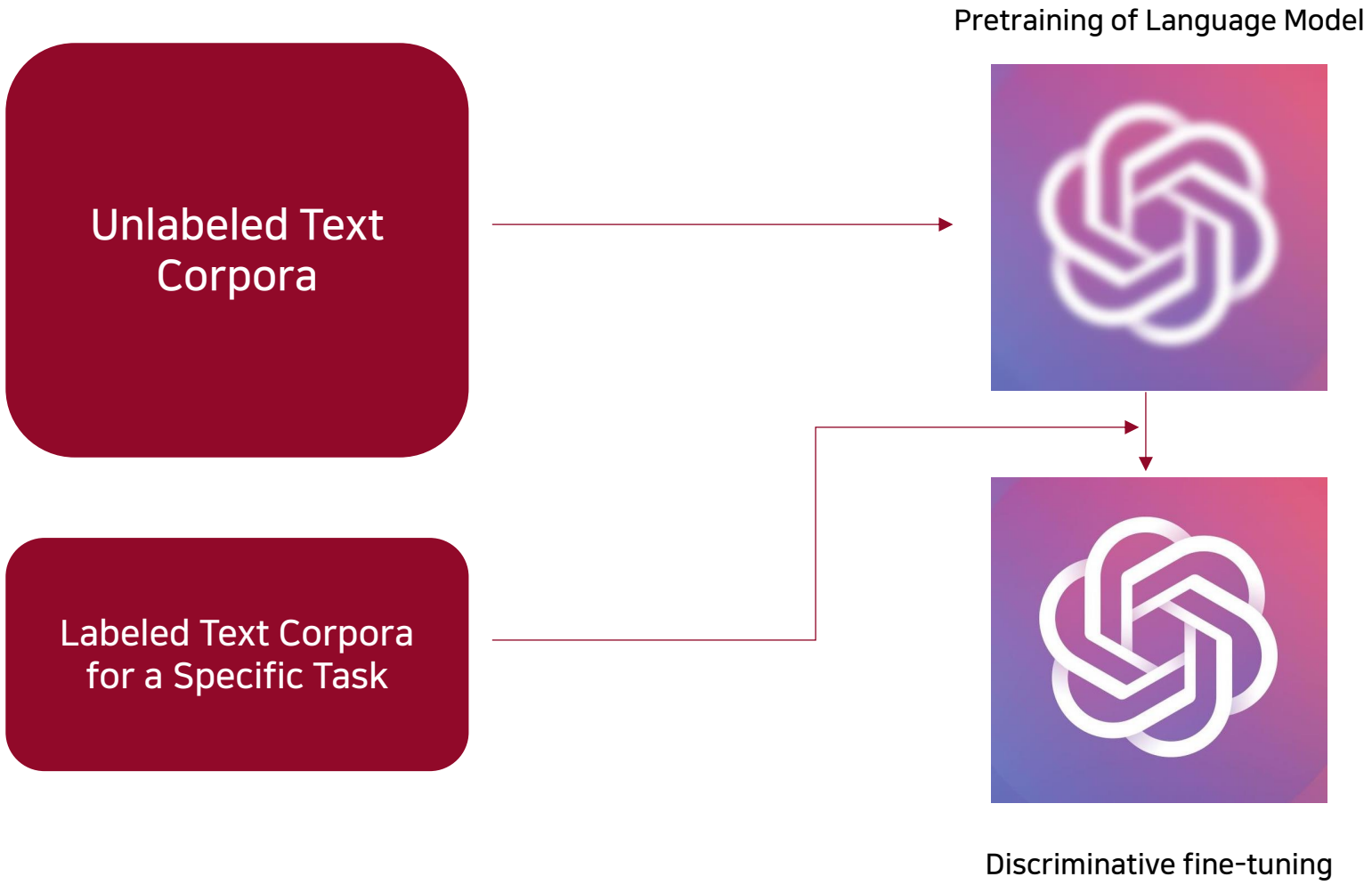
As of 24 February 2020, there are 6,020,081 articles in the English Wikipedia containing over 3.5 billion words.

Labeled dataset

- STS Benchmark for sentence similarity: 8,628 sentences
- Quora question pairs: 404,290 question pairs
- CoLA dataset: 10,657 sentences

그림: 고려대학교 산업공학과 DSBA "08-4 GPT"

2-3. Fine-tuning paradigm



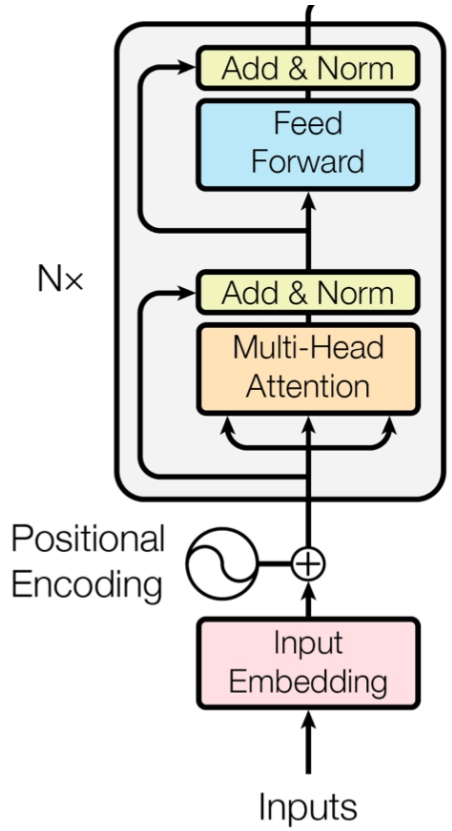
03 BERT

Fine-tuning paradigm의 시작

KUBIG DATA SCIENCE & AI

BERT (Bidirectional Encoder Representations from Transformers)

Encoder



- 초기 트랜스포머 모델 : L = 6, H = 512, A = 8
- BERT-Base : L=12, H=768, A=12 → tot params = 110M
- BERT-Large : L=24, H=1024, A=16 → tot params = 340M

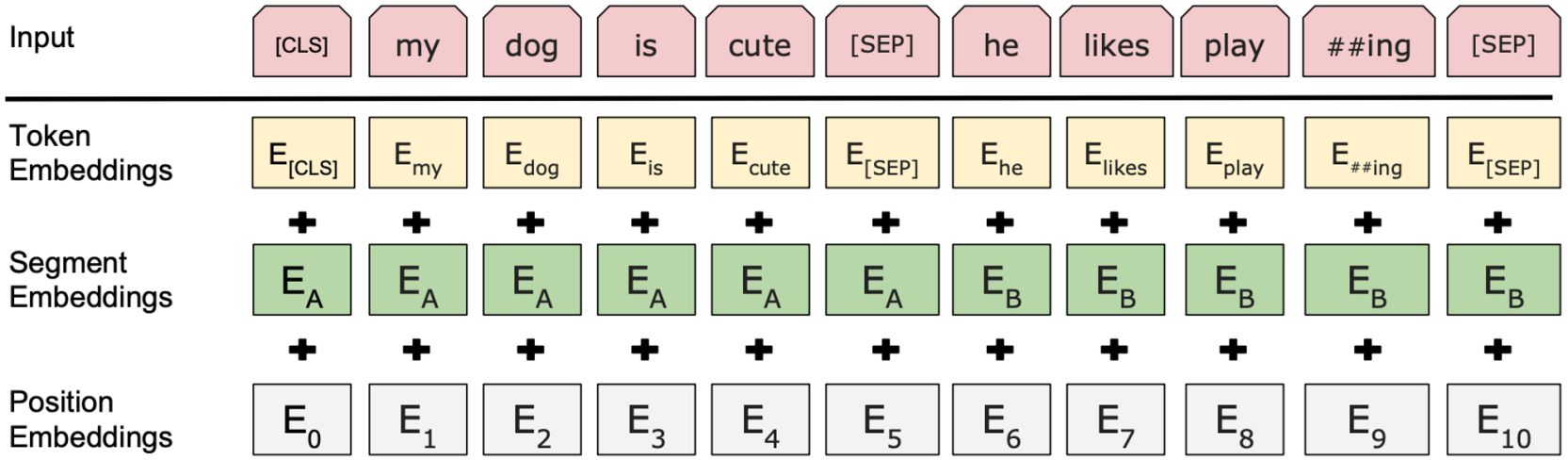
* L: # of layers (Transformer block) / H: hidden size / A: # of self attention heads

Input of BERT: sentence / sequence

To make BERT handle a variety of down-stream tasks,
our input representation is able to unambiguously represent
both a single sentence(=sentence) and a pair of sentences(=sequence)
(e.g., <Question, Answer>) in one token sequence.

- Sentence: 실제 언어 문장이 아닌 임의의 연속 텍스트 span
- Sequence: 단일 sentence거나 함께 묶인 두 sentences로, BERT의 입력 토큰 sequences

3-2. Input/Output Representations



- Token Embeddings: Word-Piece embedding을 사용.

  첫 번째 토큰은 [CLS]이고 문장 쌍을 사용할 때는 분리 토큰 [SEP]을 포함.
- Segment Embeddings: 토큰이 문장 A 또는 B에 속하는지
- Position Embeddings: Transformer와 비슷한 positional encoding

Token Embeddings: Word-Piece embedding을 사용

BPE의 변형 알고리즘으로, 빈도수 기반의 BPE와 달리
코퍼스의 likelihood를 가장 높이는 쌍을 병합하는 방식

Byte Pair Encoding(BPE)

- 대표적인 sub-word segmentation 알고리즘
- 글자(Character) 단위에서 점차 단어 집합(vocabulary)를 만들어내는 Bottom up 방식
- 가장 많이 등장(=빈도수 기반)하는 유니그램을 하나의 유니그램으로 병합하는 방식

→ OOV(Out-Of-Vocabulary) 문제 완화

3-3. Embedding of BERT

```
# dictionary
# 훈련 데이터에 있는 단어와 등장 빈도수
low : 5, lower : 2, newest : 6, widest : 3
```

low가 5회, lower가 2회, .. 등장하는 훈련 데이터
이 딕셔너리로부터 단어 집합(vocabulary)를 얻음

```
# vocabulary
low, lower, newest, widest
```

테스트 과정에서 'lowest'란 단어가 등장한다면 기계는 이 단어를 학습한 적이 없으므로
해당 단어에 대해서 제대로 대응하지 못하는 OOV 문제가 발생

BPE는 dictionary와
vocabulary가 어떻게 구성될까?

https://wikidocs.net/22592

3-3. Embedding of BERT

KUBIG DATA SCIENCE & AI

BPE

```
# dictionary
l o w : 5,  l o w e r : 2,  n e w e s t : 6,  w i d e s t : 3
```

```
# vocabulary
l, o, w, e, r, n, s, t, i, d
```

Update 1

```
# dictionary update!
l o w : 5,
l o w e r : 2,
n e w es t : 6,
w i d es t : 3
```

```
# vocabulary update!
l, o, w, e, r, n, s, t, i, d, es
```

https://wikidocs.net/22592

BPE

Update 2

```
# dictionary update!
l o w : 5,
l o w e r : 2,
n e w est : 6,
w i d est : 3
```

```
# vocabulary update!
l, o, w, e, r, n, s, t, i, d, es, est
```

Update 3

```
# dictionary update!
lo w : 5,
lo w e r : 2,
n e w est : 6,
w i d est : 3
```

```
# vocabulary update!
l, o, w, e, r, n, s, t, i, d, es, est, lo
```

https://wikidocs.net/22592

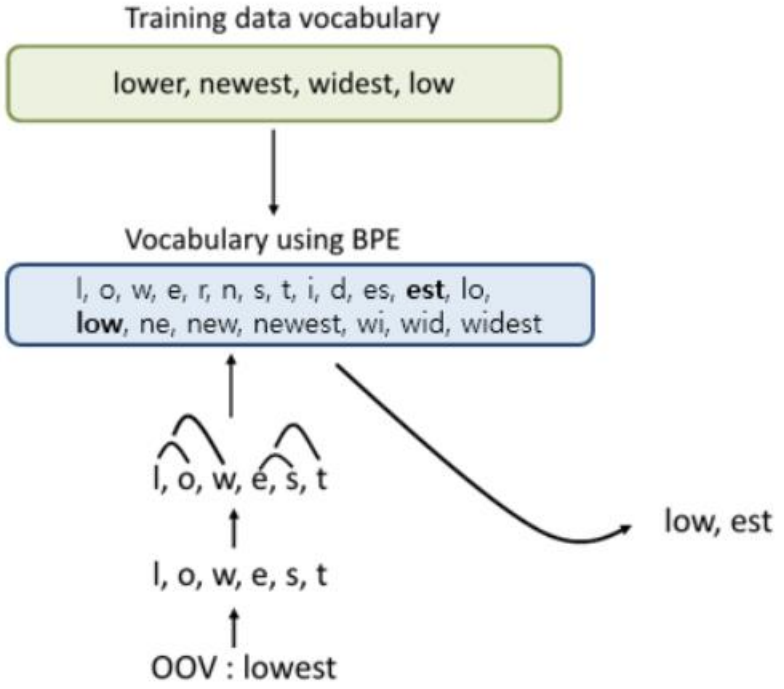
BPE

⋮

Update 10

```
# vocabulary update!
l, o, w, e, r, n, s, t, i, d, es, est, lo, low, ne, new, newest, wi, wid, widest
```

'lowest'를 'low'와 'est' 두 단어로 인코딩 → lowest는 더이상 OOV가 아님!



https://wikidocs.net/22592

3-3. Embedding of BERT

KUBIG DATA SCIENCE & AI

```python
class BertEmbeddings(nn.Module):
    """Construct the embeddings from word, position and token_type embeddings."""

    def __init__(self, config):
        super().__init__()
        self.word_embeddings = nn.Embedding(config.vocab_size, config.hidden_size, padding_idx=config.pad_token_id)
        self.position_embeddings = nn.Embedding(config.max_position_embeddings, config.hidden_size)
        self.token_type_embeddings = nn.Embedding(config.type_vocab_size, config.hidden_size)

        # self.LayerNorm is not snake-cased to stick with TensorFlow model variable name and be able to load
        # any TensorFlow checkpoint file
        self.LayerNorm = nn.LayerNorm(config.hidden_size, eps=config.layer_norm_eps)
        self.dropout = nn.Dropout(config.hidden_dropout_prob)
        # position_ids (1, len position emb) is contiguous in memory and exported when serialized
        self.position_embedding_type = getattr(config, "position_embedding_type", "absolute")

    def forward(
        self,
        input_ids: Optional[torch.LongTensor] = None,
        token_type_ids: Optional[torch.LongTensor] = None,
        position_ids: Optional[torch.LongTensor] = None,
        inputs_embeds: Optional[torch.FloatTensor] = None,
        past_key_values_length: int = 0,
    ) -> torch.Tensor:

        embeddings = inputs_embeds + token_type_embeddings
        if self.position_embedding_type == "absolute":
            position_embeddings = self.position_embeddings(position_ids)
            embeddings += position_embeddings
        embeddings = self.LayerNorm(embeddings)
        embeddings = self.dropout(embeddings)
        return embeddings
```

Token Embeddings

Position Embeddings

Segment Embeddings

BERT tokenizer

입력: I love NLP!

```python
tokens = tokenizer("I love NLP!")

print(f"input ids : {tokens['input_ids']}")
print(f"token type ids : {tokens['token_type_ids']}")
print(f"attention mask : {tokens['attention_mask']}")
```

[실행 결과]

```
input ids : [101, 1045, 2293, 17953, 2361, 999, 102]
token type ids : [0, 0, 0, 0, 0, 0, 0]
attention mask : [1, 1, 1, 1, 1, 1, 1]
```

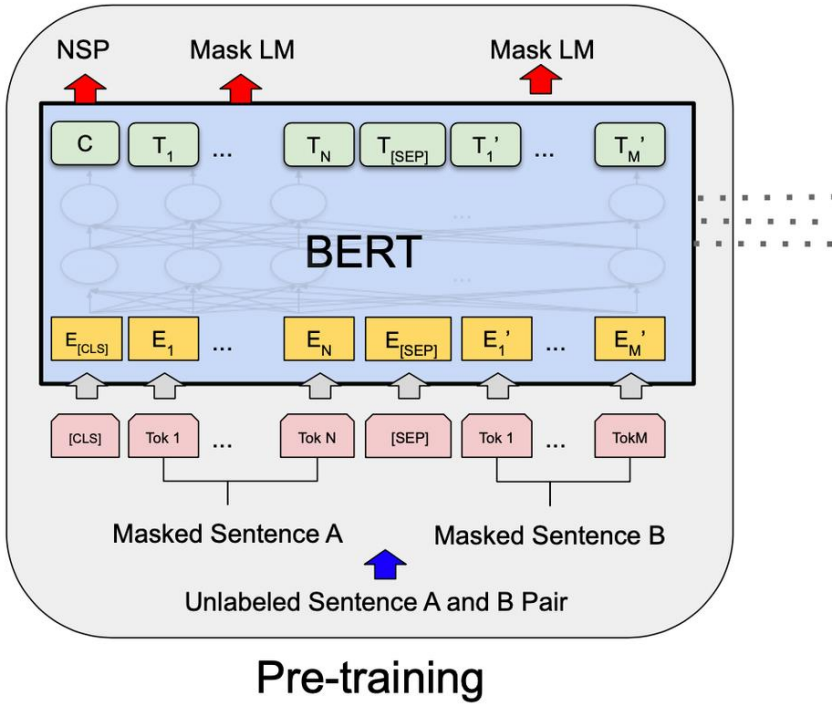
- **input_ids** : token들의 id 리스트(sequence of token id)
- **token_type_ids** : sentence A에 속하는 token에는 0을, sentence B에 속하는 token에는 1을 부여.
- **attention_mask** : attention 연산이 수행되어야 할 token과 무시해야 할 token을 구별하는 정보가 담긴 리스트. 일반적인 token에는 1을 부여하고, padding과 같이 attention 연산이 수행될 필요가 없는 token들에는 0을 부여.

```python
print(tokenizer.convert_ids_to_tokens(1045))  # 하나만 바꿀 수도 있고
print(tokenizer.convert_ids_to_tokens([101, 1045, 2293, 17953, 2361, 999, 102]))  # 여러 개를 바꿀 수도
```

[실행 결과]

```
i
['[CLS]', 'i', 'love', 'nl', '##p', '!', '[SEP]']
```

- [PAD] - 0
- [UNK] - 100
- [CLS] – 101
- [SEP] – 102
- [MASK] - 103



Pre-training

Pre-training BERT: 두가지 TASK

1. Masked Language Model (Mask LM)

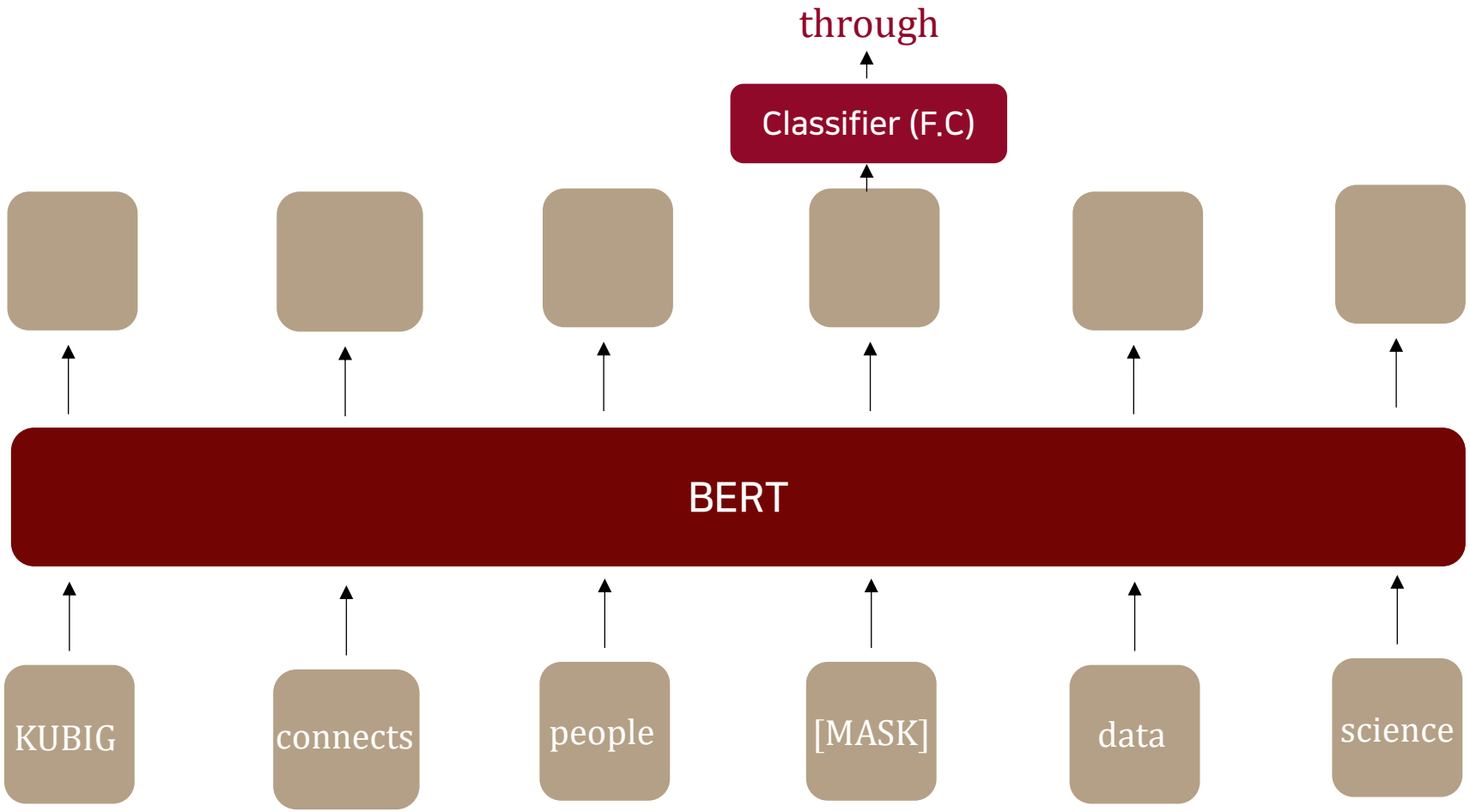
: Bidirectional pre-training for language representations

2. Next Sentence Prediction (NSP)

: Is it next sentence? Or not?

Mask LM

Bidirectional conditioning을 통해 multi-layered context에서 masked token을 예측



각 sequence의 15%를 [MASK] token으로 replaced

KUBIG DATA SCIENCE & AI

Mask LM

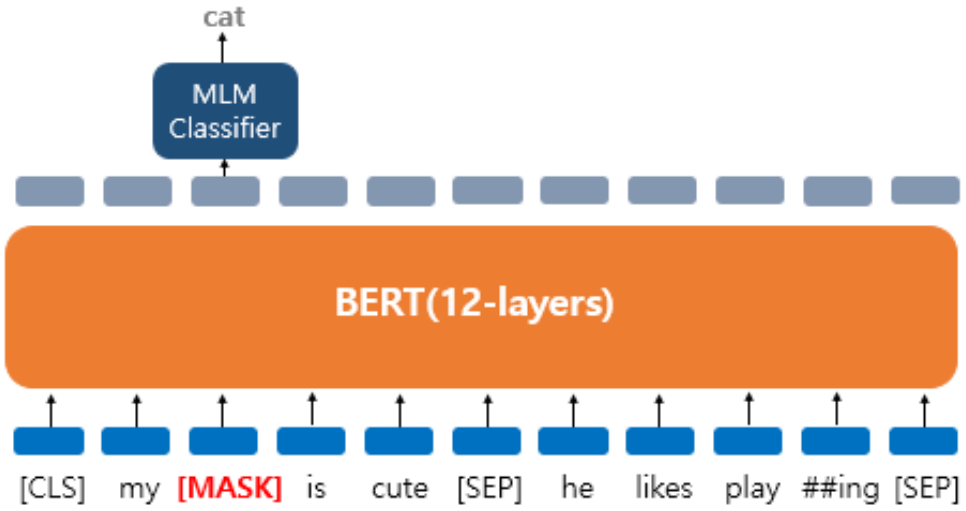
🚨 Fine-tuning 시에는 [MASK] 토큰이 나타나지 않기 때문에 pre-train과 fine-tuning 사이 mismatch가 발생하는 문제
따라서 15%의 token에 대해서..

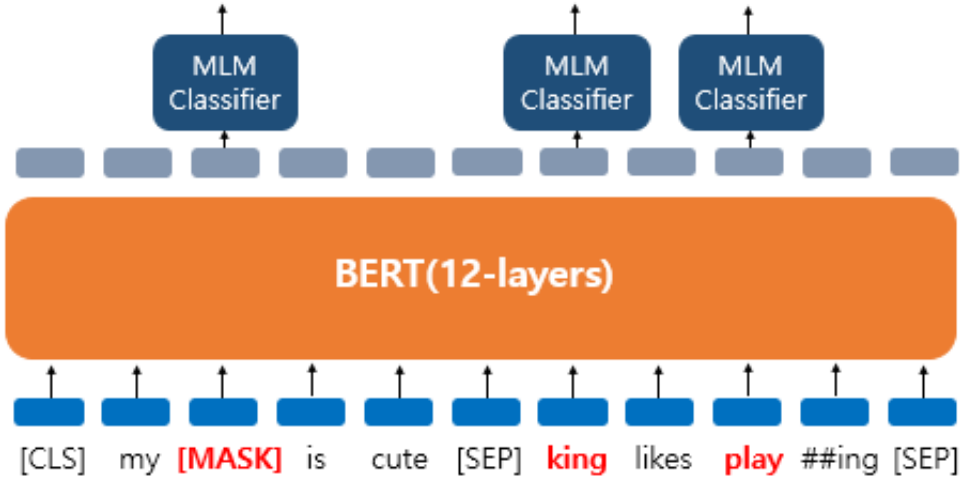
- 80% : 단어를 [MASK] 토큰으로 교체
  KUBIG connects people through data science → KUBIG connects people [MASK] data science
- 10% : 단어를 random word로 교체
  KUBIG connects people through data science → KUBIG connects people therapy data science
- 10% : 단어를 교체하지 않음
  KUBIG connects people through data science → KUBIG connects people through data science

비율은 경험적인 결과!

Masking Rates			Dev Set Results		
MASK	SAME	RND	MNLI	NER	
			Fine-tune	Fine-tune	Feature-based
80%	10%	10%	84.2	95.4	94.9
100%	0%	0%	84.3	94.9	94.0
80%	0%	20%	84.1	95.2	94.6
80%	20%	0%	84.4	95.2	94.7
0%	20%	80%	83.7	94.8	94.6
0%	0%	100%	83.6	94.9	94.6

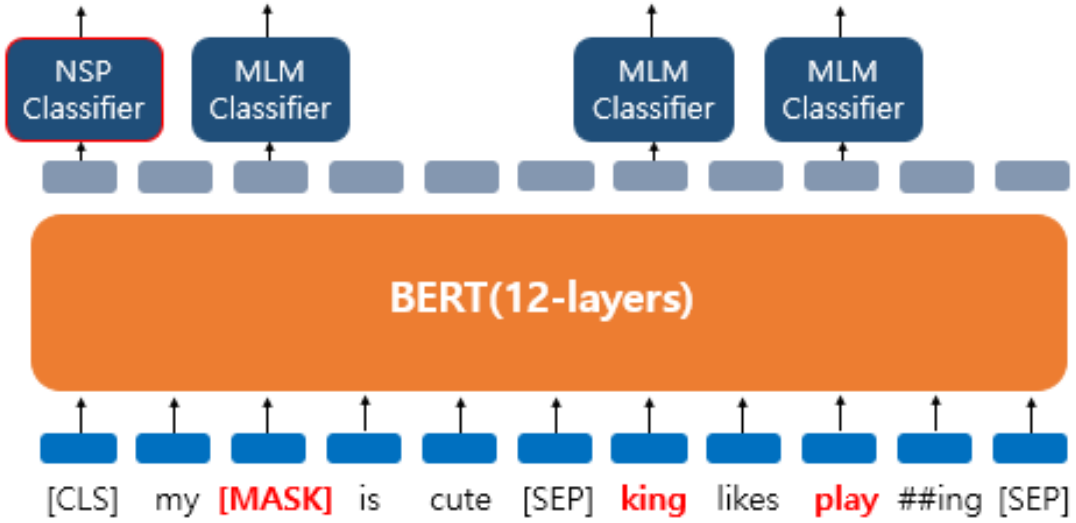
Mask LM


cat
MLM Classifier
BERT(12-layers)
[CLS] my [MASK] is cute [SEP] he likes play ##ing [SEP]


MLM Classifier
MLM Classifier
MLM Classifier
BERT(12-layers)
[CLS] my [MASK] is cute [SEP] king likes play ##ing [SEP]

NSP

Next Sentence Prediction으로 sentences간의 relationship 학습



- QA(Question-Answering)이나 NLI(Natural Language Inference)처럼 두 문장 간 관계를 이해하는 것이 중요한 task
- [CLS] sentence A [SEP] sentence B
  - 50% of the time B is the actual next sentence that follows A (label = IsNext)
  - 50% of the time it is a random sentence from the corpus (label = NotNext)

NSP

Monica: This is harder than I thought it would be.
Chandler: Oh, it is gonna be okay.
Rachel: Do you guys have to go to the new house right away, or do you have some time?
Monica: We got some time.
Rachel: Okay, should we get some coffee?
Chandler: Sure, Where?

[CLS] This is harder than I thought it would be. [SEP] Oh, it is gonna be okay.

NSP

Monica: This is harder than I thought it would be.

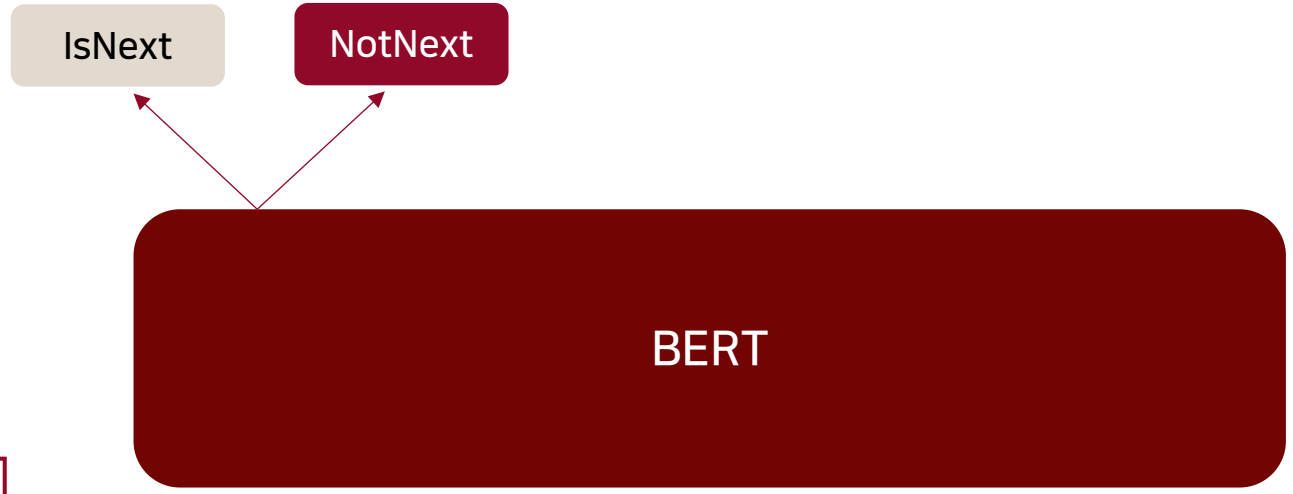
Chandler: Oh, it is gonna be okay.

Rachel: Do you guys have to go to the new house right away, or do you have some time?

Monica: We got some time.

Rachel: Okay, should we get some coffee?

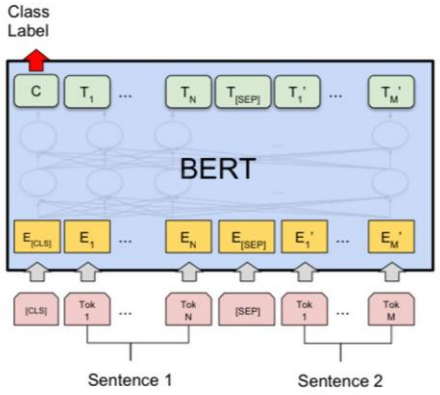
Chandler: Sure, Where?



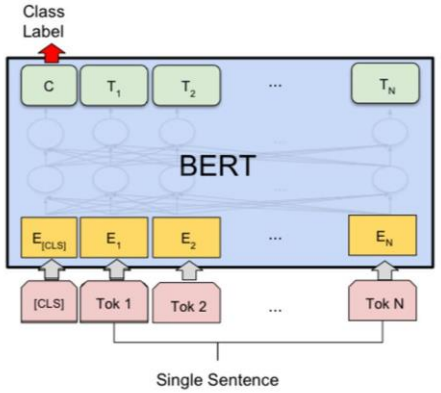
[CLS] This is harder than I thought it would be. [SEP] Okay, should we get some coffee?

3-5. Fine-tuning BERT

Classification

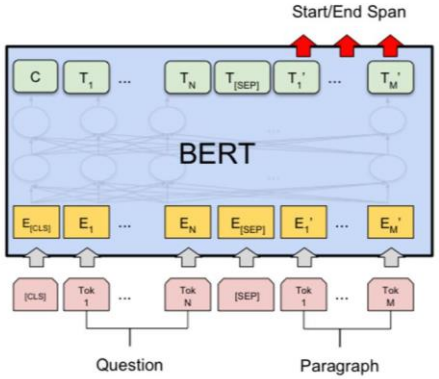


(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG

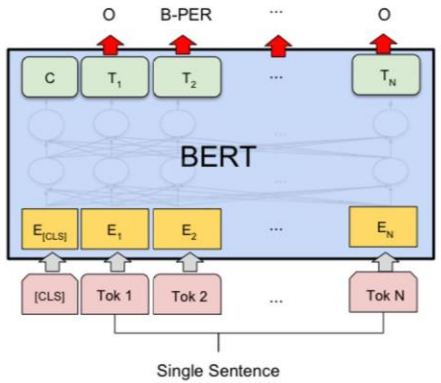


(b) Single Sentence Classification Tasks: SST-2, CoLA

Sentence



(c) Question Answering Tasks: SQuAD v1.1



(d) Single Sentence Tagging Tasks: CoNLL-2003 NER

Sentence Pair

Single sentence

Fine-tuning BERT: 11가지 TASK

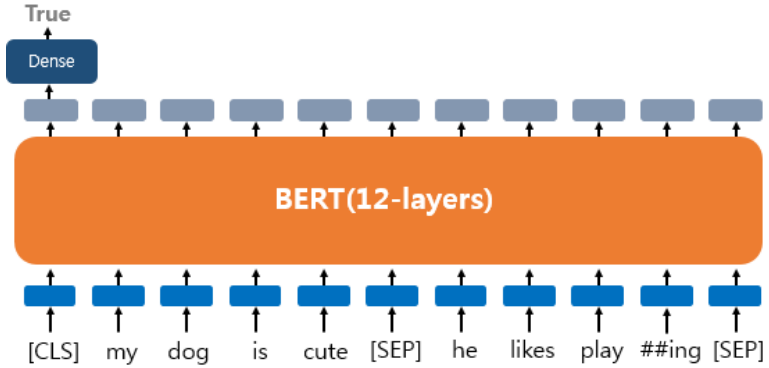
- labeled data에 대해 Fine-tuning 진행
- 모든 매개변수를 처음부터 끝까지 fine-tuning
- 각 downstream task는 별도의 fine-tuned models 를 갖게 됨 (시작은 동일한 pre-trained params)

3-5. Fine-tuning BERT

KUBIG DATA SCIENCE & AI

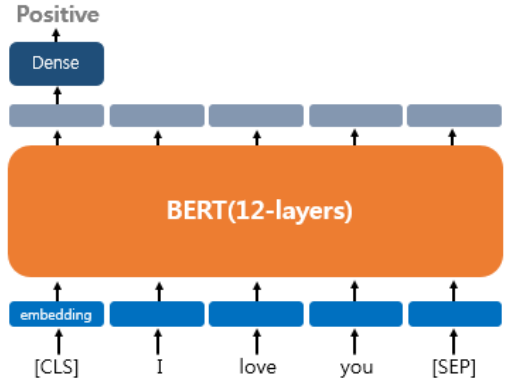
Sentence Pair Classification

Natural language inference : 두 문장이 주어졌을 때 논리적으로 어떤 관계에 있는지 분류



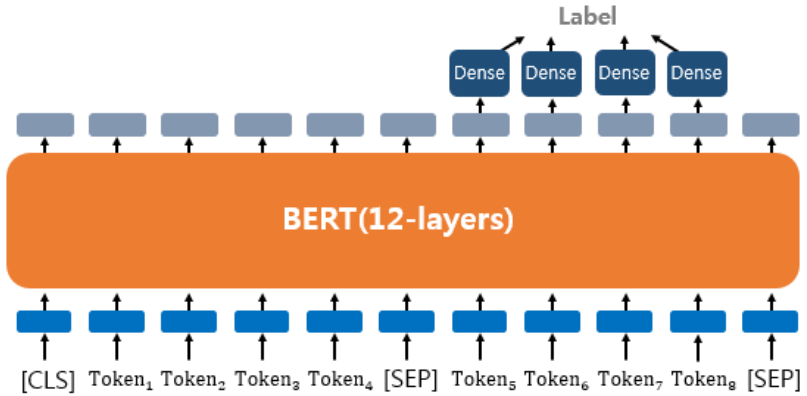
Single Sentence Classification

하나의 문서에 대한 텍스트 분류 유형



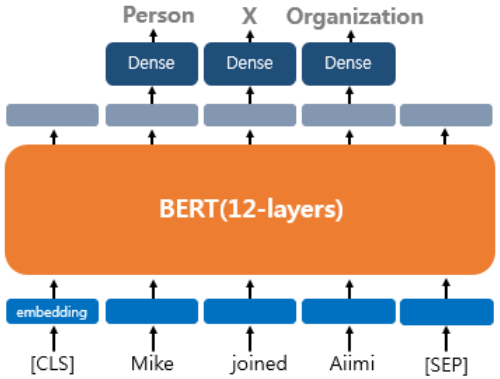
Question Answering Task

질문, 본문으로 두 개의 텍스트의 쌍을 입력 (SQuAD v1.1)

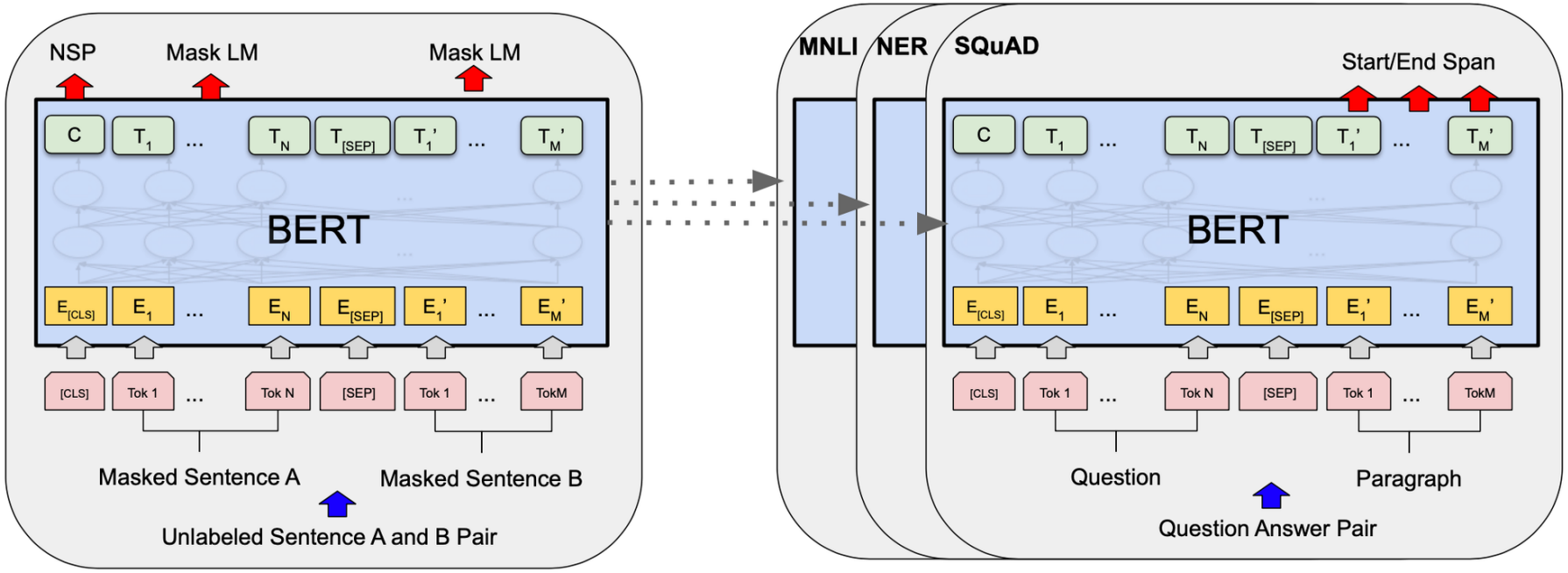


Single Sentence Tagging

각 단어에 품사를 태깅하는 품사 태깅, 개체를 태깅하는 개체명 인식 작업 등



https://wikidocs.net/115055



MLM과 NSP로 일반적인 언어표현(representation) 학습
Datasets: BooksCorpus(800M words),
English Wikipedia(2,500M words)

다양한 downstream task에 대해 미세 조정
Pre-trained model에 하나의 layer를 쌓아
다양한 NLP tasks 수행

ALBERT

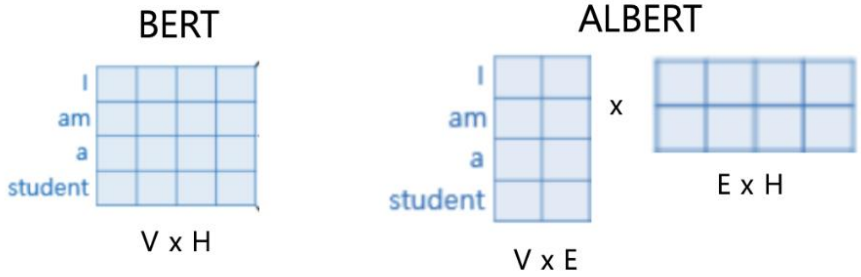
A Lite BERT for Self-supervised Learning of Language Representations

Limitations of BERT

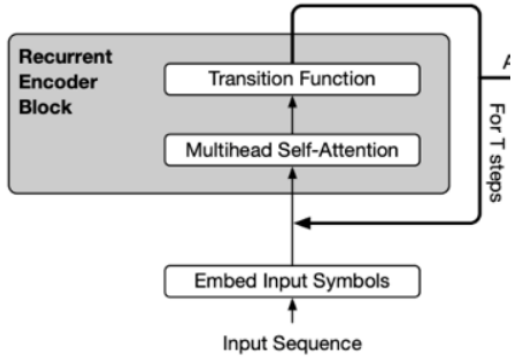
1. 110M개의 파라미터 (bert-base)
   → 메모리 부담 & 학습 속도 저하
2. NSP task의 효과가 미미함

ALBERT

1. Embedding Factorization,
   Cross-layer Parameter Sharing
2. NSP 대신 SOP(Sentence Order Prediction) 도입



Embedding Factorization



Cross-layer Parameter Sharing

A: I love you. B: I love you, too.
→ Positive
A: I love you, too. B: I love you.
→ Negative

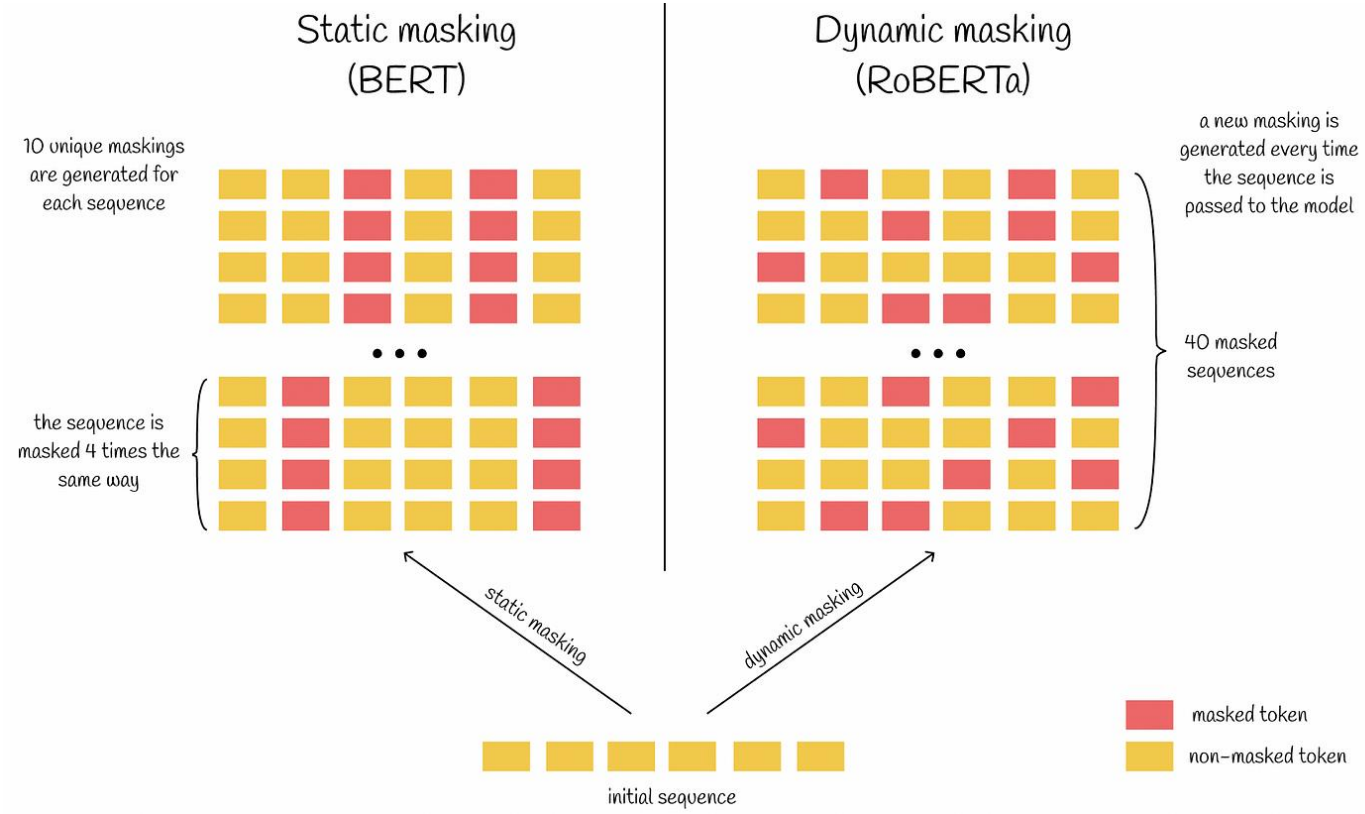
SOP

RoBERTa

Robustly optimized BERT approach

RoBERTa

1. 더 큰 배치로 모델을 더 오랫동안 훈련
2. NSP 목적함수 제거
3. 더 긴 시퀀스에 대한 훈련
4. Dynamic masking: 토큰은 각 에포크 마다 다르게 마스킹됨
5. 더 큰 데이터 세트 수집

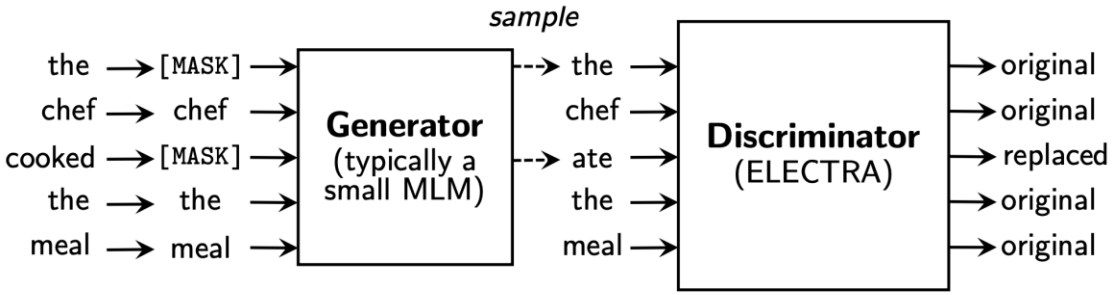


ELECTRA — Efficiently Learning an Encoder that Classifies Token Replacements Accurately

Limitation of BERT

1.전체 토큰 중 15%에 대해서만 loss가 발생
(= 하나의 example에 대해서 고작 15%만 학습함)
2. 많은 학습 비용
3. 학습 때는 [MASK] 토큰을 모델이 참고하여 예측하지만 실제(inference)로는 [MASK]토큰이 존재하지 않음

ELECTRA
: Replaced Token Detection (RTD) 도입



Generator G
: BERT의 MLM

Discriminator D
: 입력 토큰 시퀀스에 대해서 각 토큰이 original인지 replaced인지 이진분류

- generator loss와 discriminator loss의 합을 최소화하도록 학습
- 샘플링 과정이 있기 때문에 discriminator loss는 generator로 역전파 되지 않음
- 위의 구조로 pre-training을 마친 뒤에 discriminator만 취해서 downstream task으로 fine-tuning을 진행

04 GPT series

Text Generation

KUBIG DATA SCIENCE & AI

Motivation

"Improving Language Understanding by Generative Pre-Training"

Leveraging more than word-level information from unlabeled text is challenging for two main reasons.

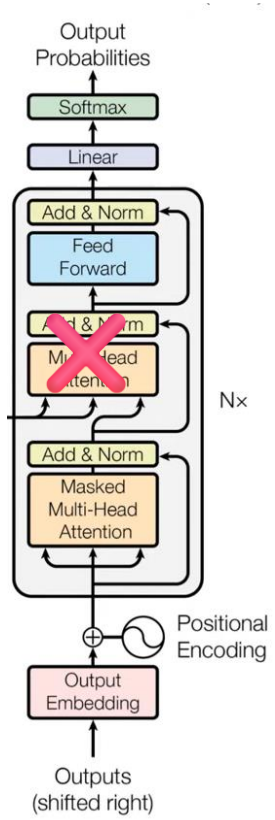
1. unclear what type of optimization objectives are most effective at learning text representations that are useful for transfer.

2. there is no consensus on the most effective way to transfer these learned representations to the target task

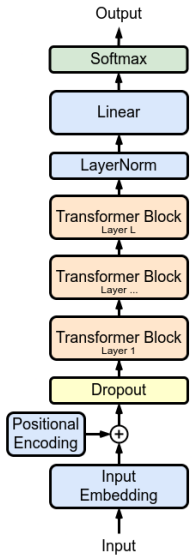
➔ **Semi-supervised approach using a combination of unsupervised pre-training and supervised fine-tuning**

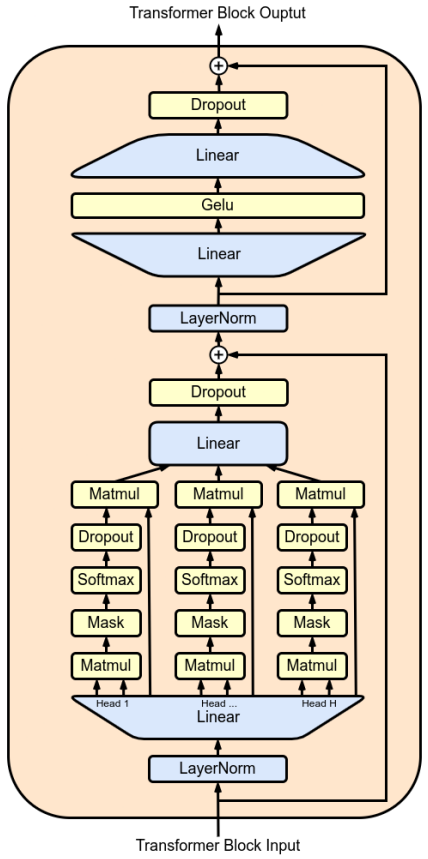
Model Architecture

A multi-layer Transformer decoder



Original decoder from transformer





GPT-1

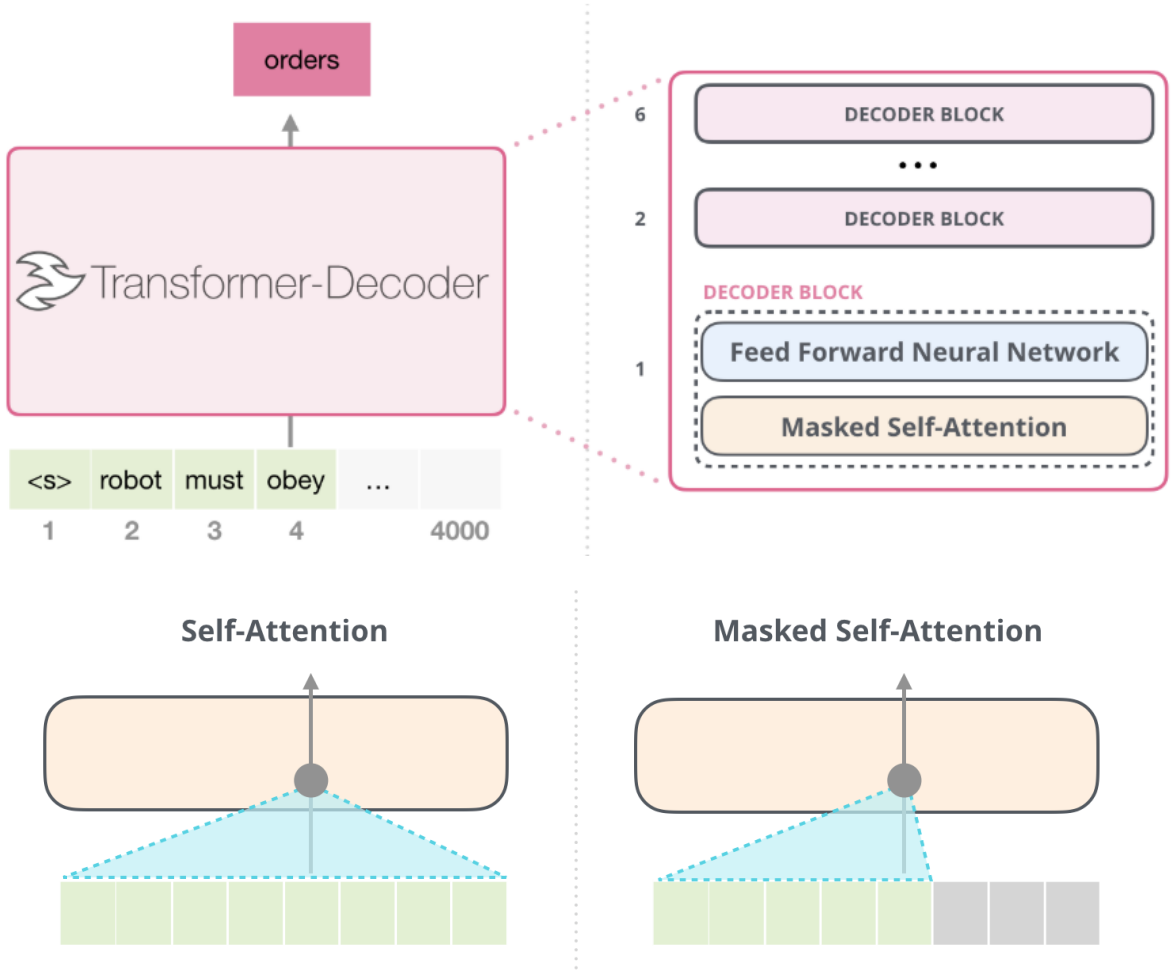
- pre-training: multi-layer Transformer decoder
- fine-tuning: pre-trained model

  → linear output layer → softmax

$$h_0 = UW_e + W_p$$
$$h_l = \texttt{transformer\_block}(h_{l-1}) \forall i \in [1, n]$$
$$P(u) = \texttt{softmax}(h_n W_e^T)$$

- $U = (u_{-k}, .., u_{-1})$: the context vector of tokens
- n: # of layers
- W_e : token embedding matrix
- W_p : position embedding matrix

Model Architecture



The Illustrated GPT-2 (Visualizing Transformer Language Models) – Jay Alammar – Visualizing machine learning one concept at a time.

Semi-supervised learning

Pre-training with unlabeled data + Fine-tuning with labeled data

$$L_1(\mathcal{U}) = \sum_i \log P(u_i|u_{i-k}, \ldots, u_{i-1}; \Theta)$$

$$L_2(\mathcal{C}) = \sum_{(x,y)} \log P(y|x^1, \ldots, x^m).$$

$$= \texttt{softmax}(h_n W_e^T)$$

$$L_3(\mathcal{C}) = L_2(\mathcal{C}) + \lambda * L_1(\mathcal{C})$$

Self-supervised pre-training

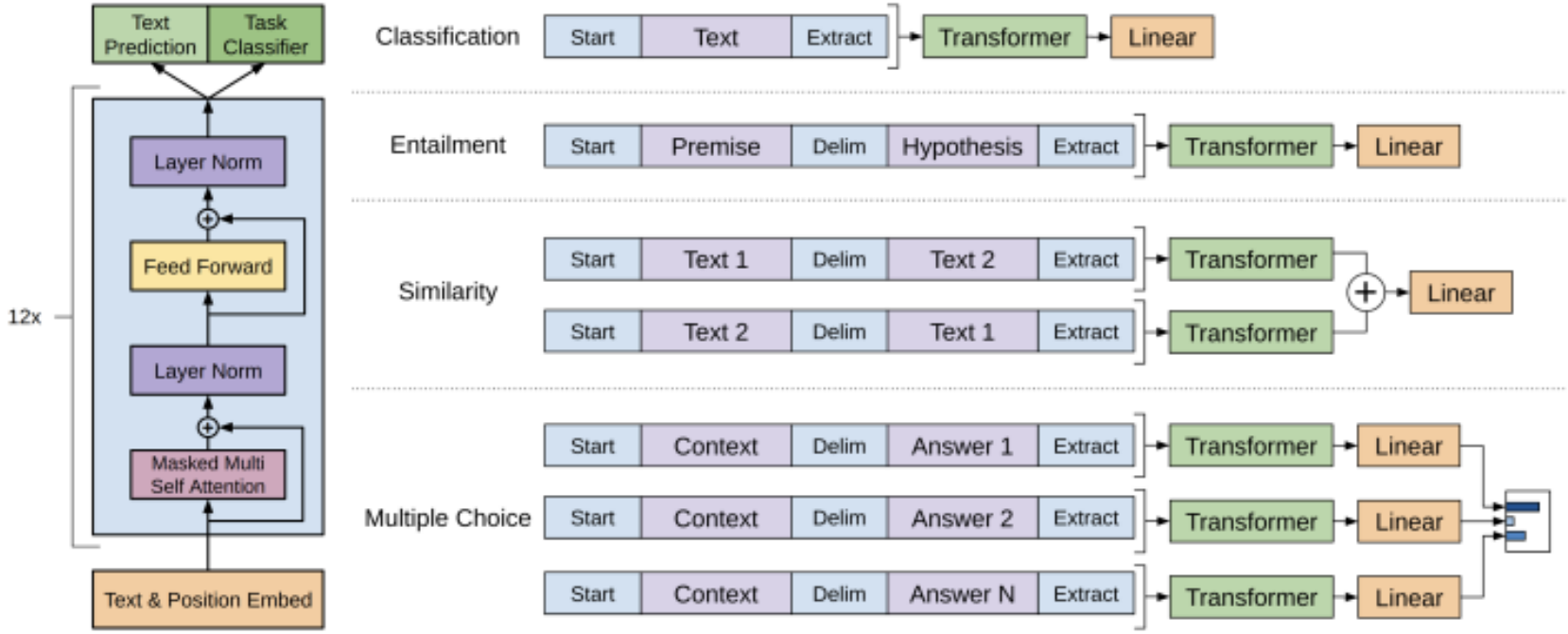
- unlabeled data(U), auto-regressive
- standard language modeling objective
  (next token prediction) : maximize L1

→ long-range dependency를 처리

Supervised fine-tuning

- labeled data (C)
- task 별 objective + language modeling objective : maximize L3
- Language modeling objective 를 같이 고려해 model의 generalization 성능을 높이고, 빠르게 수렴하도록 하는 장점 → 개별 task에 적용 가능

Task-specific input transformations



Motivation

" Language Models are Unsupervised Multitask Learners "

💡 특정 data나 task에만 뛰어난 narrow expert model이 아닌, competent generalist를 만들고 싶다!

→ Fine-tuning 없이 unsupervised pretraining만으로 down-stream task를 수행하는 General LM

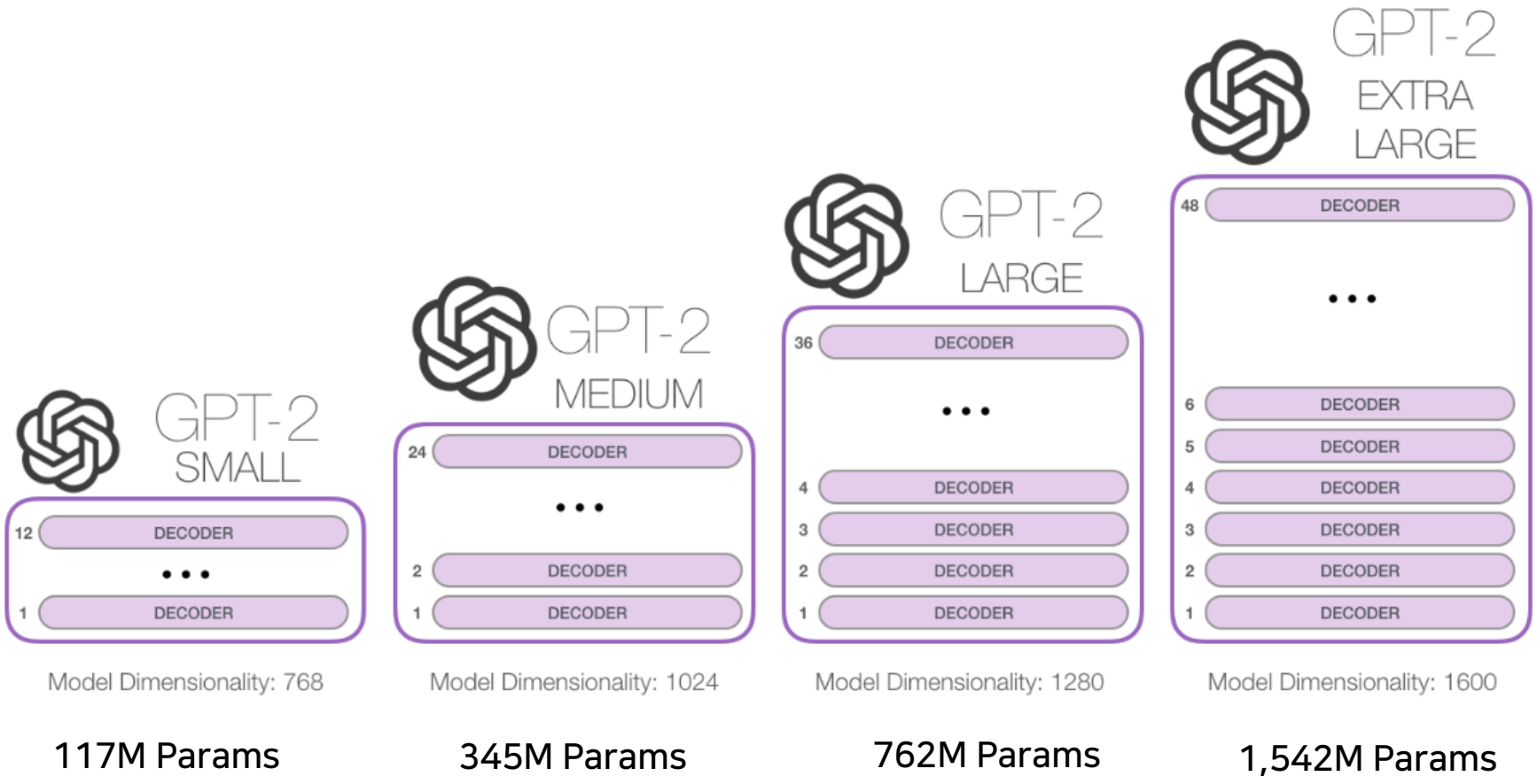
→ Zero-shot generalization

Model

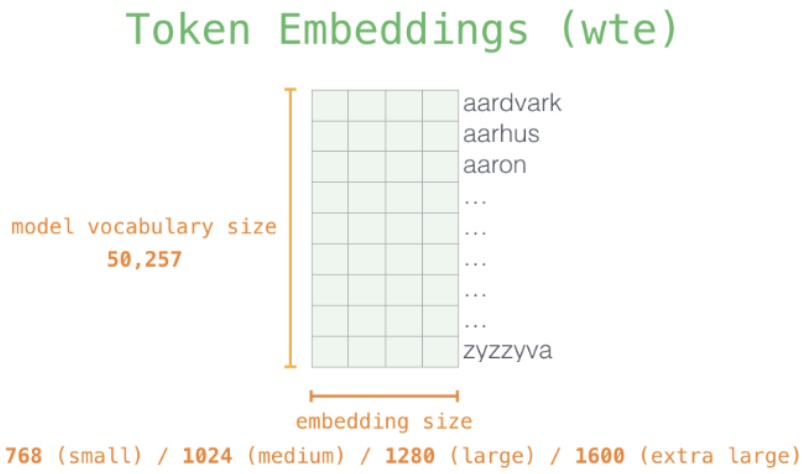
- Transformer decoder만 사용한 GPT-1의 모델 구조를 사용
- layer normalization을 먼저 하기, residual layer의 누적에 따라 initialization 변경 등 몇가지 변화
- batch size, input length(512→1024), vocab size 증가
- model size 증가

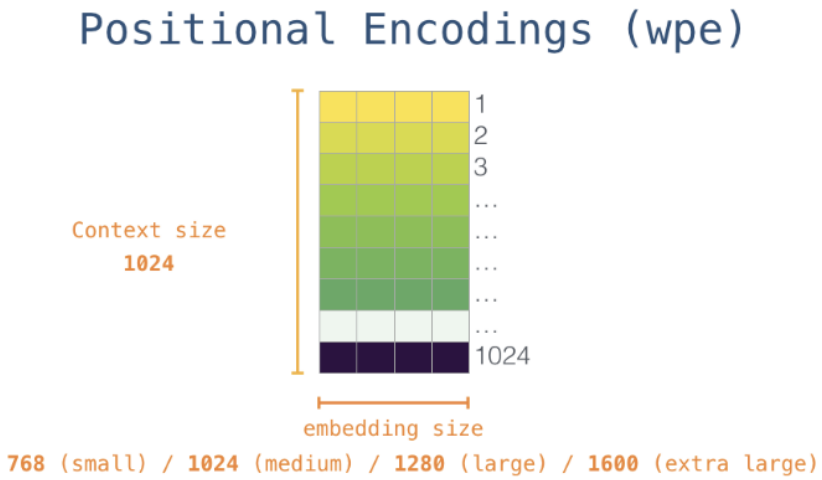
Datasets

- Web scraping dataset으로 다양한 도메인 데이터 사용
- WebText dataset: Reddit에서 링크된 글만 필터링하여 사용. 데이터에서 위키피디아 글 삭제, 중복 제거 등 전처리.
- 토큰화 byte-pair-encoding 사용



The Illustrated GPT-2 (Visualizing Transformer Language Models) – Jay Alammar – Visualizing machine learning one concept at a time.





[The Illustrated GPT-2 (Visualizing Transformer Language Models) – Jay Alammar – Visualizing machine learning one concept at a time.]

KUBIG DATA SCIENCE & AI

Results

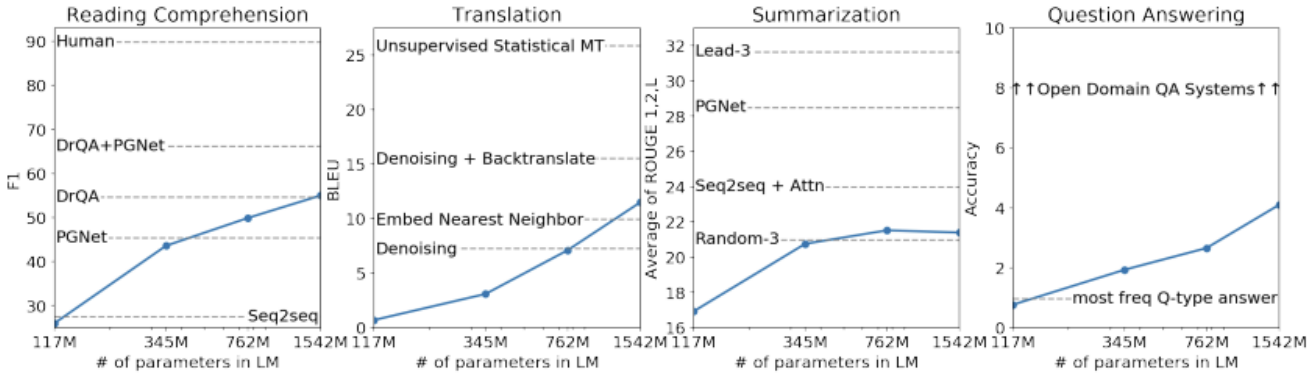
"I'm not the cleverest man in the world, but like they say in French: **Je ne suis pas un imbecile [I'm not a fool].**
In a now-deleted post from Aug. 16, Soheil Eid, Tory candidate in the riding of Joliette, wrote in French: "**Mentez mentez, il en restera toujours quelque chose**," which translates as, "**Lie lie and something will always remain.**"
"I hate the word '**perfume**,"' Burr says. 'It's somewhat better in French: '**parfum**.'
If listened carefully at 29:55, a conversation can be heard between two guys in French: "**-Comment on fait pour aller de l'autre coté? -Quel autre coté?**", which means "**- How do you get to the other side? - What side?**".
If this sounds like a bit of a stretch, consider this question in French: **As-tu aller au cinéma?**, or **Did you go to the movies?**, which literally translates as Have-you to go to movies/theater?
"**Brevet Sans Garantie Du Gouvernement**", translated to English: "**Patented without government warranty**".

Table 1. Examples of naturally occurring demonstrations of English to French and French to English translation found throughout the WebText training set.

- prompt를 추가하면 zero-shot으로 생성 가능
- 특정 task를 명시적으로 학습하지 않아도(fine-tuning)

  prompt를 활용하면 다양한 NLP task를 수행할 수 있음 !
- zero-shot generalization 가능성을 제시 💡

Results



	LAMBADA (PPL)	LAMBADA (ACC)	CBT-CN (ACC)	CBT-NE (ACC)	WikiText2 (PPL)	PTB (PPL)	enwik8 (BPB)	text8 (BPC)	WikiText103 (PPL)	1BW (PPL)
SOTA	99.8	59.23	85.7	82.3	39.14	46.54	0.99	1.08	18.3	**21.8**
117M	**35.13**	45.99	**87.65**	**83.4**	**29.41**	65.85	1.16	1.17	37.50	75.20
345M	**15.60**	55.48	**92.35**	**87.1**	**22.76**	47.33	1.01	**1.06**	26.37	55.72
762M	**10.87**	**60.12**	**93.45**	**88.0**	**19.93**	**40.31**	**0.97**	**1.02**	22.05	44.575
1542M	**8.63**	**63.24**	**93.30**	**89.05**	**18.34**	**35.76**	**0.93**	**0.98**	**17.48**	42.16

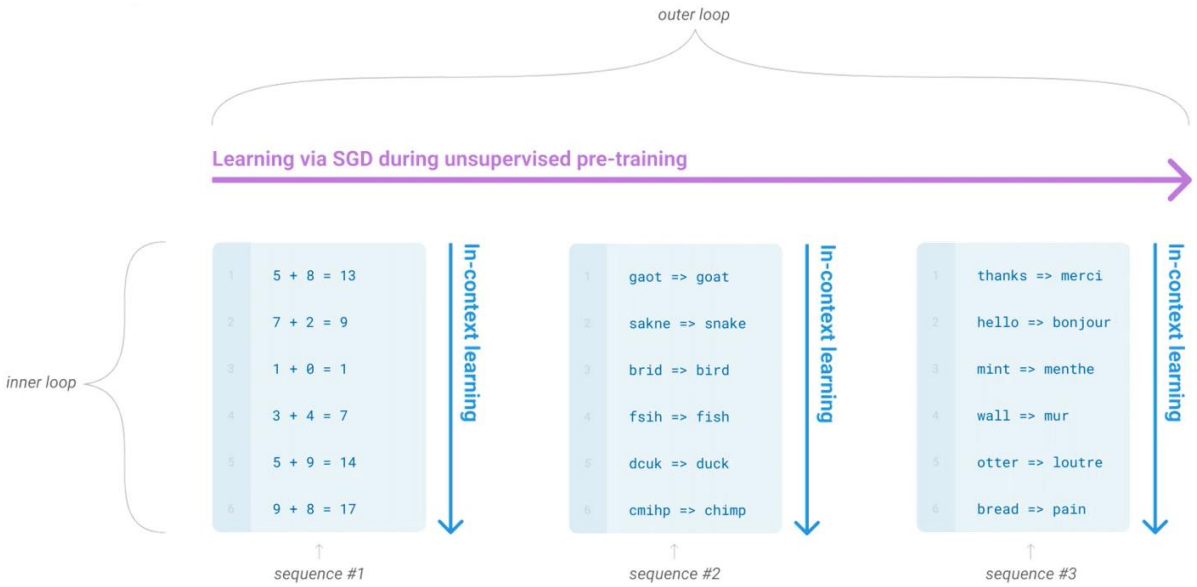
- Reading comprehension 등의 task들이 supervised baseline보다 우수한 성능을 보임
- Summarization과 같은 task에선 성능이 좋지 않음

- Sufficient model capacity일 때에만 baseline을 넘음 💡

→ "When a large language model is trained on a sufficiently large and diverse dataset it is able to perform well across many domains and datasets"

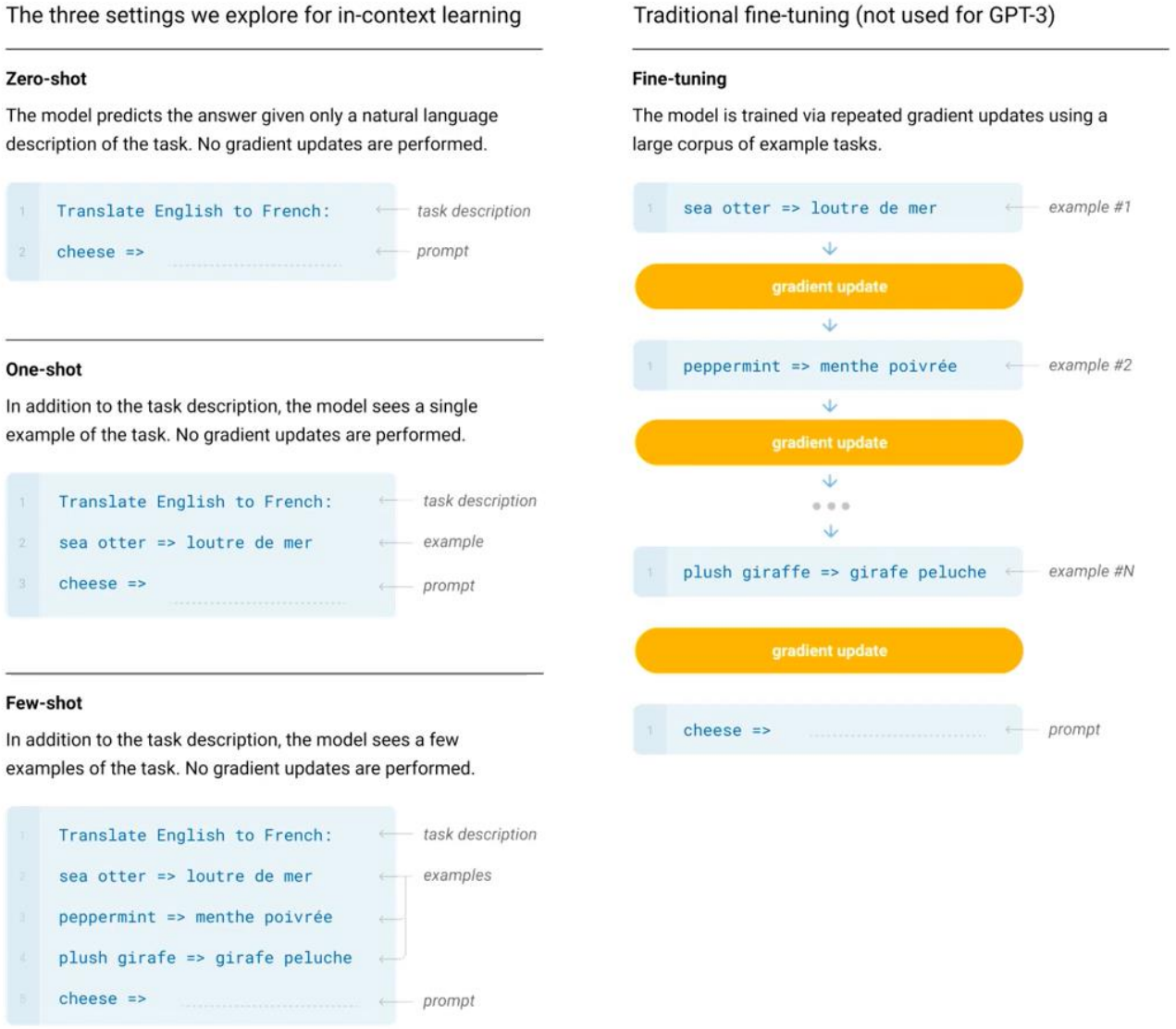
Motivation

"Language Models are Few-Shot Learners"



In-context learning

- In-context learning은 fine-tuning과 달리 gradient update를 하지 않으며, supervised dataset이 필요하지 않음.
- model이 task에 대한 정보를 참고해서 inference 할 수 있도록 input에 예제(demonstrations) 추가
- inner loop를 같은 task로 구성해서 model이 다양한 task에 대해 multi-task learning 하는 효과



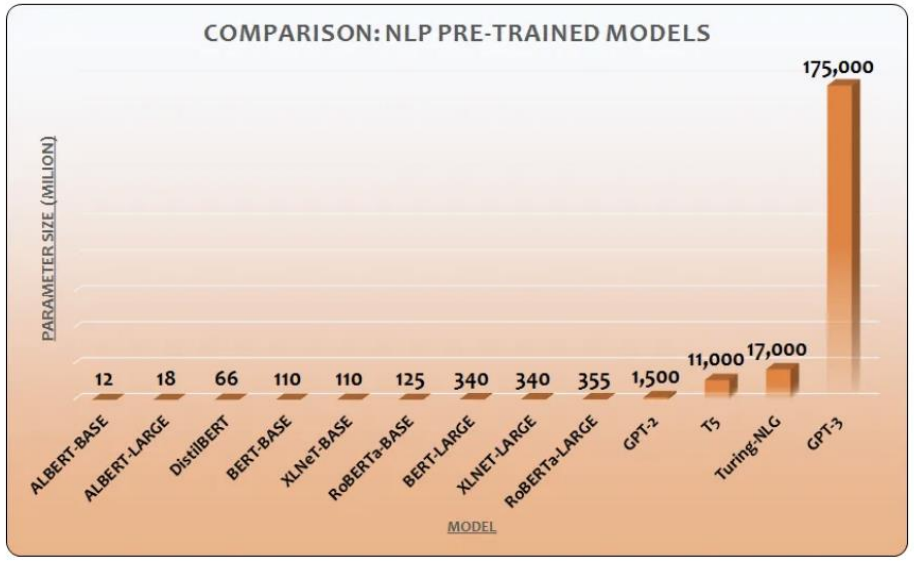
Fine-Tuning

- Pros: Strong performance on many benchmarks
- Cons: 1) 새 task에 대해선 새 large dataset이 필요
  2) out-of-distribution 에 대해선 poor generalization
  3) Gradient update 과정 필요

Few-Shot

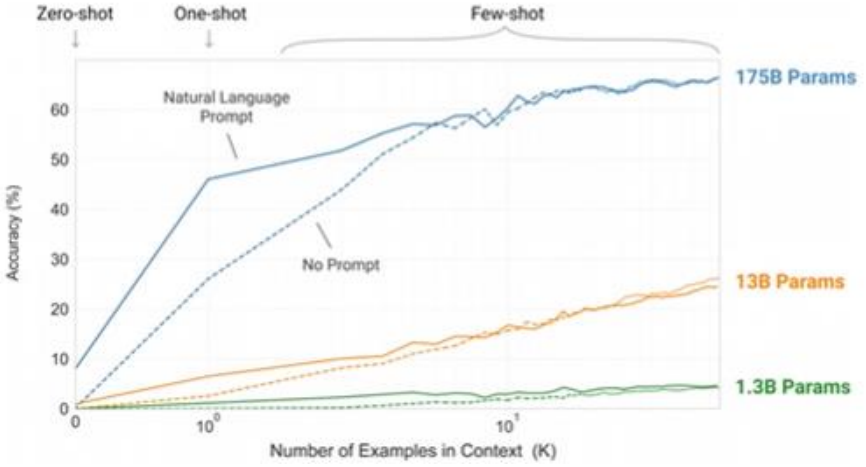
- Pros: 1) task-specific한 데이터의 필요성 감소
  2) 적은 수의 예제만 보고 작업을 수행하므로
  overly narrow distribution을 학습하지 않아도 됨
  3) training 이 아니라 inference 과정에서 예시를 제공하므로
  gradient update 과정 필요 X
- Cons: SOTA fine-tuned models보다 낮은 성능 보일 때 있음

Model Architecture

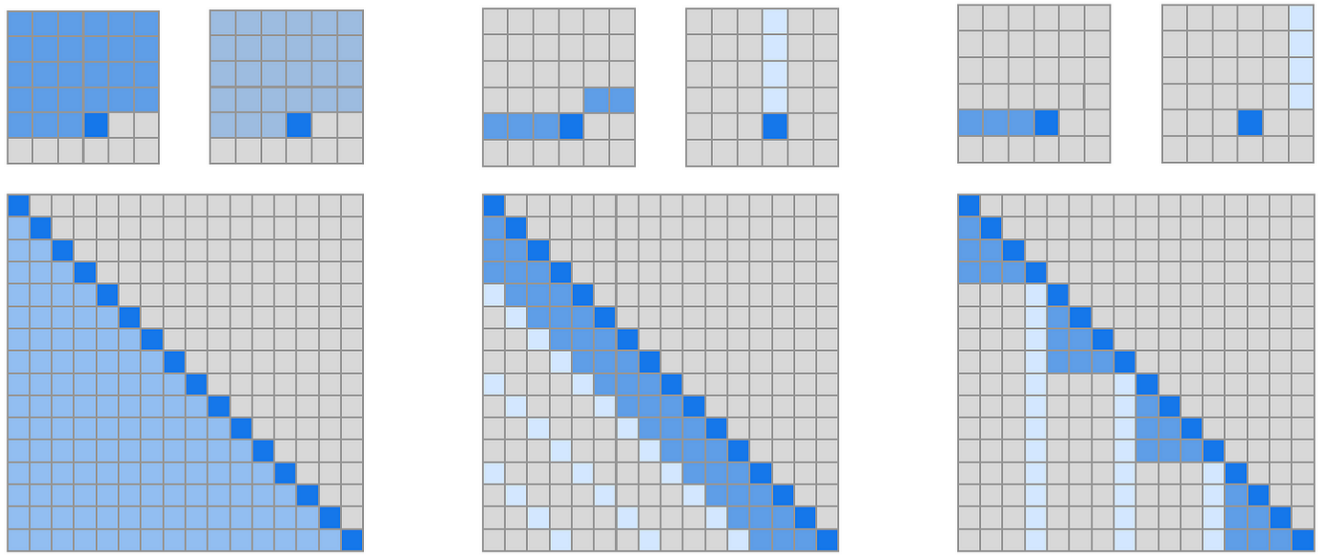


Auto-regressive language model
with 175 billion params

Model Name	n_{params}	n_{layers}	d_{model}	n_{heads}	d_{head}	Batch Size	Learning Rate
GPT-3 Small	125M	12	768	12	64	0.5M	6.0×10^{-4}
GPT-3 Medium	350M	24	1024	16	64	0.5M	3.0×10^{-4}
GPT-3 Large	760M	24	1536	16	96	0.5M	2.5×10^{-4}
GPT-3 XL	1.3B	24	2048	24	128	1M	2.0×10^{-4}
GPT-3 2.7B	2.7B	32	2560	32	80	1M	1.6×10^{-4}
GPT-3 6.7B	6.7B	32	4096	32	128	2M	1.2×10^{-4}
GPT-3 13B	13.0B	40	5140	40	128	2M	1.0×10^{-4}
GPT-3 175B or "GPT-3"	175.0B	96	12288	96	128	3.2M	0.6×10^{-4}



Model Architecture



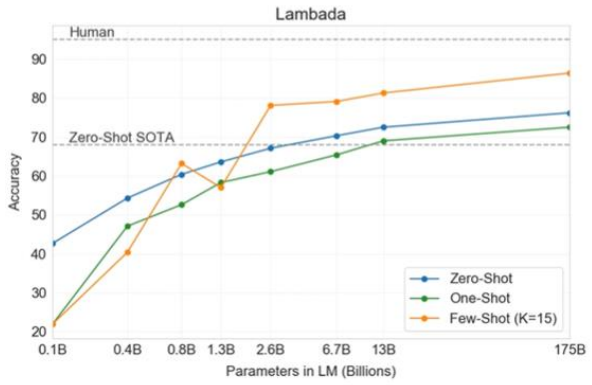
(a) Transformer (b) Sparse Transformer (strided) (c) Sparse Transformer (fixed)

Same as GPT-2 except alternating dense and locally banded

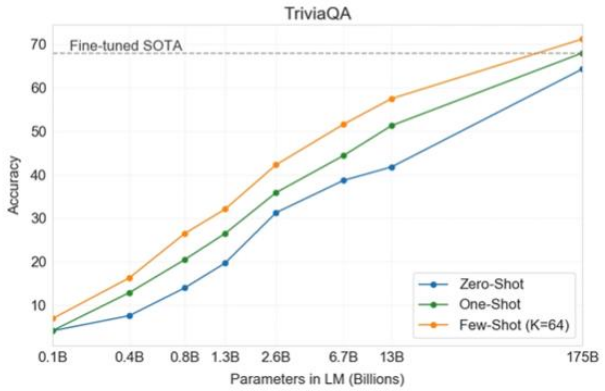
sparse attention pattern in the layers of the transformer

Results

Language Modeling

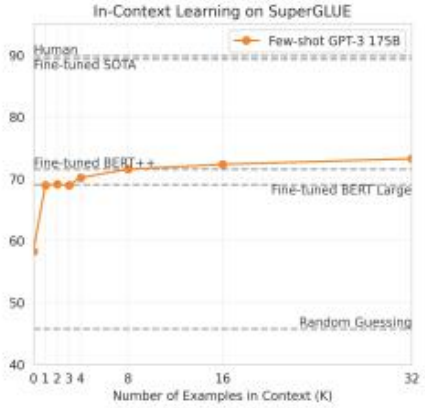


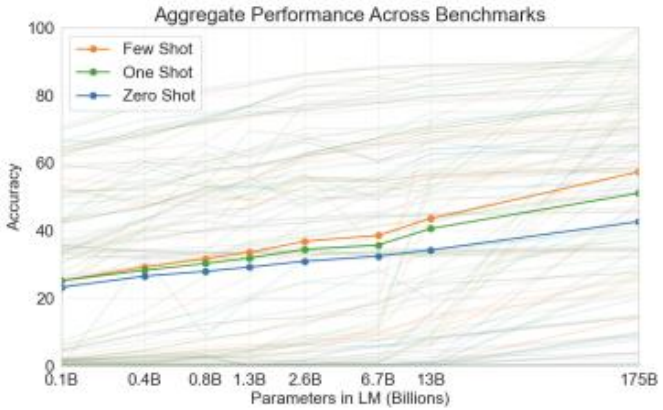
Closed Book QA



- parameter가 클 수록 성능이 좋음
- Few-shot이 fine-tuned SOTA보다 높은 성능을 보이기도
- 단어 풀이 등의 task에선 one-two shot만으로도 좋은 성능

- NLI 등 task는 few shot 이후에도 성능이 좋지 않음
- 동어 반복 현상, 일관성 부족, 모순적 내용 등





Motivation "Training language models to follow instructions with human feedback"

Limitation of GPT-3
: Alignment problem

- LM의 objective인 NTP(Next Token Prediction)이 user intention과 같지 않은 문제
- 없는 사실을 만들거나(not truthful), 편향적이고(biased), 유해한 텍스트를 만들거나(toxic, harmful) 등 사용자의 의도대로 사용되지 않을 수 있음

Objective of GPT-3*
: in accordance with user's intention

- 사용자의 instruction을 따르는 것 ➔ explicit하게 학습
- Helpful (user가 task를 달성하게 도움), Honest (거짓 정보 없이), Harmless (유해하지 않도록) ➔ implicit하게 학습
- 즉, 단순한 NTP가 아니라 user가 원하는 답변을 내뱉도록!

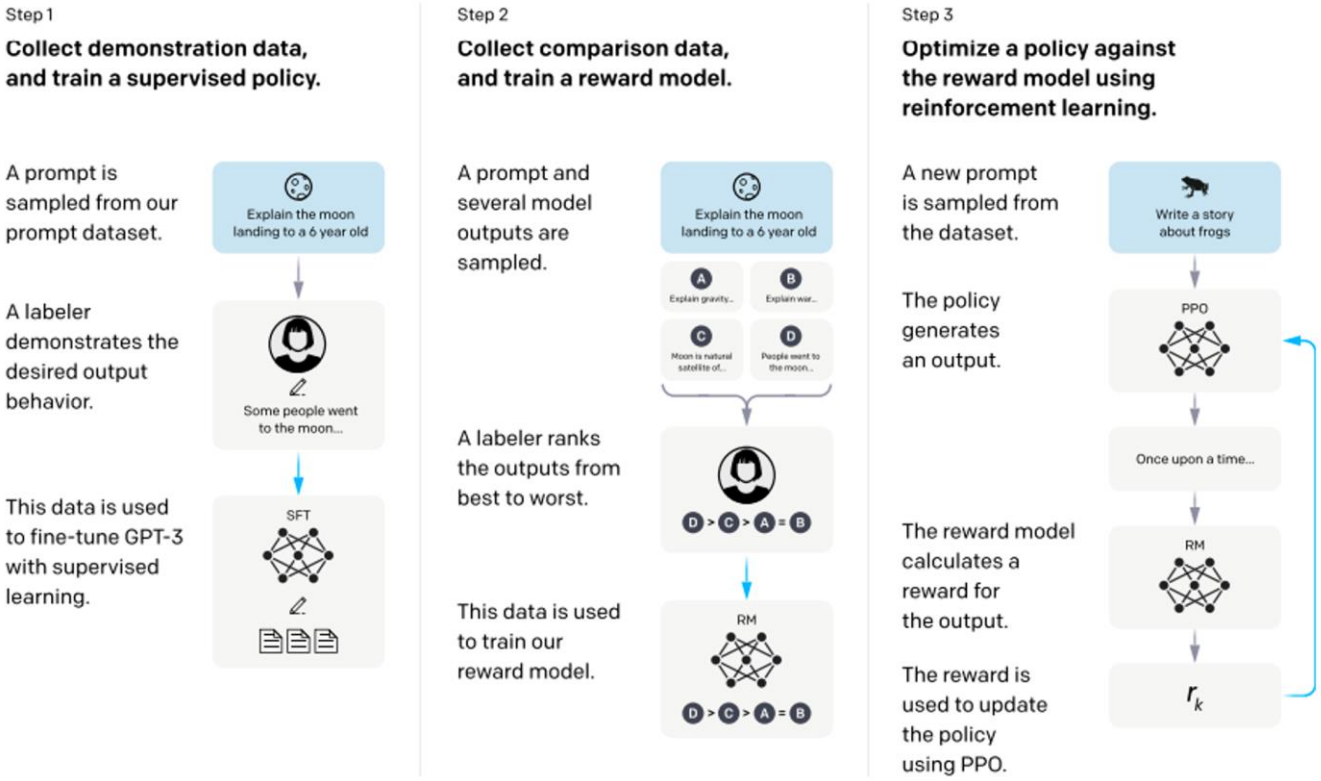
RLHF — Reinforcement Learning from Human Feedback

How?

: Fine-tuning by RLHF

- 인간의 피드백을 통한 강화학습으로 사용자의 광범위한 지시사항에 따를 수 있도록 하는 것
- 인간의 평가(preference)를 reward로 활용

- 데이터 구축을 위해 40명의 labeler를 고용
  : screening test를 통해 상위 rank된 사람들
- 연구에 참여한 저자들



→ GPT-3.5, GPT-4, …

최적이 될 때까지 step 2와 step 3를 반복

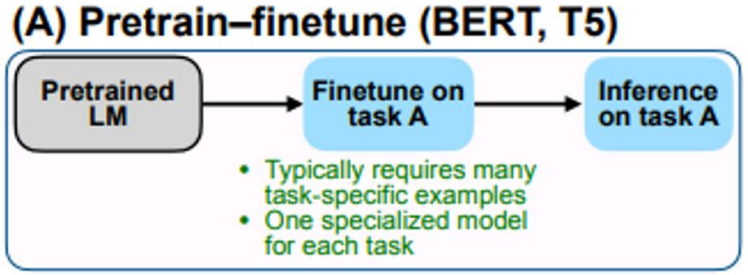
4-5. History and Future of GPTs

	날짜	특징	한계
GPT-1	2018.6	Auto-regressive pre-training objective	Zero-shot x
GPT-2	2019.2	• prompt를 추가하면 zero-shot 생성 • WebText (번역 등 다양한 downstream task)	
GPT-3	2020.5	In-context learning (few-shot learner)	Alignment Problem (hallucination, toxic, not helpful)
GPT-3* (InstructGPT)	2022.5	GPT-3 + RLHF	
GPT-3.5 (ChatGPT)	2022.11	InstructGPT의 연장선	
GPT-4 (ChatGPT+)	2023.3	Multi-modal	
GPT-4o	2024.5	Improvement for Non-English language	Mathematics, Physics and so on
GPT-5		• Complete Multi-modal? • AGI?	

05 Alpaca

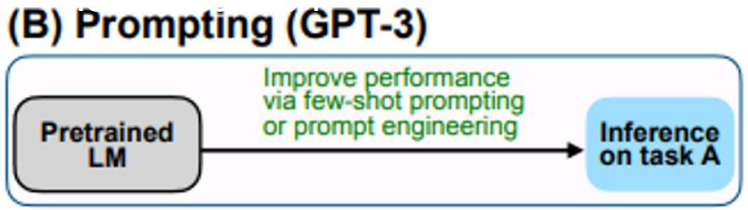
Human Alignment

KUBIG DATA SCIENCE & AI



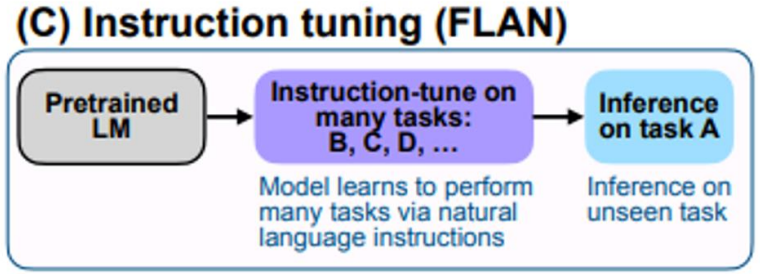
fine tuning

- pretrain된 언어 모델을 downstream task에 맞춰 supervised 방식으로 데이터셋을 학습
- pretrain 단계에서의 weight가 task-specific한 데이터셋을 학습하면서 업데이트됨



prompt learning

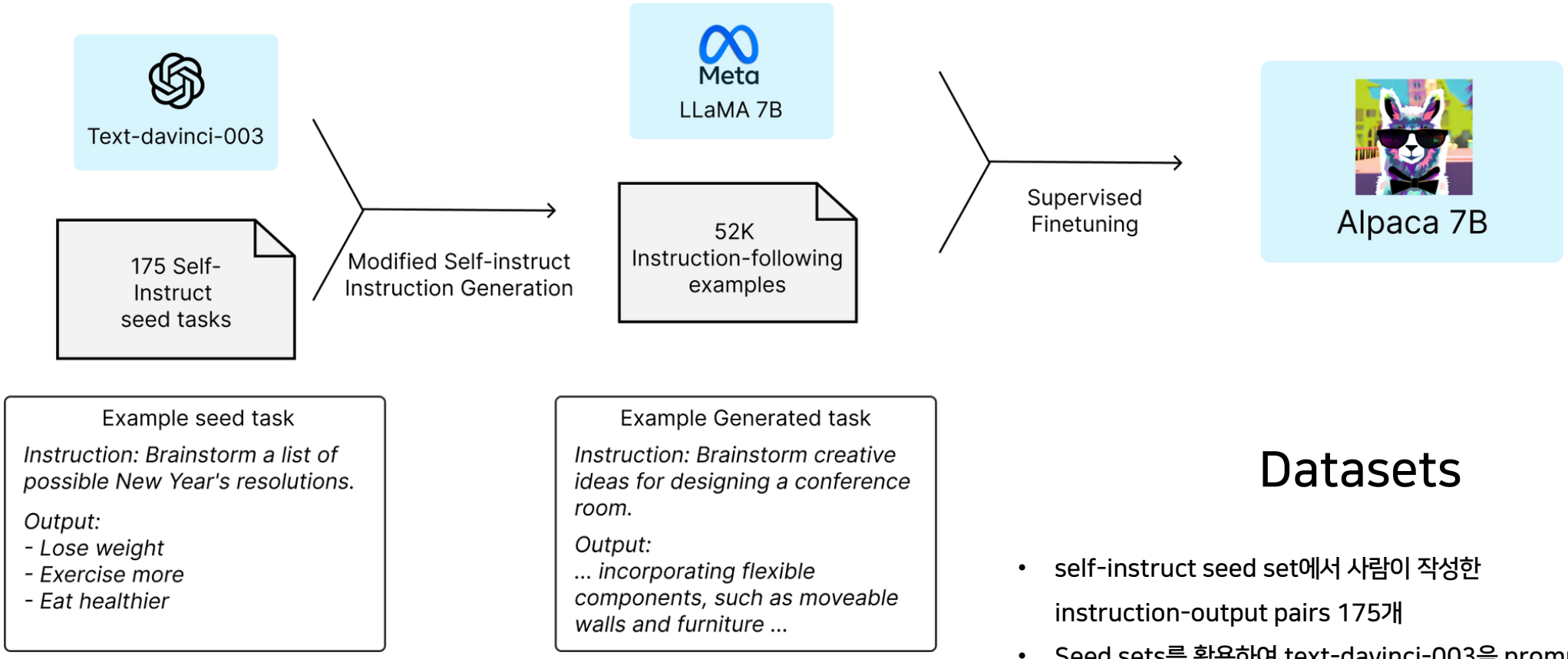
- task-specific한 데이터를 추가로 학습시키지 않음
- 사전학습을 진행할 때 모델에게 prompt (문제에 대한 설명)를 제공하여 학습
- pretrained 모델의 추가적인 가중치 업데이트가 이뤄지지 않음



Instruction tuning

- prompt learning의 prompt 방식과 finetuning의 가중치 업데이트를 결합한 방법
- 모델이 instruction이 있는 task를 supervised하게 학습해 instruction을 따르는 방식을 학습
- Instruction을 따르는 방식을 이미 배웠기 때문에 unseen task의 instruction에 따라 잘 추론할 수 있을 것이라는 아이디어

5-2. Alpaca



Datasets

- self-instruct seed set에서 사람이 작성한 instruction-output pairs 175개
- Seed sets를 활용하여 text-davinci-003을 prompting하여 더 많은 instructions 생성

Results

Evaluation

text-davinci-003과 Alpaca 7B 간의 pairwise comparison 결과

: 모델의 성능은 매우 유사.

Alpaca는 text-davinci-003과의 비교에서 90대 89

Alpaca의 size가 작고 data 양이 적은 것에 비해 놀라운 성과!

Limitation

- Alpaca는 언어 모델의 몇 가지 일반적인 결함을 보임
  : hallucination, toxicity, stereotypes 등
- Alpaca에는 underlying language model data 관련된 다른 많은 문제점이 있을 수 있음 (data filtering X)

06 KUBIG Contest 중간발표, Announcement

팀별 중간발표, week5 예복습 과제 안내, week6진도 안내

KUBIG
DATA SCIENCE & AI

6-1. KUBIG Contest 중간발표

🙋‍♂️ KUBIG Contest 중간발표

- 1팀: 김민재, 김재훈, 강서연, 남동연
- 2팀: 이예지, 이영서, 이예일
- 3팀: 이우진, 김유진, 이연호
- 4팀: 장건호, 기광민, 김정찬

3분 내외로 가볍게 발표해주시면 됩니다!

6-2. Week5 예,복습과제 안내, Week6 진도 안내

코드과제의 파일형식은 ipynb로, KUBIG 25-1 Github repo에 업로드 될 예정입니다!
Colab 환경에서 제작된 과제들이므로 google colab에서 실행하시는 것을 권장드립니다.

WEEK5 복습과제1

BERT for sequence classification

WEEK5 복습과제2

생성모델의 Decoding 전략

WEEK6 진도

- LLM 기초 (Fine-tuning, RAG)

수고하셨습니다!