DeepIntoDeep

More about Transformers and Pretraining

발표자: 조혜진

More about Transformers and Pretraining

조혜진

Artificial Intelligence in Korea University(AIKU)

Department of Computer Science and Engineering, Korea University

Contents

- Recap. Lec 3 (RNN) and 4 (Transformer)
- Model input
  - Tokenizer
  - Subword modeling
- Model pretraining
  - Why?
  - 3-ways: Decoders, Encoders and Encoder-Decoders

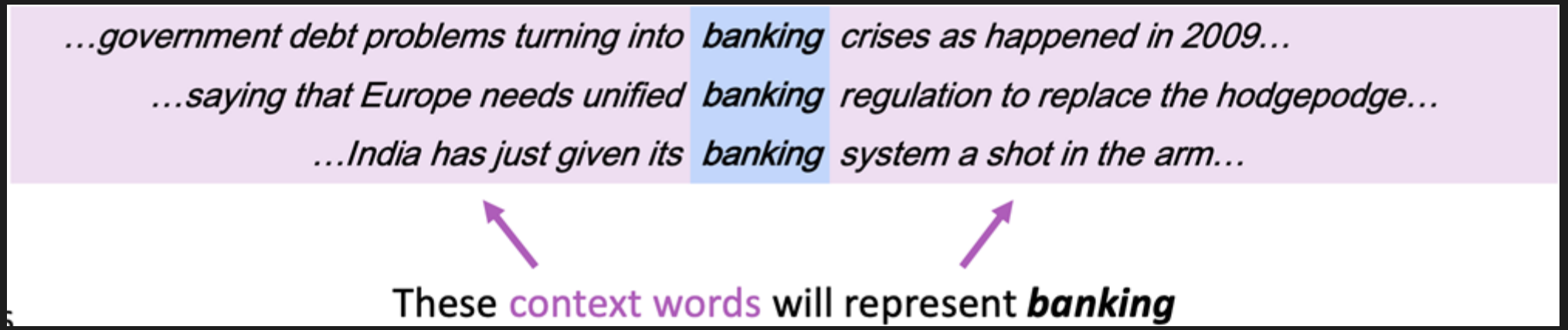
Recap. Lec 3 (RNN) and 4 (Transformer)

Recap. Lec 3 (RNN) and 4 (Transformer)

- Words Vectors (Word Embeddings)
  - Distributional Semantics: 단어의 뜻은 주변에 자주 나타나는 단어들로 표현
    - 영국의 언어학자 J.R. Firth가 제시한 개념
    - “You should know a word by the company it keeps” (J.R. Firth 1957:11)
    - 단어의 맥락은 주변에 나타나는 다른 단어들에 의해 정해진다는 뜻

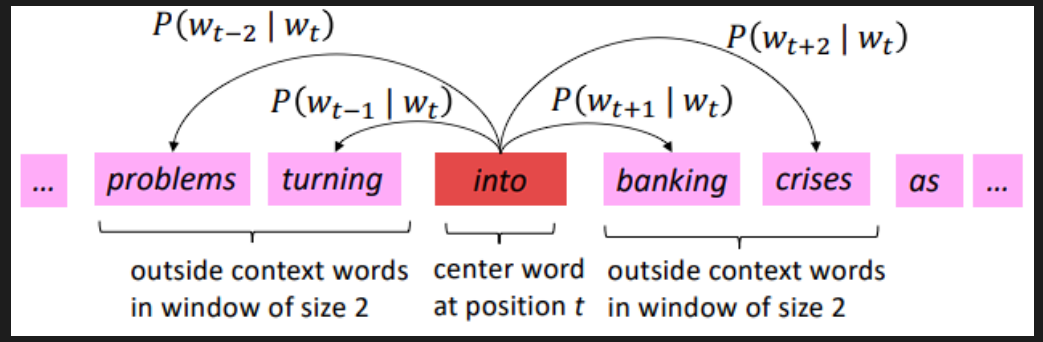

J.R. Firth

출처 : http://linguisticstheoryii.blogspot.com/2011/09/applied-linguistics-and-linguistics.html



Recap. Lec 3 (RNN) and 4 (Transformer)

- Word2Vec



출처 : CS224n (2021) Lecture 1

Problem

1. 큰 의미가 없는 단어
   - I like deep learning.
   - I like NLP.
2. Window 밖에 있는 중요한 단어
   - I went to the bank … … … … money

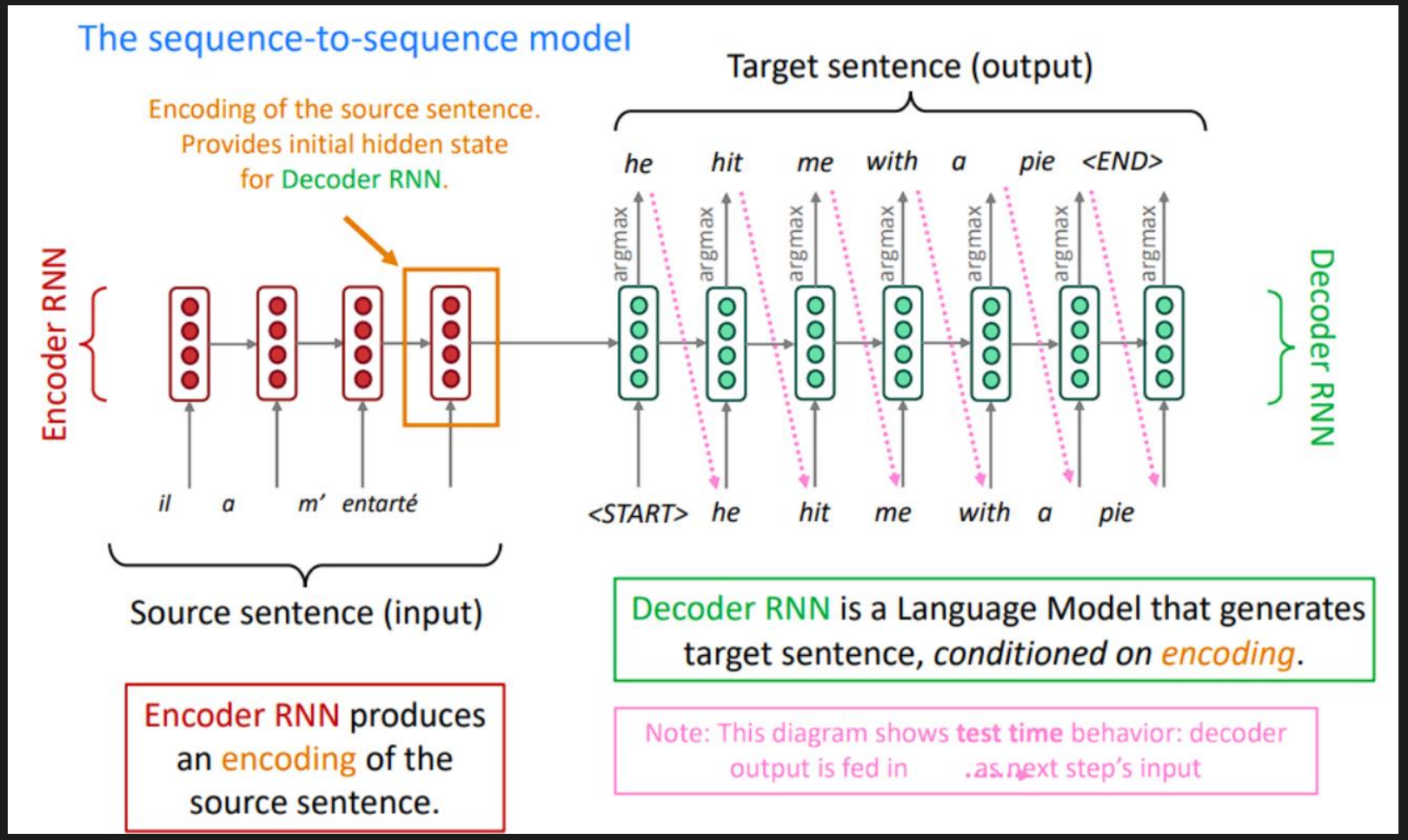
- 중심 단어를 기반으로 주변 단어를 예측하기
- 주변 단어를 기반으로 중심 단어를 예측하기

Recap. Lec 3 (RNN) and 4 (Transformer)

- n-gram language models
  - How to learn a Language Model? n-gram Language Model을 학습하자
  - Ex) the students opened their _____
  - n-gram: n개의 연속되는 단어들
    - Unigrams: “the”, “students”, “opened”, “their”
    - Bigrams: “the students”, “students opened”, “opened their”
    - Trigrams: “the students opened”, “students opened their”
    - 4-grams: “the students opened their”

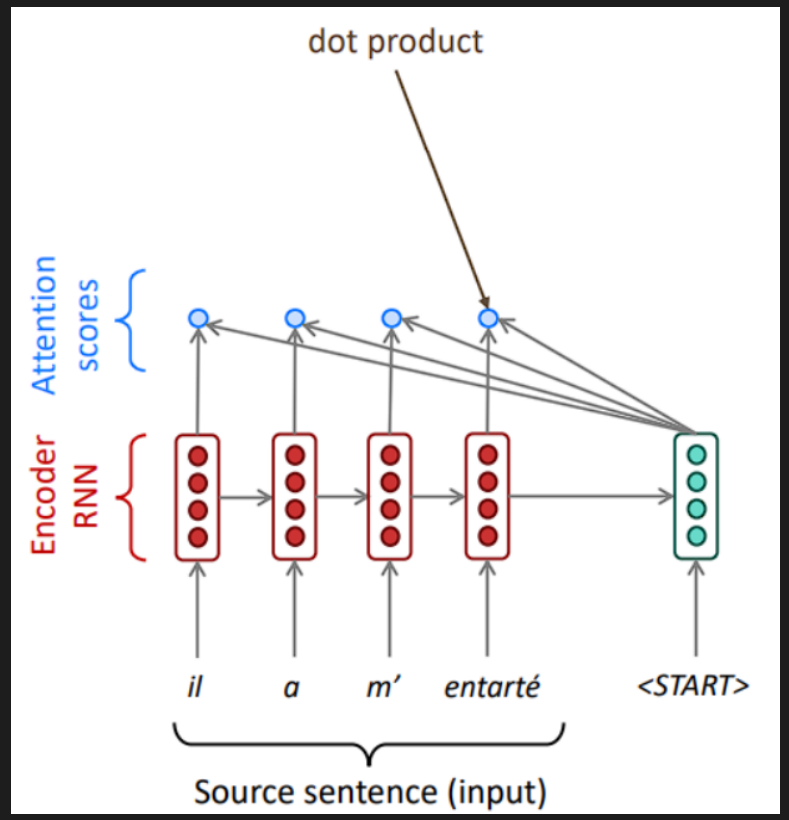
Recap. Lec 3 (RNN) and 4 (Transformer)

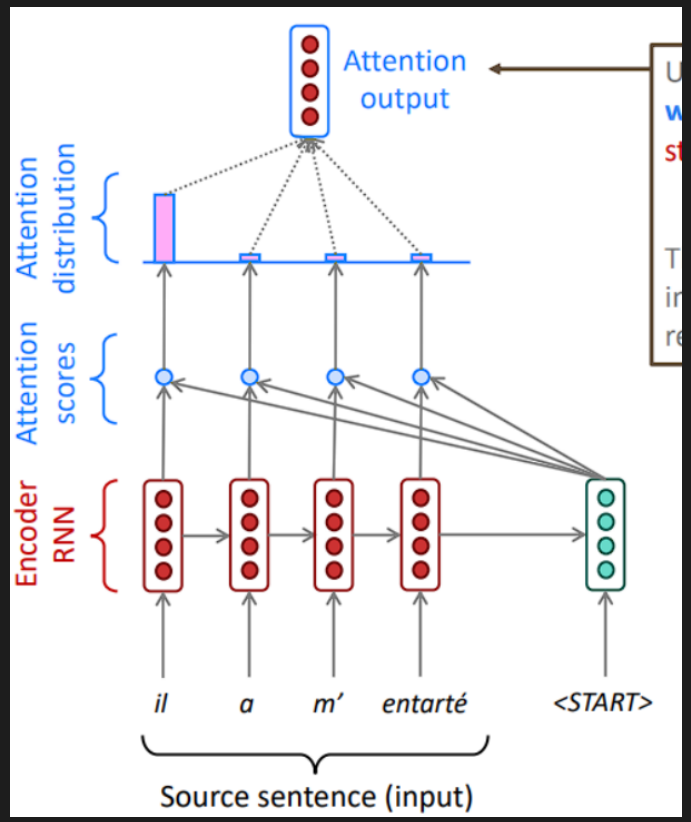
- Seq2Seq models

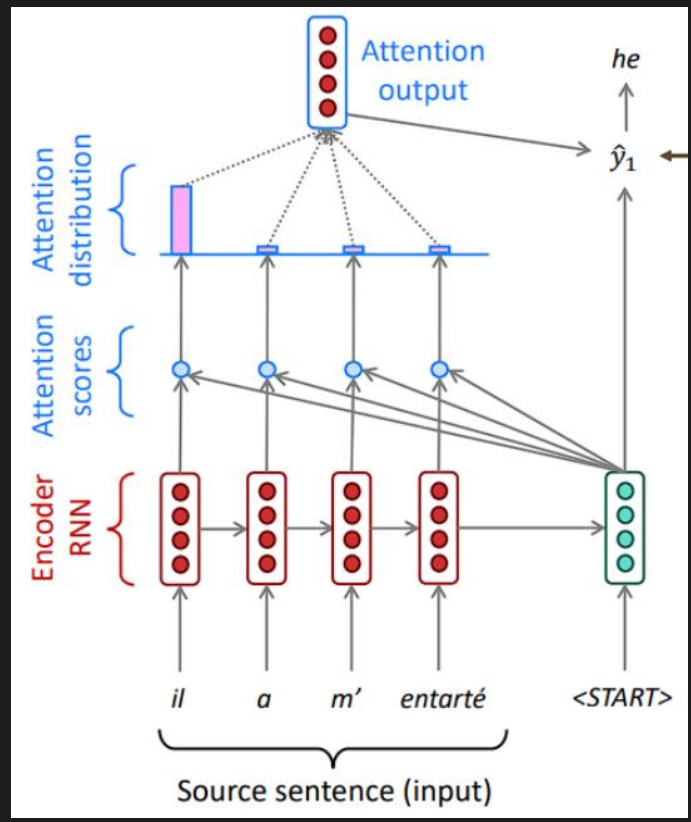


Recap. Lec 3 (RNN) and 4 (Transformer)

- Attention (+Seq2Seq)

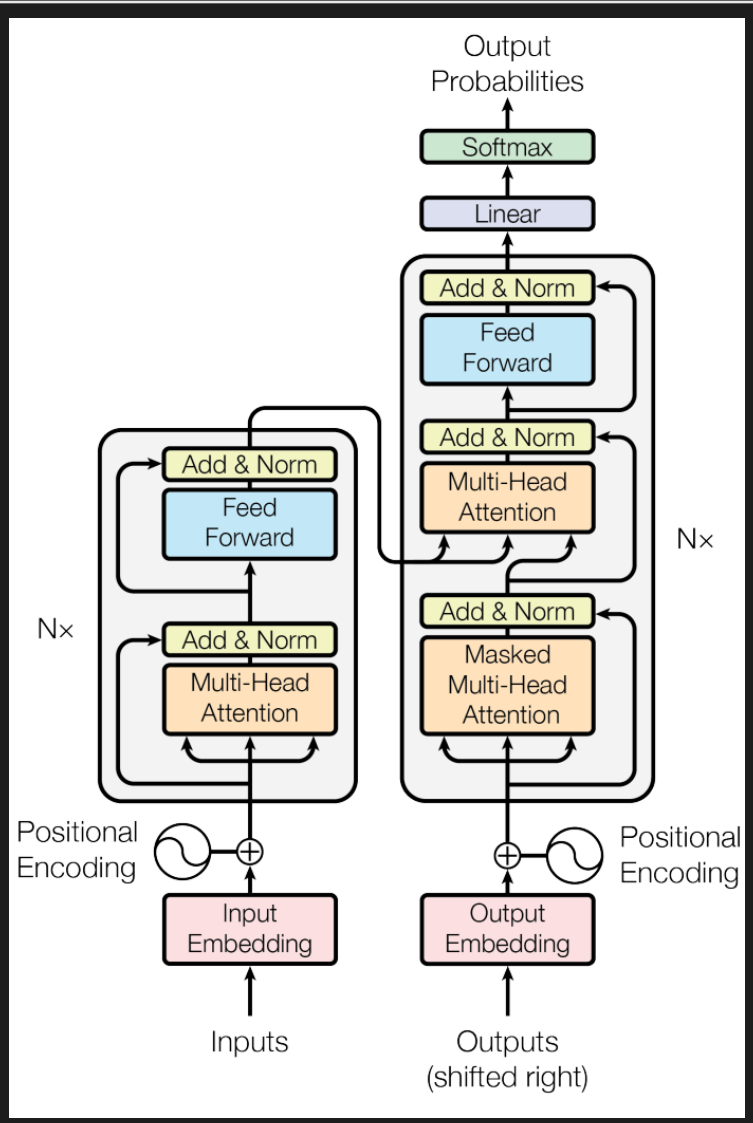






Recap. Lec 3 (RNN) and 4 (Transformer)

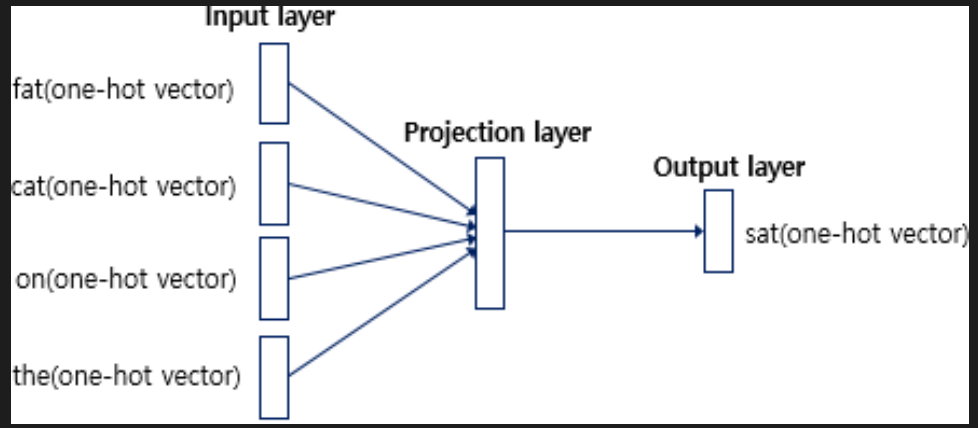
- Transformer

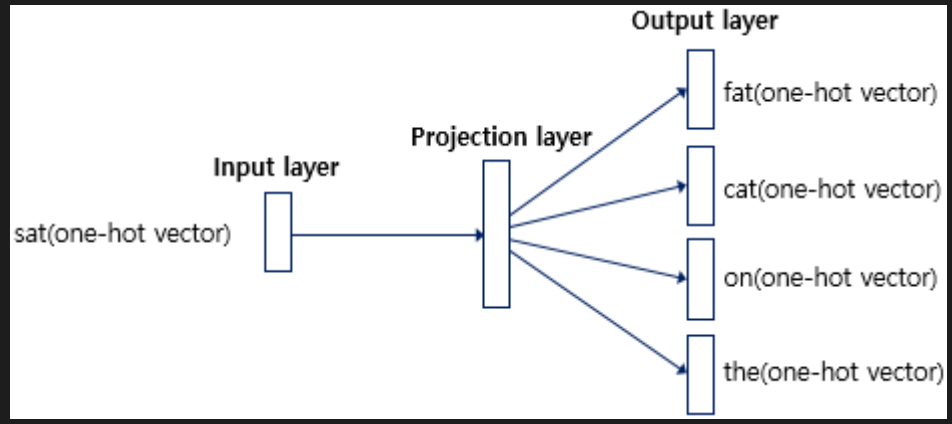


Model Input

Model input: Tokenizer

- word2vec

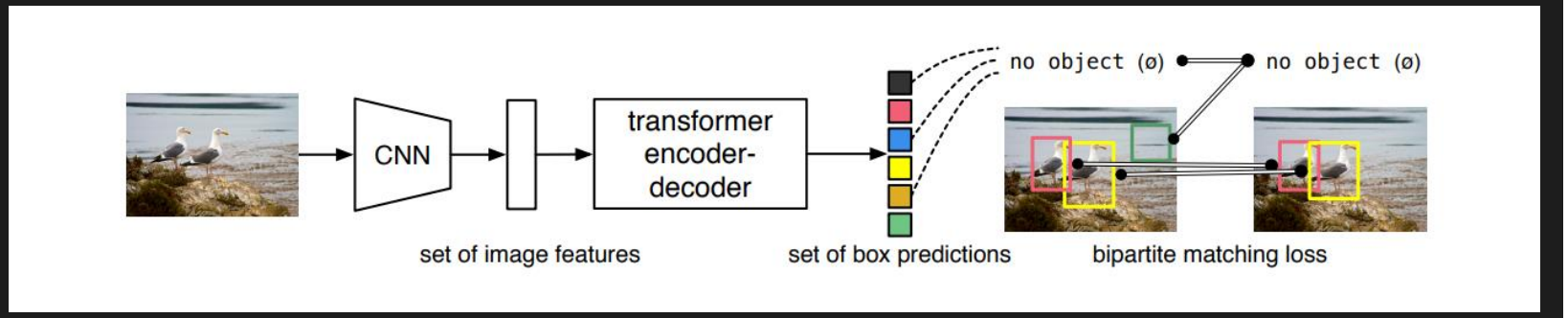




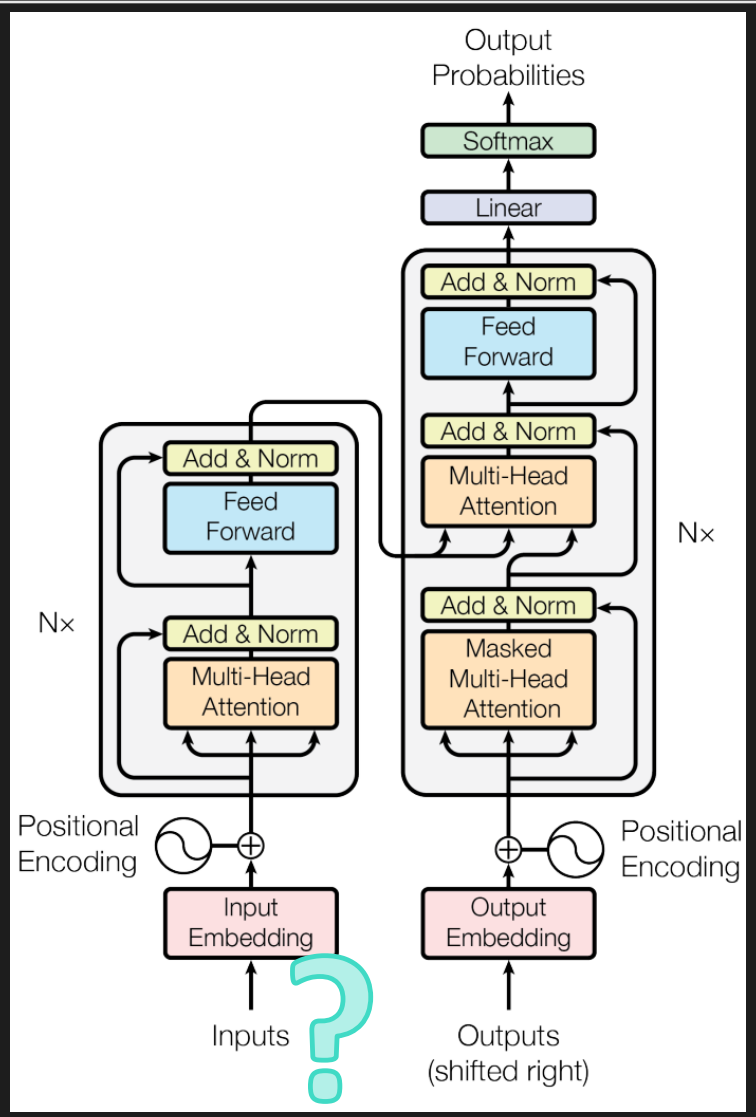
- CBOW Skip-gram

- One-hot vector로 표현된 **단어**가 neural network의 input으로 들어감
  - 단어(word)?

Model input: Tokenizer



- Image model에서는
  - 이미지 픽셀 단위의 RGB 값이 input으로 들어간다!
  - (물론 Transformer 계열일 경우 살짝 다름)
- 그러면 Language model에서는?
  - text의 token id 값이 input으로 들어간다
  - Token을 어떻게 나누는가?



Model input: Tokenizer

- Tokenizer?
  - Input text를 모델에서 처리할 수 있는 데이터로 바꾸어 주는 것
  - Text를 어떤 단위로 나누냐와 연관됨
- 예시
  - 띄어쓰기를 기준으로 token화

```
tokenized_text = "Jim Henson was a puppeteer".split()
print(tokenized_text)
```

```
['Jim', 'Henson', 'was', 'a', 'puppeteer']
```

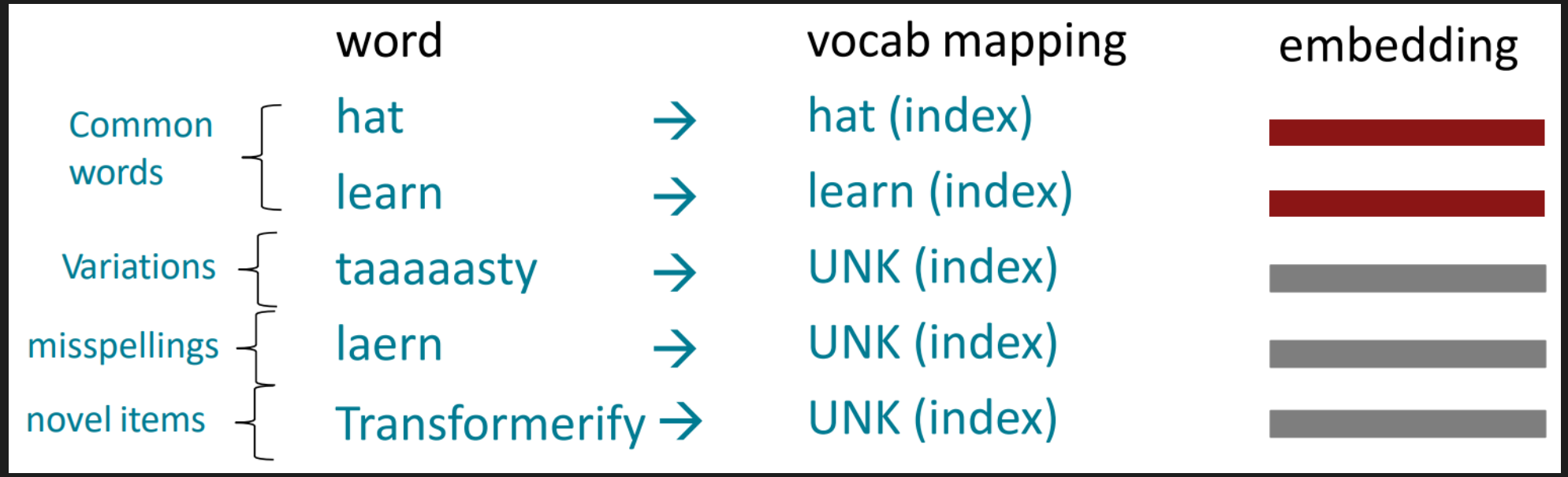
형태소를 기준으로 token화

```
print('꼬꼬마 형태소 분석 :',kkma.morphs("열심히 코딩한 당신, 연휴에는 여행을 가봐요"))
print('꼬꼬마 품사 태깅 :',kkma.pos("열심히 코딩한 당신, 연휴에는 여행을 가봐요"))
print('꼬꼬마 명사 추출 :',kkma.nouns("열심히 코딩한 당신, 연휴에는 여행을 가봐요"))
```

```
꼬꼬마 형태소 분석 : ['열심히', '코딩', '하', 'ㄴ', '당신', ',', '연휴', '에', '는', '여행', '을', '가보', '아요']
꼬꼬마 품사 태깅 : [('열심히', 'MAG'), ('코딩', 'NNG'), ('하', 'XSV'), ('ㄴ', 'ETD'), ('당신', 'NP'), (',', 'SP'), ('연휴', 'NNG'), ('에', 'JKM'), ('는', 'JX'), ('여행', 'NNG'), ('을', 'JKO'), ('가보', 'VV'), ('아요', 'EFN')]
꼬꼬마 명사 추출 : ['코딩', '당신', '연휴', '여행']
```

Model input: Tokenizer

- 단어(Word) 단위로 토큰화를 할 때 문제점
  - Like, likes, liked, … → 단어의 개수는 무한 개! 그러나 vocabulary는 유한할 수 밖에 없다.
  - Gooood, laern(오탈자), IIRC(신조어) → novel word를 만날 경우 UNK token으로 매핑
    → OOV(Out of Vocabulary) problem



Model input: Tokenizer

- 단어(Word) 단위로 토큰화를 할 때 문제점
  - 단어 형태나 구조가 복잡한 경우에는 적용이 어렵다
    - ex. 어미가 발달한 언어 특성상,
      한국어의 용언은 다양한 활용형을 가짐

Korean verb '하다' Conjugated

regular verb

Form		Conjugation
base	하	ha
base2	하	ha
base3	하	ha
declarative present informal low	해	hae
declarative present informal high	해요	hae-yo
declarative present formal low	한다	han-da
declarative present formal high	합니다	hab-ni-da
past base	했	haess
declarative past informal low	했어	haess-eo
declarative past informal high	했어요	haess-eo-yo
declarative past formal low	했다	haess-da
declarative past formal high	했습니다	haess-seub-ni-da
future base	할	hal
declarative future informal low	할 거야	hal geo-ya
declarative future informal high	할 거예요	hal geo-ye-yo
declarative future formal low	할 거다	hal geo-da
declarative future formal high	할 겁니다	hal geob-ni-da
declarative future conditional informal low	하겠어	ha-gess-eo
declarative future conditional informal high	하겠어요	ha-gess-eo-yo
declarative future conditional formal low	하겠다	ha-gess-da
declarative future conditional formal high	하겠습니다	ha-gess-seub-ni-da
inquisitive present informal low	해?	hae?
inquisitive present informal high	해요?	hae-yo?
inquisitive present formal low	하니?	ha-ni?
inquisitive present formal high	합니까?	hab-ni-gga?
inquisitive past informal low	했어?	haess-eo?
inquisitive past informal high	했어요?	haess-eo-yo?
inquisitive past formal low	했니?	haess-ni?
inquisitive past formal high	했습니까?	haess-seub-ni-gga?
imperative present informal low	해	hae
imperative present informal high	하세요	ha-se-yo
imperative present formal low	해라	hae-ra
imperative present formal high	하십시오	ha-sib-si-o
propositive present informal low	해	hae
propositive present informal high	해요	hae-yo
propositive present formal low	하자	ha-ja
propositive present formal high	합시다	hab-si-da
connective if	하면	ha-myeon
connective and	하고	ha-go
nominal ing	함	ham

Jamo Tables

Model input: Subword modeling

- 이러한 문제의 해결 방안?
  - 단어 단위에서 더 쪼개어, subword 단위로 tokenize하자!

```
sequence = "Using a Transformer network is simple"
tokens = tokenizer.tokenize(sequence)

print(tokens)
```

```
['Using', 'a', 'Trans', '##former', 'network', 'is', 'simple']
```

```
print(tokenizer.tokenize('새는 알에서 나오기 위해 투쟁한다. 알은 세계이다.'))
```

```
['_새', '는', '_알', '에서', '_나오', '기', '_위해', '_투', '쟁', '한다', '.', '_알', '은', '_세계', '이다', '.']
```

Model input: Subword modeling

- BPE (Byte-pair encoding) Algorithm
  - Subword modeling의 한 종류
  - 매우 간단하지만 효과적임
  - 최종적으로 “subword vocabulary”정의

Model input: Subword modeling

- BPE (Byte-pair encoding) Algorithm
  - 어떻게 동작할까?

  - Ex. Training dataset에 다음과 같은 빈도수를 가진 단어들이 있다고 하자.
    - low : 5, lower : 2, newest : 6, widest : 3
  1. 단어의 character들을 subword로 두고 시작한다. Subword를 vocabulary에 추가한다.
     - L o w : 5, l o w e r : 2, n e w e s t : 6, w i d e s t : 3
  2. 인접한 subword 두 개 중 가장 빈도가 높은 subword를 찾는다.
     - L o w : 5, l o w e r : 2, n e w **e s** t : 6, w i d **e s** t : 3
     - → 여기서는 (e, s)가 총 9번 등장하므로 가장 빈도가 높다.

Model input: Subword modeling

- BPE (Byte-pair encoding) Algorithm
  - 어떻게 동작할까?

  - Ex. Training dataset에 다음과 같은 빈도수를 가진 단어들이 있다고 하자.
    - low : 5, lower : 2, newest : 6, widest : 3
  3. 해당 subword들을 합쳐서 새로운 subword로 만든다. 이 subword를 vocabulary에 추가한다.
     - L o w : 5, l o w e r : 2, n e w es t : 6, w i d es t : 3
  4. 2 ~ 3의 과정을 반복한다.
     - Iteration 수를 직접 지정해주거나, 원하는 vocabulary size가 나올 때 까지 반복한다.
  5. 반복이 끝난 후, 최종적으로 만들어진 vocabulary를 얻는다.

Model input: Subword modeling

- BPE (Byte-pair encoding) Algorithm

Algorithm 1 Learn BPE operations

```
import re, collections

def get_stats(vocab):
  pairs = collections.defaultdict(int)
  for word, freq in vocab.items():
    symbols = word.split()
    for i in range(len(symbols)-1):
      pairs[symbols[i],symbols[i+1]] += freq
  return pairs

def merge_vocab(pair, v_in):
  v_out = {}
  bigram = re.escape(' '.join(pair))
  p = re.compile(r'(?<!\S)' + bigram + r'(?!\S)')
  for word in v_in:
    w_out = p.sub(''.join(pair), word)
    v_out[w_out] = v_in[word]
  return v_out

vocab = {'l o w </w>' : 5, 'l o w e r </w>' : 2,
         'n e w e s t </w>':6, 'w i d e s t </w>':3}
num_merges = 10
for i in range(num_merges):
  pairs = get_stats(vocab)
  best = max(pairs, key=pairs.get)
  vocab = merge_vocab(best, vocab)
  print(best)
```

Model input: Subword modeling

- WordPiece
  - BPE → 인접한 subword가 등장하는 횟수를 기준으로 병합
  - 병합되었을 때 likelihood가 가장 큰 subword들을 병합
- SentencePiece
  - 중국어, 일본어 등 띄어쓰기가 없어 pre-tokenize 작업이 어려운 언어에도 사용 가능
  - Raw sentence에서 바로 토큰화 수행
  - 자세한 알고리즘은… 보면 어지러우므로 생략. (참고 자료에 기재할 테니 관심있으면 찾아보기 바람)

Model input: Tokenizer

- 다시 tokenizer로 돌아가면…
- Tokenizer의 pipeline은 다음과 같다.

Tokenizer는 과제4에서 직접 다뤄 볼 예정입니다!

1. Input corpus를 토큰화

```
print(tokenizer.tokenize('새는 알에서 나오기 위해 투쟁한다. 알은 세계이다.'))
```

```
['_새', '는', '_알', '에서', '_나오', '기', '_위해', '_투', '쟁', '한다', '.', '_알', '은', '_세계', '이다', '.']
```

2. 토큰화된 subword를 id에 mapping

```
print(tokenizer.convert_tokens_to_ids(tok))
```

```
[2695, 5760, 3166, 6903, 1388, 5561, 3567, 4762, 7198, 7831, 54, 3166, 7086, 2802, 7100, 54]
```

(3. 해당 id tensor를 model에 넣어 학습 후 output 얻기)

4. Output을 tokenizer에서 decode해서 최종 결과 얻기

```
print(tokenizer.decode(res))
```

```
새는 알에서 나오기 위해 투쟁한다. 알은 세계이다.
```

Model Pretraining

Model Pretraining

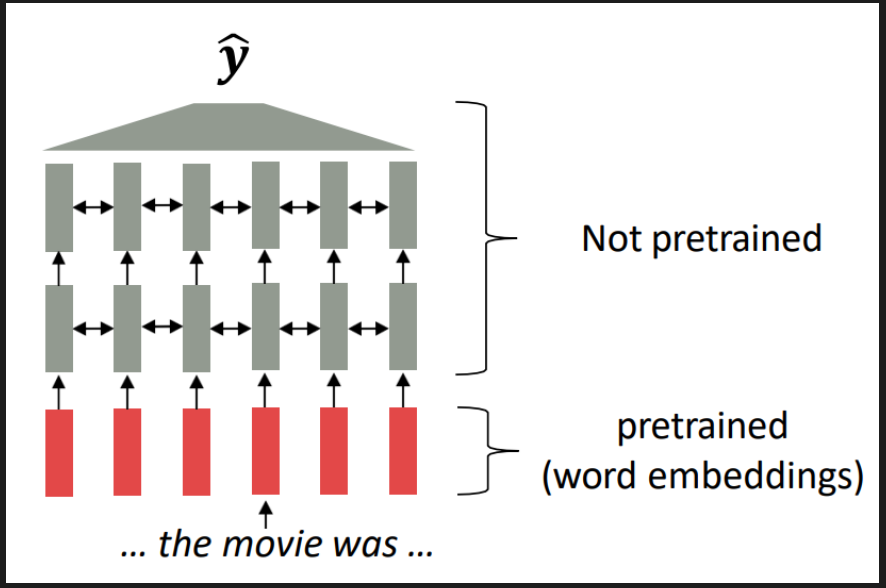

J.R. Firth

- Word2Vec을 배울 때 언급된 언어학자는
  - “You should know a word by the company it keeps”
- 외에 다음과 같은 명언도 남겼다.
  - “… the complete meaning of a word is always contextual, and no study of meaning apart from a complete context can be taken seriously.” (J. R. Firth 1935)

- 바람이 잘 불길 바람
  - 두 “바람”은 다른 의미를 가진다. 그리고 이는 문맥을 통해 파악 가능하다.
- 그러나, word embedding은 문맥 정보를 고려하지 X

Model Pretraining

- 그림과 같은 모델이 있다고 하자.
  - 우리는 이 모델을 학습시켜서 Question Answering과 같은 downstream task에 fine-tuning 하려고 한다.
  - 하지만 우리가 가지고 있는 fine-tuning용 training data의 양이 모델로 하여금 언어의 문맥적 의미를 파악하게 하는 데에 충분할까?



Model Pretraining

- 보통 dataset의 경우
  - Unlabeled data는 충분하지만 (=쉽게 만들 수 있지만)
  - 각 downstream task에 대한 Labeled data는 만드는 데 비용이 많이 듦

- 그러므로, two-stage training 방법론을 적용하자!
  1. Language modeling objective on the unlabeled data to learn the initial parameters

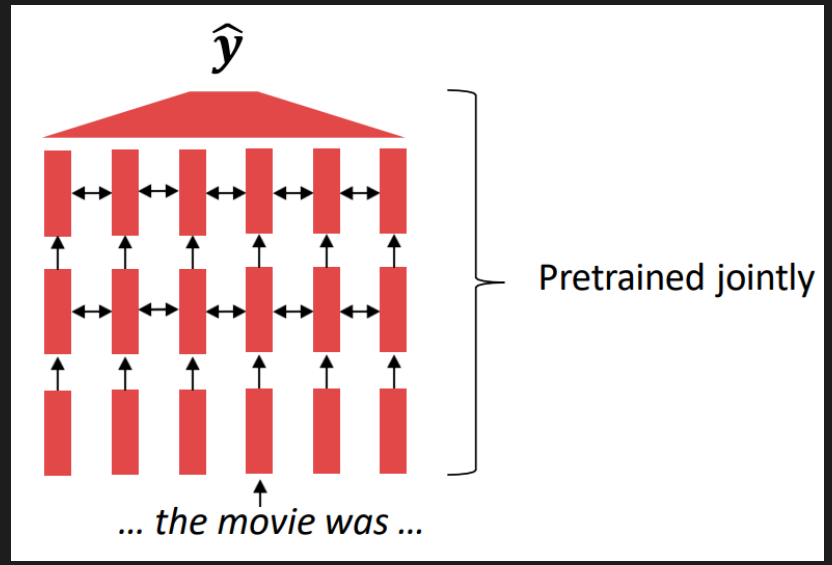
     : learning high-capacity language model on a large corpus of text
  2. Adapt the parameters to a target task using the corresponding supervised objective

     : adapt the model to a discriminative task with labeled data

Model Pretraining

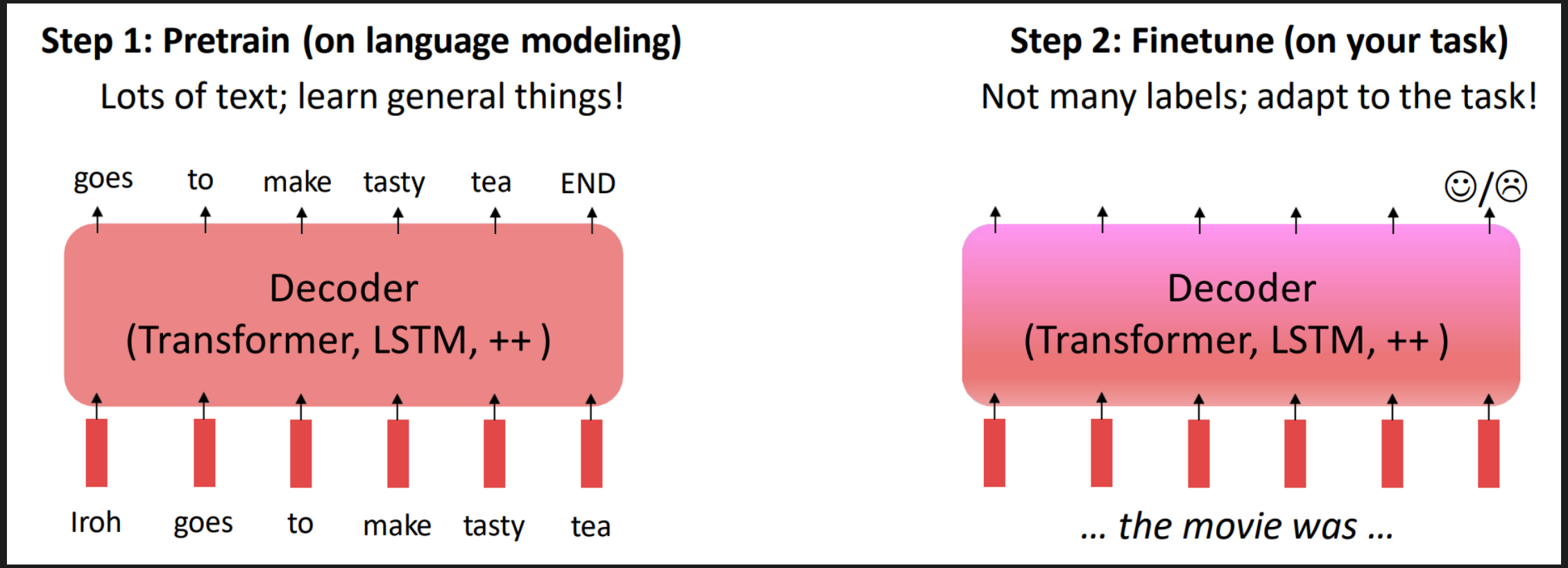
- 보통은 그렇지 않기 때문에, modern NLP 모델은 모두
  pretraining – finetuning 절차를 거침
  - 매~우 많은 양의 dataset으로 network를 initialization
  - 보통 input의 일부분을 가린 뒤에 모델에게 재구성하게 하는 task를 수행

- 이러한 pretraining 과정을 통해
  - Language에 대해 더 강한 representation을 배울 수 있음
  - 최종 output은 단어의 문맥적 의미를 잘 파악할 수 있게 됨



Model Pretraining

- The Pretraining / Fine-tuning Paradigm



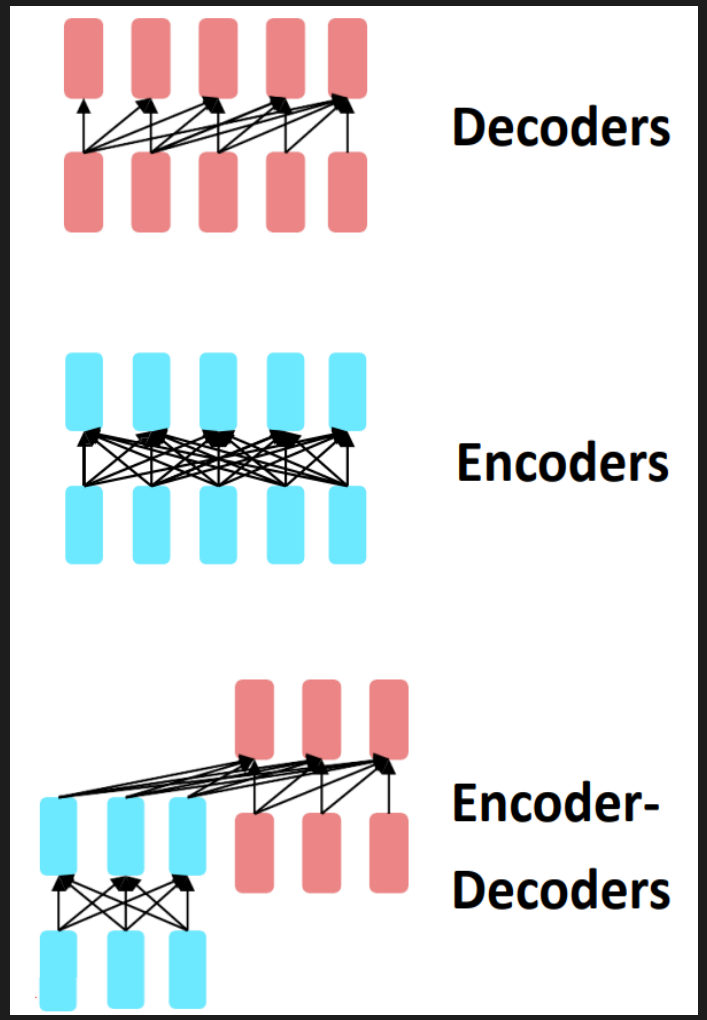
Model Pretraining

- Neural Network를 training하는 관점에서 pretrain의 역할에 대해 생각해보자!
  - Pretrain을 하면 최적의 파라미터 $\hat{\theta} = \underset{\theta}{\operatorname{argmin}} \mathcal{L}_{pretrain}(\theta)$ 를 찾을 수 있다.
  - Finetuning을 하면, $\hat{\theta}$에서 출발해서 파라미터를 업데이트
  - 최종 파라미터 $\theta^* = \underset{\theta}{\operatorname{argmin}} \mathcal{L}_{finetuning}(\hat{\theta})$

  - 상식적으로 생각해봐도 $\hat{\theta}$ 근처의 local minima가 더 최적의 파라미터일 확률이 높음

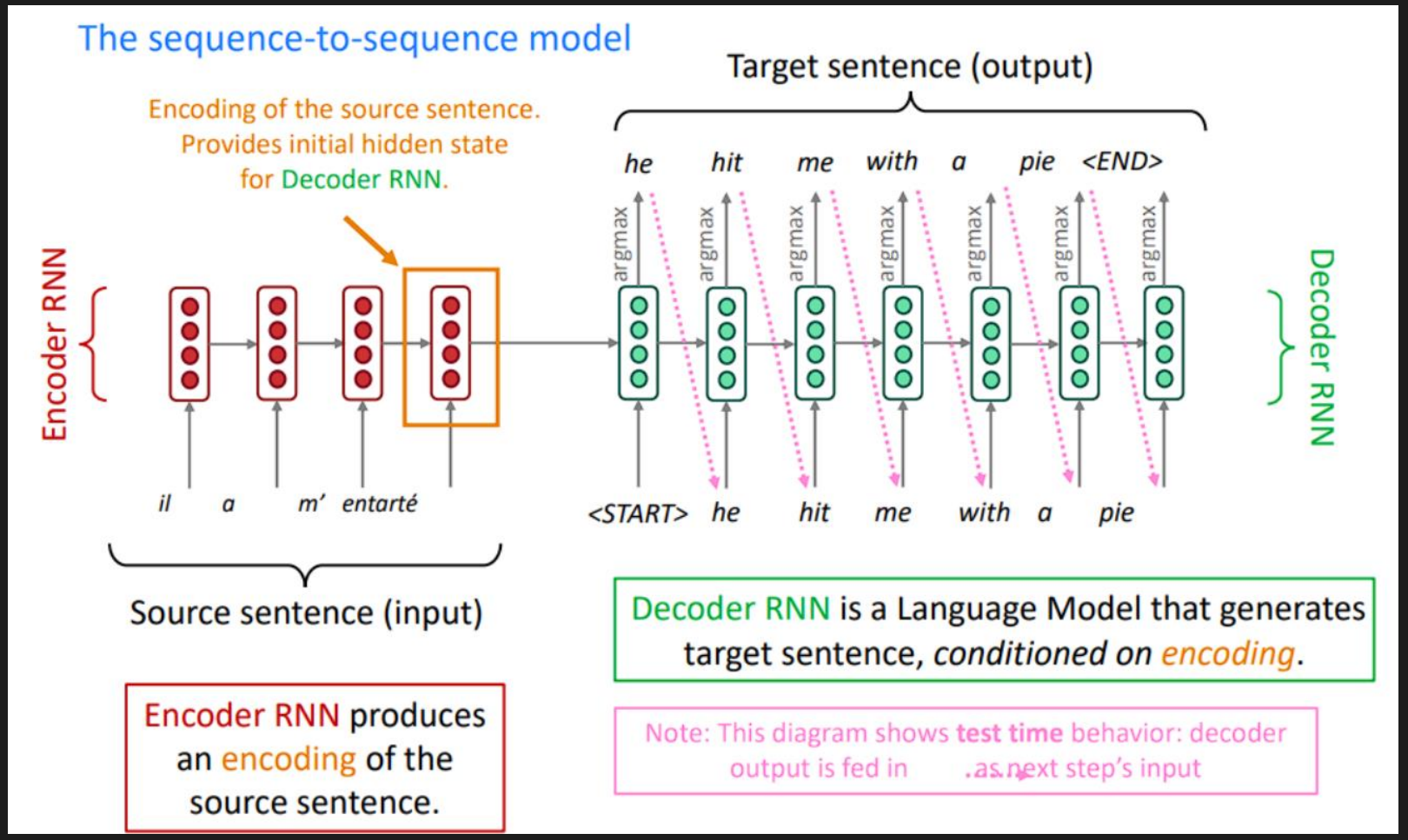
Model Pretraining: 3-ways

- Pretrained model은 보통 세 종류의 구조 가짐
  - Decoders
  - Encoders
  - Encoder-Decoders



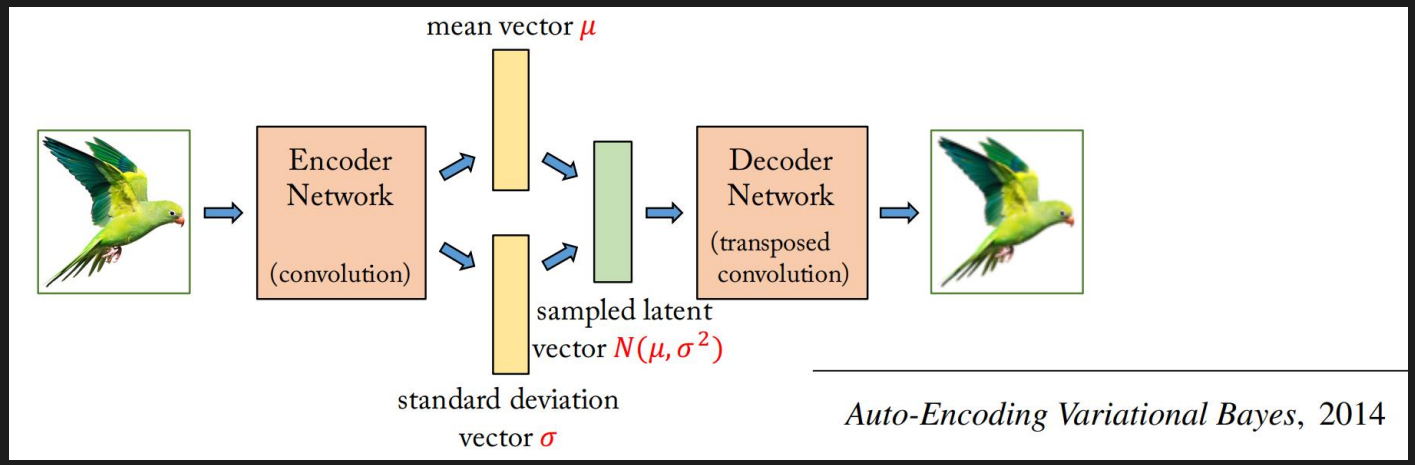
Model Pretraining: 3-ways

- Encoder, Decoder?



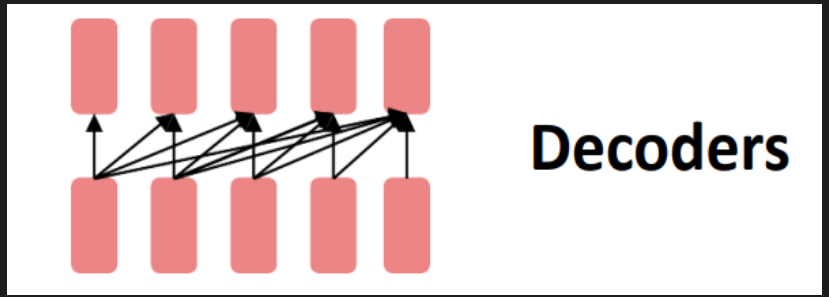
Model Pretraining: 3-ways

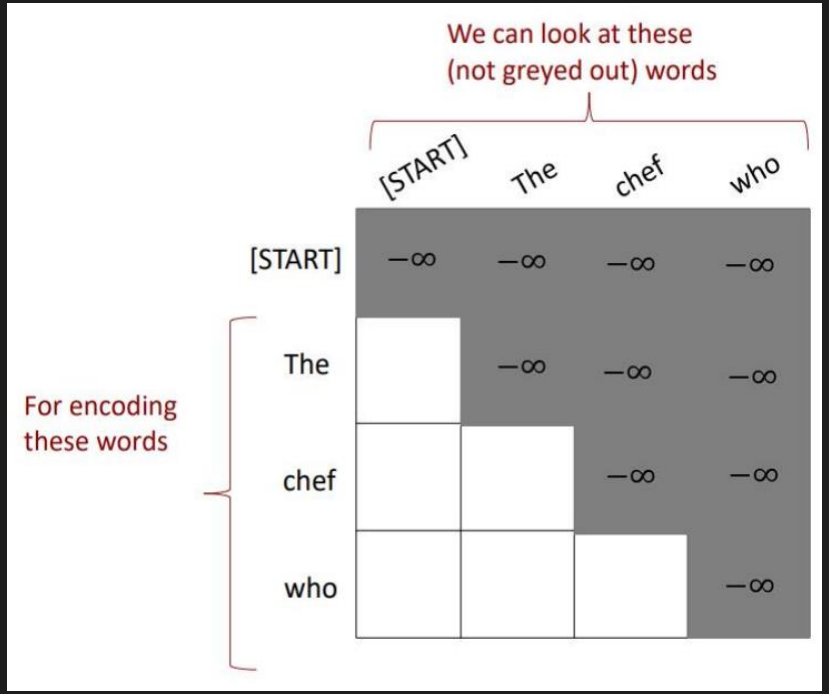
- Encoder
  - input의 정보를 압축하는 역할
  - 보통 encode의 output은 input의 정보를 담은 feature vector(s)
  - (모델 구조에 따라 output이 단일 벡터일 경우 latent vector라고도 부름)
- Decoder
  - Encoder로부터 정보를 받아 원하는 결과를 생성하는 역할



Model Pretraining: Decoders

- Decoder로만 구성된 language model
  - Output token을 “생성”해야 하므로 자기보다 뒤에 있는 token은 볼 수 없다! (masked self-attention)
  - 그러므로 단방향에서, 자기 다음에 올 단어를 예측하는 task로 pretraining 수행





Model Pretraining: Decoders

- Pretraining task for Decoder-only models

“혜진이는 점심으로 찜닭을 ___”

“동해물과 백두산이 마르고 ___”

“내 지난 날들은 눈 뜨면 잊는 ___”

“혜정아, 이사라가 니 ___”

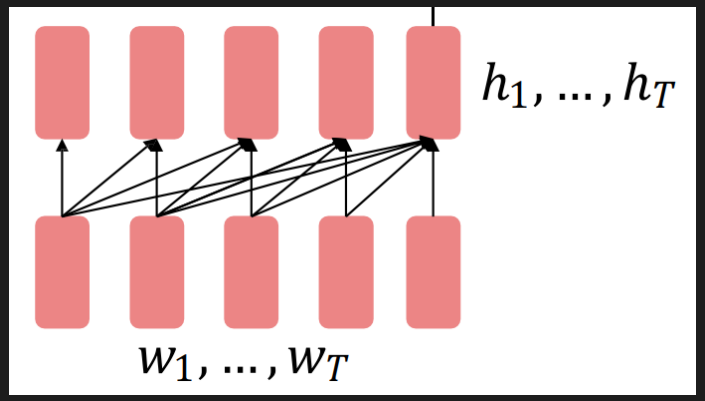
“중요한 건 꺾이지 않는 ___”

Model Pretraining: Decoders

- Pretraining task for Decoder-only models
  - Token $w_{1:t-1}$이 주어졌을 때, 그 다음 단어로 w_t(GT)가 올 확률 $p(w_t|w_{1:t-1})$ 을 최대화 (Next word prediction)
  - 이러한 확률들의 곱(=log에서는 합)을 최대화 하도록 파라미터를 학습시키자.

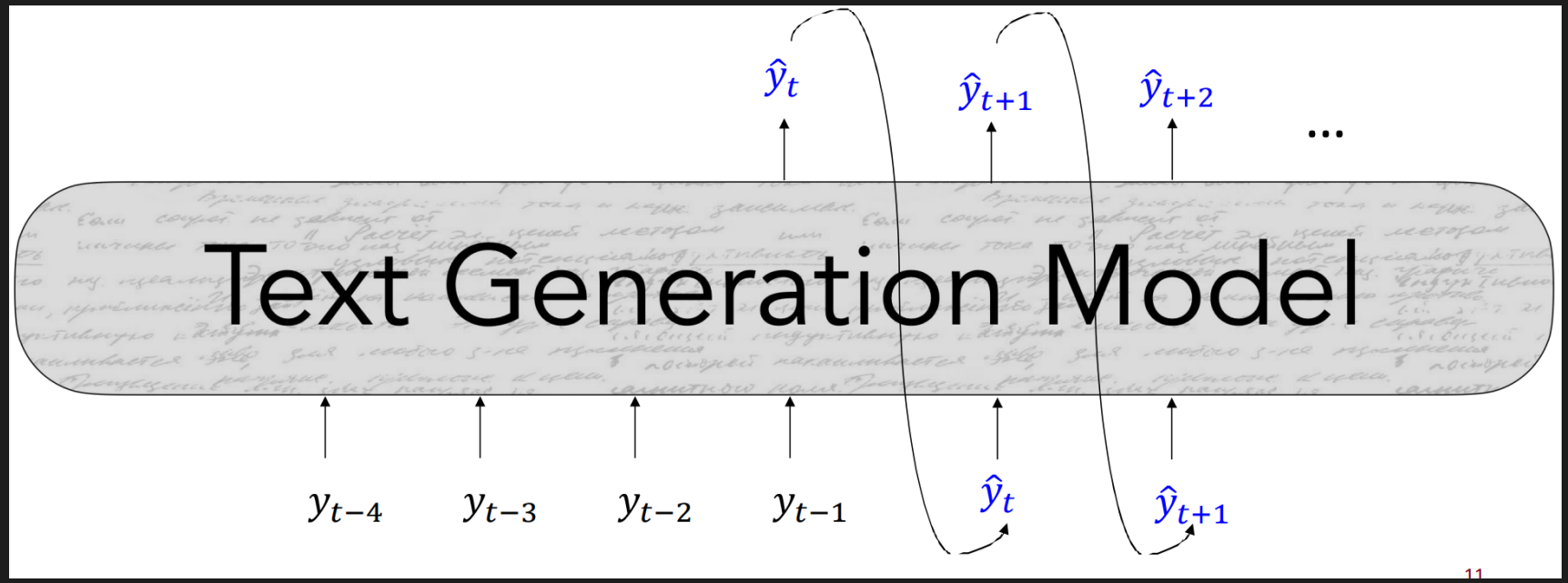
$$L_1(\mathcal{U}) = \sum_i \log P(u_i|u_{i-k}, \ldots, u_{i-1}; \Theta)$$

  - 이러한 방식으로 학습하는 것을 Causal language modeling 이라고 한다!
    - 혹은 앞 단어를 생략하고 language modeling이라고도 함



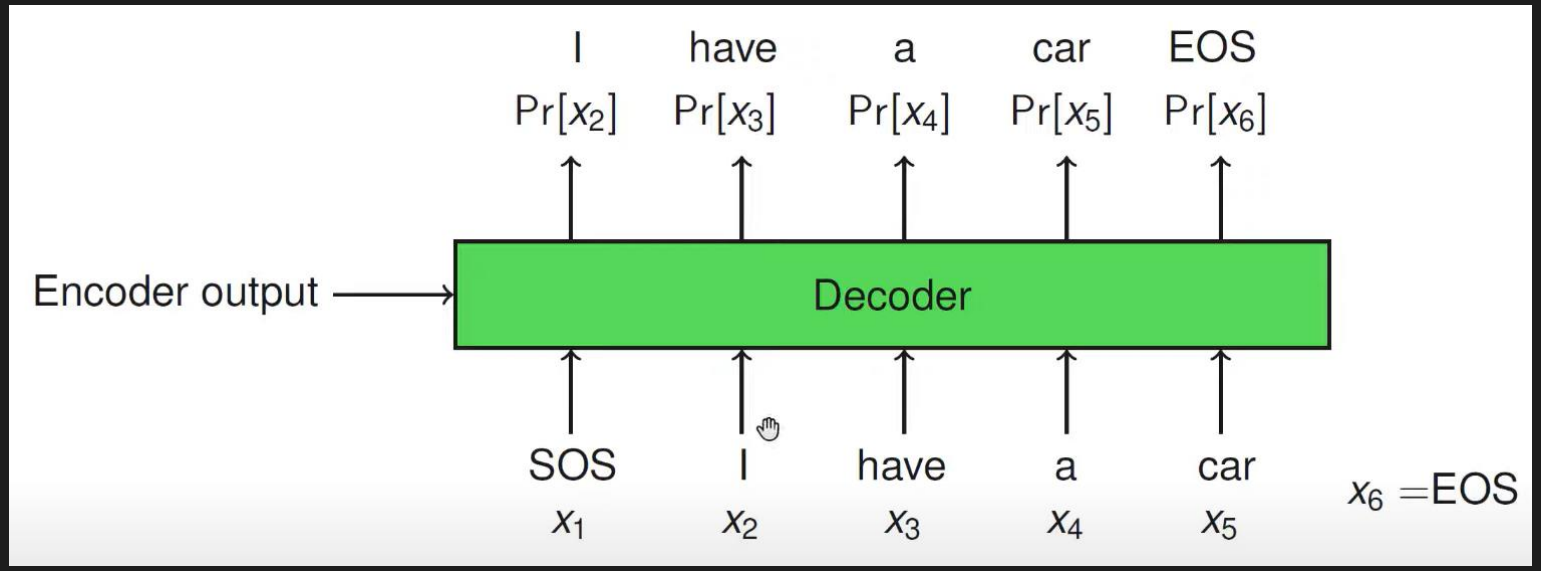
Model Pretraining: Decoders

- How to generate by Transformer decoders? (side note)
  - 자세한 내용은 nlp 집중 탐색 II 에서 다룰 예정
  - Auto-regressive model



Model Pretraining: Decoders

- How to generate by Transformer decoders? (side note)
  - 그러나 parallel한 연산을 지향하는 Transformer 특성 상,
    training에서 데이터를 한꺼번에 넣어줄 때도 있음!
    → masked self-attention이 필요한 이유



출처: https://www.youtube.com/watch?v=piT1_k8b9uM

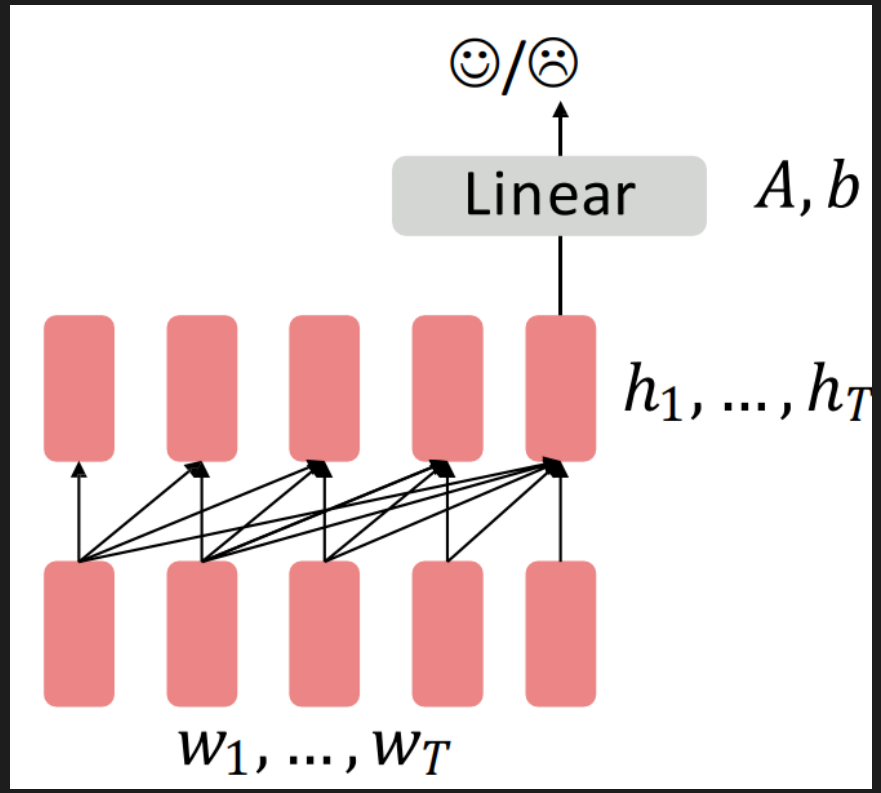
Model Pretraining: Decoders

- How to fine-tune? (output is non-sequence)
  - Pre-trained된 모델 위에 head (e.g. classifier)를 올린다.

$$h_1, \dots, h_T = \text{Decoder}(w_1, \dots, w_T)$$
$$y \sim Ah_T + b$$

  - Head를 포함해서 downstream task에 맞게 학습시킨다.

  - 해당 downstream task
    - Classification
    - Sentiment analysis
    - Question-Answering (정답이 있는 위치 예측)
    - Extractive Summarization

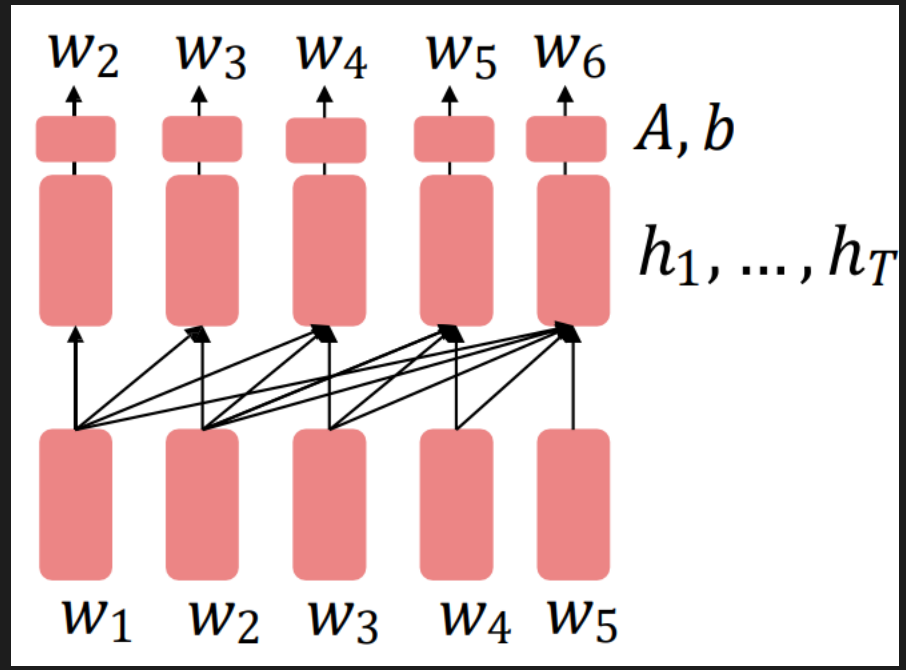


Model Pretraining: Decoders

- How to fine-tune? (output is sequence)
  - Causal language modeling대로,

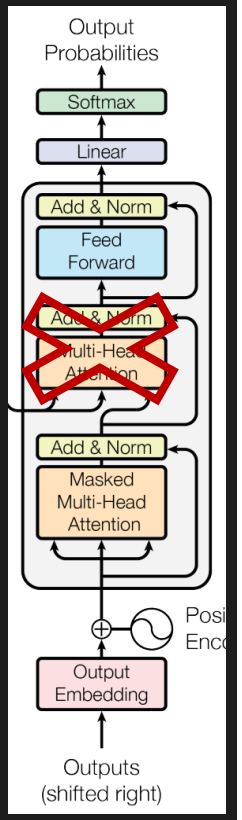
    downstream task dataset에 맞게 $p_\theta(w_t|w_{1:t-1})$ 를 최대화 하도록 훈련
    - 이 때 보통 dataset은 label이 존재 (supervised learning)

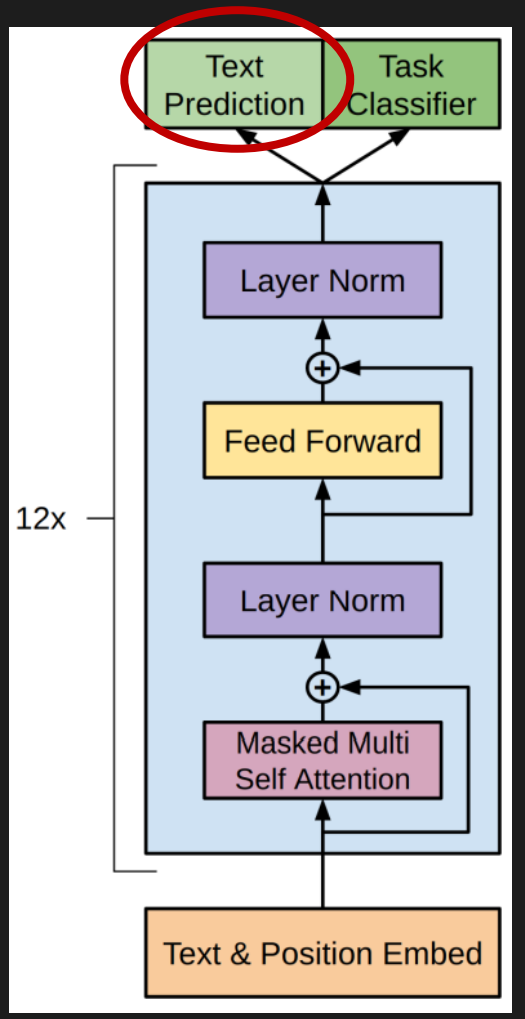
  - 해당 downstream task
    - Summarization
    - Dialogue
    - Question-Answering
    - Abstractive Summarization



Model Pretraining: Decoders

- 모델 예시: Generative Pretrained Transformer (GPT)
  - Multi-layer Transformer decoder only (L=12)
    - Apply Multi-head self-attention (heads=12)
    - 768 dimensional states
    - 3072 dimensional feed-forward hidden layers
    - Encoder가 존재하지 않으므로 cross attention층은 사라짐
  - Adam optimization
  - Use byte-pair encoding (BPE) with 4000 merges
  - Use GELU
  - Learnable position embedding parameter



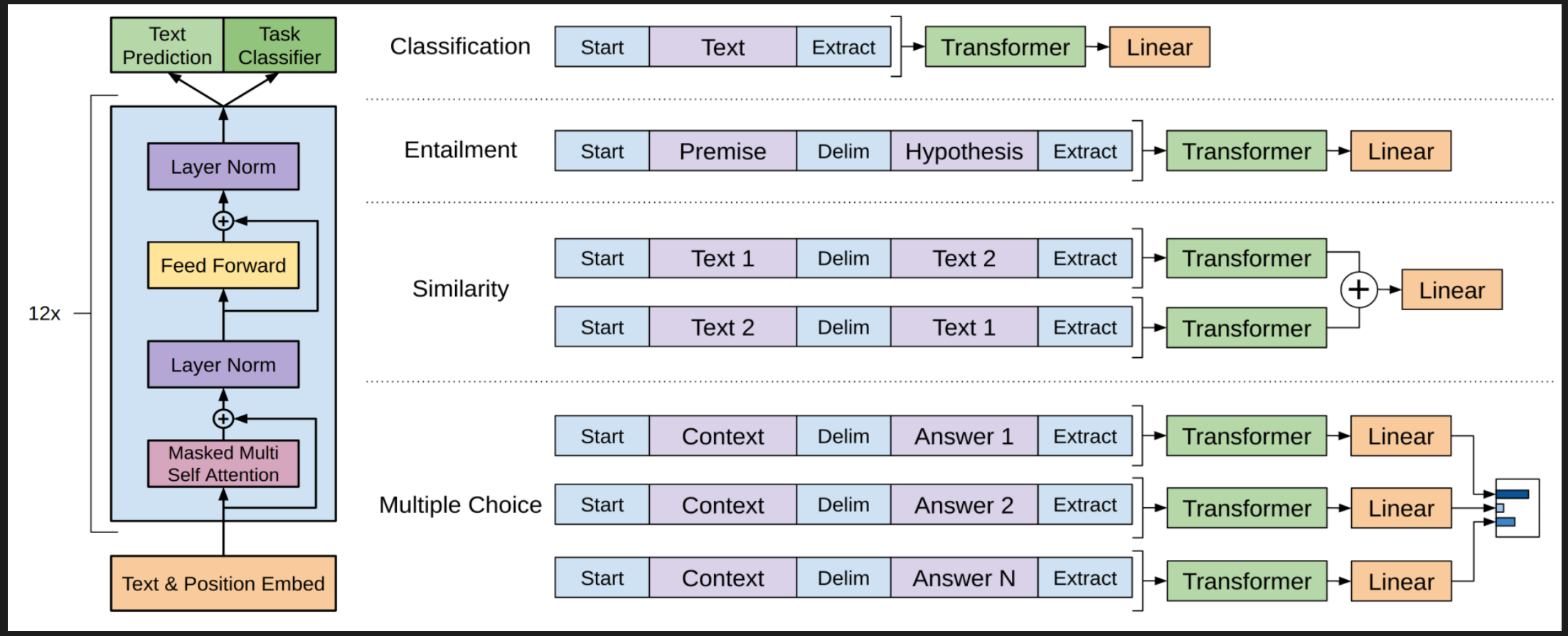


Model Pretraining: Decoders

- 모델 예시: Generative Pretrained Transformer (GPT)
  - How to finetune?
    - 이전 연구들은 backbone 위에 specifi한 architecture(head)를 붙여 fine-tune
    - GPT는 그 대신, task-specific한 input adaption을 사용
    - 즉, task에 따라 모델의 구조가 달라지는 게 아니라, input의 형태가 달라짐!
    - Enable to fine-tune effectively with minimal changes to the architecture of model

Model Pretraining: Decoders

- 모델 예시: Generative Pretrained Transformer (GPT)
  - How to finetune?



Model Pretraining: Decoders

- 모델 예시: Generative Pretrained Transformer (GPT)
  - How to finetune?
    - Linear layer의 경우 [EXTRACT] token 위치에 해당하는 output representation에 적용됨
  - Natural Language Inference (Entailment)
    - Premise(전제)와 Hypothesis(가정)이 주어졌을 때
    - 둘 사이의 관계가 entailing (함의)인지 contradictory(모순)인지 neutral(관련 없음)인지 판단
    - 예
      - 전제: AIKU는 고려대 유일 딥러닝 학회이다.
      - 가정: 고려대에 있는 딥러닝 학회의 수는 하나이다.

Model Pretraining: Decoders

- 모델 예시: Generative Pretrained Transformer (GPT)
  - Experiments
    - Natural language inference tasks

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

Model Pretraining: Decoders

- 모델 예시: Generative Pretrained Transformer (GPT)
  - Experiments
    - Question Answering task

Method	Story Cloze	RACE-m	RACE-h	RACE
val-LS-skip [55]	76.5	-	-	-
Hidden Coherence Model [7]	77.6	-	-	-
Dynamic Fusion Net [67] (9x)	-	55.6	49.4	51.2
BiAttention MRU [59] (9x)	-	60.2	50.3	53.3
Finetuned Transformer LM (ours)	**86.5**	**62.9**	**57.4**	**59.0**

Model Pretraining: Decoders

- 모델 예시: Generative Pretrained Transformer (GPT)
  - Experiments
    - Semantic similarity & classification task

Method	Classification		Semantic Similarity			GLUE
	CoLA (mc)	SST2 (acc)	MRPC (F1)	STSB (pc)	QQP (F1)	
Sparse byte mLSTM [16]	-	**93.2**	-	-	-	-
TF-KLD [23]	-	-	**86.0**	-	-	-
ECNU (mixed ensemble) [60]	-	-	-	81.0	-	-
Single-task BiLSTM + ELMo + Attn [64]	35.0	90.2	80.2	55.5	66.1	64.8
Multi-task BiLSTM + ELMo + Attn [64]	18.9	91.6	83.5	72.8	63.3	68.9
Finetuned Transformer LM (ours)	**45.4**	91.3	82.3	**82.0**	**70.3**	**72.8**

Model Pretraining: Decoders

- 모델 예시: Increasingly convincing generations (GPT-2)
  - GPT와 구조 거의 동일
    - Layer normalization 순서 변동만 있을 뿐
  - Unicode가 아닌 Byte 수준의 BPE 사용
  - 모델의 크기가 커짐
  - 학습 데이터셋으로 저자들이 직접 만든 WebText 채택
  - 어떤 explicit supervision 없이 downstream task를 수행하는 것이 목표
    - = zero-shot task transfer

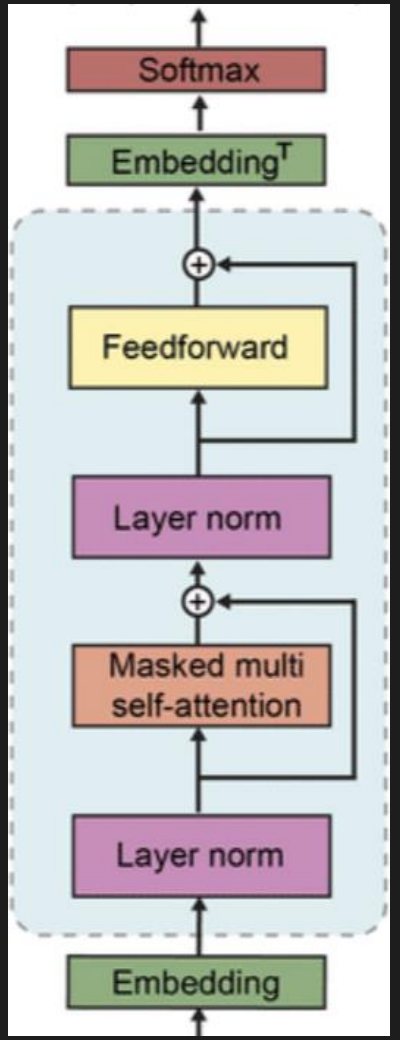


사진 출처: https://supkoon.tistory.com/25

Model Pretraining: Decoders

- 모델 예시: Increasingly convincing generations (GPT-2)

Context (passage and previous question/answer pairs)

The 2008 Summer Olympics torch relay was run from March 24 until August 8, 2008, prior to the 2008 Summer Olympics, with the theme of "one world, one dream". Plans for the relay were announced on April 26, 2007, in Beijing, China. The relay, also called by the organizers as the "Journey of Harmony", lasted 129 days and carried the torch 137,000 km (85,000 mi) – the longest distance of any Olympic torch relay since the tradition was started ahead of the 1936 Summer Olympics.

After being lit at the birthplace of the Olympic Games in Olympia, Greece on March 24, the torch traveled to the Panathinaiko Stadium in Athens, and then to Beijing, arriving on March 31. From Beijing, the torch was following a route passing through six continents. The torch has visited cities along the Silk Road, symbolizing ancient links between China and the rest of the world. The relay also included an ascent with the flame to the top of Mount Everest on the border of Nepal and Tibet, China from the Chinese side, which was closed specially for the event.

Q: What was the theme
A: "one world, one dream".

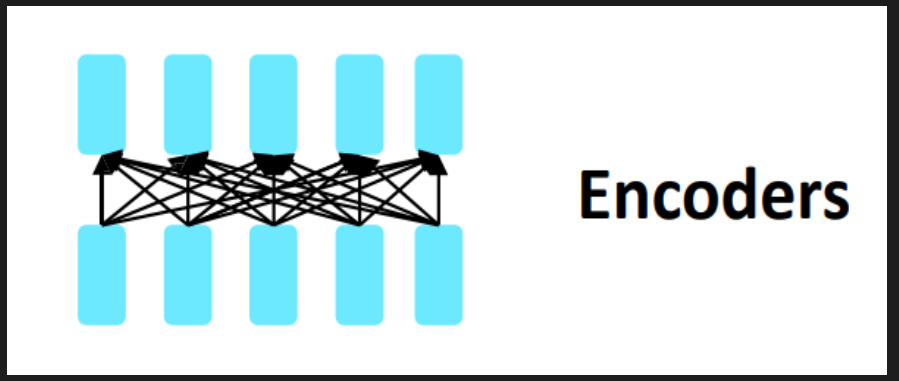
Q: What was the length of the race?
A: 137,000 km

Q: Was it larger than previous ones?
A: No

Q: Where did the race begin?
A: Olympia, Greece

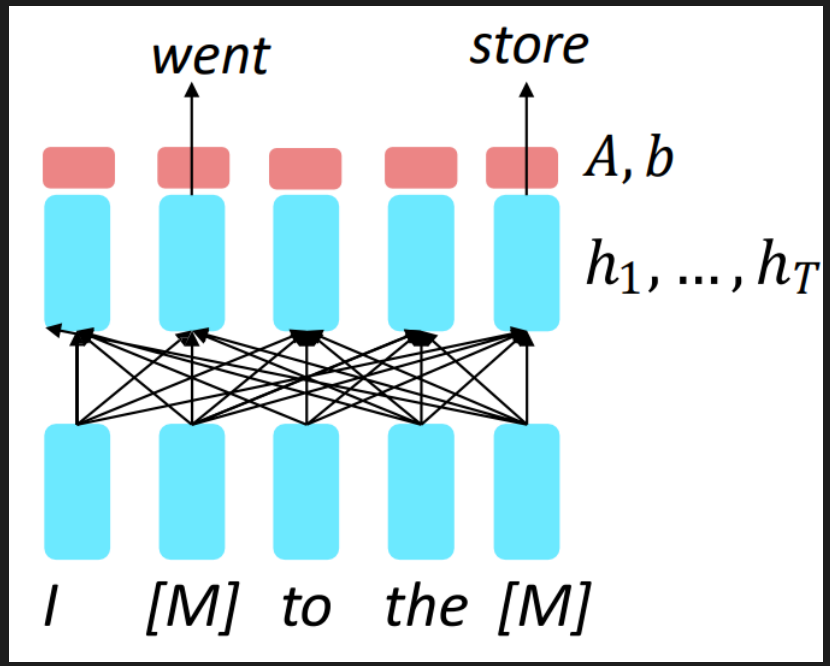
Model Pretraining: Encoders

- Encoder로만 구성된 language models
  - 디코더와는 달리, 양방향(bidirection) context를 얻을 수 있다.
    → 미래 단어를 볼 수 있게 된다.
  - 그러므로 causal language modeling으로 학습 불가
    → 새로운 방법론 필요!



Model Pretraining: Encoders

- A novel task for encoder-only models
  - Token의 일부를 마스킹한 후, ([MASK] token으로 치환) 마스킹된 부분을 예측
    → Masked Language Modeling
  - 해당 방법을 사용해 pretrain한 모델을
    Masked Language Model (MLM) 이라고 함

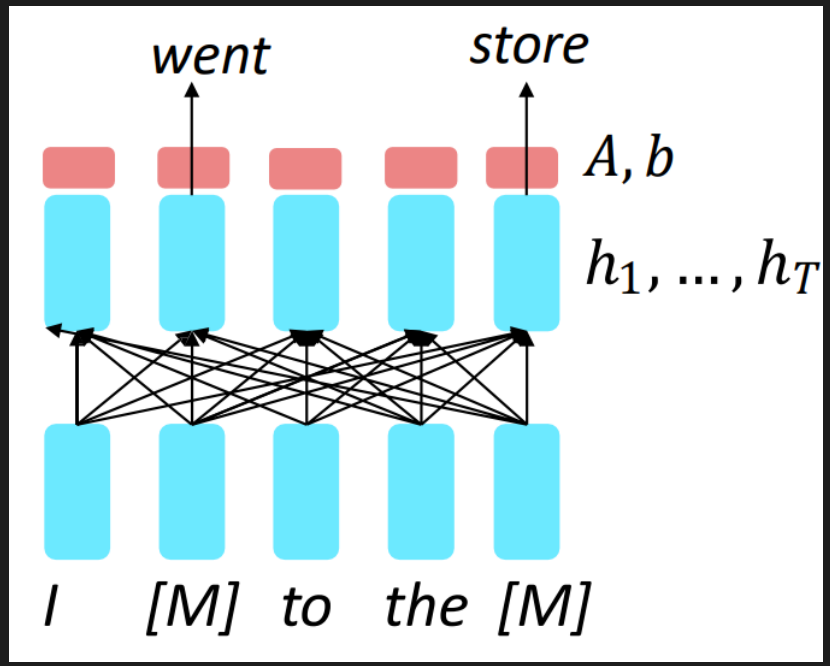


Model Pretraining: Encoders

- Mask Language modeling
  - 식으로 나타내면 다음과 같음

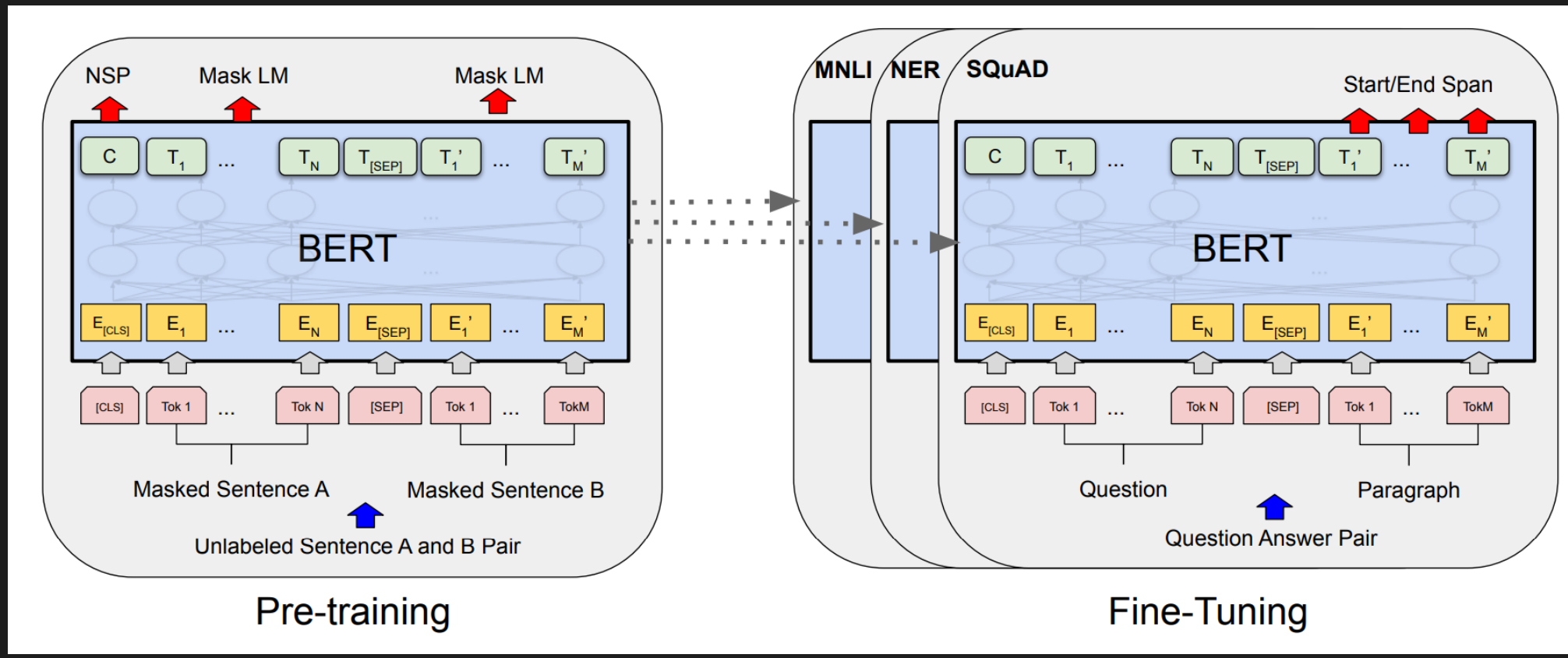
$$h_1, \ldots, h_T = \text{Encoder}(w_1, \ldots, w_T)$$
$$y_i \sim Aw_i + b$$

  - $x: input\ words, \tilde{x}: masked\ version\ of\ x$ 일 때, $p_\theta(x|\tilde{x})$를 최대화하도록 모델 파라미터 학습



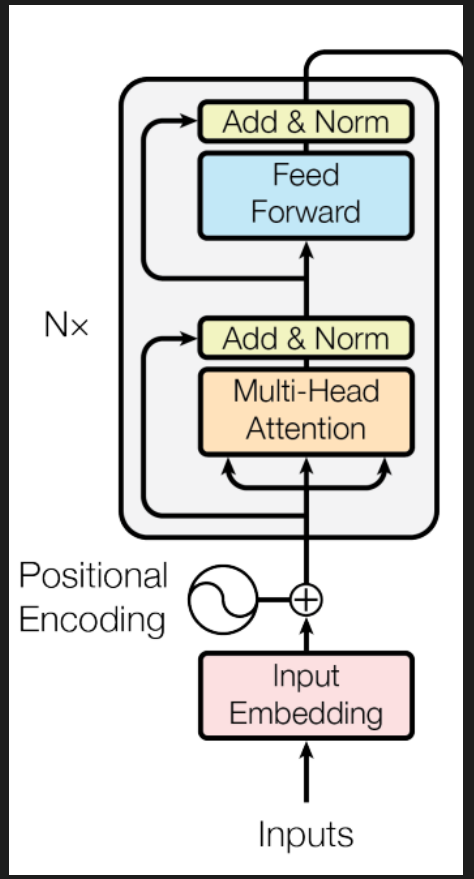
Model Pretraining: Encoders

- 모델 예시: Bidirectional Encoder Representations from Transformers (BERT)



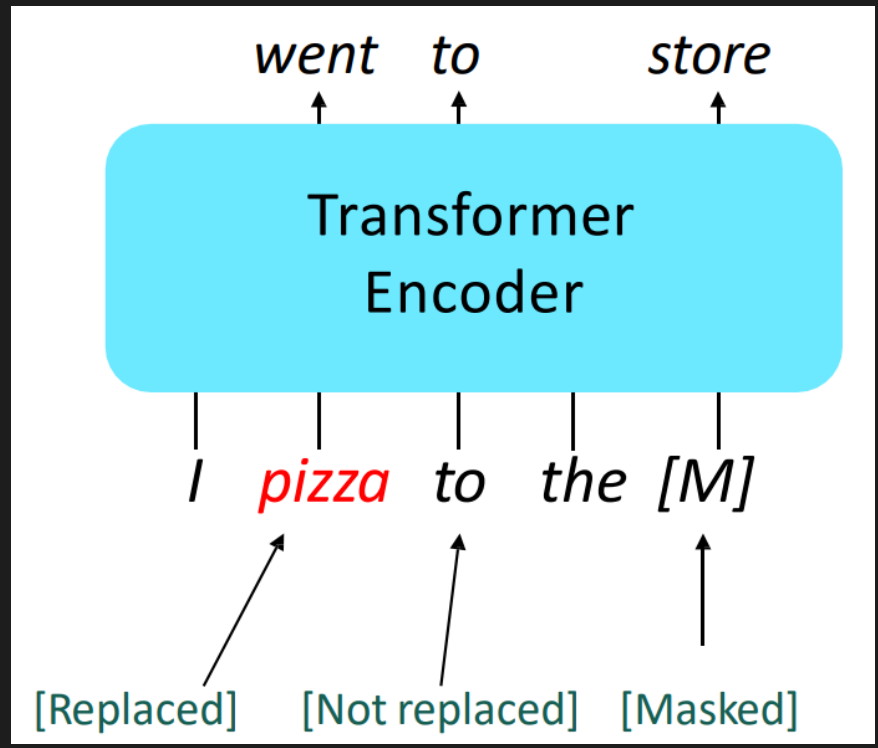
Model Pretraining: Encoders

- 모델 예시: Bidirectional Encoder Representations from Transformers (BERT)
  - Encoder-only models
  - Base: 12 layers, hidden dimensions: 768, Attention head: 12
    - 총 110M개의 파라미터
  - Large : 24 layers, hidden dimensions: 1024, Attention head : 16
    - 총 340M개의 파라미터
  - WordPiece Tokenizer
  - Learnable Position encoding



Model Pretraining: Encoders

- 모델 예시: Bidirectional Encoder Representations from Transformers (BERT)
  - Pre-training for BERT: MLM
    - 랜덤하게 전체 중 15%의 token을 택해서 예측한다.
    - 선택된 token 중
      - 80%는 [MASK] token으로 교체
      - 10%는 엉뚱한 token으로 교체
      - 10%는 그대로 둠
    - 왜 그대로 둔 token도 예측하는가?
      - 해당 작업을 제거할 경우 모델이 non-maked token에 대해 strong representation을 얻지 못한다!
        → pretraining과 finetuning간 불일치 발생
        (fine-tuning time에서는 mask token이 없음)

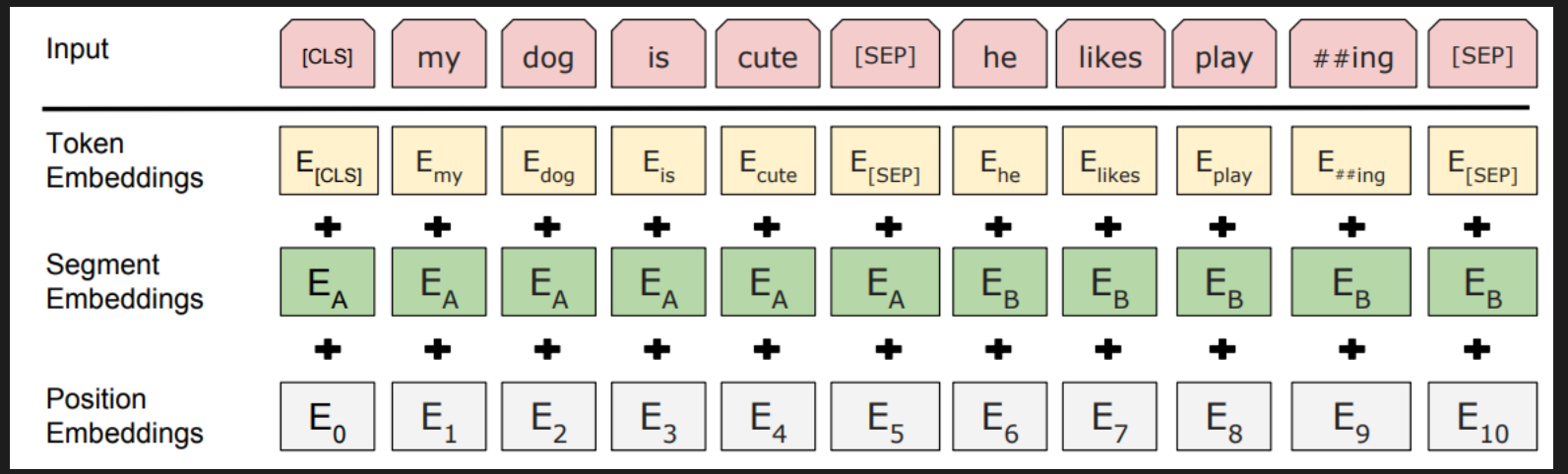


Model Pretraining: Encoders

- 모델 예시: Bidirectional Encoder Representations from Transformers (BERT)
  - Pre-training for BERT: Next Sentence Prediction (NSP)
    - Many important downstream tasks such as Question Answering (QA) and Natural Language Inference (NLI) are based on understanding the relationship between two sentences, which is not directly captured by language modeling
    - 이러한 문제점을 보완하기 위해 NSP를 이용해 pretraining 수행

Model Pretraining: Encoders

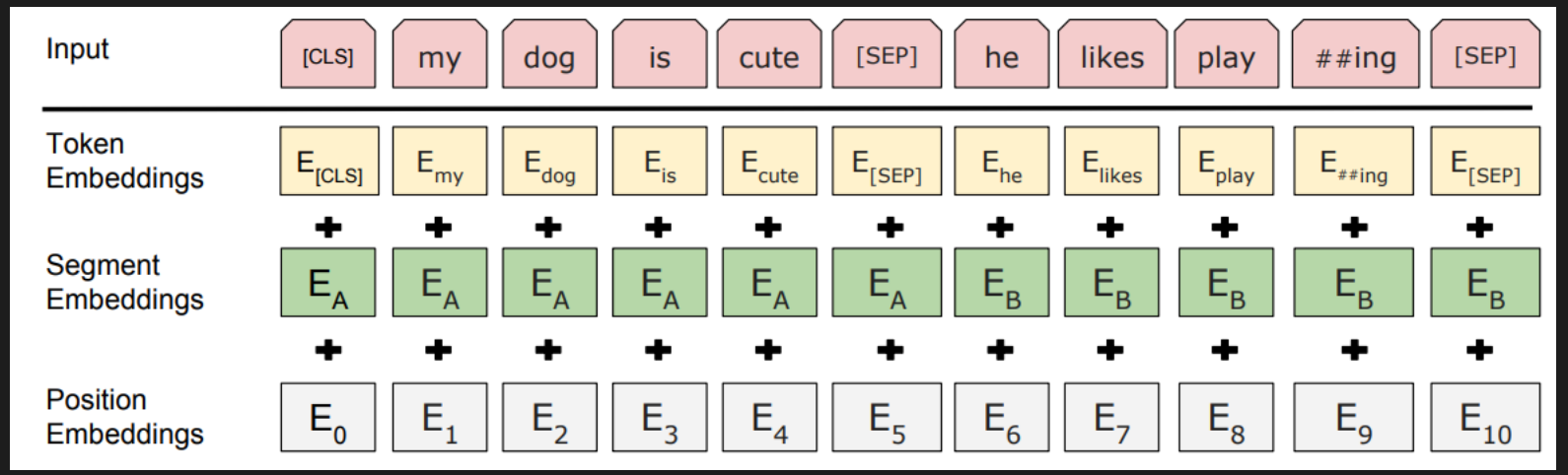
- 모델 예시: Bidirectional Encoder Representations from Transformers (BERT)



  - 두 문장 A와 B를 [SEP] token으로 구분해서 넣기
    - 이 때 50%의 확률로 A와 B는 이어지는 문장 (label: IsNext)
    - 나머지 50%의 확률로 A와 B는 이어지는 문장이 아님 (label: NotNext)
  - [CLS] token에서 나온 representation을 classifier에 태워 prediction

Model Pretraining: Encoders

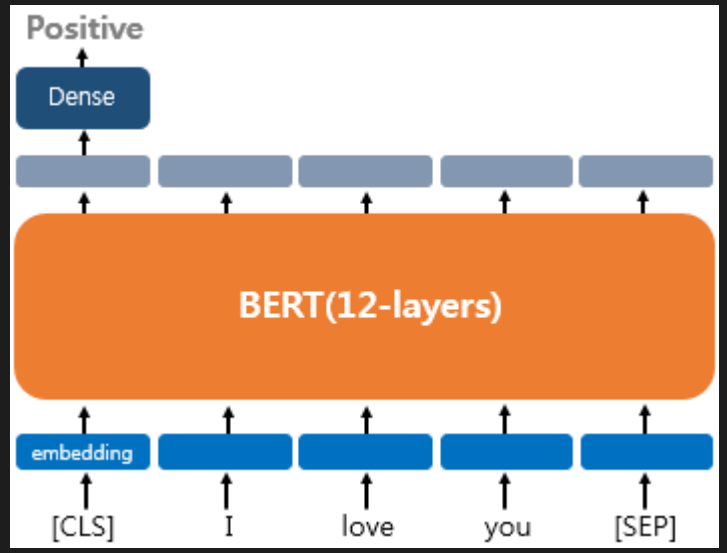
- 모델 예시: Bidirectional Encoder Representations from Transformers (BERT)



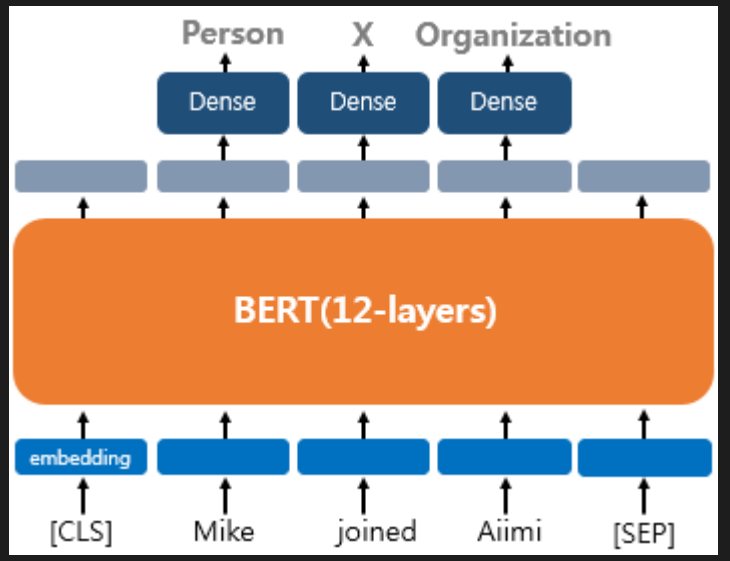
- 이 때 [SEP] token 외에도, segment embedding을 추가해 A와 B가 확실하게 분리되도록 함
- Training할 때, classifier뿐만 아니라 모델 전체 parameter가 다 학습되도록 함 (freeze X)

Model Pretraining: Encoders

- 모델 예시: Bidirectional Encoder Representations from Transformers (BERT)
  - Fine-tuning for BERT



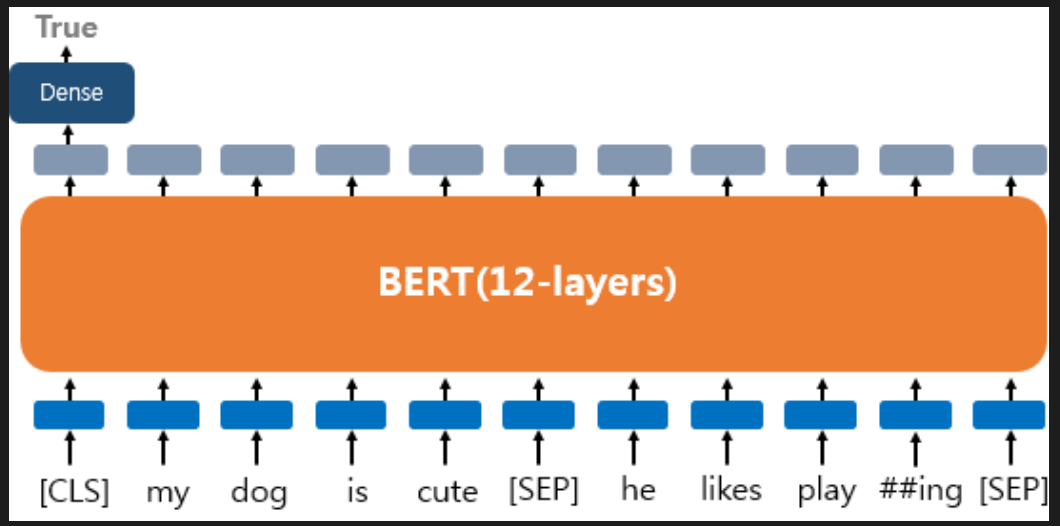
Classification



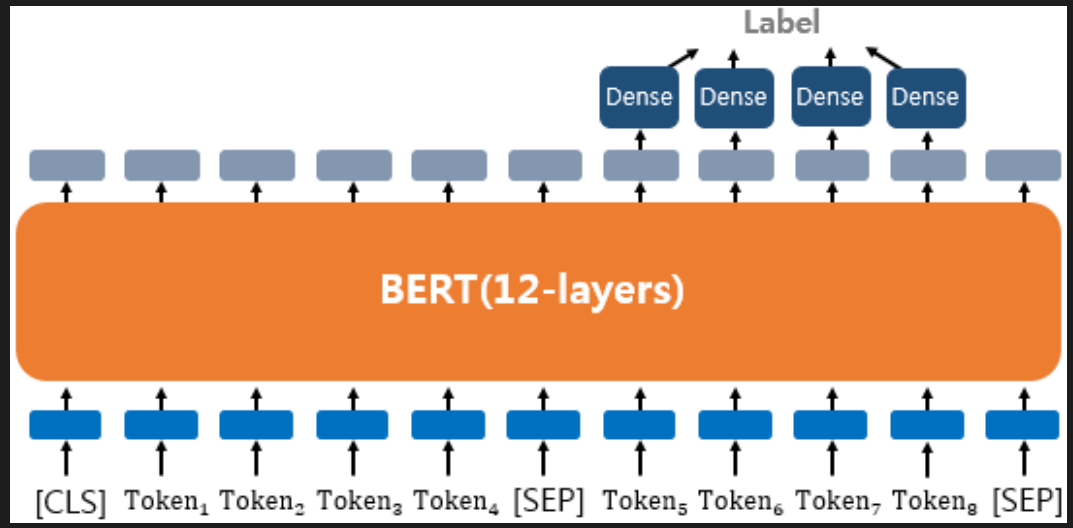
Tagging

Model Pretraining: Encoders

- 모델 예시: Bidirectional Encoder Representations from Transformers (BERT)
  - Fine-tuning for BERT



Text Pair Classification



Question Answering

Model Pretraining: Encoders

- 모델 예시: Bidirectional Encoder Representations from Transformers (BERT)
  - Experiment
    - QQP: Quora Question Pairs (detect paraphrase questions)
    - QNLI: natural language inference over question answering data
    - SST-2: sentiment analysis
    - CoLA: corpus of linguistic acceptability (detect whether sentences are grammatical.)
    - STS-B: semantic textual similarity
    - MRPC: microsoft paraphrase corpus
    - RTE: a small natural language inference corpus

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
$\text{BERT}_{\text{BASE}}$	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
$\text{BERT}_{\text{LARGE}}$	**86.7/85.9**	**72.1**	**92.7**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**82.1**

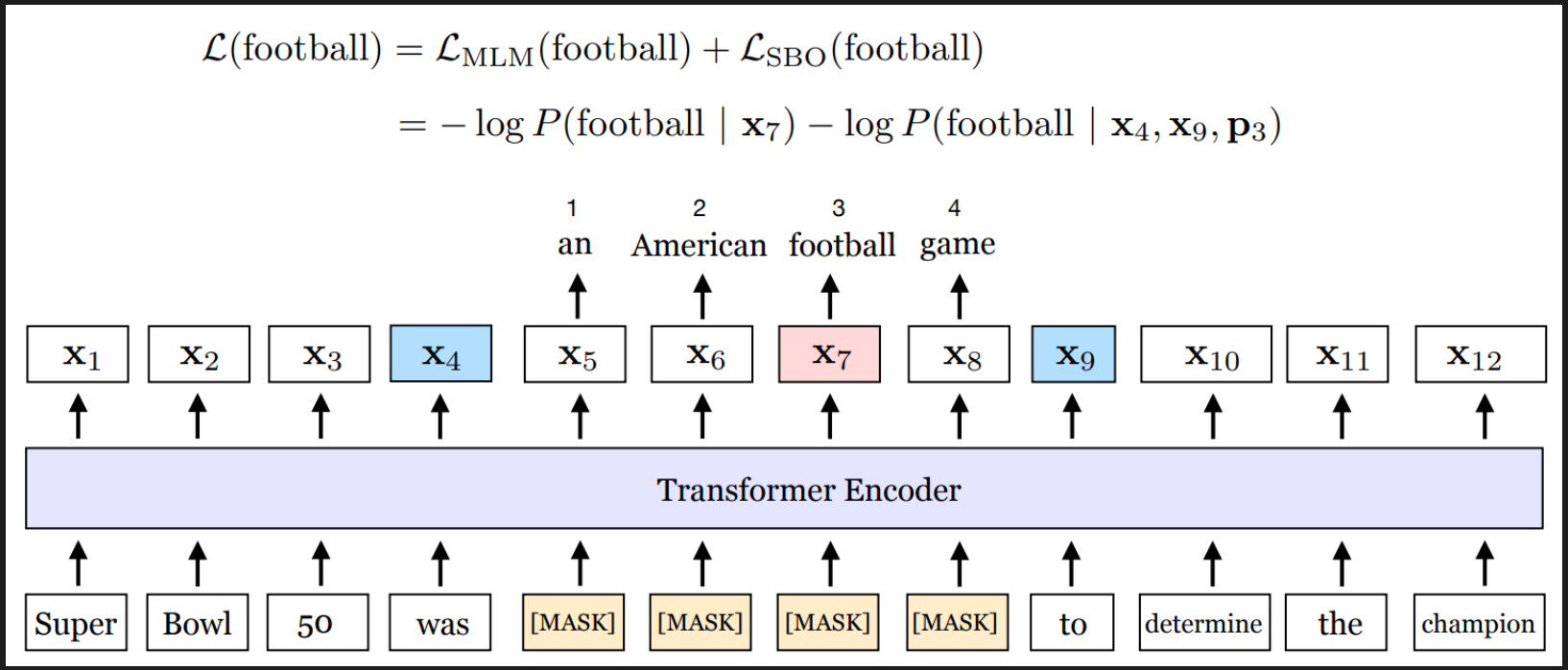
Model Pretraining: Encoders

- Extensions of BERT
  - BERT의 variants 등장
    - RoBERTa, SpanBERT 등
  - RoBERTa
    - “BERT는 undertrained되었다”
    - Pretrain 중 NSP 과정 제거
    - 다양한 train 방법론을 통해 BERT의 성능 향상

Model Pretraining: Encoders

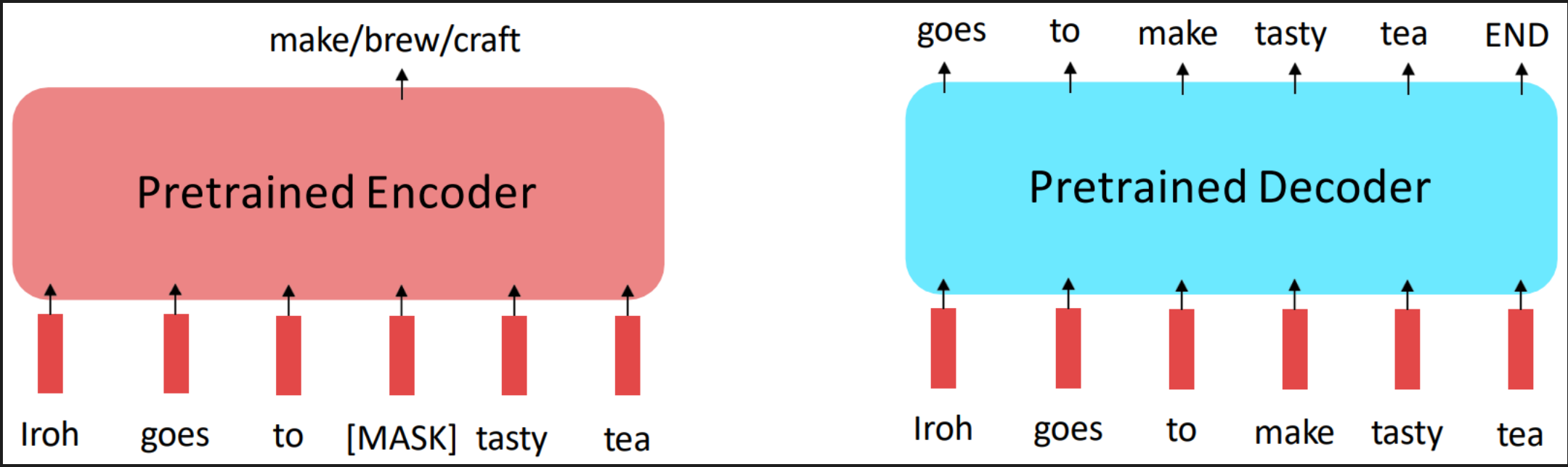
- Extensions of BERT
  - SpanBERT
    - Random token이 아닌 continuous한 random span을 예측하게 함으로써 성능 향상 도모

$$\mathcal{L}(\text{football}) = \mathcal{L}_{\text{MLM}}(\text{football}) + \mathcal{L}_{\text{SBO}}(\text{football})$$
$$= -\log P(\text{football} \mid \mathbf{x}_7) - \log P(\text{football} \mid \mathbf{x}_4, \mathbf{x}_9, \mathbf{p}_3)$$



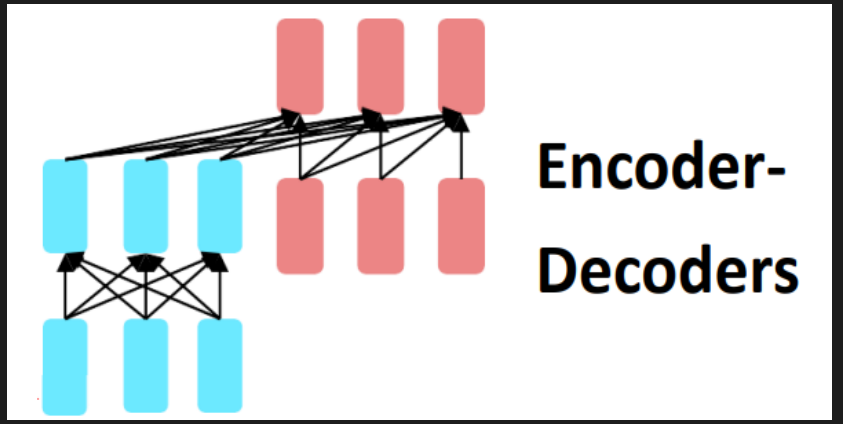
Model Pretraining: Encoders

- Encoder 모델의 한계
  - 양방향에서 문맥을 살필 수 있는 장점이 있긴 하지만, Encoder는 generation에 특화된 모델이 X
  - BERT같은 encoder-only model은 autoregressive generation에서 좋은 성능을 보이지X



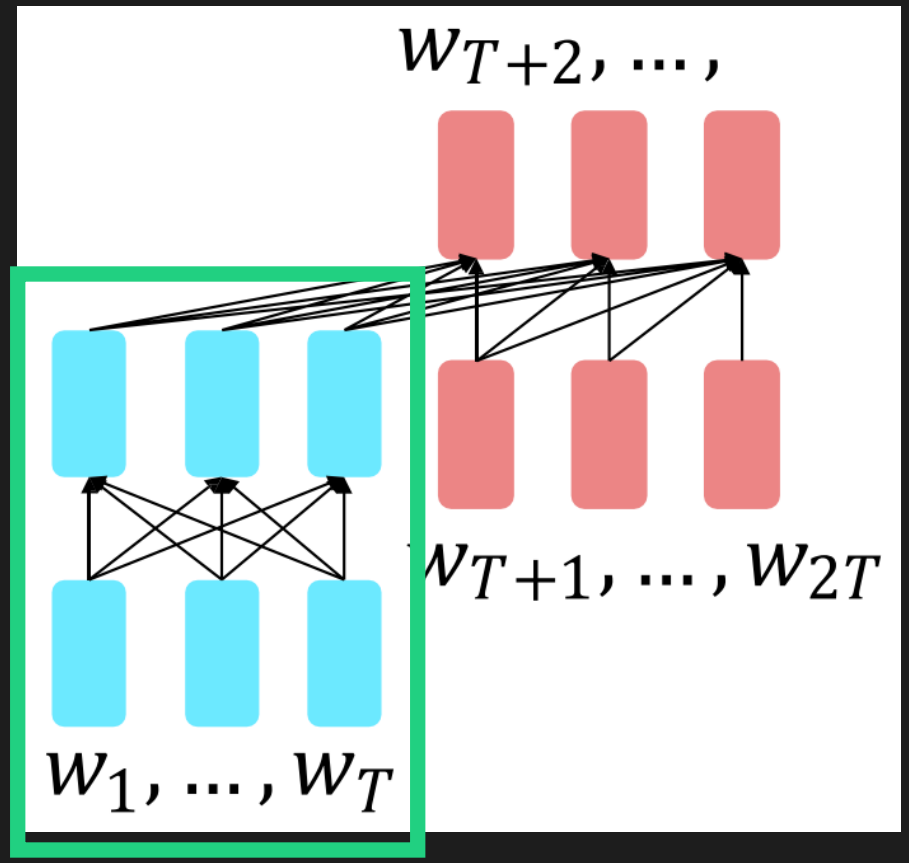
Model Pretraining: Encoder-Decoders

- Encoder와 Decoder의 장점만 취할 순 없을까?
  - Encoder와 Decoder를 붙이자
- 하지만 Encoder-decoder model은 어떻게 pretrain하지?



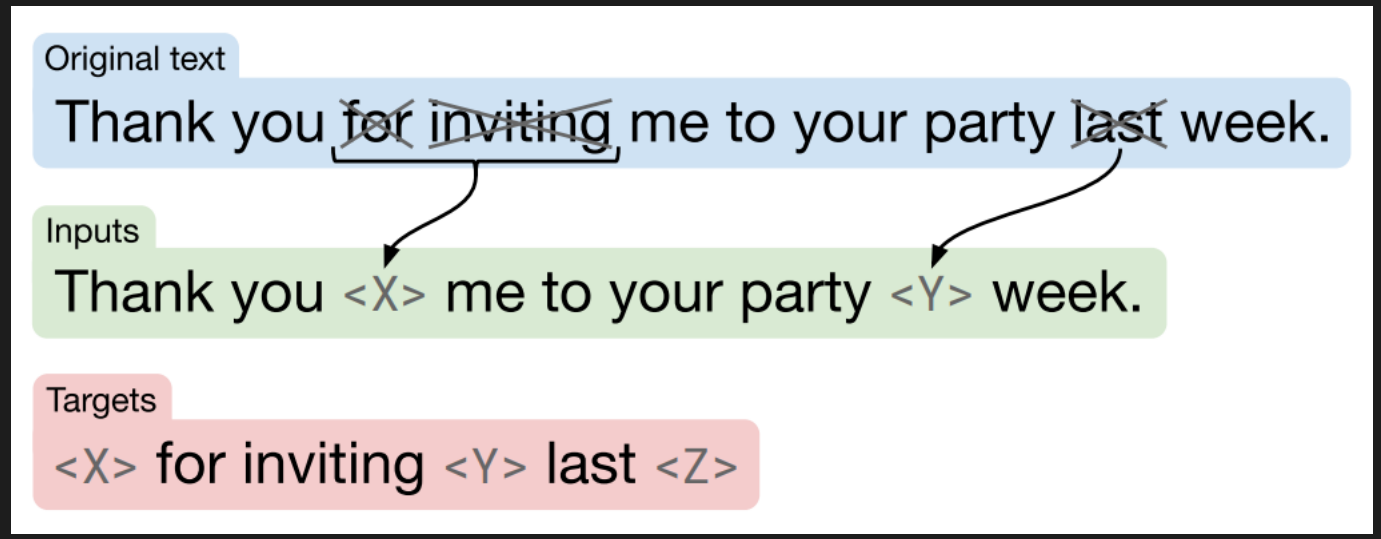
Model Pretraining: Encoder-Decoders

- Language modeling을 하면, encoder의 input은 prediction이 불가



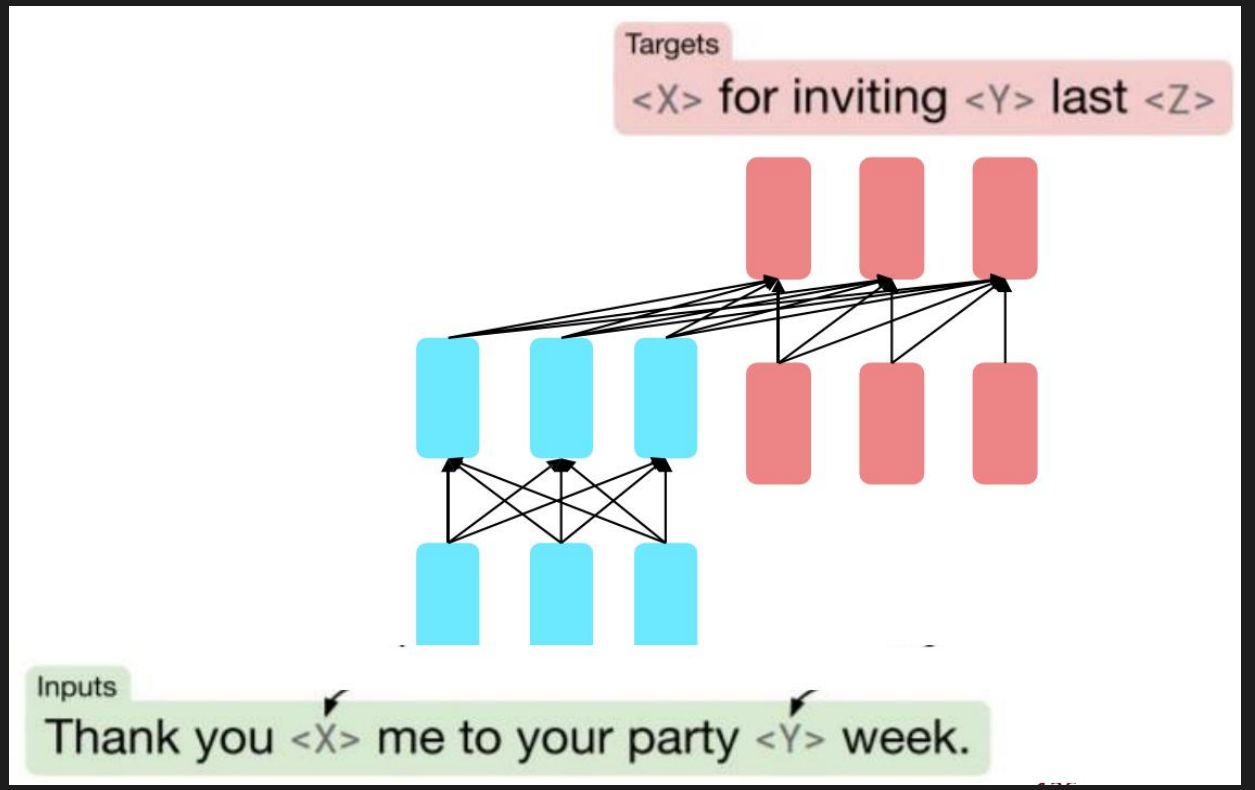
Model Pretraining: Encoder-Decoders

- 그럼 어떻게 해야하는가?
- T5 model에서는 Denoising span corruption task를 제시



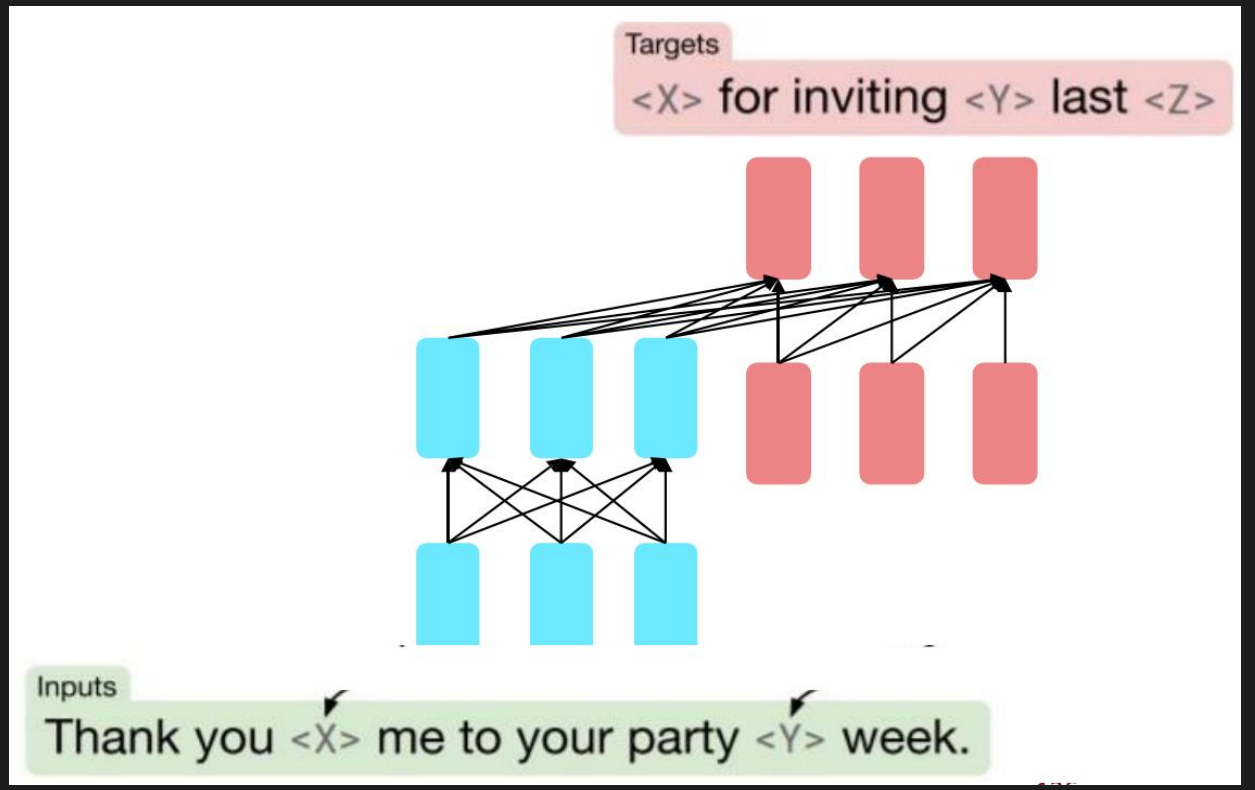
Model Pretraining: Encoder-Decoders

- 그럼 어떻게 해야하는가?
- T5 model에서는 Denoising span corruption task를 제시



Model Pretraining: Encoder-Decoders

- 그럼 어떻게 해야하는가?
- T5 model에서는 Denoising span corruption task를 제시



Model Pretraining: Encoder-Decoders

- T5 논문의 저자들은 해당 task가 pretrain 할 때 가치가 있음을 실험을 통해 증명!

Objective	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
Prefix language modeling	80.69	18.94	77.99	65.27	**26.86**	39.73	**27.49**
BERT-style (Devlin et al., 2018)	**82.96**	**19.17**	**80.65**	**69.85**	**26.78**	**40.03**	**27.41**
Deshuffling	73.17	18.59	67.61	58.47	26.11	39.30	25.62

Objective	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
BERT-style (Devlin et al., 2018)	82.96	19.17	**80.65**	69.85	26.78	**40.03**	27.41
MASS-style (Song et al., 2019)	82.32	19.16	80.10	69.28	26.79	**39.89**	27.55
★ Replace corrupted spans	83.28	**19.24**	**80.88**	**71.36**	**26.98**	39.82	**27.65**
Drop corrupted tokens	**84.44**	**19.31**	**80.52**	68.67	**27.07**	39.76	**27.82**

Table 5: Comparison of variants of the BERT-style pre-training objective. In the first two variants, the model is trained to reconstruct the original uncorrupted text segment. In the latter two, the model only predicts the sequence of corrupted tokens.

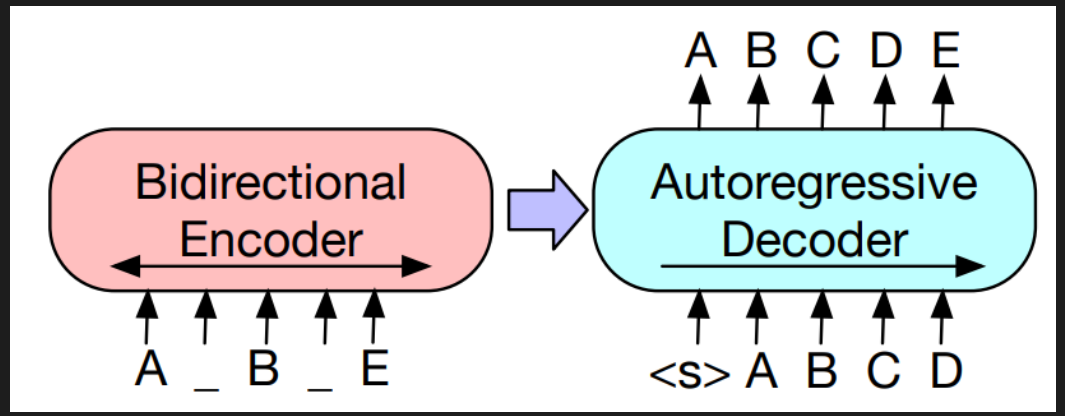
Model Pretraining: Encoder-Decoders

- T5 논문의 저자들은 해당 task가 pretrain 할 때 가치가 있음을 실험을 통해 증명!

Architecture	Objective	Params	Cost	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ Encoder-decoder	Denoising	$2P$	M	**83.28**	**19.24**	**80.88**	**71.36**	**26.98**	**39.82**	**27.65**
Enc-dec, shared	Denoising	P	M	82.81	18.78	**80.63**	**70.73**	26.72	39.03	**27.46**
Enc-dec, 6 layers	Denoising	P	$M/2$	80.88	18.97	77.59	68.42	26.38	38.40	26.95
Language model	Denoising	P	M	74.70	17.93	61.14	55.02	25.09	35.28	25.86
Prefix LM	Denoising	P	M	81.82	18.61	78.94	68.11	26.43	37.98	27.39
Encoder-decoder	LM	$2P$	M	79.56	18.59	76.02	64.29	26.27	39.17	26.86
Enc-dec, shared	LM	P	M	79.60	18.13	76.35	63.50	26.62	39.17	27.05
Enc-dec, 6 layers	LM	P	$M/2$	78.67	18.26	75.32	64.06	26.13	38.42	26.89
Language model	LM	P	M	73.78	17.54	53.81	56.51	25.23	34.31	25.38
Prefix LM	LM	P	M	79.68	17.84	76.87	64.86	26.28	37.51	26.76

Model Pretraining: Encoder-Decoders

- 모델 예시: BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension



  - Inputs : arbitrary noise transformations
    - replacing spans of text with mask symbols
  - The corrupted document : encoded with a bidirectional model
  - Predict the original document in an autoregressive decoder

Model Pretraining: Encoder-Decoders

- 모델 예시: BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension
  - For fine-tuning: an uncorrupted document is input to both the encoder and decoder
    - use representations from the final hidden state of the decoder
  - Base: x6 layers in each encoder and decoder
  - Large: x12 layers in each encoder and decoder
  - T5 이전 모델

Model Pretraining: Encoder-Decoders

- 모델 예시: BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension
  - Training by corrupting documents

    → optimizing a reconstruction loss (cross-entropy)

$$Loss(ours) = -\sum\sum p(x)log(q(x))$$

  - 기존 denoising autoencoders: 특정 noising schemes에만 특화

    BART allows us to apply any type of document corruption

Model Pretraining: Encoder-Decoders

- 모델 예시: BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension
  - Training by corrupting documents

    → optimizing a reconstruction loss (cross-entropy)

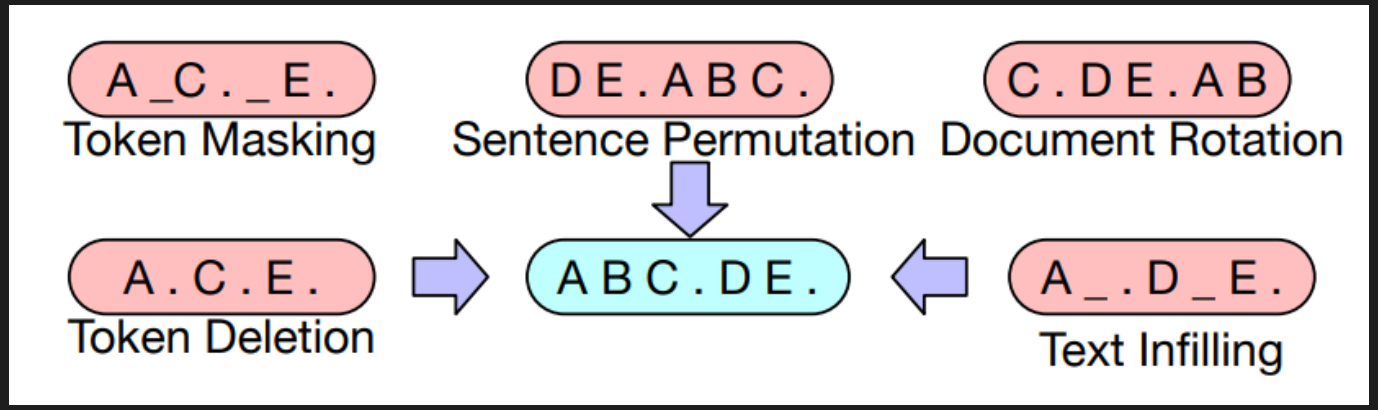
$$Loss(ours) = -\sum\sum p(x)log(q(x))$$

  - 기존 denoising autoencoders: 특정 noising schemes에만 특화

    BART allows us to apply any type of document corruption

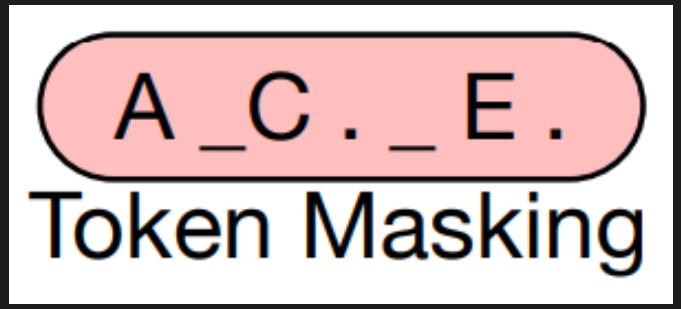
Model Pretraining: Encoder-Decoders

- 모델 예시: BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension
  - 5개의 pretraining task로 구성

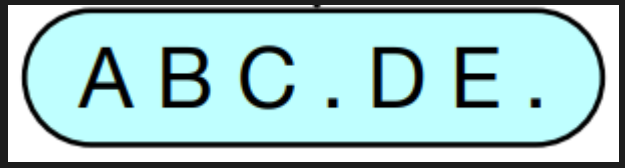


Model Pretraining: Encoder-Decoders

- 모델 예시: BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension
  - Token Masking
    - Same as BERT
    - Random tokens are sampled and replaced with [MASK] elem.

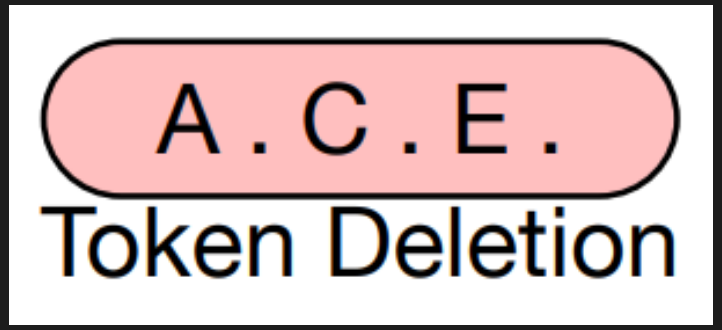


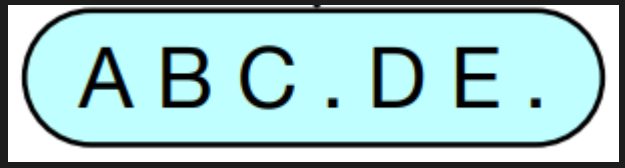
여기부터는 SKIP.
관심 있으신 분들은 자세히 읽어보세요.



Model Pretraining: Encoder-Decoders

- 모델 예시: BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension
  - Token Deletion
    - Random tokens are deleted from the input
    - Token masking과 다르게 model은 어느 위치의 token이 유실되었는지 알아야 함

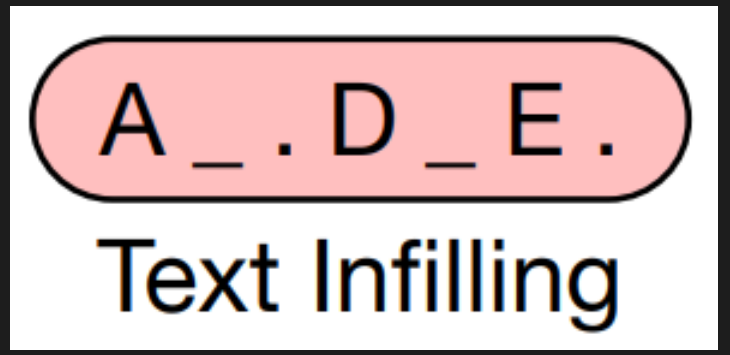


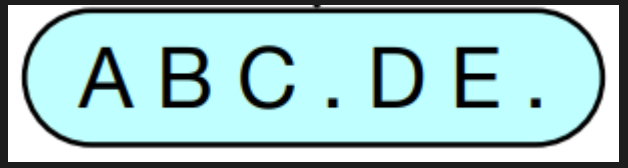


Model Pretraining: Encoder-Decoders

- 모델 예시: BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension
  - Text Infilling
    - Sampling “Span” (length drawn from Poisson distribution)
    - Each span is replaced with a single [MASK] token (0-len span도 포함)

    (SpanBERT: span 길이와 동일한 [MASK] token으로 대체)

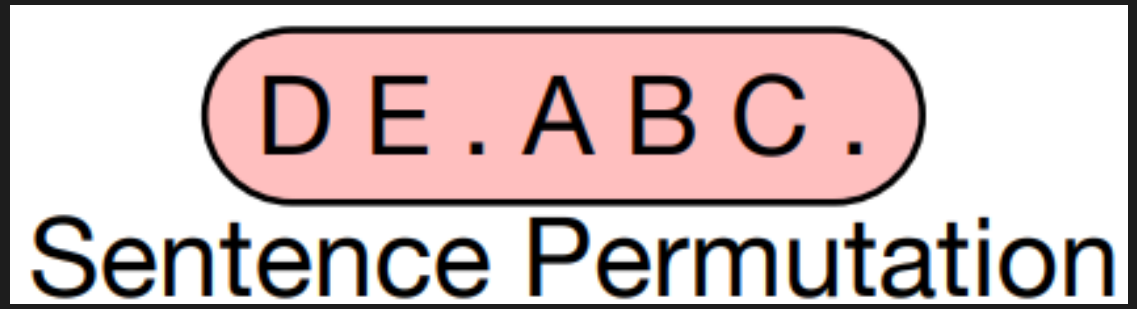


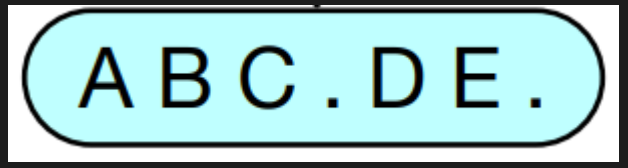


Model Pretraining: Encoder-Decoders

- 모델 예시: BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension
  - Sentence Permutation
    - 마침표 기준, documents를 sentence로 분리

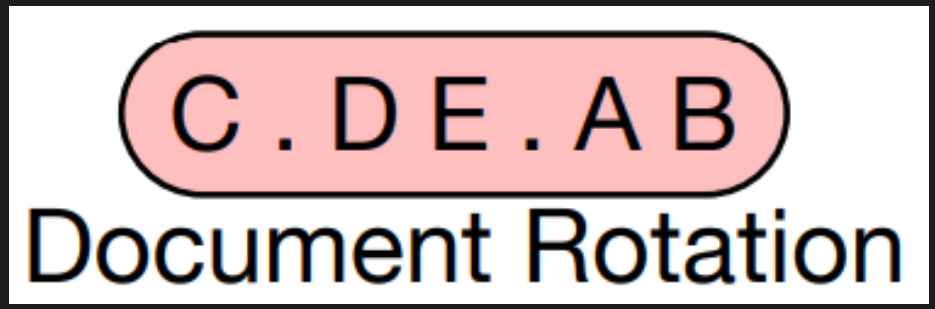
    → shuffled in a random order

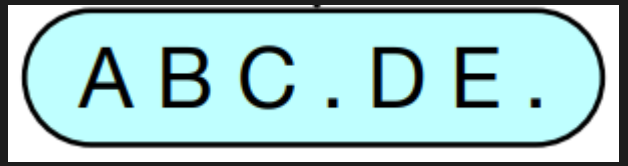




Model Pretraining: Encoder-Decoders

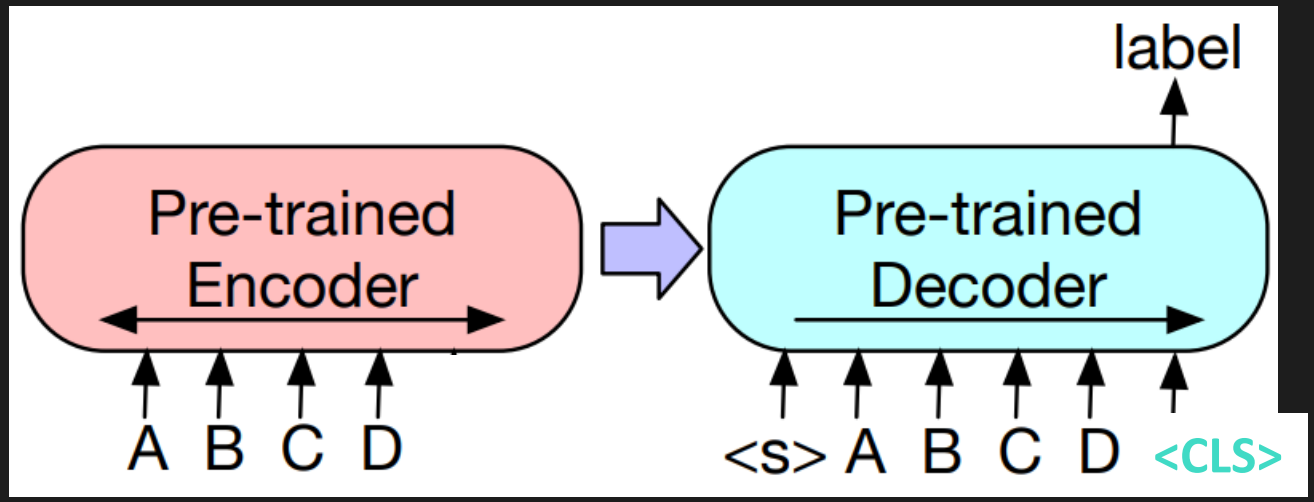
- 모델 예시: BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension
  - Sentence Document Rotation
    - A token is chosen uniformly at random
    - The document is rotated so that it begins with that token





Model Pretraining: Encoder-Decoders

- 모델 예시: BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension
  - Fine-tuning: Sequence Classification Tasks
    - The same input is fed into the encoder & decoder
    - Final hidden state of the final decoder token: fed into new multi-class linear classifier
    - BERT와는 다르게 CLS token을 끝에 붙임

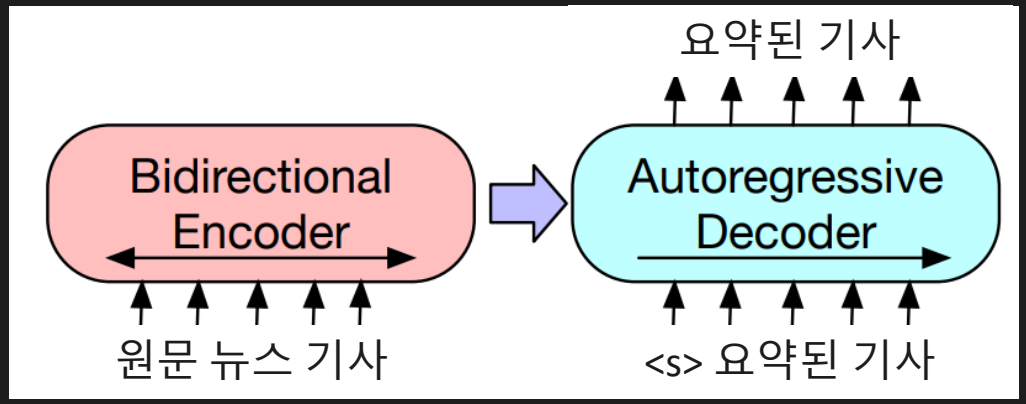


Model Pretraining: Encoder-Decoders

- 모델 예시: BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension
  - Fine-tuning: Sequence Generation Tasks
  - BART has an autoregressive decoder

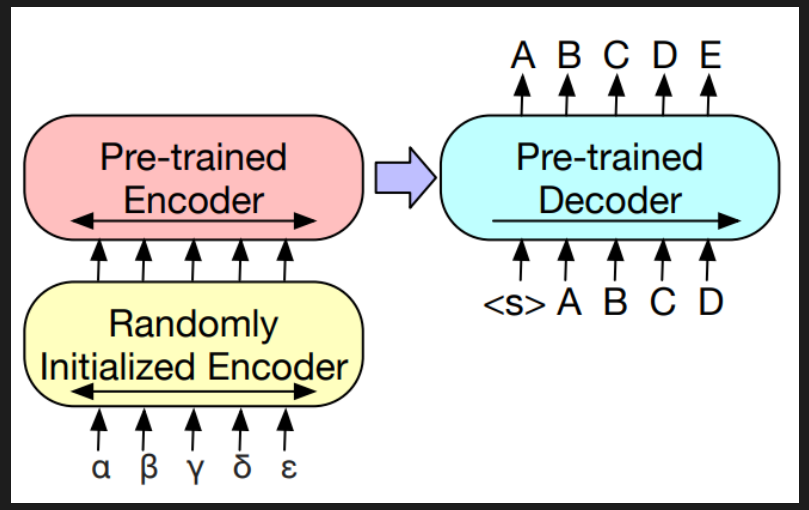
    → can be directly fine tuned for this task
  - Information is copied from the input

    → changed to the form closely related to the denoising pre-training objective



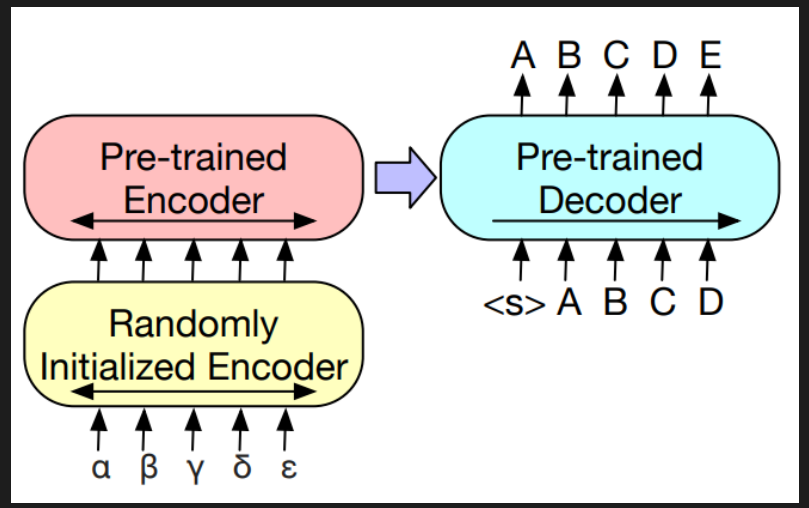
Model Pretraining: Encoder-Decoders

- 모델 예시: BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension
  - Fine-tuning: Machine Translation
    - 이전 연구 → decoder에서 pre-trained language model을 쓰는 것은 제한적
    - BART에서는 이 문제를 해결 (possible to use the entire BART model)
    - How? By adding a new set of encoder parameters (learned from bitext)



Model Pretraining: Encoder-Decoders

- 모델 예시: BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension
  - Fine-tuning: Machine Translation
    - Replace BART's encoder embedding layer with a new randomly initialized encoder.
    - Trains the new encoder to map **source words** into an input that BART can de-noise to **target words**.



Summary

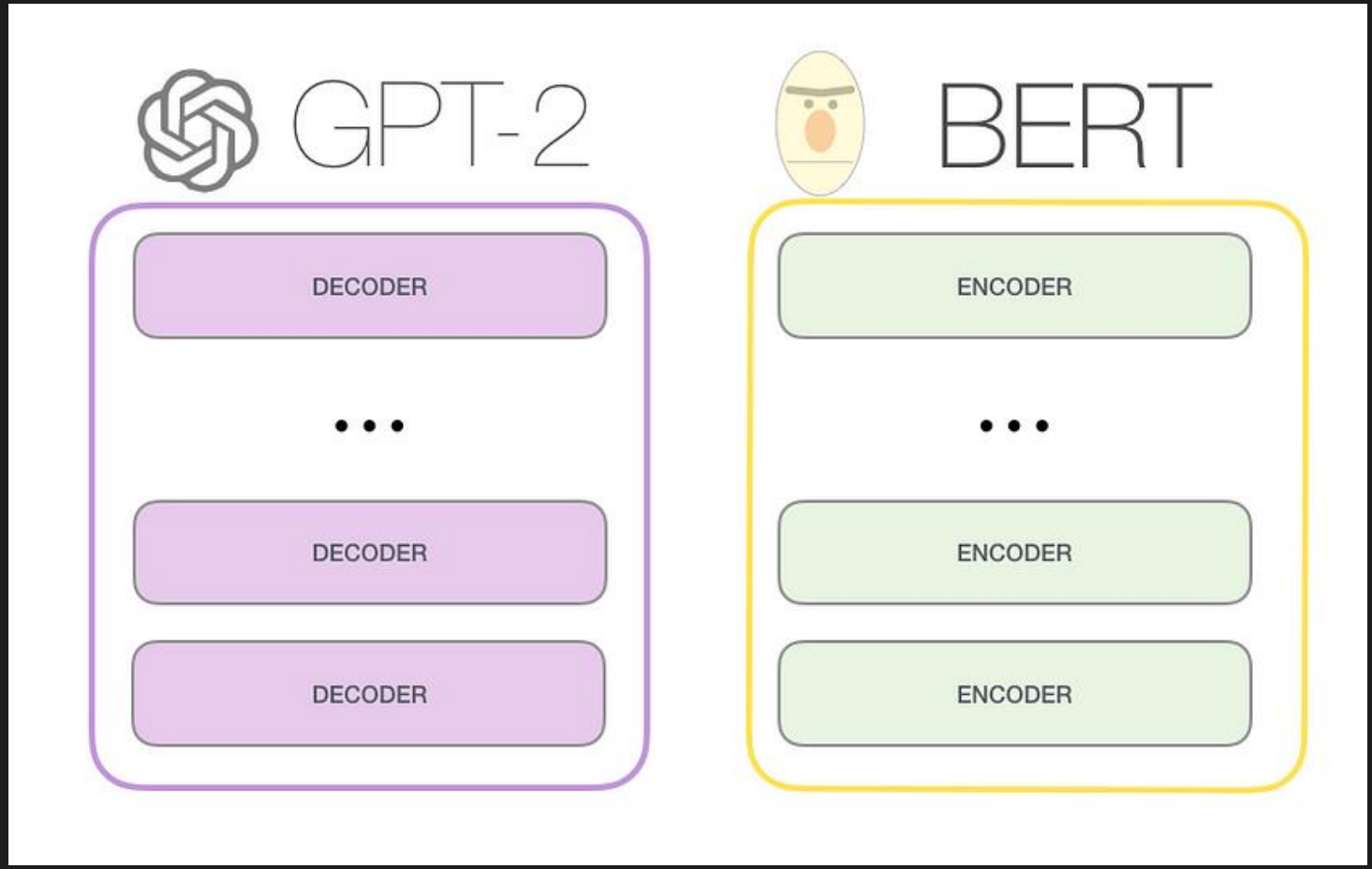
Summary

- 저희는 오늘
  - Tokenizer가 무엇이고, 어떤 역할을 하는지
  - 그리고 그 중 BPE 알고리즘에 대해 배웠습니다.

  - Language model에서 Pretraining-finetuning 패러다임을 살펴보고
  - 왜 pretrain이 필요한지 알아보았습니다.

  - Language model의 대표적인 세 가지 구조 (Decoders, Encoders, Encoder-decoders)를 알아보고
  - 각각의 장단점과
  - 각 구조를 대표하는 모델 예시를 살펴보았습니다.

Summary


GPT-2
DECODER
DECODER
DECODER
BERT
ENCODER
ENCODER
ENCODER

Discussion (~ 이번 주 목요일 수업 전까지)

- Q1. 데이터셋의 단어 빈도를 체크해봤더니 다음과 같은 결과가 나왔다고 합시다.
  - {‘사랑합니다’: 2, ‘애들이’: 1, ‘재미있다’: 3, ‘재미없다’: 4, ‘사랑해’: 3, ‘좋아합니다’: 1}
  - 이 데이터를 바탕으로 (1)BPE를 수행해보고, (2)기존 단어(어절)와 수행된 subword 중 어느 것이 더 나은지 짧게 의견을 내 주세요!
    - 힌트: 한국어의 음절을 영어의 알파벳 문자와 대응시켜 수행하면 됩니다. 자세한 사항은 추가 자료의 ‘한국어 BPE’를 참고해주세요.

- Q2. Finetuning을 할 때는 backbone model을 freeze하고 head만 학습시킬 때도 있고, 전체 모델을 학습시킬 때도 있습니다. 두 방법의 장단점은 무엇일까요?

프로그래밍 과제 (~8월 20일까지)

- 아직 프로그래밍 과제는 제작 중입니다!
- 이번 시간에 배운 model로 다음 수업 시간에 배울 task를 수행하는 과제가 될 예정입니다.
- 자세한 공지는 다음 수업 시간에 하겠습니다 ☺

참고자료

자료 출처

- Standford Univ. cs224n
- 위키독스 딥 러닝을 이용한 자연어처리 입문
- 위키독스 🤗Transformers (신경망 언어모델 라이브러리) 강좌
- 고려대 수학과 오승상 교수님 강의자료
- 고려대 컴퓨터학과 이병준 교수님 강의자료
- 논문 BPE, Attention is all you need, GPT, GPT-2, BERT, T5, BART

추가 자료

- 한국어 BPE
- SentencePiece

감사합니다.