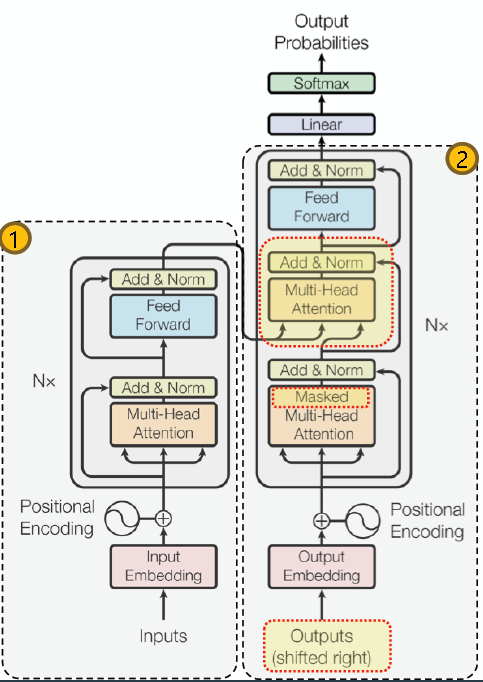
# **BERT**

* By Google (2018.10 논문이 공개, 11월에 오픈 소스로 공개 됨)
* BERT는 **B**idirectional **E**ncoder **R**epresentations from **T**ransformers
* “Attention is all you need” (2017년 논문)에서 소개한 Transformer의 Encoder 부분 사용
* Github : <https://github.com/google-research/bert>
* Paper : <https://arxiv.org/abs/1810.04805>



**[Transformer Architecture]**

1. Transformer Encoder
2. Transformer Decoder

Bert에서는 Transformer의 Encoder Block만을 사용

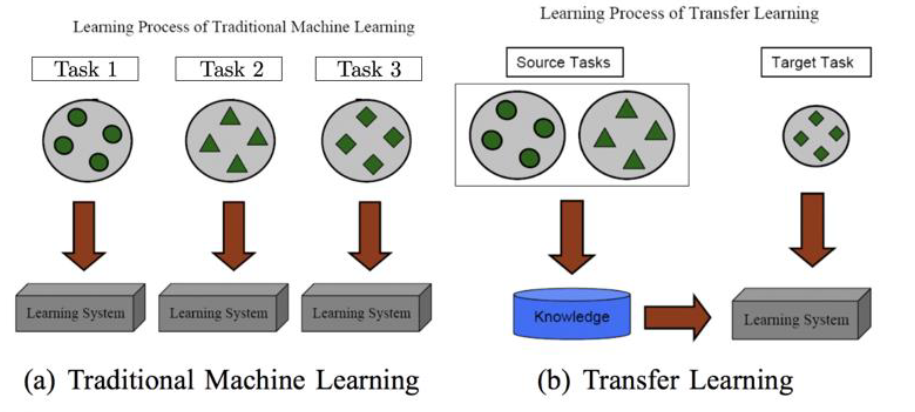
Bert base 모델은 Encoder Block을 12개, Bert large 모델은 24개 쌓아서 만듬.

(Transformer의 Decoder Block을 쓰는 모델은 GPT 계열 모델, Encoder / Decoder Block을 모두 쓰는 모델은 T5 등이 있다)

## **주요 개념**

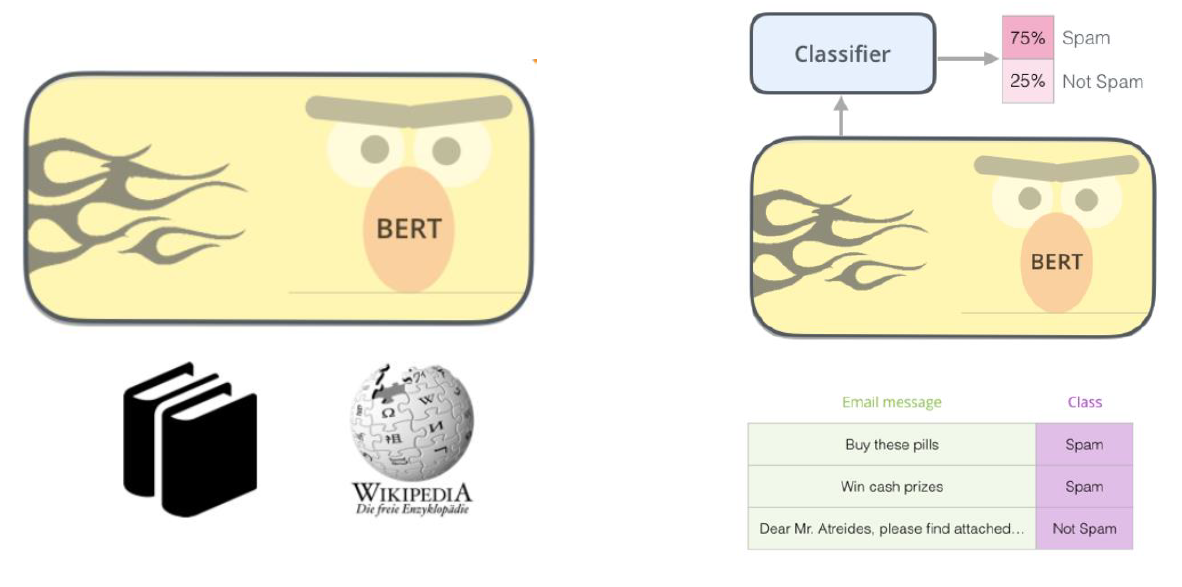
Bert의 자세한 내용을 살펴보기 전에, Bert를 이해하기 위한 주요 개념들에 대해 소개한다.

### **Transfer Learning**



**패러다임의 전환!**

* BERT는 기본적으로, wiki나 book data와 같은 대용량 unlabeled data로 모델을 미리 학습 시킨 후, 특정 task를 가지고 있는 labeled data로 transfer learning을 하는 모델이다. 이때 unlabeled data로 학습하는 과정을 pre-training (upstream task), labeled data로 학습하는 과정을 fine-tuning (downstream task)라고 지칭한다.

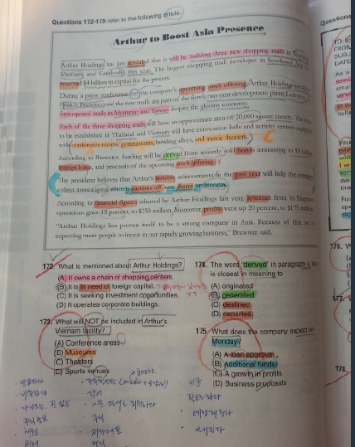
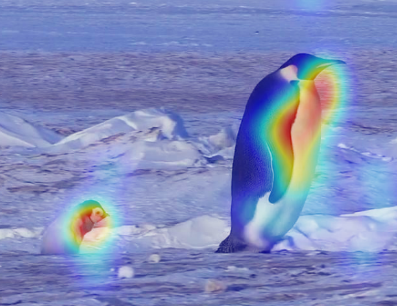


Bert의 Pre-training / Fine-tuning

Bert는 wiki, book corpus 데이터로 pre-training을 수행하여 General purpose의 언어모델 (Language Model)을 학습하고, 다양한 task에 따라 (Glue, Super Glue 등) fine-tuning 모델을 생성하는 방식이다. 이로써 적은 labeling data로도 SOTA 모델을 만들 수 있다.

Language Model은 어떤 방식으로 문맥을 학습할까? (Transformer 계열 모델)

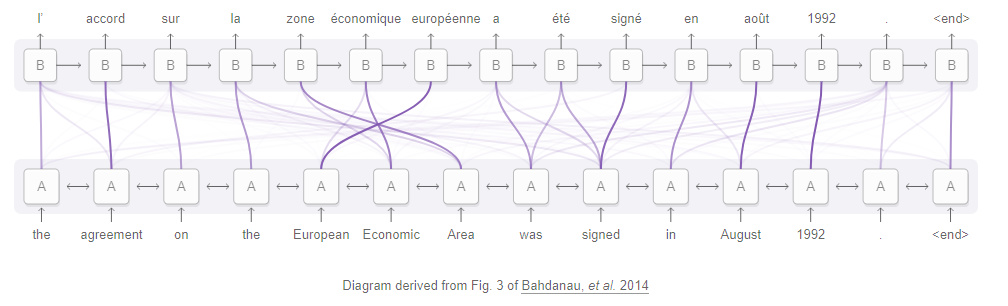
### **Attention**

Attention이란?

인간은 정보처리를 할 때 모든 sequence를 고려하지 않음. 찾아야 하는 주요 정보가 무엇인지 알 수 있다면 이에 가중치를 주어, 주요 정보를 중심으로 맥락을 파악할 수 있음. Attention의 모티브는 자신에게 의미 있는 feature만 골라내서 중요하게 판단하겠다는 것임.

* 논문 : Neural Machine Translation by Jointly Learning to Align and Translate (<https://arxiv.org/abs/1409.0473>), 조경현 교수님
* 어텐션 매커니즘은 기계번역(machine translation)을 위한 sequence-to-sequence 모델(S2S)에 처음 도입
* 번역 시 소스랭귀지와 타겟랭귀지의 길이가 길어질 수록 모델의 성능이 나빠짐, 이를 방지하기 위해 모델로 하여금 ‘중요한 부분만 집중(attention)하게 만들자’는 것이 어텐션 매커니즘의 핵심 아이디어
* 어텐션은 weights의 중요도 벡터, 이미지의 픽셀값이나 문장에서 단어 등 어떤 요소를 예측하거나 추정하기 위해, 다른 요소들과 얼마나 강하게 연관되어 있는지 확인하고 어텐션 백터로 가중 합산된 값의 합계를 타겟값으로 추정

Input 중에서 무엇이 중요한지는 attention weight로 학습이 가능하며, input과 output이 어떻게 연결되어 있는지 시각화가 가능함. (어떤 feature들이 연결되어 있는지 알 수 있음)



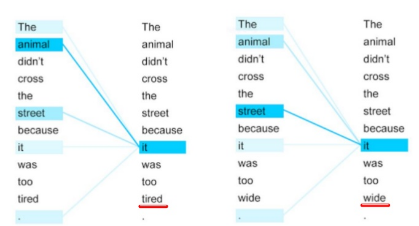
**기존 Attention 방식의 단점**

* 번역모델(Seq2Seq) Attention 메커니즘의 핵심은 decoder의 특정 time-step의 output이 encoder의 모든 time-step의 output 중 어떤 time-step과 가장 연관이 있는가임. 일반적인 Seq2Seq-Attention 모델에서의 번역 태스크의 문제는 원본 언어(Source Language), 번역된 언어(Target Language)간의 대응 여부는 Attention을 통해 찾을 수 있었음. (input 나는 소년이다 <-> output I am a boy 간에 ‘나’ 와 ‘I’, ‘소년’과 ‘boy’의 대응관계)
* **그러나 각 문장에서의 관계를 나타낼 수 없음.** ‘I love tiger but it is scare’에서 ‘it’이 무엇을 나타내는지 와 같은 문제는 기존 Encoder-Decoder 기반의 Attention메커니즘에서는 찾을 수 없었음.
* Attention 모델은 Seq2seq의 성능을 높였으나, RNN이 순차로 이뤄져 연산이 느리다는 구조적인 문제점이 있음. 또한 병렬처리가 불가능함. 이런 단점을 해결하기 위해 RNN을 없애는 아이디어가 나오게 됨.

### **Self-attention**

Self-attention이란?

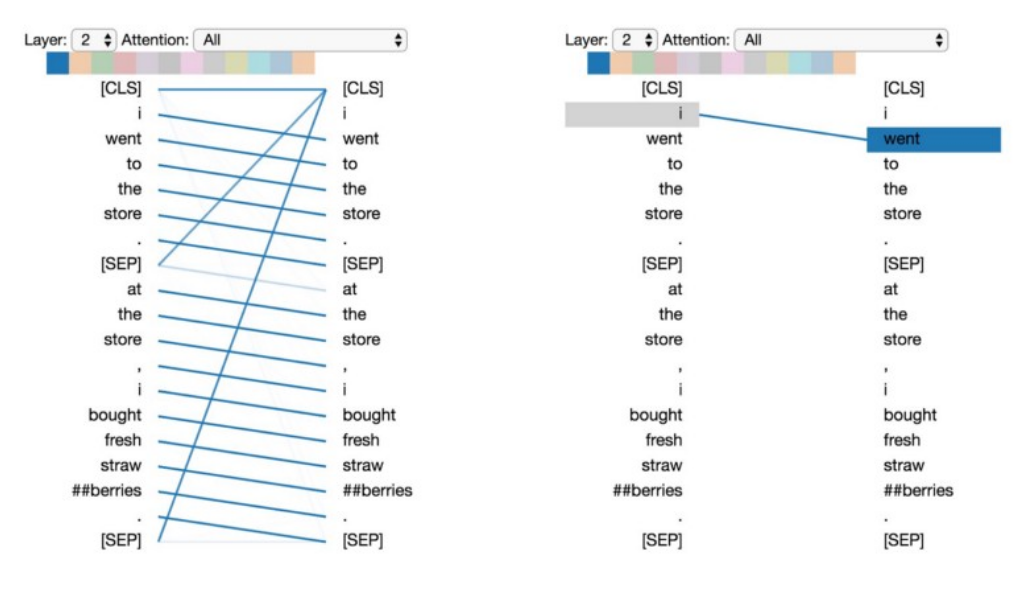
* 2017년 구글이 발표한 논문인 "Attention is all you need" (https://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf)에서 제안된 **transformer model**에서 소개된 방법
* 기존의 seq2seq의 구조인 인코더-디코더 따르면서도 순환신경만 기반으로 구성된 기존 모델과 다르게 단순히 어텐션 구조만으로 전체 모델을 만들어 어텐션 기법의 중요성을 강조
* 문장 내에서 token들이 스스로를 잘 표현하는 context vector를 갖게 만듦.



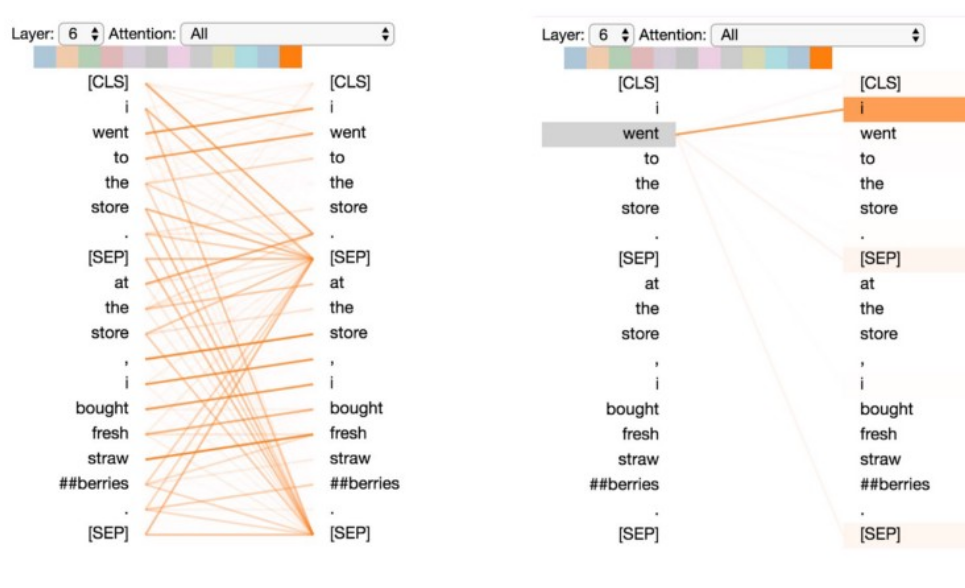
밑줄 친 부분에 따라 it에 대한 가중치가 달라짐

Bert has multi-head attention : Bert is multi-headed beast!

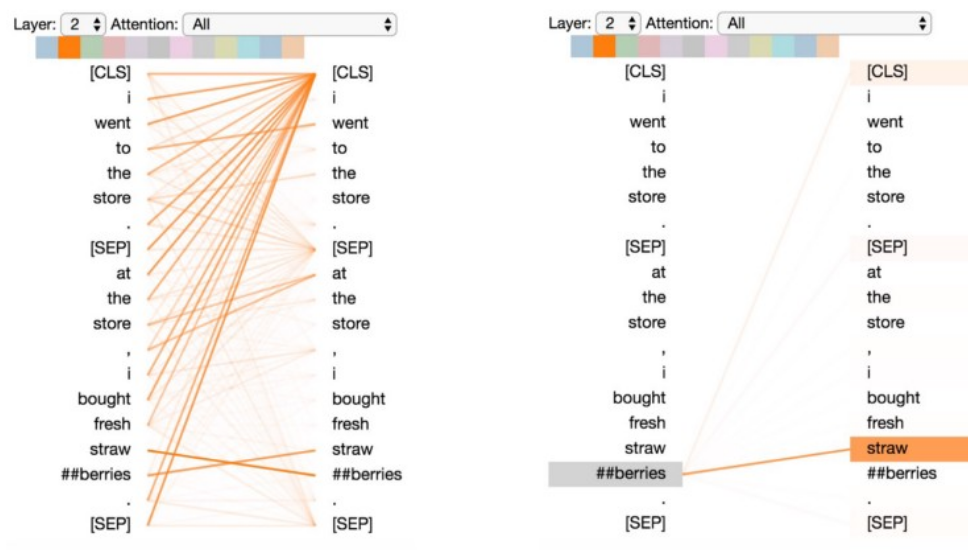
* Bert base 모델의 경우 Layer 12, attention head 12개 architecture를 가지고 있다.
* 따라서 12\*12 = 144, 144개의 distinct attention을 갖게 됨



L2, H0: Attention to next word



L6, H11: Attention to previous word



L2, H1: Attention to other words predictive of word

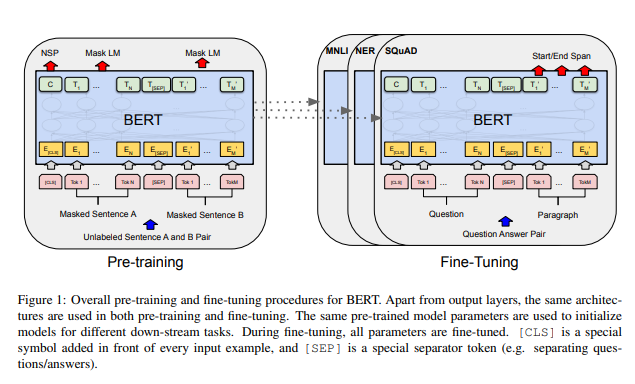
기존의 word2vec 같은 경우 하나의 단어는 하나의 vector 값만을 표현하였지만, transformer 계열 모델의 경우 multi-headed attention을 이용함으로써 하나의 단어가 여러 개의 “representation 공간”을 가질 수 있게 됨.

## **2. Pre-training**

Bert의 pre-training 과정은 크게 Input Embedding, MLM, NSP 3가지 과정으로 나눠진다.

### **Pre-training data (BERT base)**

* BookCorpus (Zhu et al., 2015) (800M words) + English Wikipedia (2,500M words)
* Unsupervised Learning으로 진행됨
* long contiguous sequence만을 학습시키기 위해 Wikipedia는 text passage만 추출하여 사용



BERT의 Pre-training과 Fine-tuning

BERT의 Pre-training은 tokenization을 수행하여 embedding 된 Sentence 2개를 Input Data로 사용한다. (Input Data 생성에 대해서는 후술) Input 값은 Transformer layer를 거치며 문장 내에서 단어(token)스스로를 가장 잘 표현하는 Self-attention된 token vector(Contextual representation of token), 그리고 후술할 NSP(Next Sentence Prediction)와 MLM(Masked Language Model)을 거치며 언어의 맥락 정보를 저장한 모델을 구축함.

이 후 Fine-tuning 과정에서는 pre-training 된 모델을 로드하여, MNLI, NER, SQuAD와 같은 다양한 NLP Classification task를 수행하는 labeled train data를 학습시켜 분류를 하게 됨.

### **2) Tokenization**

**Subword segmentation**

* 단어 분리(Subword segmenation) 작업은 하나의 단어는 (단어보다 작은 단위의) 의미있는 여러 내부 단어들(subwords)의 조합으로 구성된 경우가 많기 때문에, 하나의 단어를 여러 내부 단어로 분리해서 단어를 이해해보겠다는 의도를 가진 전처리 작업
* 더 적은 vocab으로 더 많은 단어를 표현하고 out of vocabulary 방지

Bert의 Tokenization은 Subword segmentation의 하나인 WPM(word piece model, by Google) 기반으로 tokenization 된다.

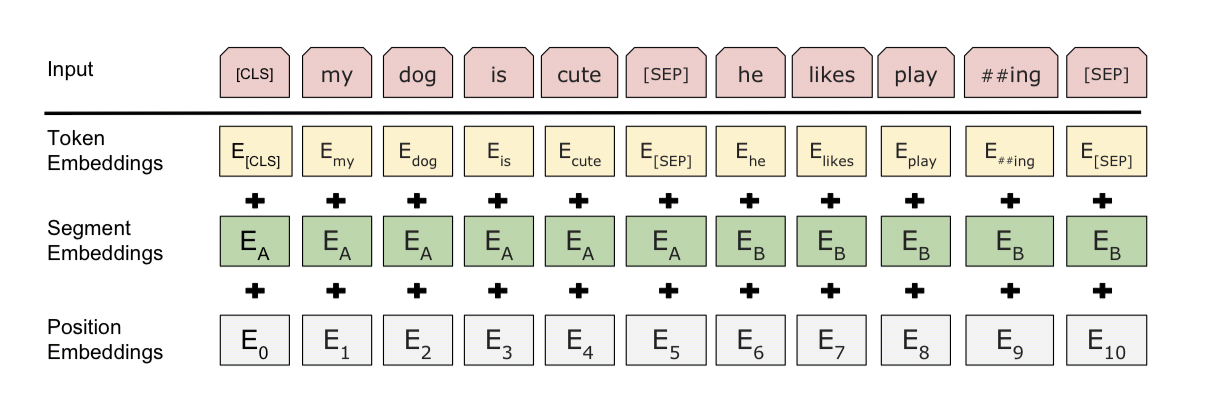
[Bert Tokenizer 모음](https://huggingface.co/transformers/v1.2.0/_modules/pytorch_transformers/tokenization_bert.html)

Tokenization 과정

* Space 단위로 token 분리
* Pretrained 된 WPM tokenizer 기준으로 subwords segmentation
* 문장 시작에 [CLS], 문장 끝에 [SEP], max\_seq\_len에 따라 [PAD] special token 붙이기

### **3) Input Embedding**

Tokenization된 문장은 Token Embeddings, Sentence Embeddings, Positional Embedings를 합쳐 모델에 입력할 수 있는 Input Embedding 값으로 변환된다.

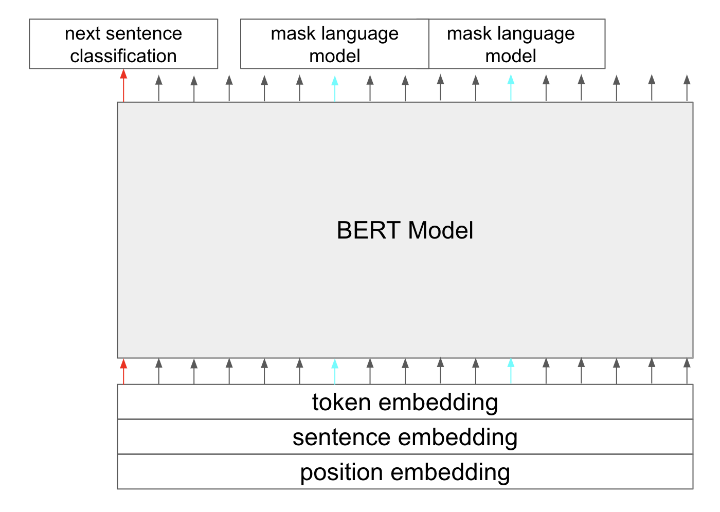


BERT는 Input data로 3 가지의 vector들이 합쳐서 만들어짐

1) Token Embeddings : Input에 들어갈 2개의 문장을 단어 token 단위로 자르고 token을 수학적으로 표현함. 이후 NSL와 MLM을 실시하기 위해, sentence 순서를 50% 확률로 변경하고, 15%의 token을 masking함. Masked token의 80%는 masked([MSK])되고, 10%는 그대로, 10%는 다른 token으로 치환하게 됨. special token으로는 문장 가장 앞([CLS])과 문장이 분리되는 부분([SEP])에 들어가게 됨.

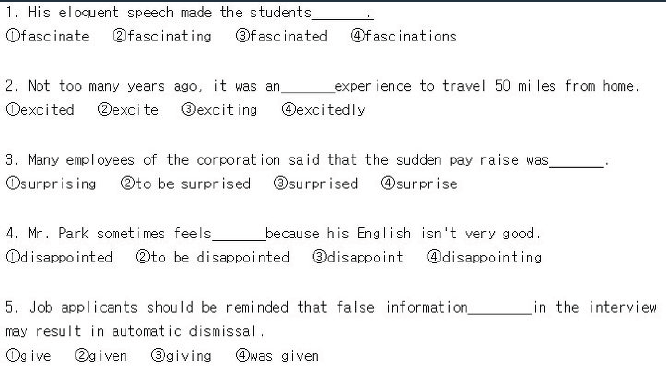
2) Segment Embeddings : 문장의 순서를 Embedding함. 그림 상에서 A, B로 표현됨.

3) Position Embeddings : 단어의 순서를 Embedding함. (Transformer는 RNN이나 CNN처럼 data를 순차적으로 연산하지 않기 때문에 단어와 문장의 순서정보를 token에 별도로 넣어줘야 함) 또한 같은 단어라도 쓰인 위치에 따라 다른 임베딩 값을 가질 수 있음

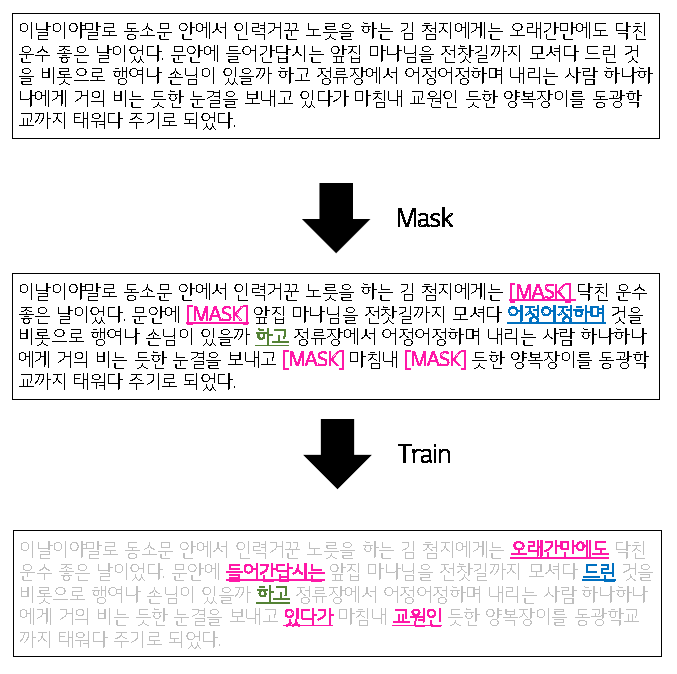


### **4 )MLM (Masked Language Model)**

token embedding 과정에서 각 문장 token을 15% 확률로 masked 하게 되며, (문맥을 계산하여) masked token이 무엇인지를 예측하는 학습을 진행함.



입력 token의 15%를 선택하여 80% [MASK], 10% 랜덤토큰대체, 10% 유지

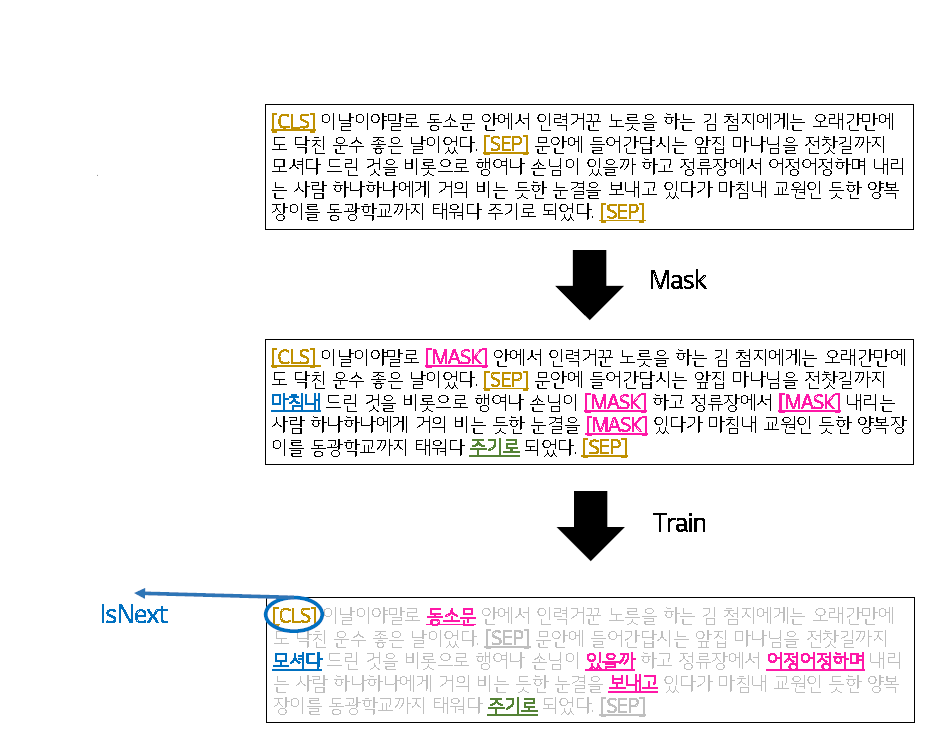
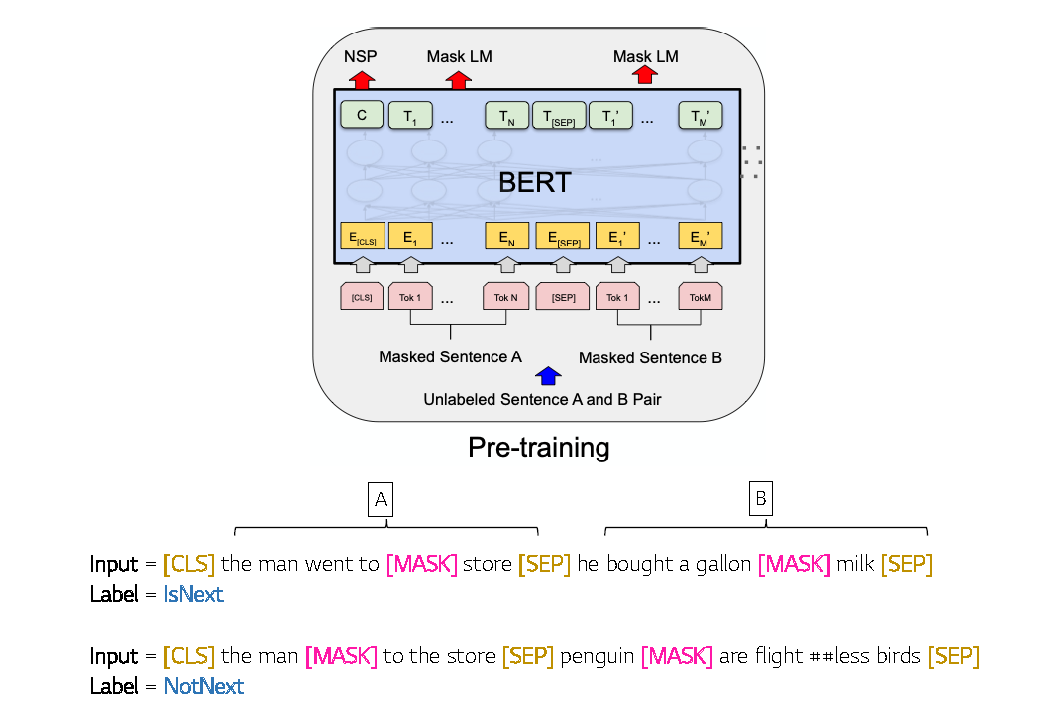


### **5) NSP (Next Sentence Prediction)**

NSP를 위해 corpus에서 실제 next sentence와 random sentence를 50% 확률로 가져오게 되며, 이를 가지고 IsNext인지 NotNext인지를 예측하는 학습을 진행. (Segment embedding으로 문장의 순서 정보를 보존)

|  |
| --- |
| sentence\_a = “신용 대출을 신청 하고 싶어요”  sentence\_b = “대출 관련 서류를 준비해 오셨어요?”  Label: IsNext  sentence\_a = “신용 대출을 신청 하고 싶어요”  sentence\_b = ”어떤 팀이 올해 프로야구 우승 했나요?”  Label: NotNext |

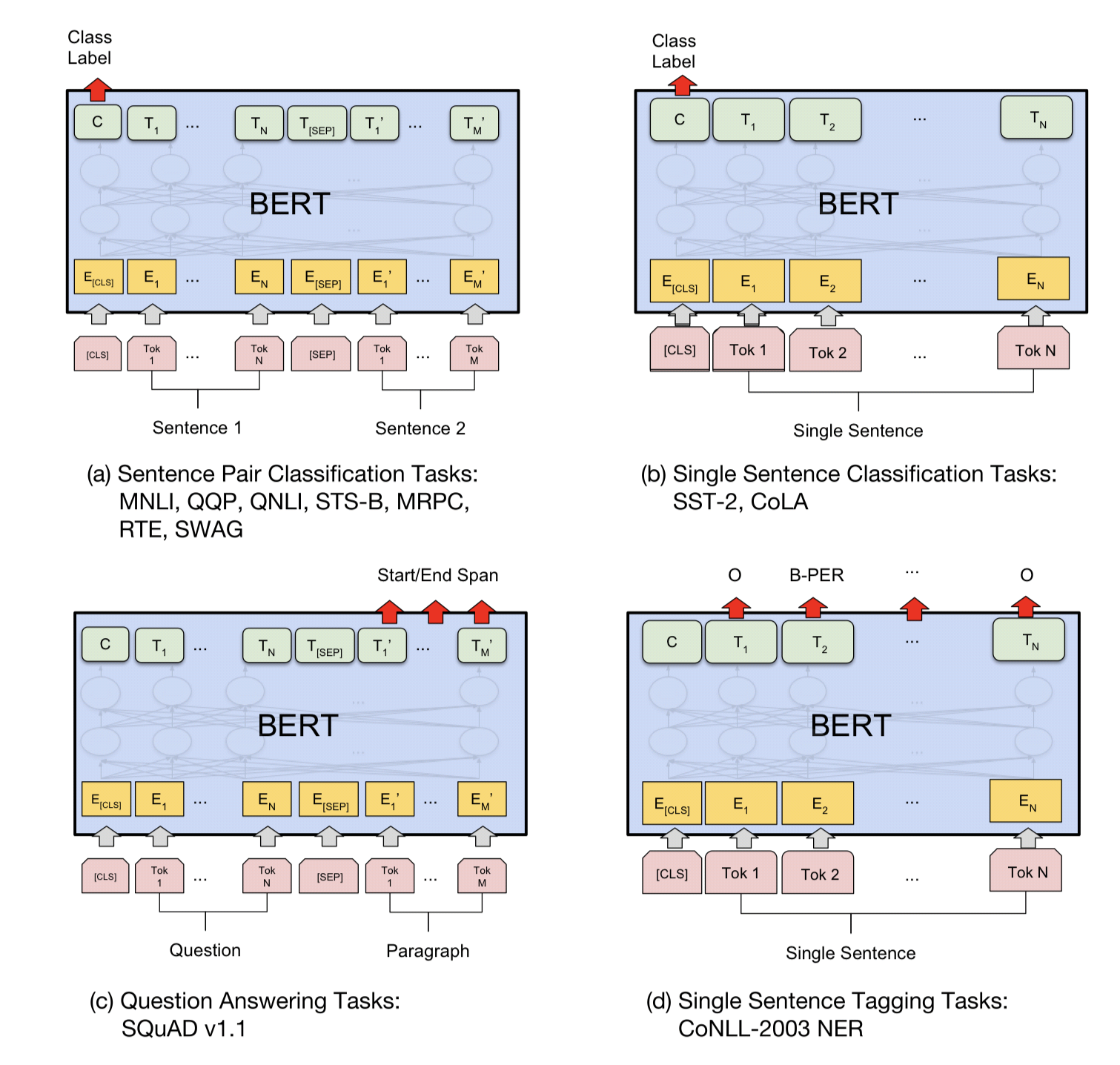
MLM & NSP 방식



## **3. Fine-tuning(Classification Layer)**

Pre-training된 모델을 불러와 NLP task를 하는데 사용할 수 있음. 크게 4가지 유형으로 구분된다.

모델은 pre-trained parameters로 초기화 되고 각각 downstream task 별로 다른 모델이 만들어진다. 문맥을 이해하는 Language Model위에 fine-tuning 방식으로 학습되므로 적은 label 데이터로도 높은 정확도를 갖는 모델을 생성할 수 있다.



(a), (b)는 sequence-level task, (c)와 (d)는 token-level task로 각각의 특징은 다음과 같음.

(a)는 2개 sentence를 넣고 그 관계(pair classification)를 분류하는 task임. 문장의 유사 관계 분류 등에 사용되며 **(b)는 1개 sentence를 넣고 감성과 같은 문장의 특성을 분류함**

(c)는 질문(Question)과 본문(Paragraph)을 넣고 문장 단위의 답을 호출함. paragraph에서 Q와 유사도가 높은 문장을 답으로 내놓기 위해, [SEP] token 이후의 00번째 token부터 00번째 token까지를 호출함.

d)는 1개 문장을 넣고, 문장 내 token의 정보를 분류함. NER(Named Entity Recognition, 개체명 인식)이나 형태소 분석과 같이 single sentence에서 각 token이 어떤 class를 갖는지 모두 classifier 적용하여 정답을 찾아냄.