BERT

기존의 모델들

BERT 이전의 Pre-trained 모델은 크게 두 가지로 구성된다.

- Feature-based (사전 훈련된 모델의 파라미터를 고정)
  - **ELMo**
  - Shallowly 양방향 언어 모델(biLM)
  - **ELMo의 경우 양방향이지만 진정한 양방향이 아니다.**
  - 언어 모델의 파라미터는 Freezing하고,
    각 layer에 대한 비율 조정 파라미터(γ)를 계산
    - $[\gamma_{input}, \gamma_{layer1}, \gamma_{layer} \quad \gamma_{layer}]$

- Fine-tuning (사전 훈련된 모델의 파라미터를 함께 학습)
  - **GPT**
  - 단방향(Unidirectional) 언어 모델(language models)
  - **GPT의 경우 단방향 모델이다.**

기존의 모델들

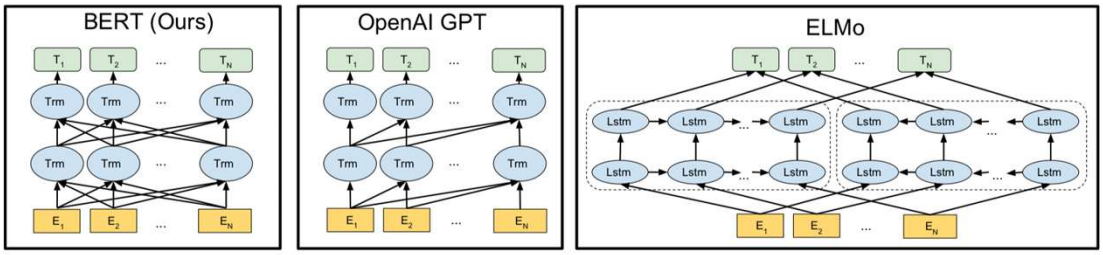
BERT 이전의 Pre-trained 모델은 크게 두 가지로 구성된다.

- Feature-based (사전 훈련된 모델의 파라미터를 고정)
  - **ELMo**
  - Shallowly 양방향 언어 모델(biLM)
  - **ELMo의 경우 양방향이지만 진정한 양방향이 아니다.**
  - 언어 모델의 파라미터는 Freezing하고,
    각 layer에 대한 비율 조정 파라미터(γ)를 계산
    - $[\gamma_{input}, \gamma_{layer1}, \gamma_{layer2}\ \gamma_{layer3}]$

BERT
Fine-tuning 방식을 취하되,
GPT와는 달리 양방향을 실현한다.

- Fine-tuning (사전 훈련된 모델의 파라미터를 함께 학습)
  - **GPT**
  - 단방향(Unidirectional) 언어 모델(language models)
  - **GPT의 경우 단방향 모델이다.**

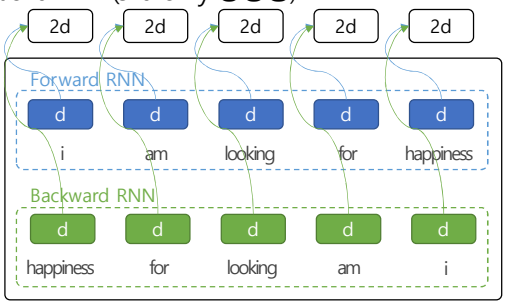
기존의 모델들



Un-directional Vs. Bi-directional

- RNN Language Model, Transformer: 이전 단어들을 보고 다음 단어를 예측(**Uni**-directional)
- BiLM : Shallowly Bi-directional
- BERT
  - Masked Language Model : 문장에 Masking15% 처리한 후 Mask된 단어를 예측
  - 1. 주변 단어들을 보고 Masked 단어를 예측 (**Bi**-directional)

Bidirectional RNN(shallowly 양방향)



Bidirectional BERT(deeply 양방향)

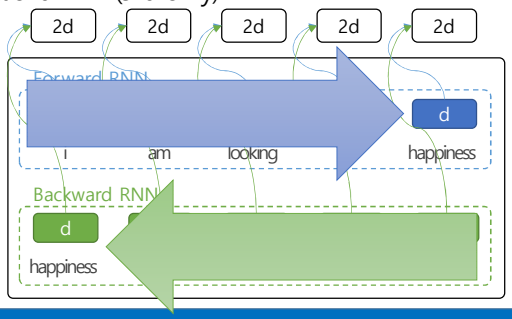
Input: i am looking for [MASK1] . I am [MASK2]

Labels: [MASK1] = happiness; [MASK2] = Donghwa

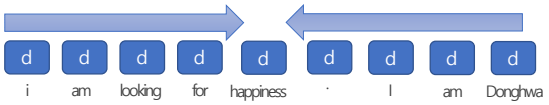
Un-directional Vs. Bi-directional

- RNN Language Model, Transformer: 이전 단어들을 보고 다음 단어를 예측(**Uni**-directional)
- BiLM : Shallowly Bi-directional
- BERT
  - Masked Language Model : 문장에 Masking15% 처리한 후 Mask된 단어를 예측
  - 1. 주변 단어들을 보고 Masked 단어를 예측 (**Bi**-directional)

Bidirectional RNN(shallowly)



Bidirectional BERT(deeply)



Input: i am looking for [MASK1] . I am [MASK2]

Labels: [MASK1] = happiness; [MASK2] = Donghwa

Next Sentence Prediction

- BERT
  - 2. 문장들 사이의 관계를 학습하기 위해 "다음 문장 예측 태스크를 도입

BERT에서는 훈련 데이터에서 두 문장을 이어붙여 원래의 훈련 데이터에서 붙여져 있던 문장인지를 맞추는 다음 문장 예측 문제를 수행한다.

50% : Sentence A, B가 실제 다음 문장
50% : Sentence A, B가 랜덤으로 뽑힌 관계가 없는 두 문장

Sentence A: i am looking for happeiness
Sentence B: i am Wonjoon

Labels: IsNextSentence

Sentence A: i am looking for happeiness
Sentence B: i am Sana

Labels: NotNextSentence

BERT의 Pre-training

결론적으로 BERT는 사전 학습 단계에서 두 가지 방법으로 학습된다.

- 1. Masked Language Model : 문장에 Masking15% 처리한 후 Mask된 단어를 예측
- 2. 문장들 사이의 관계를 학습하기 위해 "다음 문장 예측 태스크를 도입"

사실 BERT의 성능의 대부분은 1. Masked Language Model에서 나오는 편이고,
2.의 경우에는 BERT가 향후 파인 튜닝 시에 두 개의 문장을 입력으로 받는 태스크를 풀 경우를 위해 추가한

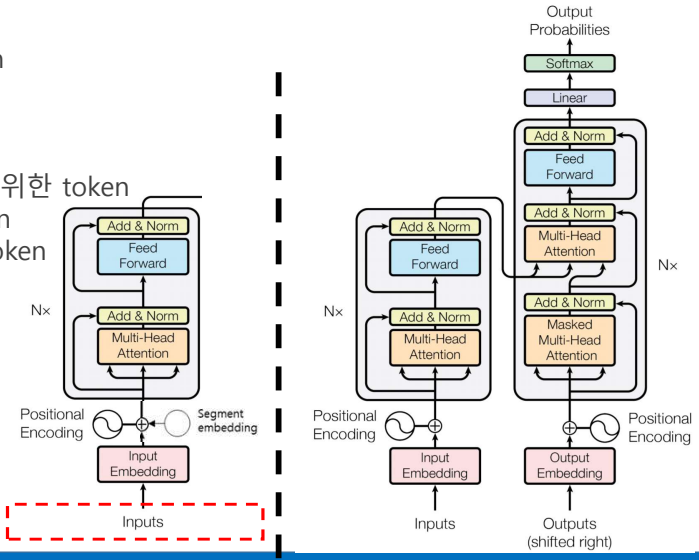
Ex) 두 개의 문장을 입력받고 이 문장이 모순 관계인지, 중립 관계인지를 맞추는 태스크 등.

실제로 BERT 이후의 연구들에서는 2.를 그다지 중요하게 생각하지 않는 경우가 많음.

Transformer Vs. BERT

- Transformer
  - [SOS]: 문장의 시작을 알리는 token
  - [EOS]: 문장의 끝을 알리는 token

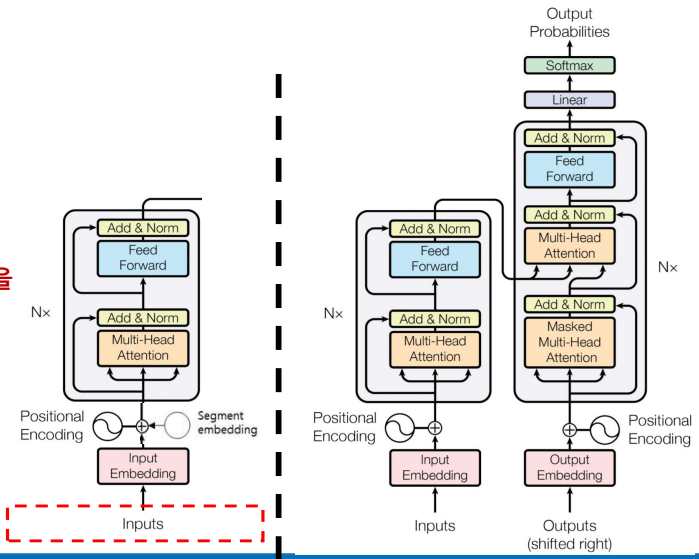
- BERT
  - [CLS]: Task-specific한 정보를 주기 위한 token
  - [SEP]: Token A, B를 구분하는 token
  - [SEP]: Token A, B의 끝을 알리는 token
  - [MASK]: 예측하는 Target



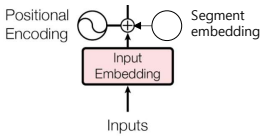
Transformer Vs. BERT

- Transformer
  - Input embedding
  - Positional embedding

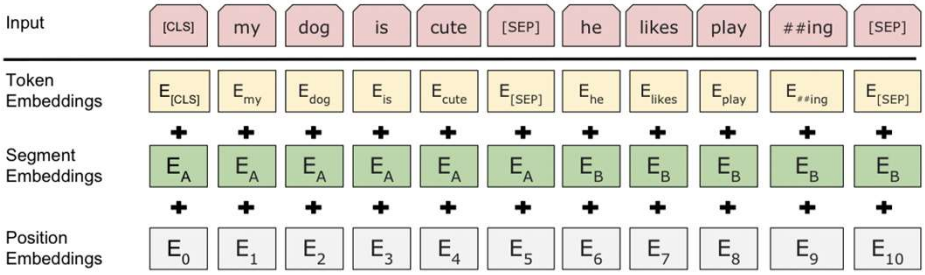
- BERT
  - Input embedding
  - Positional embedding
  - **Segment embedding 추가**
    - **Token A group, Token B group을 구분하는 정보**



BERT의 세 가지 Embedding

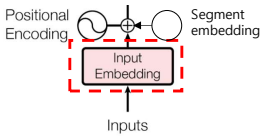


- **BERT에는 총 세 가지 Embedding이 사용된다.**
- Token Embedding은 우리가 알고있는 Input Word Embedding
- Segment Embeddign은 두 개의 문장을 구분하기 위한 Embedding
- Position Embeddding은 트랜스포머의 Positional Encoding을 대체할 수 있는 방법



Token Embedding

- Encoder
  - Sent1: [CLS] i am looking for happiness [SEP], B type sentence [SEP]



looking x_j

Vocab size(Encoder)

<PAD>	<s>	</s>	<UNK>	am	for	...	i	looking	...	w	...	z
0	0	0	0	0	0	0	0	1	0	0	0	0

Embed size

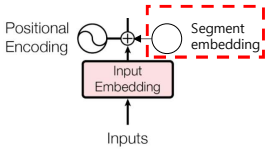
<PAD>	0.0	0.0	0.0	0.0
⋮	0.0	0.3	0.7	0.0
looking	0.2	0.1	0.7	0.2
⋮	0.9	0.1	0.7	0.9
	0.5	0.7	0.1	0.5
	0.2	0.9	0.4	0.2
z	0.2	0.0	0.7	0.2

0.2	0.1	0.7	0.2

w_j looking

Segment Embedding

- Encoder
  - Sent1: [CLS] i am looking for happiness [SEP], B type sentence [SEP]



looking x_j

Vocab size(Encoder)

A	B
1	0

Embed size

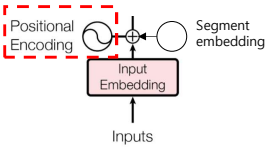
A				
B				

0.1	0.1	0.1	0.3

w_j looking

Position Embedding

- Encoder
  - Sent1: [CLS] i am looking for happiness [SEP], B type sentence [SEP]



3th position

Position vocab

0	...	3	...	9	10
0	...	1	0	0	0

Embed size

0	0.1	0.4	0.7	0.1
⋮	0.0	0.3	0.7	0.0
3	0.1	0.1	0.9	0.1
⋮	...	...	...	...
10	0.2	0.0	0.7	0.2

3th position embedding

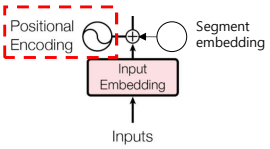
0.1	0.1	0.9	0.1

Position Embedding

- Encoder
  - Sent1: [CLS] i am looking for happiness [SEP], B type sentence [SEP]

0 1 2 3 4 5 6 7 8 9 10

샘플의 길이가 11이라고 가정해보자.



3th position

Position vocab

0	...	3	...	9	10
0	...	1	0	0	0

Embed size

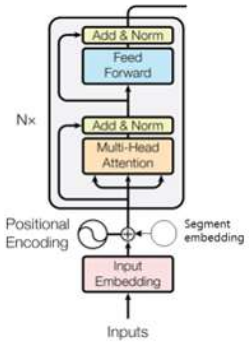
0	0.1	0.4	0.7	0.1
⋮	0.0	0.3	0.7	0.0
3	0.1	0.1	0.9	0.1
⋮	...	...	...	...
10	0.2	0.0	0.7	0.2

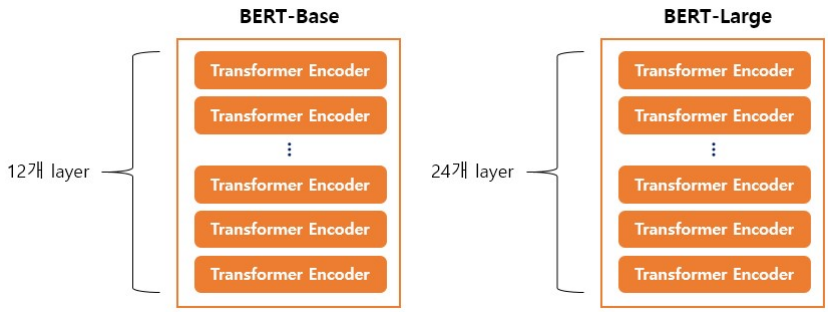
3th position embedding

0.1	0.1	0.9	0.1

BERT-Base Vs. BERT-Large

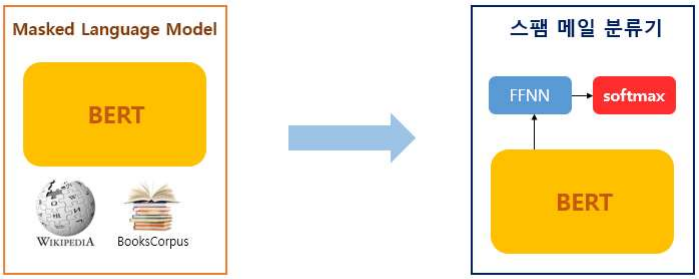
- BERT-BASE는 트랜스포머의 인코더를 12층을 적재.
- BERT-LARGE는 트랜스포머의 인코더를 24층을 적재.





BERT의 적용

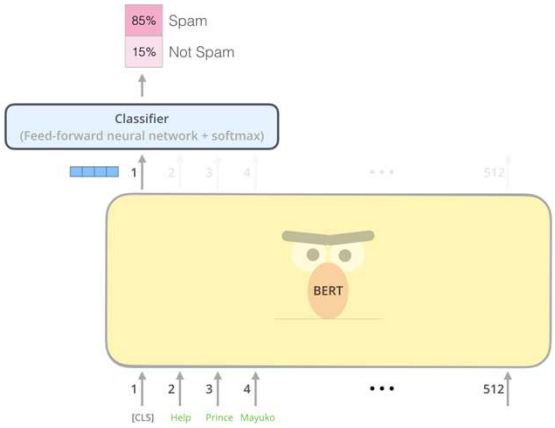
- 방대한 데이터로 사전 훈련된 언어 모델 BERT.
- BERT에 풀고자 하는 태크스에 맞는 추가적인 신경망을 추가.
- 풀고자 하는 태스크의 오차에 맞추어서 BERT를 학습(fine-tuning)



33억 단어에 대해서 4일간 학습시킨 언어 모델

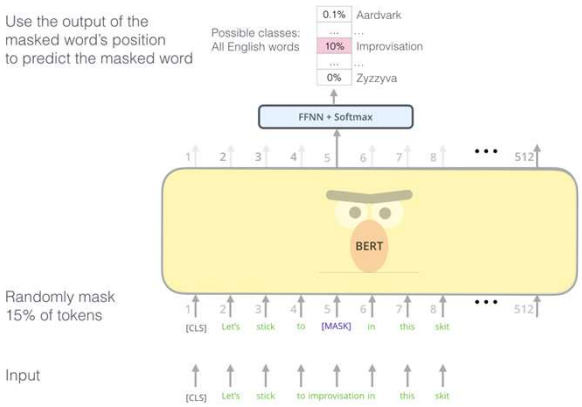
BERT의 적용

- BERT의 윗단의 신경망을 통해서 원하는 Task를 수행.
- 텍스트 분류의 경우 <CLS>의 토큰을 최종 Classification을 위한 입력으로 사용.



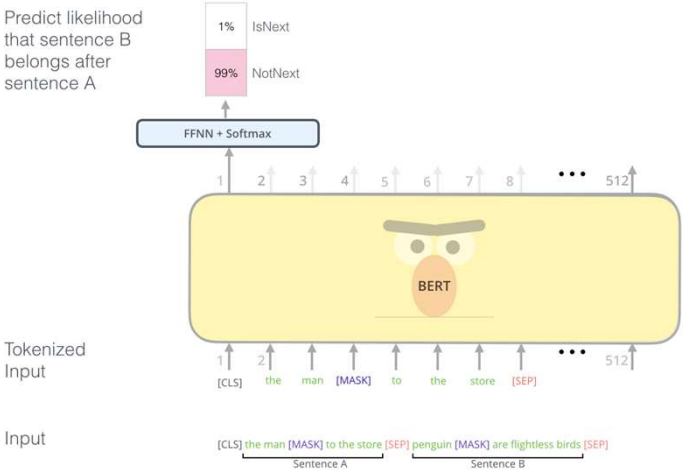
BERT의 사전 훈련

- BERT는 사전 훈련 시 첫번째 Task로 Masked Language Model을 학습



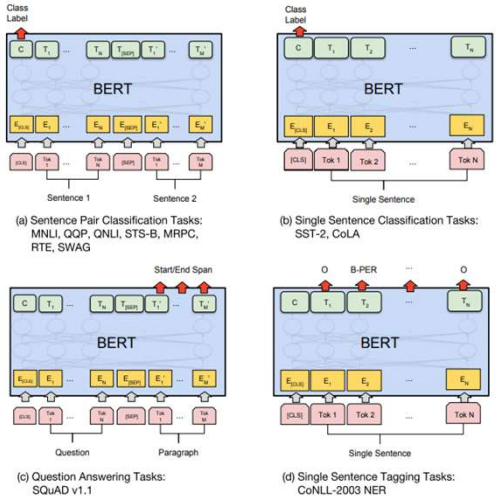
BERT의 사전 훈련

- BERT는 사전 훈련 시 두번째 Task로 다음 문장 예측

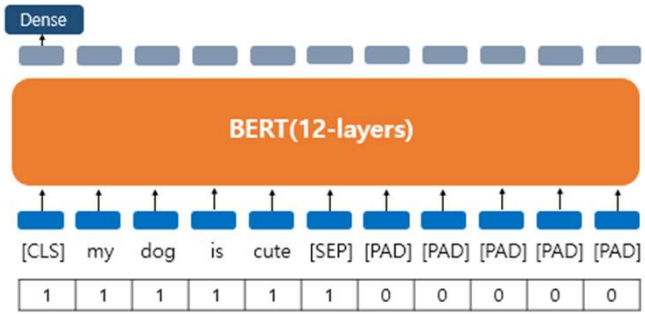


BERT의 다양한 적용

- **BERT를 다양한 태스크에 적용 가능.**
- Text Classification
- Named Entity Recognition
- QA
- NLI



어텐션 마스크(Attention Mask)



- 앞서 논문 설명에서는 언급되지 않았지만, 실제 BERT를 사용할 경우에 필요한 입력.
- 숫자 1은 해당 토큰은 실제 단어이므로 마스킹을 하지 않는다라는 의미이고, 숫자 0은 해당 토큰은 패딩 토큰이므로 마스킹을 한다는 의미. 위의 그림과 같이 실제 단어의 위치에는 1, 패딩 토큰의 위치에는 0의 값을 가지는 시퀀스를 만들어 BERT의 또 다른 입력으로 사용하면 된다.