Week3. NLP models

3-3. Transformer, Pre-trained Language Models (PLM)

신승진

Contents

Goal

Contextual Embedding이 무엇인지 설명할 수 있고 transformer와 bert의 모델 구조를 이해한다.

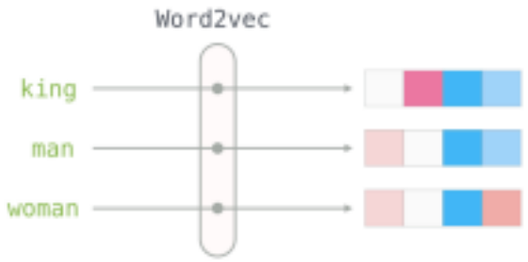
1. Contextual Embedding

“모델이 문맥에 맞는 vector를 구해줄거야”

- Word2vec 모델의 단점
  - 단어의 의미를 vector화 할 수 있지만, 한 단어가 가지고 있는 여러 의미를 vector로 표현할 수 없음. →Fixed Embedding
  - Input: Words (BOW) ex. “눈”

- Output: Fixed Word Embedding

- Contextual Embedding 등장
  - 하나의 단어가 문맥에 따라 여러가지 vector로 표현
  - Input: Sentence (word1, word2, word3, …) ex. “너 참 눈이 예쁘다” • Output: Contextualized Word Embedding
    - By: Model Weight



2. ELMO

“Embedding from Language MOdels” • biLSTM

- Deep representation (linear combination of 2 layers)

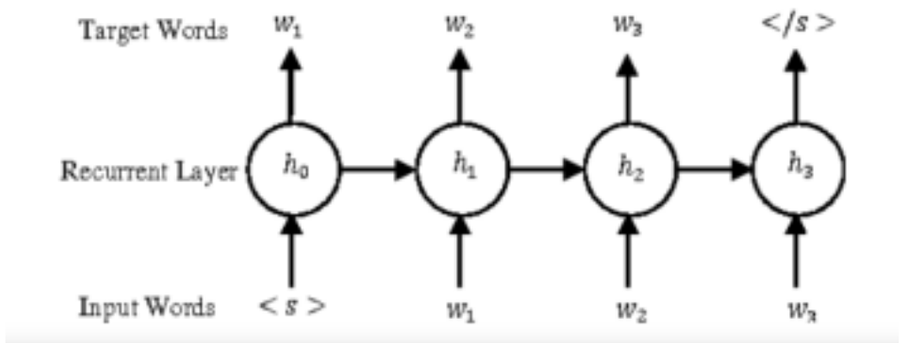
- Pre-trained Language Model
- 30M (30,000,000) 문장 학습
- LM (un-supervised)

Concat

Forward

Backward

문제



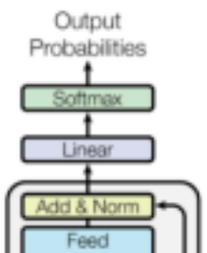
Linear Combination of 2 Layer Embeddings

charCNN

- 다음 단어 예측

3. Transformer

“Attention is all you need”



- LSTM 단점 (recurrent model의 단점)
  - Long term dependency
  - 병렬 학습이 불가능
- Transformer
  - Encoder + Decoder → 기계 독해
  - Encoder → BERT (MaskedLM 사용해 bi-directional 가능케) • Decoder → GPT

3. Transformer

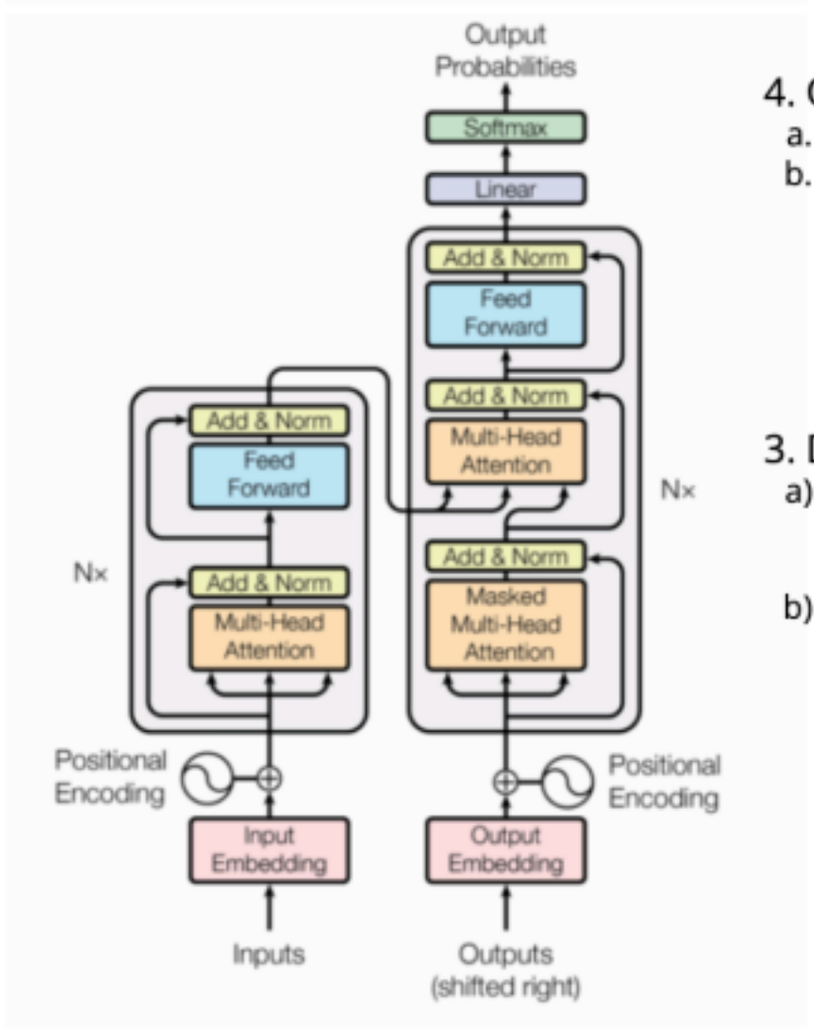
“Attention is all you need”

- Positional encoding
- Multi-head self-attention

Transformer

1. Input (dim = 512)
   a) Byte-pair encoding
   b) Positional encoding

2. Encoder (6 layers)
   a) layer = Attention + FFNN
   b) Multi-Head Attention



3. Decoder (6 layers)
   a) layer = Masked Attention + Attention (Encoder + Decoder) + FFNN
   b) Multi-Head Attention

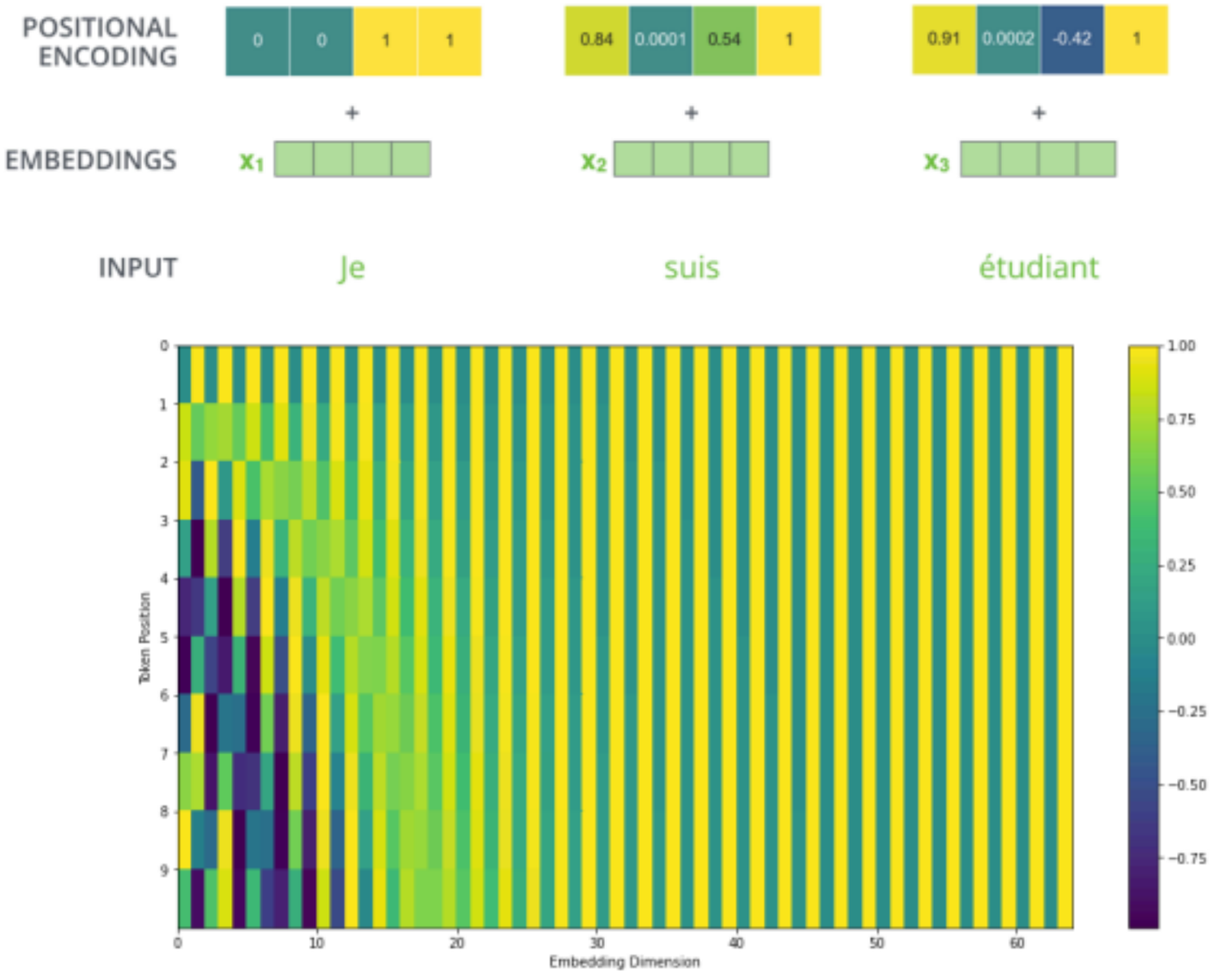
4. Output
   a. Linear
   b. Softmax

*base기준

3. Transformer

“Attention is all you need”

- Positional encoding
  - 단어에 “순서”를 주기 위해

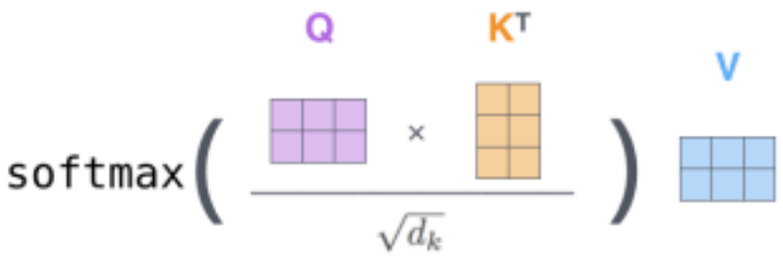


3. Transformer

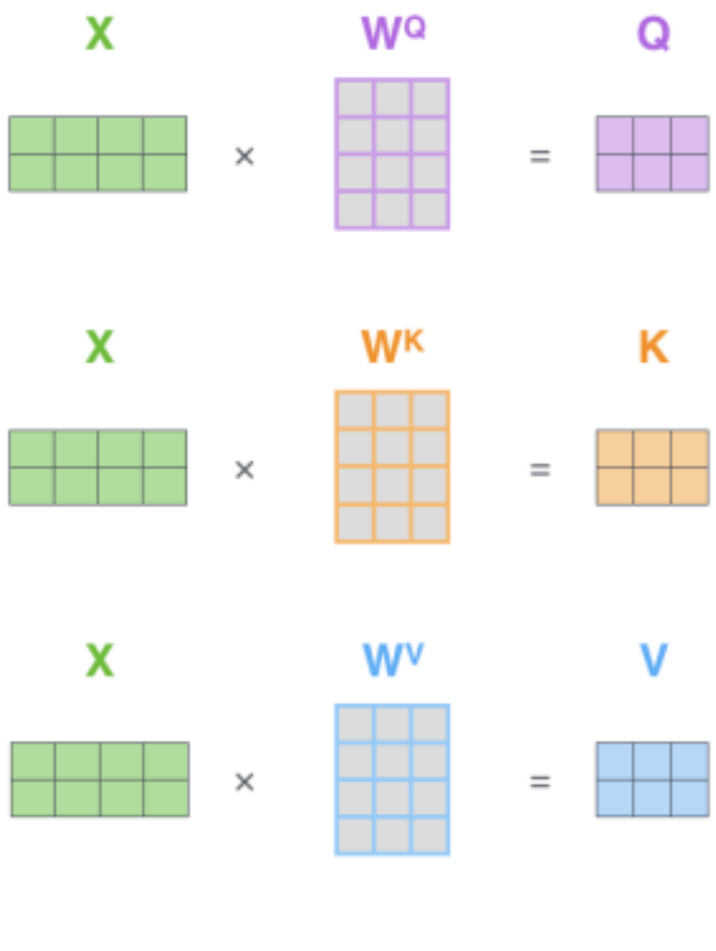
“Attention is all you need”

- Multi-head self-attention
  1. sequence의 Key, Query, Value matrix 생성

2. Key * Query를 weight 삼아 Value와 곱해 attention matrix



생성 = Z



Dimension

Token

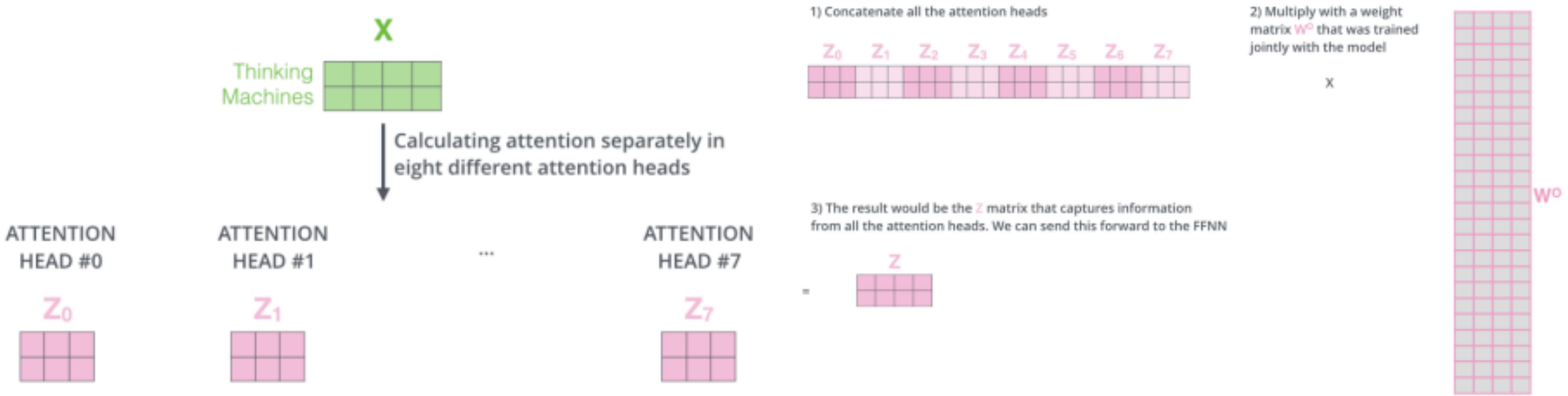
3. Transformer

“Attention is all you need”

- Multi-head self-attention
  3. Self attention을 여러번 계산 해 8개 (base 기준)의 attention

matrix 생성

4. 8개의 attention matrix를 concat한 후 weight matrix와 곱해 차원을 맞춰줌

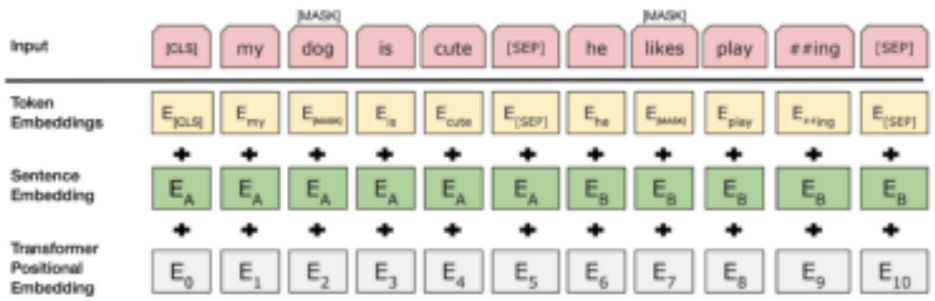


4. BERT

“Attention으로 양방향 ”

- 모델 구조
  - Input Layer ← Token Embedding + Sentence Embedding + Positional Embedding(≠positional encoding)

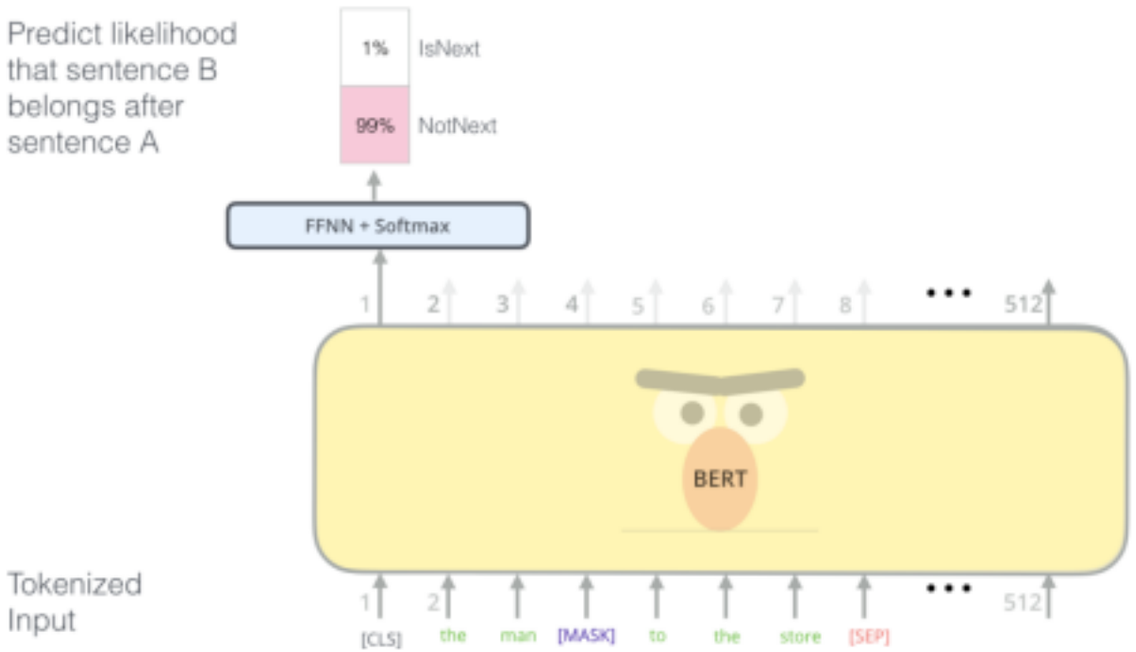
- 12 Encoder Layers

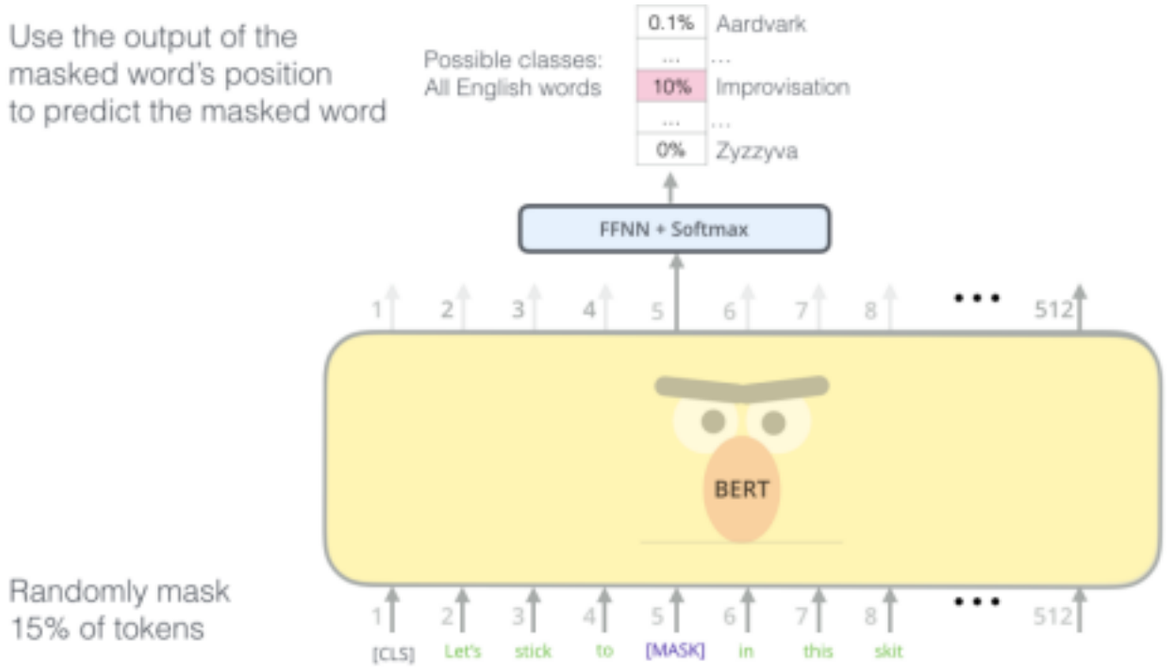


4. BERT

“Attention으로 양방향 ”

- Pre-training 2 tasks
  - Language Model → Masked LM (15% 토큰에 [MASK])
  - Next Sentence Prediction (50%:50%)





4. BERT

Corpus: “나는 코딩이 좋아요! 꺍꺍꺍”

{“나” : 0,
“##는” : 1,
“코딩” : 2,
“##이” : 3, “좋” : 4,
“##아” : 5, “##요” : 6, “##!” : 7,
“꺍꺍꺍” : 8}

“Attention으로 양방향 ”

- 자주 사용되는 parameters
  - vocab_size = 토크나이저 학습으로 만든 token lookup

table • hidden_size = 토큰 벡터의 차원

- num_hidden_layers = # 인코더
- num_attention_heads = 각 인코더 레이어의 # attention head
- max_position_embedding = 최대 입력 토큰의 개수

⚠

6. Reference

- [Illustrated Transformer](https://jalammar.github.io/illustrated-transformer/) ❓❓❓
- [Illustrated ELMo & BERT](https://jalammar.github.io/illustrated-bert/) ❓❓❓

- [Annotated Transformer](https://nlp.seas.harvard.edu/2018/04/03/attention.html) ◆ ◆

- [HuggingFace](https://huggingface.co) ◆ ◆

- [BERT embeddings](https://medium.com/@_init_/why-bert-has-3-embedding-layers-and their-implementation-details-9c261108e28a) ◆

- [Attention is all you need](https://arxiv.org/pdf/1706.03762.pdf) ◆ ◆ ◆

- [BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding](https:// arxiv.org/pdf/1810.04805.pdf) ◆ ◆ ◆