Attention is All You Need

Ashish Vaswani 12 Jun 2017

논문 구현

CloseAI팀 이상헌 김유철 박준혁 이정훈

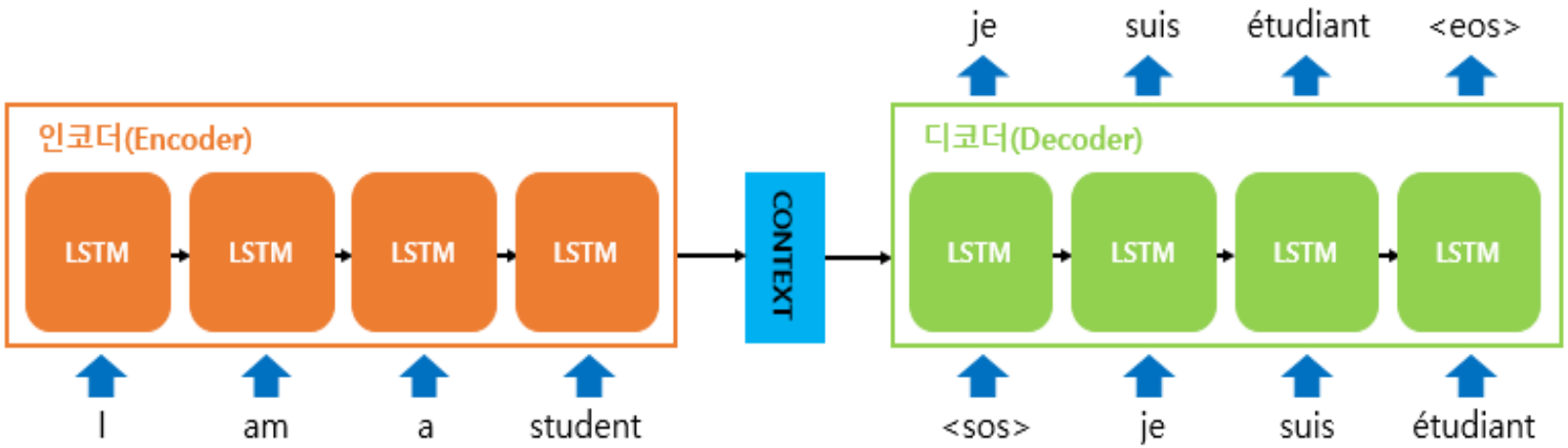
09 May 2024

목차

- Seq2Seq
- Attention
- Transformer
- 코드 구현

Sequence-to-Sequence

Seq2Seq 모델

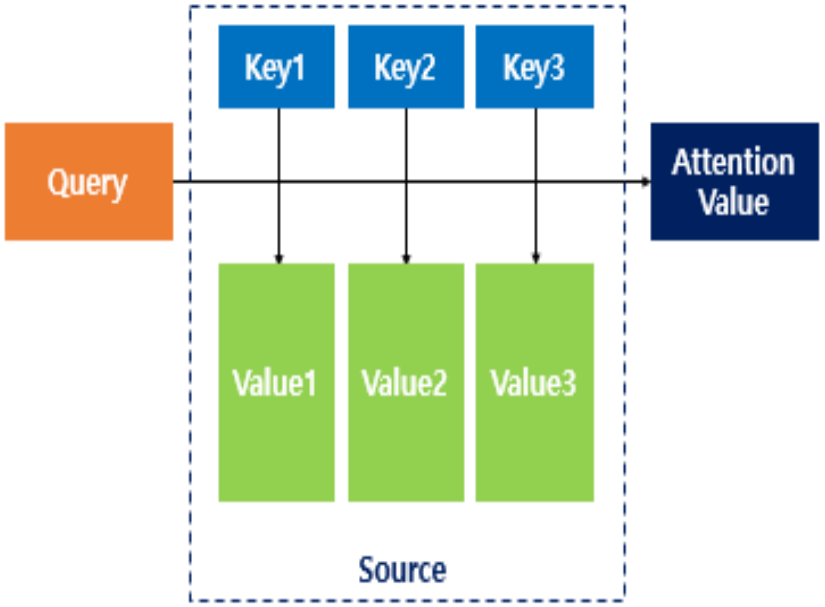


- Encoder
- Context vector
- Decoder

- 컨텍스트 벡터의 크기 제한
- 병렬 처리 X
- 디코더의 정보 부족

Attention

Attention 메커니즘



- **Seq2Seq 어텐션의 Q,K,V**

Query : t 시점의 디코더 셀에서의 은닉 상태

Key : 모든 시점의 인코더 셀의 은닉 상태들

Value : 모든 시점의 인코더 셀의 은닉 상태들

- **트랜스포머 셀프 어텐션 Q,K,V**

Query : 입력 문장의 모든 단어 벡터들

Keys : 입력 문장의 모든 단어 벡터들

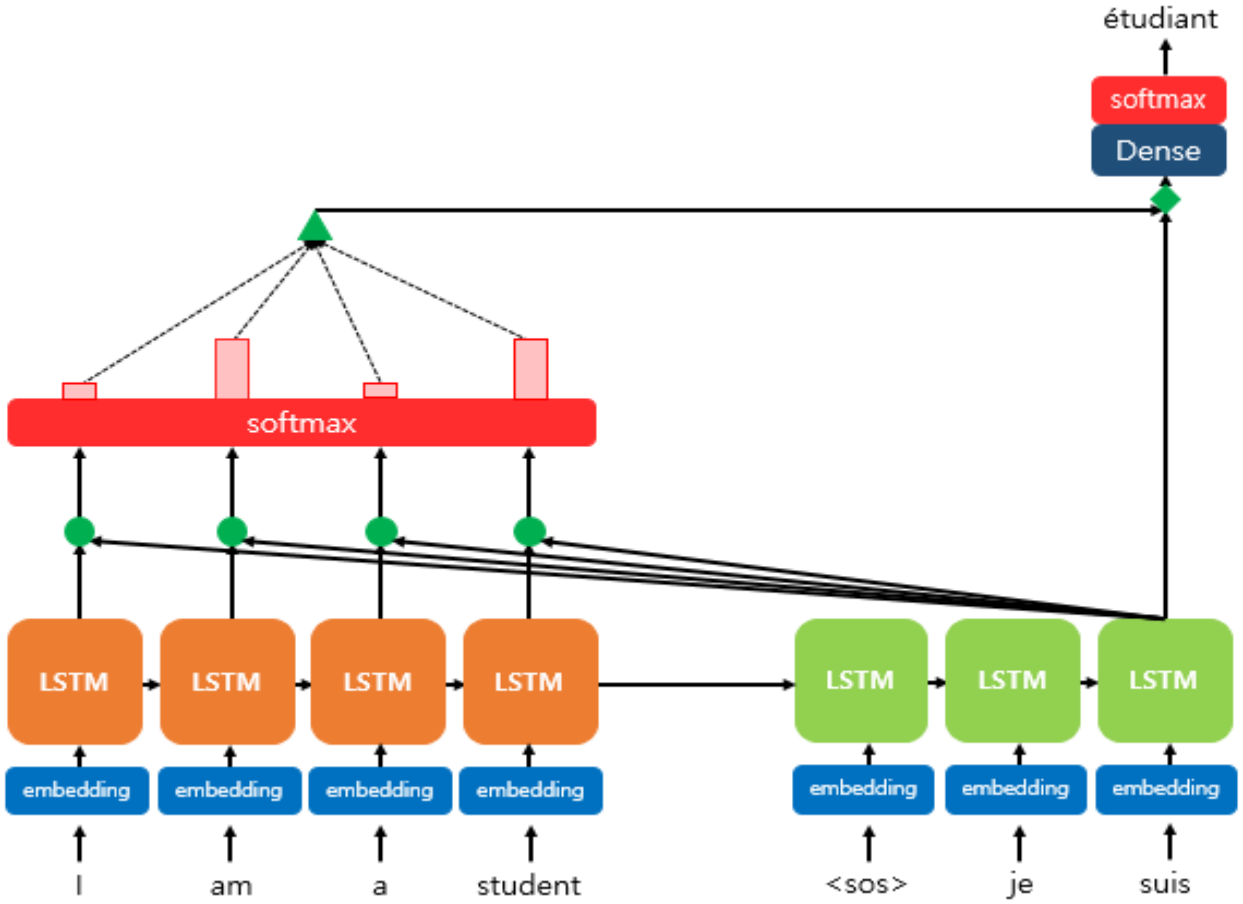
Values : 입력 문장의 모든 단어 벡터들

Attention 메커니즘

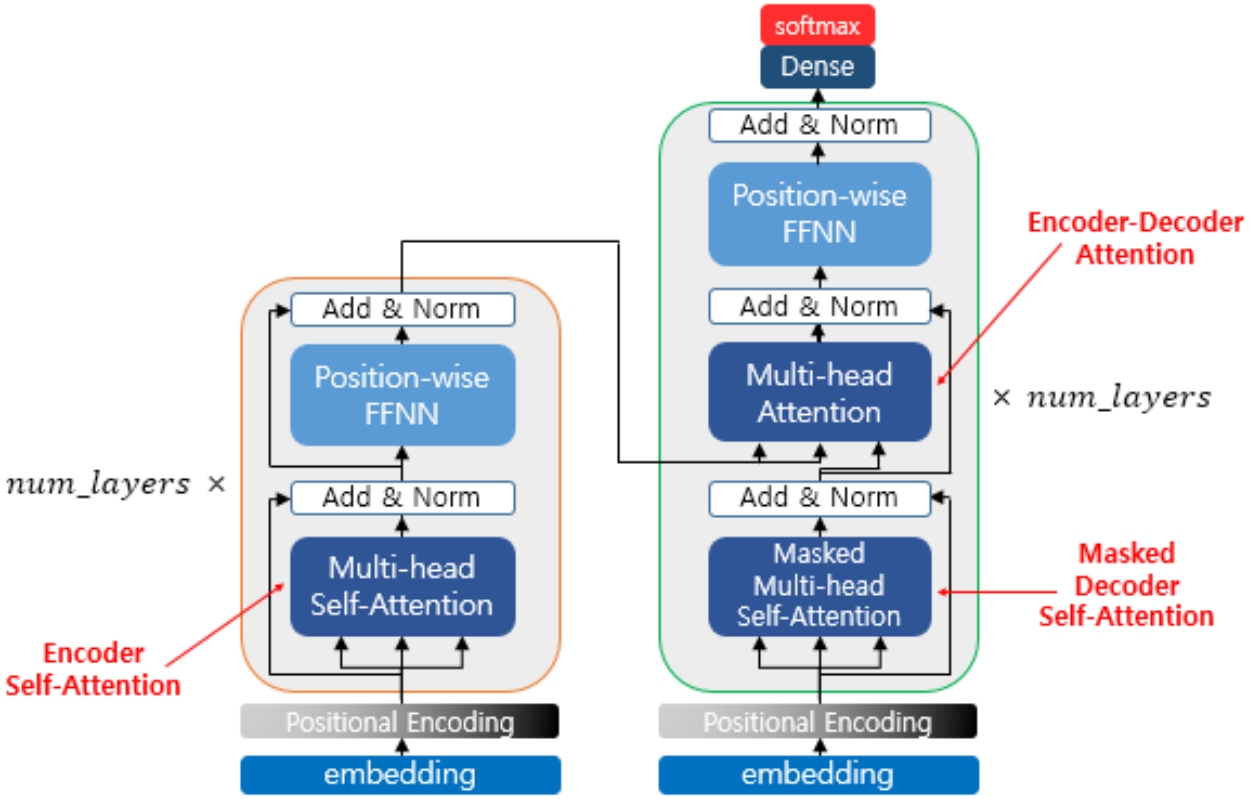
- 디코더에서 출력 단어를 예측하는 매 시점(time step)마다, 인코더에서의 전체 입력 문장을 다시 한 번 참고

- 전체 입력 문장을 전부 다 동일한 비율로 참고하는 것이 아니라, 해당 시점에서 예측해야할 단어와 연관이 있는 입력 단어 부분을 좀 더 집중(attention)해서 보게 됩니다

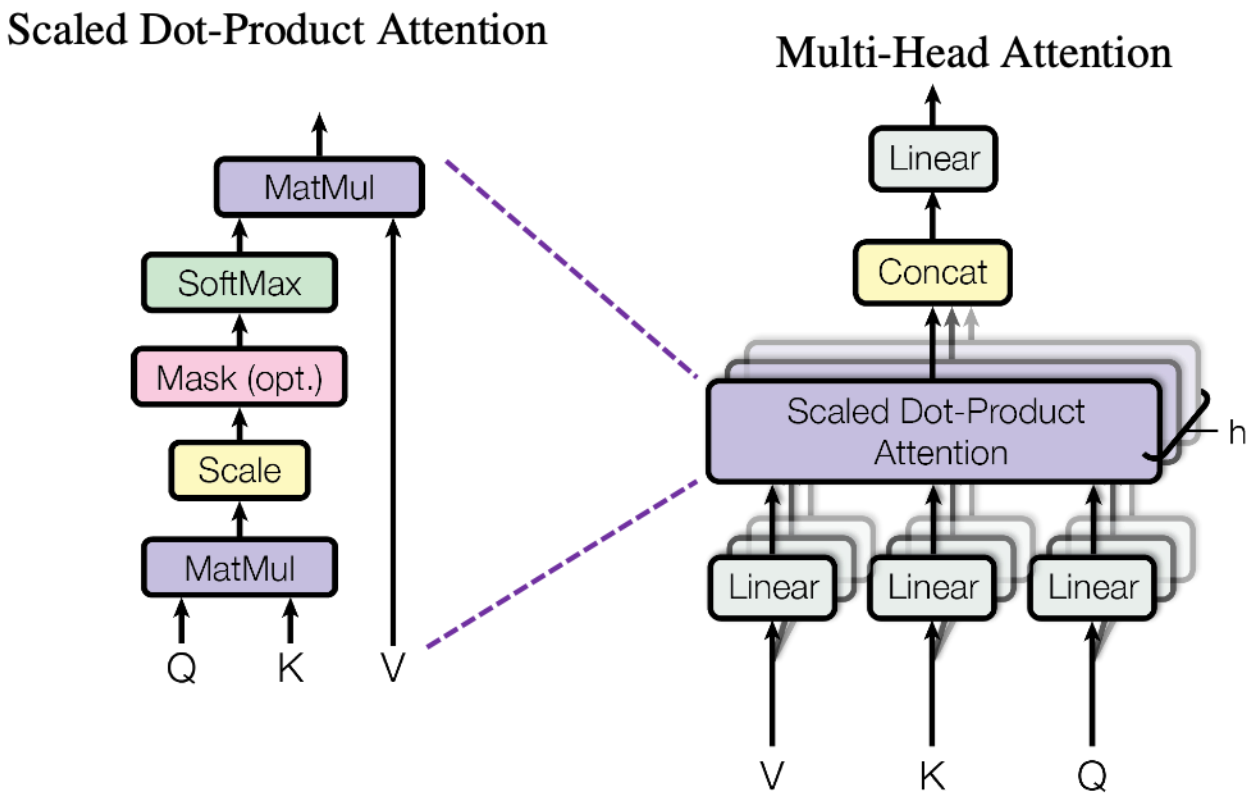
Seq2Seq + Attention



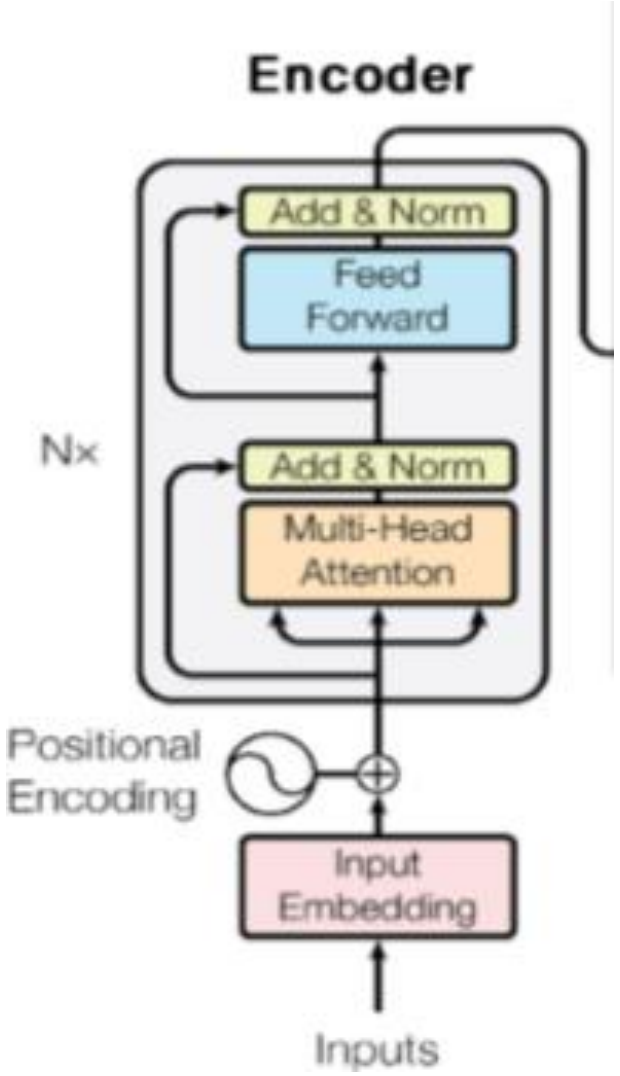
Transformer 등장



Multi-Head Attention

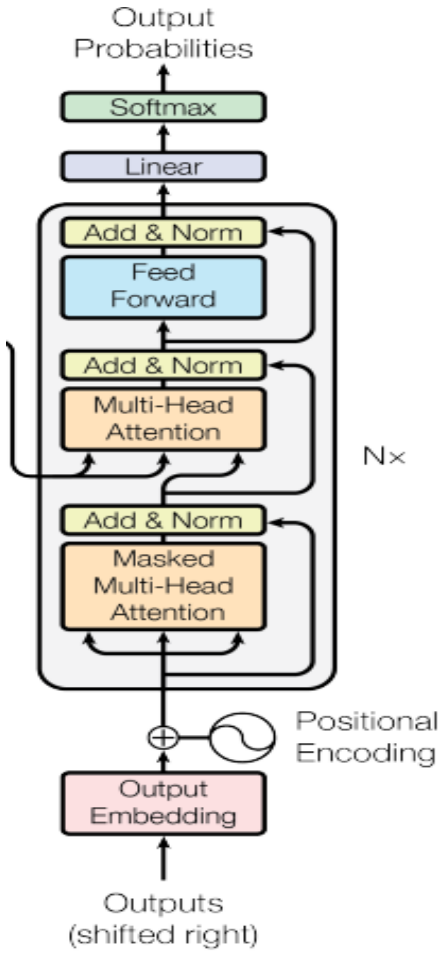


Transformer encoder



- Positional encoding
- 2개의 서브층으로 구성
- (MultiHeadAttention층, Feedforward층)
- 각 서브층 이후 dropout, residual, layer_norm진행
- Padding Mask

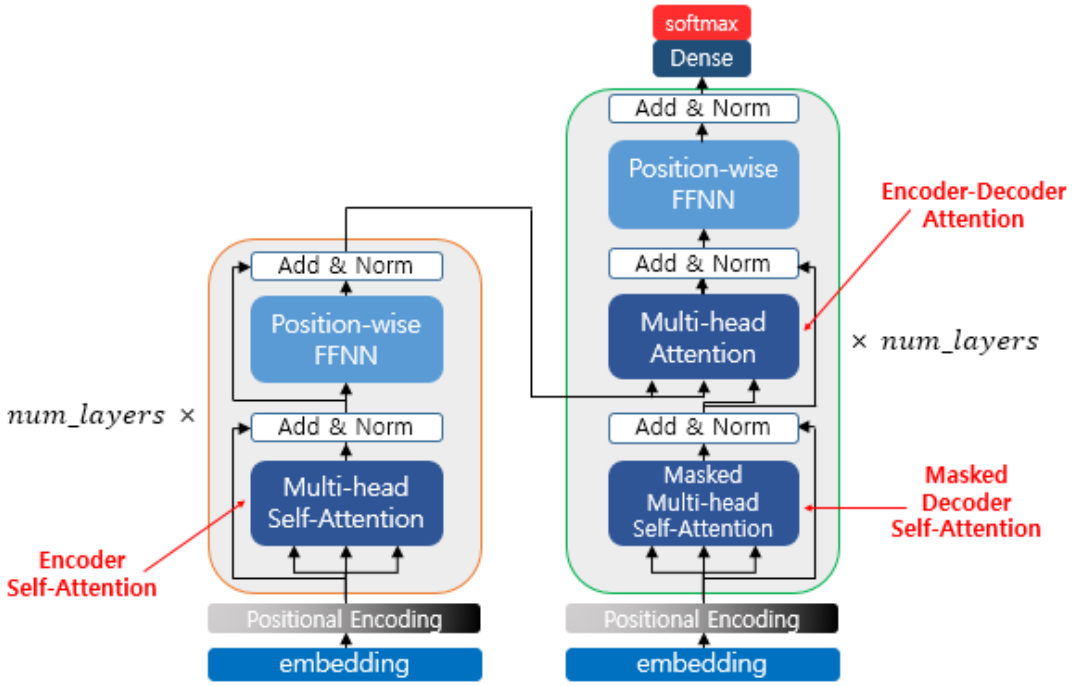
Transformer decoder



- Positional encoding
- 3개의 서브층으로 구성
- 각 서브층 이후 dropout, residual, layer_norm진행
- Look-Ahead Mask
- 인코더의 최종 출력에 대해 cross attention수행

Transformer 3가지 Attention

- 인코더의 self attention
- 디코더의 masked self attention
- 디코더의 인코더-디코더 attention



코드 구현

코드 구현(1)

```
class Transformer(nn.Module):
    # Constructor
    def __init__( self, num_tokens, dim_model, num_heads, num_encoder_layers, num_decoder_layers, dropout_p, ):
        super().__init__()

        # INFO
        self.model_type = "Transformer"
        self.dim_model = dim_model

        # LAYERS
        self.positional_encoder = PositionalEncoding(dim_model=dim_model, dropout_p=dropout_p, max_len=5000)
        self.embedding = nn.Embedding(num_tokens, dim_model)
        self.transformer = nn.Transformer(
            d_model=dim_model,
            nhead=num_heads,
            num_encoder_layers=num_encoder_layers,
            num_decoder_layers=num_decoder_layers,
            dropout=dropout_p,
        )
        self.out = nn.Linear(dim_model, num_tokens)

    def forward(self, src, tgt, tgt_mask=None, src_pad_mask=None, tgt_pad_mask=None):
```

코드 구현(2)

```
def forward(self, src, tgt, tgt_mask=None, src_pad_mask=None, tgt_pad_mask=None):
    # Src, Tgt size 는 반드시 (batch_size, src sequence length) 여야 합니다.

    # Embedding + positional encoding - Out size = (batch_size, sequence length, dim_model)
    src = self.embedding(src) * math.sqrt(self.dim_model)
    tgt = self.embedding(tgt) * math.sqrt(self.dim_model)
    src = self.positional_encoder(src)
    tgt = self.positional_encoder(tgt)

    src = src.permute(1, 0, 2)
    tgt = tgt.permute(1, 0, 2)

    # Transformer blocks - Out size = (sequence length, batch_size, num_tokens)
    transformer_out = self.transformer(src, tgt, tgt_mask=tgt_mask,
                                       src_key_padding_mask=src_pad_mask,
                                       tgt_key_padding_mask=tgt_pad_mask)
    out = self.out(transformer_out)

    return out
```

구현 결과

```
input_sentences = ["나는 28살이다."]
```

```
1/1 [==============================] - 0s 9ms/step
1/1 [==============================] - 0s 9ms/step
1/1 [==============================] - 0s 9ms/step
I am so the store and sometimes and sometimes.......
BLEU score: 2.9083177105737 57
```

Q&A