

Translator 번역기

2조 | 김기목 / 김선규 / 박솔이 / 이상정 / 주원진

Deep Learning Team Project 2020. 04. 23

언어연습

How do you feel today?

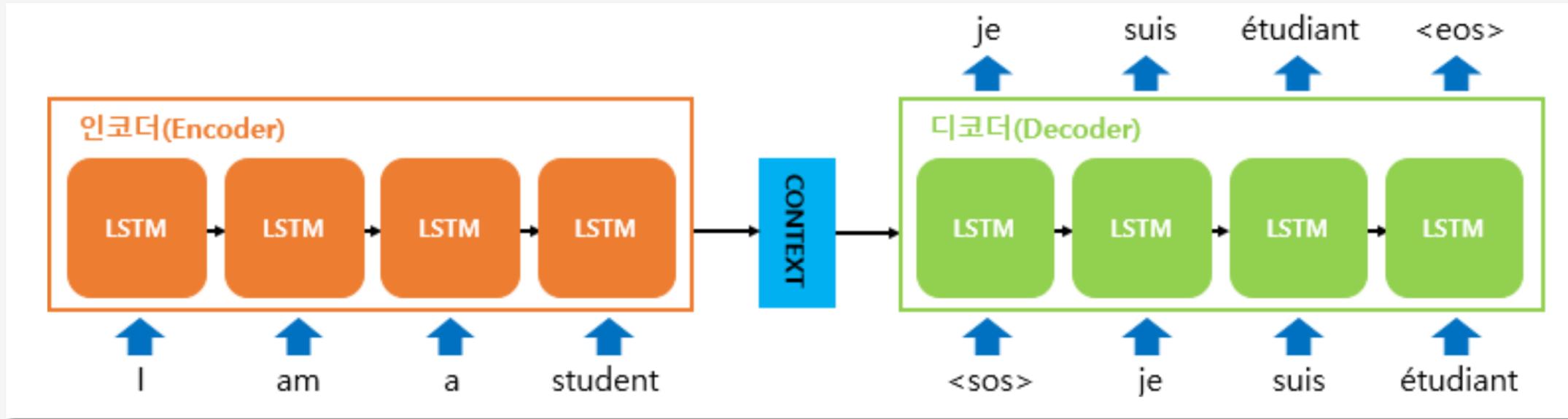
오늘 기분이 어때요?

목차

- 1, Seq To Seq 모델
- 2, Transformer 모델
- 3, 모델 성능 비교
- 4, qnA

Seq To Seq 모델

Seq To Seq 구조



- **LSTM** : RNN 의 코드성능으로 인해 사용
- **CONTEXT Vector**: Hidden Layer

마지막에 등장할 확률이 높은 단어 예측

데이터

◆ 한국어-영어 데이터

- AI Hub, “한국어-영어 번역말뭉치 Sample Data”
- 대화체 및 구어체 data 약 550,000개(55만)

전처리

◆ 순서

① 문장 정리 (정규 표현식 활용)

② 토크나이즈

- 영어 - NLTK Tokenize 활용

- 한국어 - KONLP의 OKT 활용

③ 정수 인코딩

④ 패딩

⑤ 원 핫 인코딩

[문장정리]

- 정규 표현식 - 불필요한 노이즈 제거

```
1 normalized = []
2 for line in en:
3     normal = re.sub(r"[^a-zA-Z]+", " ", line.lower())
4     normalized.append(normal)

1 kor = kor.str.replace("[^ㄱ-ㅎ-ㅣㅋ-ㅌ]", "")
```

- 시작점('₩t')과 종료점('₩n') 추가

```
5344      ₩t 한국에서 카카오톡 할 수 있어요. ₩n
7444      ₩t 이 차는 저 차보다 더 좋아 보이네요. ₩n
1731      ₩t 누군가 요구하기 전에는 도움을 주는 일을 참으세요. ₩n
8719      ₩t 말괄량이 할리퀸들의 역습, 살아있네! ₩n
4521      ₩t 그건 고양증합동장은 인천에 있는 경기장이 아니기 때문이에요. ₩n
.
.
.
5581      ₩t 너 언제 나한테 카페 줄 거야? ₩n
1074      ₩t 가장 순수한 혈통을 지닌 마이들만 가르치도록 하겠어. ₩n
3063      ₩t 마트에서 내가 원하는 펜을 할인받고 살 수 있어요. ₩n
5861      ₩t 나는 오늘 하루 수업을 안 해도 괜찮아요. ₩n
4704      ₩t 풀업한 후에 독일에 가고 싶어서. ₩n
Name: 한국어, Length: 3000, dtype: object
```

토크나이즈

- 한국어는 Konlpy 패키지의 OKT 모듈 사용
- 영어는 NLTK 패키지의 Tokenize 활용
- 토큰화 단어 사전 (인덱스 부여)

✓ 빈도가 높은 것 순으로 내림차순

✓ Ex) 가장 빈도가 높은 단어 = 1

한국어 토큰화

```
[['₩t'],  
['검토'],  
['를'],  
['한'],  
['미후'],  
['에'],  
['고치다'],  
['마하다'],  
['내용'],  
['이'],  
['있다'],  
['해주다'],  
['₩n'],  
['₩t'],  
['미'],  
['것'],  
['은'],  
['을'],  
['홀를하다'],  
['앱'],  
['미지만'],  
['마직'],  
['은'],  
['조금'],  
['더'],  
['보완'],  
['미'],  
['필요하다'],
```

토큰화 단어 사전

```
'areas': 947,  
'illegal': 948,  
'log': 949,  
'trees': 950,  
'kidding': 951,  
'laughing': 952,  
'nauseous': 953,  
'twenty': 954,  
'roasts': 955,  
'beans': 956,  
'strange': 957,  
'million': 958,  
'tons': 959,  
'coal': 960,  
'mined': 961,  
'telegram': 962,  
'saying': 963,  
'uncle': 964,  
'persuaded': 965,  
'policeman': 966,  
'shoot': 967,  
'monkey': 968,  
'pretty': 969,  
'bluffing': 970,  
'vegetarian': 971,  
'various': 972,  
'topics': 973,  
'saving': 974,
```

정수 인코딩

- 단어 사전 활용

- ✓ 단어 사전의 인덱스대로 숫자 할당

- Decoder input에서는 <END> 제거
- 실제 단어(답)에서는 <START> 제거

```
[1, 7, 219, 26, 1175, 15, 7, 1176, 10, 1177],  
[1, 11, 12, 9, 91, 241],  
[1, 16, 463, 188, 70],  
[1, 13, 6, 39, 6, 30, 13, 11, 275, 118, 83],  
[1, 58, 52, 16, 223],  
[1, 18, 29, 1178, 46, 1179],  
[1, 36, 6, 109, 68, 88, 5, 37, 246],  
[1, 3, 268, 6, 69, 54, 5, 567, 37, 140, 479],  
[1, 2, 221, 22, 220, 21, 222, 1501]
```

‘1’은 <START> 을 의미(Decoder input)

```
[3, 98, 37, 1231, 15, 1232, 29, 1233, 2],  
[61, 34, 6, 54, 2],  
[17, 8, 6, 32, 5, 159, 5, 4, 2],  
[4, 65, 8, 28, 23, 2],  
[20, 6, 237, 382, 37, 386, 2],  
[3, 29, 1234, 1235, 2],  
[16, 20, 66, 2],  
[3, 39, 4, 47, 208, 9, 221, 326, 2],  
[16, 1236, 1237, 2],
```

‘2’는 <END> 을 의미(실제 단어)

패딩 및 원 핫 인코딩

- **패딩**

- ✓ 문장의 길이가 모두 다르므로 이를
같게 해주기 위함
- ✓ 가장 긴 문장의 길이를 기준으로

- **원 핫 인코딩**

-> 모두 Tensorflow 활용

```
1 from tensorflow.keras.preprocessing.sequence import pad_sequences
2 encoder_input = pad_sequences(encoder_input, maxlen=max_po_len, padding='post')
3 decoder_input = pad_sequences(decoder_input, maxlen=max_en_len, padding='post')
4 decoder_target = pad_sequences(decoder_target, maxlen=max_en_len, padding='post')
```

```
1 encoder_input.shape
```

```
(1200, 96)
```

```
1 decoder_input.shape
```

```
(1200, 93)
```

```
1 decoder_target.shape
```

```
(1200, 93)
```

```
1 from tensorflow.keras.utils import to_categorical
2 encoder_input = to_categorical(encoder_input)
3 decoder_input = to_categorical(decoder_input)
4 decoder_target = to_categorical(decoder_target)
```

```
1 encoder_input.shape
```

```
(1200, 96, 1724)
```

```
1 decoder_input.shape
```

```
(1200, 93, 1369)
```

모델 구성 및 결과

• 모델 구성

✓ LSTM 사용

→ 결과는 매우 좋지 않았음

```
1 from tensorflow.keras.layers import Input, LSTM, Embedding, Dense
2 from tensorflow.keras.models import Model
3
4 encoder_inputs = Input(shape=(None, num_encoder_tokens))
5 encoder_lstm = LSTM(units=256, return_state=True)
6 encoder_outputs, state_h, state_c = encoder_lstm(encoder_inputs)
7 # encoder_outputs도 같이 리턴받기는 했지만 여기서는 필요없으므로 이 값은 버림.
8 encoder_states = [state_h, state_c]
9 # LSTM은 바닐라 RNN과는 달리 상태가 두 개. 바로 은닉 상태와 셀 상태.
```

```
1 decoder_inputs = Input(shape=(None, num_decoder_tokens))
2 decoder_lstm = LSTM(units=256, return_sequences=True, return_state=True)
3 decoder_outputs, _, _ = decoder_lstm(decoder_inputs, initial_state=encoder_states)
4 # 디코더의 첫 상태를 인코더의 은닉 상태, 셀 상태로 합니다.
5 decoder_softmax_layer = Dense(num_decoder_tokens, activation='softmax')
6 decoder_outputs = decoder_softmax_layer(decoder_outputs)
7
8 model = Model([encoder_inputs, decoder_inputs], decoder_outputs)
9 model.compile(optimizer="rmsprop", loss="categorical_crossentropy")
```

```
1 model.fit(x=[encoder_input, decoder_input], y=decoder_target, batch_size=64, epochs=100, validation_split=0.2)
15/15 [=====] - 18s 1s/step - loss: 0.2229 - val_loss: 0.3962
Epoch 62/100
15/15 [=====] - 18s 1s/step - loss: 0.2189 - val_loss: 0.3912
Epoch 63/100
15/15 [=====] - 18s 1s/step - loss: 0.2154 - val_loss: 0.3940
Epoch 64/100
15/15 [=====] - 18s 1s/step - loss: 0.2111 - val_loss: 0.3939
Epoch 65/100
15/15 [=====] - 18s 1s/step - loss: 0.2078 - val_loss: 0.3950
Epoch 66/100
```

포르투갈어-영어 변환

- 한국어-영어 변환 실패
- 영어와 유사성이 낮은 한국어 대신 **포르투갈어 사용** (ManyThings.org, “Tab-delimited Bilingual Sentence Pairs” / 포르투갈어-영어 pair data 약 150,000개)
- 포르투갈어를 영어로 번역
- 전체적인 과정은 한-영 변환과 동일

- ✓ 포-영 변환에서는 글자 토큰화도 시도
- ✓ 글단위자 토큰화(vs. 단어단위 토큰화)에서 더 좋은 성능을 냄

글자 토큰화

```
1 # 글자 집합 구축
2 src_vocab=set()
3 for line in lines.src: # 1줄씩 읽음
4     for char in line: # 1개의 글자씩 읽음
5         src_vocab.add(char)
6
7 tar_vocab=set()
8 for line in lines.tar:
9     for char in line:
10        tar_vocab.add(char)
```

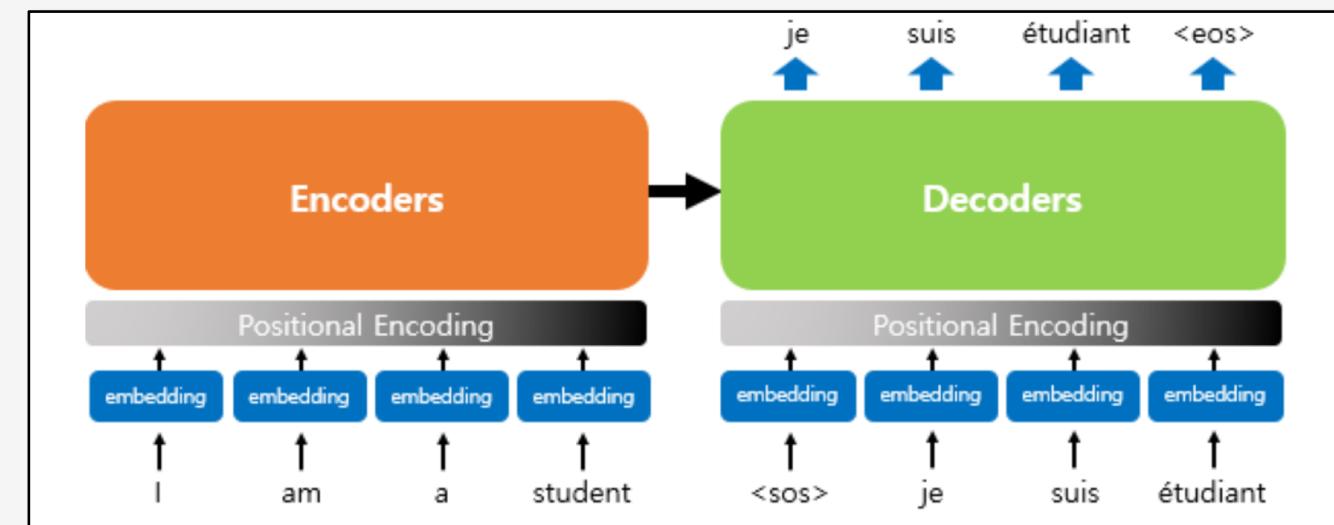
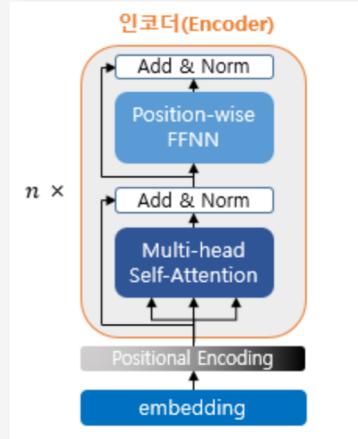
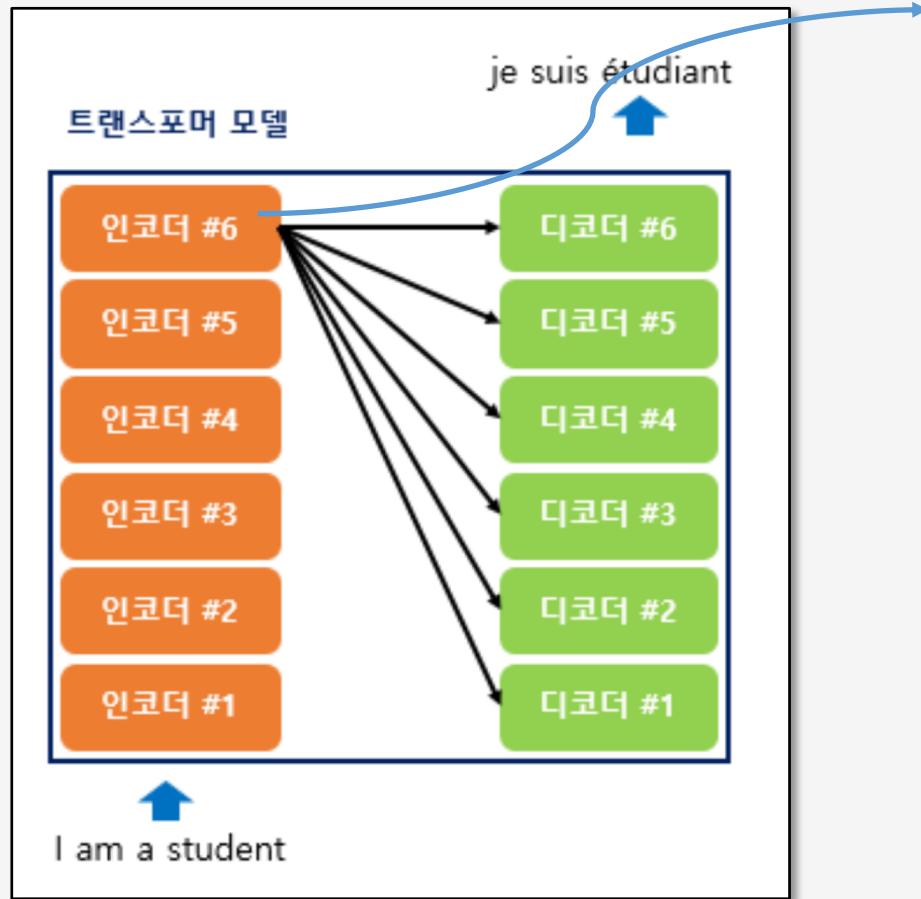
```
['X', 'Y', 'Z', 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i'],
['Y', 'W', 'X', 'Y', 'Z', 'a', 'b', 'c', 'd', 'e', 'f', 'g']
```

```
1 src_to_index = dict([(word, i+1) for i, word in enumerate
2 tar_to_index = dict([(word, i+1) for i, word in enumerate
3 print(src_to_index)
4 print(tar_to_index)

{' ': 1, '!': 2, "'": 3, '$': 4, '%': 5, '"': 6, ',': 7, '-':
{'₩': 1, '₩n': 2, '₩₩': 3, '₩₩': 4, '"': 5, '$': 6, '%': 7, '"':
```

Transformer 모델

Transformer 구조



입력 파이프 라인 설정

Data Set

- TFDS 를 사용, TED Talks Open Translation Project에서 포르투갈어-영어 번역 데이터 세트를 로드
- 데이터 세트에는 약 50,000개의 Training 예제, 1100개의 Validation 예제 및 2000개의 Test 예제 포함

[데이터 전처리] Training 데이터 세트에서 맞춤 하위 단어 1) tokenizer를 만듬

```
[] tokenizer_en = tfds.features.text.SubwordTextEncoder.build_from_corpus(  
    (en.numpy() for pt, en in train_examples), target_vocab_size=2**13)  
  
tokenizer_pt = tfds.features.text.SubwordTextEncoder.build_from_corpus(  
    (pt.numpy() for pt, en in train_examples), target_vocab_size=2**13)
```

2) 입력 및 대상에 시작 및 종료 토큰을 추가

```
def encode(lang1, lang2):  
    lang1 = [tokenizer_pt.vocab_size] + tokenizer_pt.encode(  
        lang1.numpy()) + [tokenizer_pt.vocab_size+1]  
  
    lang2 = [tokenizer_en.vocab_size] + tokenizer_en.encode(  
        lang2.numpy()) + [tokenizer_en.vocab_size+1]  
  
    return lang1, lang2
```

```
for ts in tokenized_string:  
    print ('{} ----> {}'.format(ts, tokenizer_en.decode([ts])))
```

7915	---->	T
1248	---->	ran
7946	---->	s
7194	---->	former
13	---->	is
2799	---->	awesome
7877	---->	.

3) Masking (데이터 전처리)

- 일련의 순서로 모든 패드 토큰을 Mask
- 모델이 패딩을 입력으로 처리하지 않도록 함
- Mask는 패드 값 0이 있는 위치를 나타냄
- 해당 위치에서 1을 출력하고 그렇지 않으면 0을 출력

```
[ ] def create_padding_mask(seq):
    seq = tf.cast(tf.math.equal(seq, 0), tf.float32)

    # add extra dimensions to add the padding
    # to the attention logits.
    return seq[:, tf.newaxis, tf.newaxis, :] # (batch_size, 1, 1, seq_len)
```

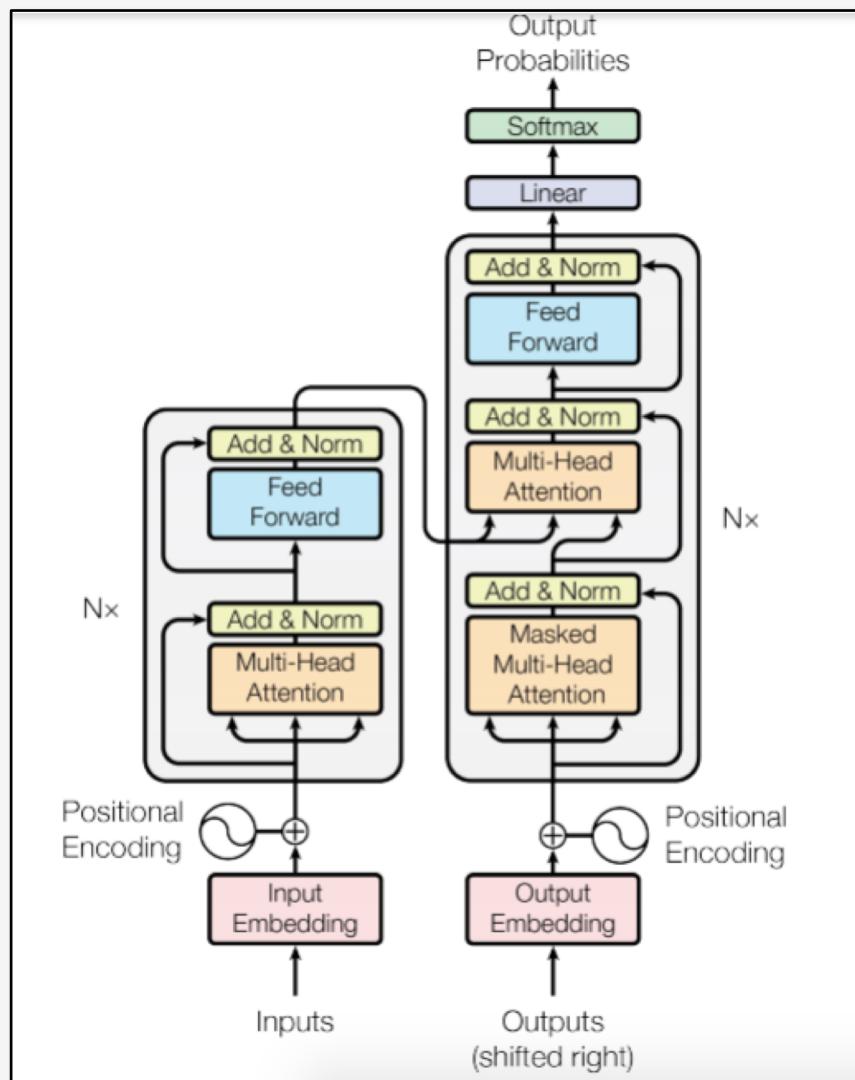
▶ x = tf.constant([[7, 6, 0, 0, 1], [1, 2, 3, 0, 0], [0, 0, 0, 4, 5]])
create_padding_mask(x)

⠃ <tf.Tensor: shape=(3, 1, 1, 5), dtype=float32, numpy=
array([[[[0., 0., 1., 1., 0.]]],

 [[[0., 0., 0., 1., 1.]]],

 [[[1., 1., 1., 0., 0.]]]], dtype=float32>

Encoder and decoder: self-attention



Encoder

```
[] sample_encoder_layer = EncoderLayer(512, 8, 2048)
```

```
sample_encoder_layer_output = sample_encoder_layer(  
    tf.random.uniform((64, 43, 512)), False, None)
```

```
sample_encoder_layer_output.shape # (batch_size, input_seq_len, d_model)
```

TensorShape([64, 43, 512])

Decoder

```
[] sample_decoder_layer = DecoderLayer(512, 8, 2048)
```

```
sample_decoder_layer_output, _ = sample_decoder_layer(  
    tf.random.uniform((64, 50, 512)), sample_encoder_layer_output,  
    False, None, None)
```

```
sample_decoder_layer_output.shape # (batch_size, target_seq_len, d_model)
```

TensorShape([64, 50, 512])

모델 성능 비교

BLEU

- BLEU(Bilingual Evaluation Understudy)
 - 기계 번역 결과와 사람이 직접 번역한 결과를 비교하여 번역에 대한 성능을 측정하는 방법
 - “모델이 번역한 문장이 여러 번역가가 실제 번역한 글들과의 상관성을 측정한 것”

- 특징

- 점수가 높을 수록 성능이 좋음
- 언어에 구애 받지 않고 사용 가능

BLEU 점수	해석
10점 미만	거의 의미 없음
10~19점	핵심을 파악하기 어려움
20~29점	요점은 명확하지만 많은 문법적 오류가 있음
30~40점	이해할 수 있는 양호한 번역
40~50점	고품질 번역
50~60점	매우 우수한 품질의 적절하고 유창한 번역
60점 초과	대체적으로 사람보다 우수한 품질

결과 및 성능 비교 (포어-영어 번역기)

정답 문장	seq2seq	Transformer	BLEU(seq2seq)	BLEU(Transformer)
Could you please turn on the heat?	Could you please tell me what's going on?	would i get a flap up of the heat , please ?	0.3656	0.6389
Don't throw away a good opportunity.	Don't throw away a good or uset here.	you get cheese out a good opportunity .	0.5170	0.4347
We've got a big day ahead of us tomorrow.	Let's see if we can get the gate of more.	We fixed a long day to have a three-dimensional day .	0.5623	0.6530
I don't know when he came back from France.	I don't know where Tom had to do that.	i get on my way that he returned to the sovereign questions .	0.3303	0.5266
This book is very good except for a few mistakes.	This book is not about three housre.	fifty book have been so good , except for some errors .	0.2678	0.4367

✓ 기계 번역 결과와 실제 번역 문장(100개)의 BLEU 평균 점수
트랜스포머의 성능이 높음



BLEU(seq2seq)	BLEU(Transformer)
0.42	0.45

BLEU score 세부정보

✓ BLEU score 수식은

축약 패널티(Brevity Penalty)와
N-그램 중복(N-gram Overlap)으로 구성됨

✓ 축약 패널티

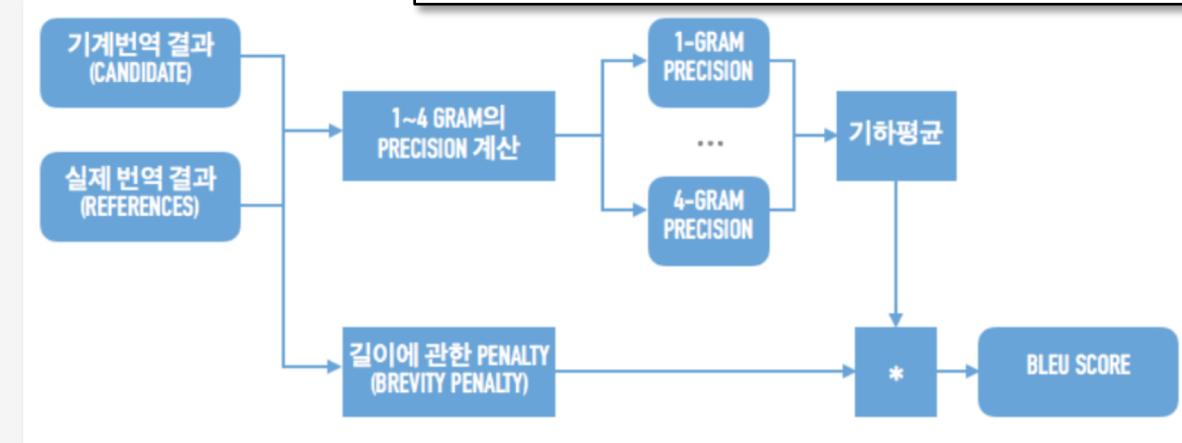
- 지나치게 짧게 생성된 번역에 패널티 적용

✓ N-gram 중복

- 참조 번역에서 유니그램, 바이그램, 트라이그램, 4-gram($i=1, \dots, 4$)이 그에 해당하는 N-그램과의 일치 정도 측정
- 과잉 계산 방지를 위해 참조에서 발생하는 최대 N-그램 수에 맞게 N-그램 수를 자름

$$\text{BLEU} = \underbrace{\min\left(1, \exp\left(1 - \frac{\text{reference-length}}{\text{output-length}}\right)\right)}_{\text{brevity penalty}} \underbrace{\left(\prod_{i=1}^4 \text{precision}_i\right)^{1/4}}_{\text{n-gram overlap}}$$

$$\text{precision}_i = \frac{\sum_{\text{snt} \in \text{Cand-Corpus}} \sum_{i \in \text{snt}} \min(m_{\text{cand}}^i, m_{\text{ref}}^i)}{w_t^i} = \sum_{\text{snt}' \in \text{Cand-Corpus}} \sum_{i' \in \text{snt}'} m_{\text{cand}}^{i'}$$



Q & A