Deep Learning Model for Natural Language Translation

Seonkyu Kim

Gimok Kim

Sol-yi Park

Sangjeong Lee

Wonjin Joo

Language Translation

How do you feel today?

오늘 기분이 어때요?

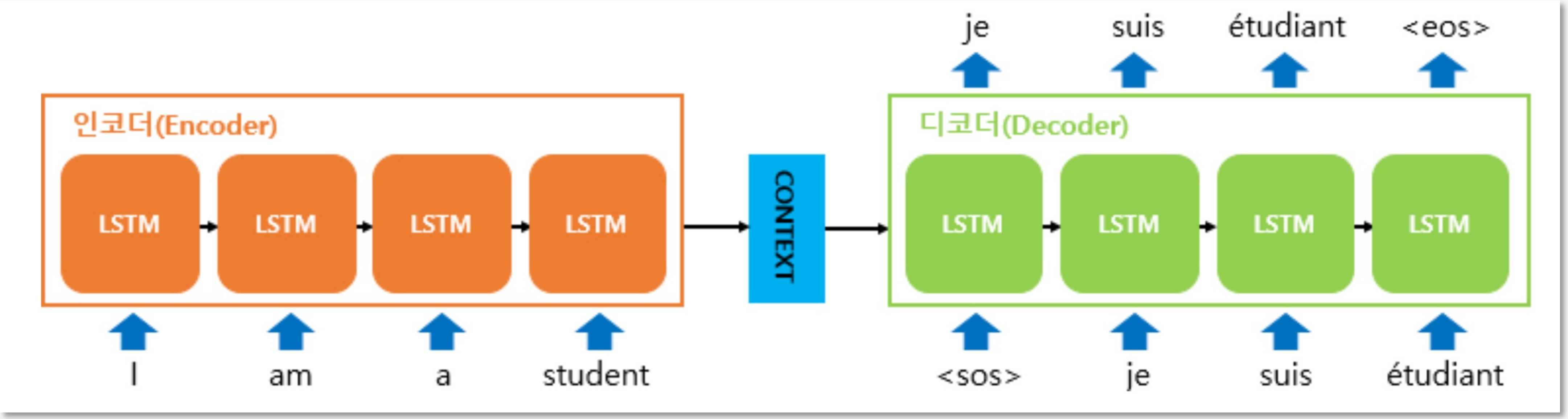
Como você se sente hoje?

Agenda

1. Seq To Seq Model
2. Transformer Model
3. Model Comparison
4. QnA

Seq To Seq Model

Seq to seq structure predicts words that are likely to appear next.



- **LSTM** : Used for performance of RNN
- **CONTEXT Vector**: Hidden Layer

Korean-English Dataset

- AI Hub, "Korean-English Corpus Sample Data"
- 550K sets of conversational and colloquial data

Preprocessing

1. Removing unnecessary noise using Regular Expression
2. Tokenization
3. Integer Encoding
4. Padding
5. One Hot Encoding
6. Adding start token and end token

Remove unnecessary noise

```python
normalized = []
for line in en:
  normal = re.sub(r"[^a-z0-9]+", " ", line.lower())
  normalized.append(normal)
```

```python
kor = kor.str.replace("[^ㄱ-하-ㅣrk-힣]","")
```

Add starting token, ‘\t’, and end token, ‘\n’.

```
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 고양종합운동장은 인천에 있는 경기장이 아니기 때문이에요. ₩n
                            ...
5581                  ₩t 너 언제 나한테 카페 줄 거야? ₩n
1074          ₩t 가장 순수한 혈통을 지닌 아이들만 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 졸업한 후에 독일에 가고 싶어서. ₩n
Name: 한국어, Length: 3000, dtype: object
```

Tokenization

- Korean—using OKT module of Konlpy
- English—using Tokenize of NLTK
- Generate a dictionary for tokenized vocabulary (with index)
  - Sort from highest frequency
  - e.g., Highest frequency = 1

Tokenizing Korean

```
['₩t ',
 '검토',
 '를',
 '한',
 '이후',
 '에',
 '고치다',
 '야하다',
 '내용',
 '이',
 '있다',
 '해주다',
 '₩n'],
['₩t ',
 '이',
 '것',
 '은',
 '훌륭하다',
 '앱',
 '이지만',
 '아직',
 '은',
 '조금',
 '더',
 '보완',
 '이',
 '필요하다',
```

Dictionary for Tokenized Vocabulary

```
'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,
```

Integer Encoding

- Using the dictionary
  - Allocate numbers according to the index of each vocabulary
- Remove <END> from the decoder input.
- Remove <START> from the actual vocabulary (i.e., the answer).

```
[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’ means <START> (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’ means <END> (Actual vocabulary)

Padding and One Hot Encoding

- **Padding**
  - To make the length of sentences uniform
  - Using the longest sentences as the standard
- **One Hot Encoding**
  - Using Tensorflow

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

```python
encoder_input.shape
```

```
(1200, 96)
```

```python
decoder_input.shape
```

```
(1200, 93)
```

```python
decoder_target.shape
```

```
(1200, 93)
```

```python
from tensorflow.keras.utils import to_categorical
encoder_input = to_categorical(encoder_input)
decoder_input = to_categorical(decoder_input)
decoder_target = to_categorical(decoder_target)
```

```python
encoder_input.shape
```

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

```python
decoder_input.shape
```

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

LSTM showed poor performance.

```python
from tensorflow.keras.layers import Input, LSTM, Embedding, Dense
from tensorflow.keras.models import Model

encoder_inputs = Input(shape=(None, num_encoder_tokens))
encoder_lstm = LSTM(units=256, return_state=True)
encoder_outputs, state_h, state_c = encoder_lstm(encoder_inputs)
# encoder_outputs도 같이 리턴받기는 했지만 여기서는 필요없으므로 이 값은 버림.
encoder_states = [state_h, state_c]
# LSTM은 바닐라 RNN과는 달리 상태가 두 개. 바로 은닉 상태와 셀 상태.
```

```python
decoder_inputs = Input(shape=(None, num_decoder_tokens))
decoder_lstm = LSTM(units=256, return_sequences=True, return_state=True)
decoder_outputs, _, _= decoder_lstm(decoder_inputs, initial_state=encoder_states)
# 디코더의 첫 상태를 인코더의 은닉 상태, 셀 상태로 합니다.
decoder_softmax_layer = Dense(num_decoder_tokens, activation='softmax')
decoder_outputs = decoder_softmax_layer(decoder_outputs)

model = Model([encoder_inputs, decoder_inputs], decoder_outputs)
model.compile(optimizer="rmsprop", loss="categorical_crossentropy")
```

```
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
```

Translating Portuguese-English

- Use Portuguese instead of Korean for its similarity to English (ManyThings.org, “Tab-delimited Bilingual Sentence Pairs” / 150K sets of Portuguese-English data pair)
- Translate Portuguese to English
- Use the same process as in the Korean-English translation
    - Character tokenization has been added
    - Character tokenization showed better performance.

Character Tokenization

```
# 글자 집합 구축
src_vocab=set()
for line in lines.src: # 1줄씩 읽음
    for char in line: # 1개의 글자씩 읽음
        src_vocab.add(char)

tar_vocab=set()
for line in lines.tar:
    for char in line:
        tar_vocab.add(char)
```

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

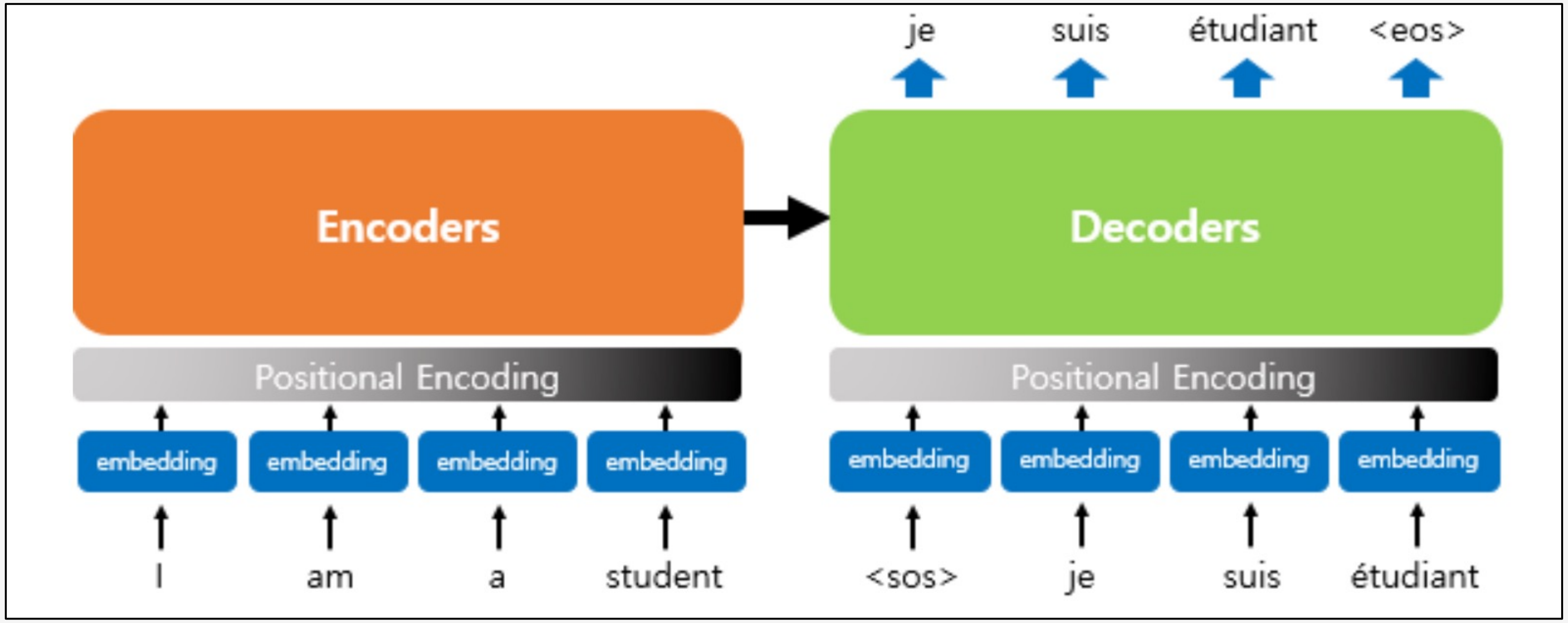
```
src_to_index = dict([(word, i+1) for i, word in enumerate
tar_to_index = dict([(word, i+1) for i, word in enumerate
print(src_to_index)
print(tar_to_index)
```

```
{' ': 1, '!': 2, '"': 3, '$': 4, '%': 5, "'": 6, ',': 7, '-'
{'₩t': 1, '₩n': 2, ' ': 3, '!': 4, '"': 5, '$': 6, '%': 7, "
```

Transformer Model

Transformer structure adds positional information.

- **Positional Encoding**



Setting the Input Pipeline

- Load Portuguese-English dataset from TED Talks Open Translation Project, using TFDS
- The dataset includes 50K training examples, 1.1K validation examples, and 2K Test examples
- Generate customized sub-word tokenizer from the training dataset.

```
[ ]  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)
```

```
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 ----> .
```

- Add start and end tokens to the input and target

```
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
```

Masking

- Mask all pad tokens in sequential order.
- Prevent the model from processing padding as input.
- Mask indicates where pad value 0 is located.
- Print 1 at the pad value 0 location, other wise print 0.

```
[ ] 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)
```

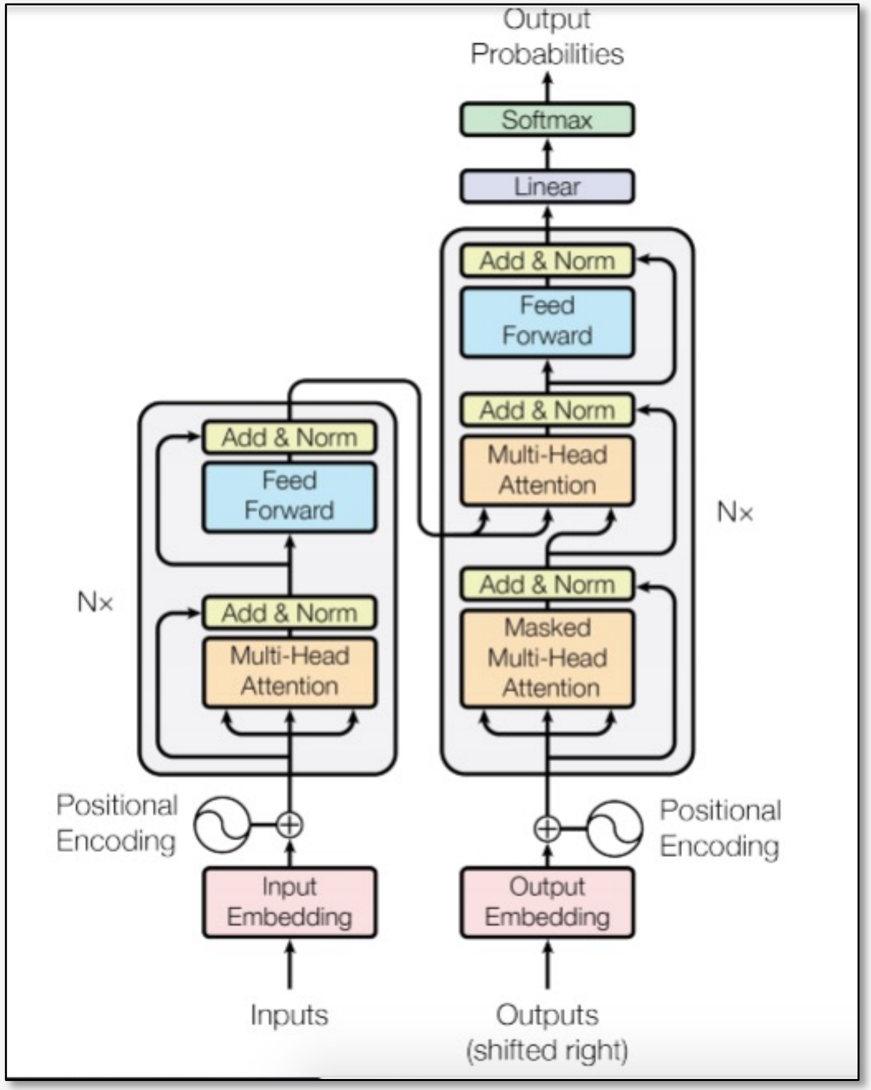
```
x = tf.constant([[7, 6, 0, 0, 1], [1, 2, 3, 0, 0], [0, 0, 0, 4, 5]])
create_padding_mask(x)
```

```
<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



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])
```

Model Comparison

BLEU

- **BLEU(Bilingual Evaluation Understudy)**

- Measures translation performance by comparing outcomes of machine translation and human translation
  - “measuring correlation between the sentence translated by the model versus sentences actually translated by human translators”
- Higher score indicates better performance.
- Can be used regardless of language

BLEU Score	Interpretation
<10	Almost meaningless
10–19	Difficult to get to the point
20–29	Clear to the point but with many grammatical errors
30–40	Good, understandable translation
40–50	High-quality translation
50–60	Appropriate, fluent translation of very good quality
60<	Generally better than human

BLEU score formula consists of brevity penalty and n-gram overlap.

- **Brevity Penalty:** Penalizes translations that are too short
- **N-gram Overlap**
  - Measures the degree to which n-grams (n = 1, …, 4) in the candidate translation match n-grams in the reference translation
  - The number of n-grams is limited to the maximum number of n-grams generated from the reference to prevent over-calculation.

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

$$precision_i = \frac{\sum_{\text{snt}\in\text{Cand-Corpus}} \sum_{i\in\text{snt}} \min(m^i_{cand}, m^i_{ref})}{w^i_t = \sum_{\text{snt'}\in\text{Cand-Corpus}} \sum_{i'\in\text{snt'}} m^{i'}_{cand}}$$

Transformer showed better performance than seq to seq.

Correct Answer	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 hourse.	fifty book have been so good , except for some errors .	0.2678	0.4367

We compared 100 machine translated sentences with the correct sentences to generate the average BLEU score.

BLEU(seq2seq)	BLEU(Transformer)
0.42	**0.45**