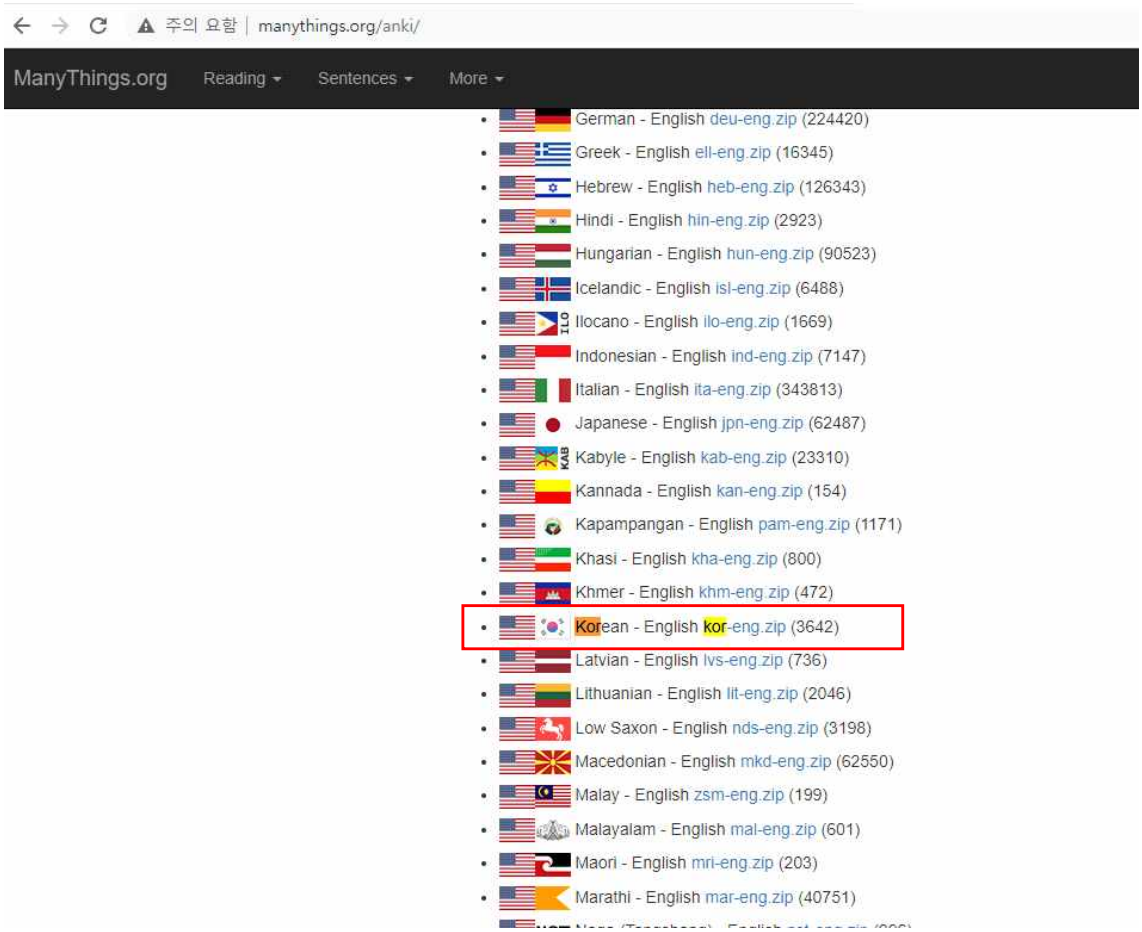
딥러닝 기말과제

20152658 빅데이터경영통계전공 원동욱

1. 데이터 수집

교수님이 주신 데이터와 'manythings' 사이트의 데이터를 활용



2. 전처리

2-1. 외부 내부 공통 데이터 전처리

```
def unicode_to_ascii(s):
    return ''.join(c for c in unicodedata.normalize('NFD', s)
       if unicodedata.category(c) != 'Mn')
```

```
def preprocess_sentence(w):
    w = unicode_to_ascii(w.lower().strip())

    w = re.sub(r"([?.!,])", r" ₩1 ", w)
    w = re.sub(r'[" "]+', " ", w)

    w = re.sub(r"[^a-zA-Z[가-힣]?.!]+", " ", w)
    w = re.sub('[-=+,#/₩:^$@*₩"※~&%・『₩₩ '|₩(₩)₩[₩]₩<₩>`₩'…》]', '', w)

    w = w.strip()

    w = '<s> ' + w + ' </s>'
    return w
```

<단어와 구두점 사이에 공백을 만들고 특수문자는 정규표현식으로 제거>
<시작을 의미하는 토큰 <s>와 종료를 의미하는 토큰 </s> 추가>

2-2. 외부 데이터 전처리

```
encoder_input, decoder_input = [], []

with open('kor.txt','r',encoding = 'utf-8') as lines:
    for i, line in enumerate(lines):

            # source 데이터와 target 데이터 분리
            src_line, tar_line, _ = line.strip().split('₩t')

            # source 데이터 전처리
            src_line_input = [w for w in preprocess_sentence(src_line).split()]

            # target 데이터 전처리
            tar_line_input = [w for w in preprocess_sentence(src_line).split()]

            encoder_input.append(src_line_input)
            decoder_input.append(tar_line_input)
```

<영어 데이터와 한글 데이터를 분리하여 따로 전처리 수행>

```
d  = [' '.join(decoder_input[i]) for i in range(len(decoder_input))]
e = [' '.join(encoder_input[i]) for i in range(len(encoder_input))]
```

```
d[:5]
```

```
['<s> 가 . </s>',
 '<s> 안녕 . </s>',
 '<s> 뛰어 ! </s>',
 '<s> 뛰어 . </s>',
 '<s> 누구 ? </s>']
```

```
e[:5]
```

```
['<s> go . </s>',
 '<s> hi . </s>',
 '<s> run ! </s>',
 '<s> run . </s>',
 '<s> who ? </s>']
```

<나뉘어진 단어 join을 수행하여 문장으로 구성>

```
new
```

	eng	kor
0	<s> go . </s>	<s> 가 . </s>
1	<s> hi . </s>	<s> 안녕 . </s>
2	<s> run ! </s>	<s> 뛰어 ! </s>
3	<s> run . </s>	<s> 뛰어 . </s>
4	<s> who ? </s>	<s> 누구 ? </s>
...	...	...
3637	<s> science fiction has undoubtedly been the i...	<s> 공상 과학 소설은 의심의 여지 없이 ᄋ...
3638	<s> i started a new blog . ill do my best not ...	<s> 난 블로그를 시작했어 . 블로그를 초ᄇ...
3639	<s> i think its a shame that some foreign lang...	<s> 몇몇 외국어 선생님이 한 번도 원어...
3640	<s> if someone who doesnt know your background...	<s> 만일 네 사정도 잘 모르는 사람이 원ᄋ...
3641	<s> doubtless there exists in this world preci...	<s> 의심의 여지 없이 세상에는 어떤 남자ᄋ...

<하나의 DataFrame 형성>

2-3. 내부 데이터 전처리

		sentences = [
("He came to London by way of Siberia."	"그 사람은 시베리아를 거쳐서 런던에 왔어.")	NaN
("Is there bus service to the airport?"	"공항까지 가는 버스가 있나요?")	NaN
("Japan and South Korea are neighbors."	"일본과 남한은 이웃 국가다.")	NaN
("My favorite singer is Kylie Minogue."	"내가 좋아하는 가수는 카일리 미노그야.")	NaN
	"내가 좋아하는 가수는 카일리 미노그입니다.")	NaN
...	...	...
("I can not answer your question."	"나는 네 질문에 답할 수 없어.")	NaN
("I really hate myself right now."	"바로 지금 난 내 자신이 싫어.")	NaN
("Paris is the capital of France."	"파리는 프랑스의 수도다.")	NaN
("Please take me to the hospital."	"병원으로 데려다 주세요.")	NaN
]	NaN	NaN

<DataFrame 정리>

```
def data_modify(data) :
    df = data.reset_index()

    del df['sentences = [']

    df = df.iloc[:len(df)-1]
    df.rename(columns = {'level_0' : 'eng','level_1' :'kor'},inplace = True)
    return df
```

```
d = d[:1500 - len(df)]
e = e[:1500 - len(df)]
```

```
new = pd.DataFrame(d,e).reset_index().rename(columns = {'index' : 'eng',0 : 'kor'})
```

```
df = pd.concat([df,new]).reset_index()
```

```
del df['index']
```

```
df.head()
```

	eng	kor
0	<s> he came to london by way of siberia . </s>	<s> 그 사람은 시베리아를 거쳐서 런던에 와...
1	<s> is there bus service to the airport ? </s>	<s> 공항까지 가는 버스가 있나요 ? </s>
2	<s> japan and south korea are neighbors . </s>	<s> 일본과 남한은 이웃 국가다 . </s>
3	<s> my favorite singer is kylie minogue . </s>	<s> 내가 좋아하는 가수는 카일리 미노그야 ....
4	<s> my favorite singer is kylie minogue . </s>	<s> 내가 좋아하는 가수는 카일리 미노그입ㄴ...

<학습 속도로 인하여 외부 데이터 포함 1500개만 선정>

3. 토큰화

SentencePiece란?

실무에서 바로 적용 가능하고, BPE를 포함한 기타 서브워드 토크나이징 알고리즘들을 내장
또한, 사전 토큰화 작업없이 단어 분리 토큰화를 수행 → 언어에 종속되지 않는다.

3-1. 영어 SentencePiece

English

```
with open('eng_translation.txt','w',encoding = 'utf8') as f:
    f.write('\n'.join(df['eng']))
```

```
spm.SentencePieceTrainer.Train('--input=eng_translation.txt --model_prefix=english --vocab_size=1600 --user_defined_symbols=<s>,</s>')
```

```
True
```

```
sp_eng = spm.SentencePieceProcessor()
vocab_file = "english.model"
sp_eng.load(vocab_file)
```

```
True
```

<vocab_size는 1600으로 지정>
<시작 · 출력 토큰인 <s>,</s>는 나누어지지 않게 정의>
input에 학습시킬 파일, model_prefix는 모델명, vocab_size는 단어 집합의 크기
user_defined_symbols는 사용자 정의 토큰이다.

```
print(df['eng'][0])
print(sp_eng.encode_as_pieces(df['eng'][0]))
print(sp_eng.encode_as_ids(df['eng'][0]))
```

```
<s> he came to london by way of siberia . </s>
['_', '<s>', '_he', '_came', '_to', '_lon', 'don', '_', 'by', '_', 'way', '_of', '_si', 'ber', 'i', 'a',
'_', '.', '_', '</s>']
[3, 1, 47, 219, 21, 812, 701, 3, 205, 3, 105, 77, 316, 189, 51, 44, 3, 4, 3, 2]
```

<예시>
위의 user_defined_symbols에 시작 출력 토큰을 지정해주었기에 나누어지지 않았다.

3-2. 한글 SentencePiece

Kor

```
with open('kor_translation.txt','w',encoding = 'utf8') as f:
    f.write('\n'.join(df['kor']))
```

```
spm.SentencePieceTrainer.Train('--input=kor_translation.txt --model_prefix=kor --user_defined_symbols=<s>,</s> --vocab_size=2800')
```

```
True
```

```
sp_kor = spm.SentencePieceProcessor()
vocab_file = "kor.model"
sp_kor.load(vocab_file)
```

<vocab_size는 2800으로 지정>

<시작·출력 토큰인 <s>,</s>는 나누어지지 않게 정의>

```
print(df['kor'][900])
print(sp_kor.encode_as_pieces(df['kor'][900]))
print(sp_kor.encode_as_ids(df['kor'][900]))
```

```
<s> 우리가 사과할게 . </s>
['_', '<s>', '_우리가', '_사과', '할게', '_.', '_', '</s>']
[2147, 1, 257, 290, 993, 3, 2147, 2]
```

<예시>

한글 또한 마찬가지

3-3. Padding(패딩)

padding

```
]:
eng_tokenizer = pad_sequences(eng_tokenizer,padding = 'post')
kor_tokenizer = pad_sequences(kor_tokenizer,padding = 'post')
```

```
]:
print('eng_tokenizer의 크기 : {}'.format(eng_tokenizer.shape))
print('kor_tokenizer의 크기 : {}'.format(kor_tokenizer.shape))
```

```
eng_tokenizer의 크기 : (1500, 28)
kor_tokenizer의 크기 : (1500, 21)
```

길이가 가장 긴 문서의 길이를 기준으로 패딩

<Padding 후 Shape>

3-4. 데이터 분리

```
indices = np.arange(eng_tokenizer.shape[0])
np.random.shuffle(indices)
```

```
encoder_input = eng_tokenizer[indices]
decoder_input = kor_tokenizer[indices]
```

```
encoder_train,encoder_test,decoder_train,decoder_test = train_test_split(encoder_input,decoder_input,test
```

```
print(decoder_train.shape)
print(encoder_train.shape)
print(decoder_test.shape)
print(encoder_test.shape)
```

```
(1200, 21)
(1200, 28)
(300, 21)
(300, 28)
```

<적절한 분포를 갖도록 shuffle 진행 후 Train : Test 8:2 비율>

4. 인코더 및 디코더 모델 작성 및 학습

4-1. Dataset 및 모델 구축

```
BUFFER_SIZE = len(encoder_train)
BATCH_SIZE = 64
steps_per_epoch = len(encoder_train)//BATCH_SIZE
embedding_dim = 256
units = 1024
vocab_inp_size = max([max(l) for l in encoder_input])+1
vocab_tar_size = max([max(l) for l in decoder_input])+1

dataset = tf.data.Dataset.from_tensor_slices((encoder_train, decoder_train)).shuffle(BUFFER_SIZE)
dataset = dataset.batch(BATCH_SIZE, drop_remainder=True)
```

```
example_input_batch, example_target_batch = next(iter(dataset))
example_input_batch.shape, example_target_batch.shape
```

```
(TensorShape([64, 28]), TensorShape([64, 21]))
```

4-2 인코더 및 디코더

```
class Encoder(tf.keras.Model):
      def __init__(self, vocab_size, embedding_dim, enc_units, batch_sz):
        super(Encoder, self).__init__()
        self.batch_sz = batch_sz
        self.enc_units = enc_units
        self.embedding = tf.keras.layers.Embedding(vocab_size, embedding_dim)
        self.gru = tf.keras.layers.GRU(self.enc_units,
                                       return_sequences=True,
                                       return_state=True,
                                       recurrent_initializer='glorot_uniform')

      def call(self, x, hidden):
        x = self.embedding(x)
        output, state = self.gru(x, initial_state = hidden)
        return output, state

      def initialize_hidden_state(self):
        return tf.zeros((self.batch_sz, self.enc_units))
```

```
encoder = Encoder(vocab_inp_size, embedding_dim, units, BATCH_SIZE)

# sample input
sample_hidden = encoder.initialize_hidden_state()
sample_output, sample_hidden = encoder(example_input_batch, sample_hidden)
print ('Encoder output shape: (batch size, sequence length, units) {}'.format(sample_output.shape))
print ('Encoder Hidden state shape: (batch size, units) {}'.format(sample_hidden.shape))
```

```
Encoder output shape: (batch size, sequence length, units) (64, 28, 1024)
Encoder Hidden state shape: (batch size, units) (64, 1024)
```

<인코더 설정>

```python
class BahdanauAttention(tf.keras.layers.Layer):
    def __init__(self, units):
        super(BahdanauAttention, self).__init__()
        self.W1 = tf.keras.layers.Dense(units)
        self.W2 = tf.keras.layers.Dense(units)
        self.V = tf.keras.layers.Dense(1)

    def call(self, query, values):
        query_with_time_axis = tf.expand_dims(query, 1)

        score = self.V(tf.nn.tanh(self.W1(query_with_time_axis) + self.W2(values)))

        attention_weights = tf.nn.softmax(score, axis=1)

        context_vector = attention_weights * values
        context_vector = tf.reduce_sum(context_vector, axis=1)

        return context_vector, attention_weights
```

```python
attention_layer = BahdanauAttention(10)
attention_result, attention_weights = attention_layer(sample_hidden, sample_output)

print("Attention result shape: (batch size, units) {}".format(attention_result.shape))
print("Attention weights shape: (batch_size, sequence_length, 1) {}".format(attention_weights.shape))
```

```
Attention result shape: (batch size, units) (64, 1024)
Attention weights shape: (batch_size, sequence_length, 1) (64, 28, 1)
```

<바다나우 어텐션 정의>

어텐션 함수의 Query가 디코더 셀의 t 시점의 은닉 상태가 아니라 t-1시점의 은닉 상태이다.

```python
class Decoder(tf.keras.Model):
    def __init__(self, vocab_size, embedding_dim, dec_units, batch_sz):
        super(Decoder, self).__init__()
        self.batch_sz = batch_sz
        self.dec_units = dec_units
        self.embedding = tf.keras.layers.Embedding(vocab_size, embedding_dim)
        self.gru = tf.keras.layers.GRU(self.dec_units,
                                       return_sequences=True,
                                       return_state=True,
                                       recurrent_initializer='glorot_uniform')

        self.fc = tf.keras.layers.Dense(vocab_size)


        self.attention = BahdanauAttention(self.dec_units)

    def call(self, x, hidden, enc_output):

        context_vector, attention_weights = self.attention(hidden, enc_output)

        x = self.embedding(x)

        x = tf.concat([tf.expand_dims(context_vector, 1), x], axis=-1)

        output, state = self.gru(x)

        output = tf.reshape(output, (-1, output.shape[2]))

        x = self.fc(output)

        return x, state, attention_weights
```

```python
decoder = Decoder(vocab_tar_size, embedding_dim, units, BATCH_SIZE)

sample_decoder_output, _, _ = decoder(tf.random.uniform((BATCH_SIZE, 1)),
                                      sample_hidden, sample_output)

print ('Decoder output shape: (batch_size, vocab size) {}'.format(sample_decoder_output.shape))
```

```
Decoder output shape: (batch_size, vocab size) (64, 2799)
```

<디코더 설정>

4-3 모델

```
optimizer = tf.keras.optimizers.Adam()
loss_object = tf.keras.losses.SparseCategoricalCrossentropy(
    from_logits=True, reduction='none')

def loss_function(real, pred):
    mask = tf.math.logical_not(tf.math.equal(real, 0))
    loss_ = loss_object(real, pred)

    mask = tf.cast(mask, dtype=loss_.dtype)
    loss_ *= mask

    return tf.reduce_mean(loss_)
```

```
checkpoint_dir = './training_checkpoints'
checkpoint_prefix = os.path.join(checkpoint_dir, "ckpt")
checkpoint = tf.train.Checkpoint(optimizer=optimizer,
                                 encoder=encoder,
                                 decoder=decoder)
```

<옵티마이저 및 손실 함수 정의>

```
@tf.function
def train_step(inp, targ, enc_hidden):
    loss = 0
    with tf.GradientTape() as tape:
        enc_output, enc_hidden = encoder(inp, enc_hidden)

        dec_hidden = enc_hidden

        dec_input = tf.expand_dims([sp_kor.piece_to_id('</s>')] * BATCH_SIZE, 1)

        for t in range(1, targ.shape[1]):
            predictions, dec_hidden, _ = decoder(dec_input, dec_hidden, enc_output)

            loss += loss_function(targ[:, t], predictions)

      # using teacher forcing
            dec_input = tf.expand_dims(targ[:, t], 1)

    batch_loss = (loss / int(targ.shape[1]))

    variables = encoder.trainable_variables + decoder.trainable_variables

    gradients = tape.gradient(loss, variables)

    optimizer.apply_gradients(zip(gradients, variables))

    return batch_loss
```

<체크 포인트>

4-4 학습

```
EPOCHS = 30

for epoch in range(EPOCHS):
    start = time.time()

    enc_hidden = encoder.initialize_hidden_state()
    total_loss = 0

    for (batch, (inp, targ)) in enumerate(dataset.take(steps_per_epoch)):
        batch_loss = train_step(inp, targ, enc_hidden)
        total_loss += batch_loss

        if batch % 100 == 0:
            print('Epoch {} Batch {} Loss {:.4f}'.format(epoch + 1,
                                                         batch,
                                                         batch_loss.numpy()))


    print('Epoch {} Loss {:.4f}'.format(epoch + 1,total_loss / steps_per_epoch))

    print('Time taken for 1 epoch {} sec\n'.format(time.time() - start))
```

```
Epoch 27 Batch 0 Loss 0.2151
Epoch 27 Loss 0.2261
Time taken for 1 epoch 32.2049036026001 sec

Epoch 28 Batch 0 Loss 0.1804
Epoch 28 Loss 0.2018
Time taken for 1 epoch 32.409356117248535 sec

Epoch 29 Batch 0 Loss 0.1686
Epoch 29 Loss 0.1767
Time taken for 1 epoch 32.050315380096436 sec

Epoch 30 Batch 0 Loss 0.1299
Epoch 30 Loss 0.1598
Time taken for 1 epoch 32.92041778564453 sec
```

<Loss 값 : 0.1299>

5. 평가

```python
def evaluate(sentence):

    max_length_targ = decoder_input.shape[1]
    max_length_inp = encoder_input.shape[1]

    attention_plot = np.zeros((max_length_targ, max_length_inp))

    sentence = preprocess_sentence(sentence)

    inputs = sp_eng.encode_as_ids(sentence)
    inputs = tf.keras.preprocessing.sequence.pad_sequences([inputs],
                                                           maxlen=max_length_inp,
                                                           padding='post')
    inputs = tf.convert_to_tensor(inputs)

    result = ''

    hidden = [tf.zeros((1, units))]
    enc_out, enc_hidden = encoder(inputs, hidden)

    dec_hidden = enc_hidden
    dec_input = tf.expand_dims([sp_kor.piece_to_id('<sos>')], 0)

    for t in range(max_length_targ):
        predictions, dec_hidden, attention_weights = decoder(dec_input,
                                                             dec_hidden,
                                                             enc_out)

    # storing the attention weights to plot later on
        attention_weights = tf.reshape(attention_weights, (-1, ))
        attention_plot[t] = attention_weights.numpy()

        predicted_id = tf.argmax(predictions[0]).numpy()

        result += sp_kor.decode_ids([int(predicted_id)]) + ' '

        if sp_kor.decode_ids([int(predicted_id)]) == '<eos>':
            return result, sentence, attention_plot

    # the predicted ID is fed back into the model
        dec_input = tf.expand_dims([predicted_id], 0)

    return result, sentence, attention_plot
```

```python
def plot_attention(attention, sentence, predicted_sentence):
    fig = plt.figure(figsize=(10,10))
    ax = fig.add_subplot(1, 1, 1)
    ax.matshow(attention, cmap='viridis')

    fontdict = {'fontsize': 14}

    ax.set_xticklabels([''] + sentence, fontdict=fontdict, rotation=90)
    ax.set_yticklabels([''] + predicted_sentence, fontdict=fontdict)

    ax.xaxis.set_major_locator(ticker.MultipleLocator(1))
    ax.yaxis.set_major_locator(ticker.MultipleLocator(1))

    plt.show()
```

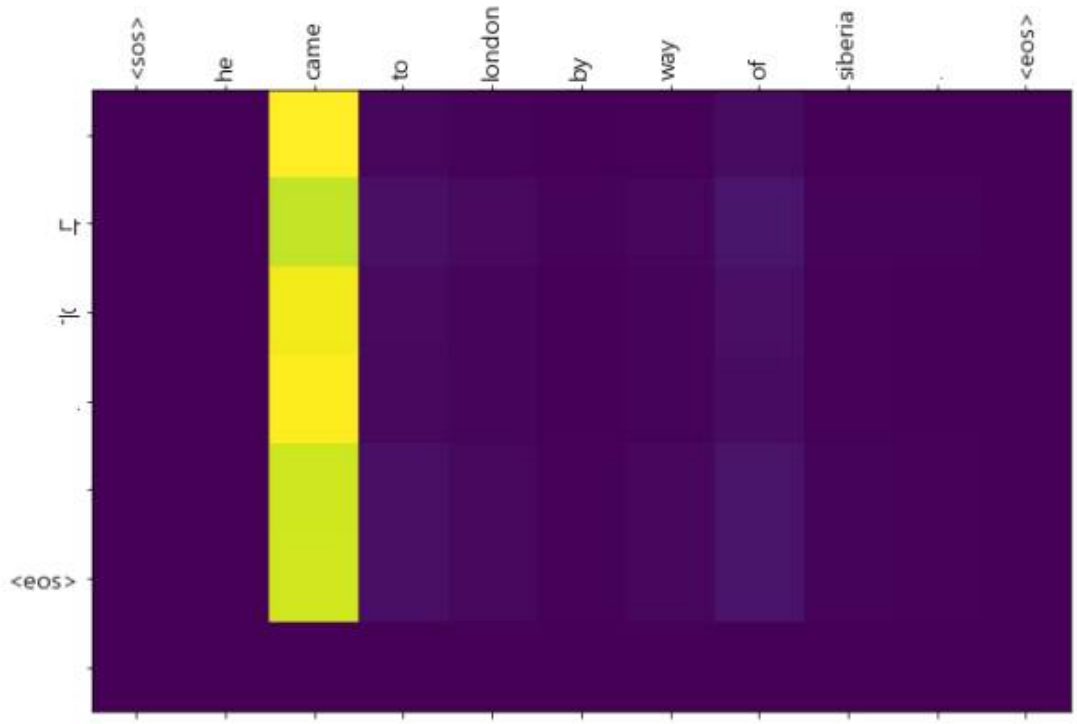
```python
def translate(sentence):
    result, sentence, attention_plot = evaluate(sentence)

    print('Input: %s' % (sentence))
    print('Predicted translation: {}'.format(result))

    attention_plot = attention_plot[:len(result.split(' ')), :len(sentence.split(' '))]
    plot_attention(attention_plot, sentence.split(' '), result.split(' '))
```

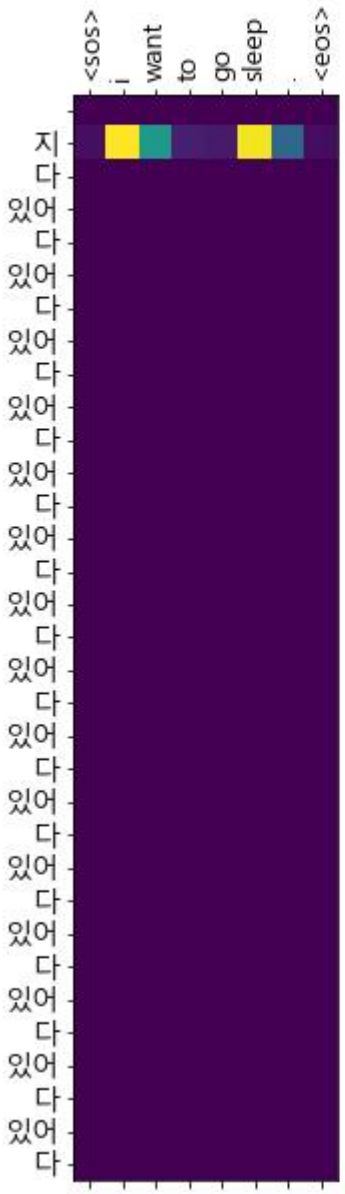
5-1. 모든 데이터를 가지고 한 모델링 예제

```
Input: <sos> he came to london by way of siberia . <eos>
Predicted translation:  다 은 . 는 <eos>
```



```
translate('i want to go sleep.')
```

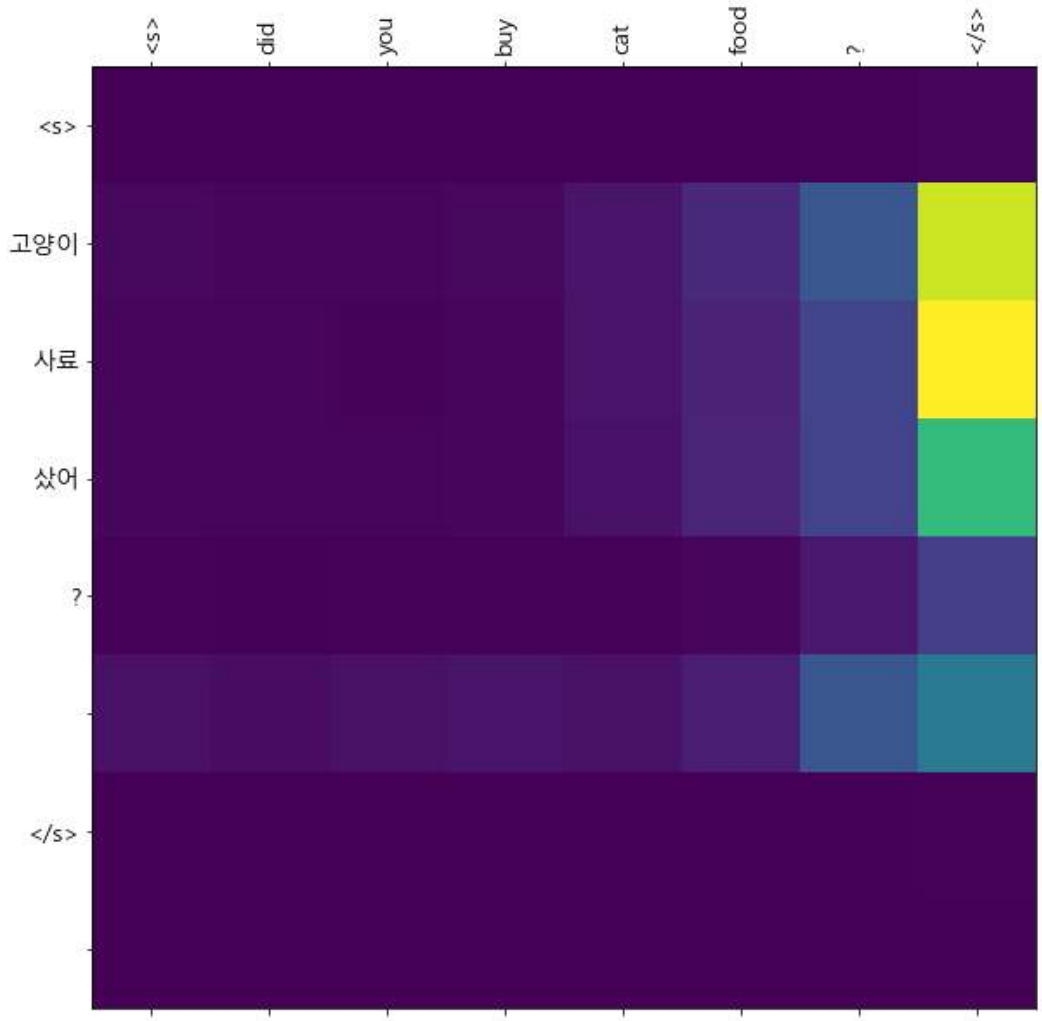
```
Input: <sos> i want to go sleep . <eos>
Predicted translation:  지 다 있어 다 있어 다
다 있어 다 있어 다 있어 다 있어 다 있어 다 있
```



5-2. 잘 된 예제

```
translate(u'did you buy cat food ?')
```

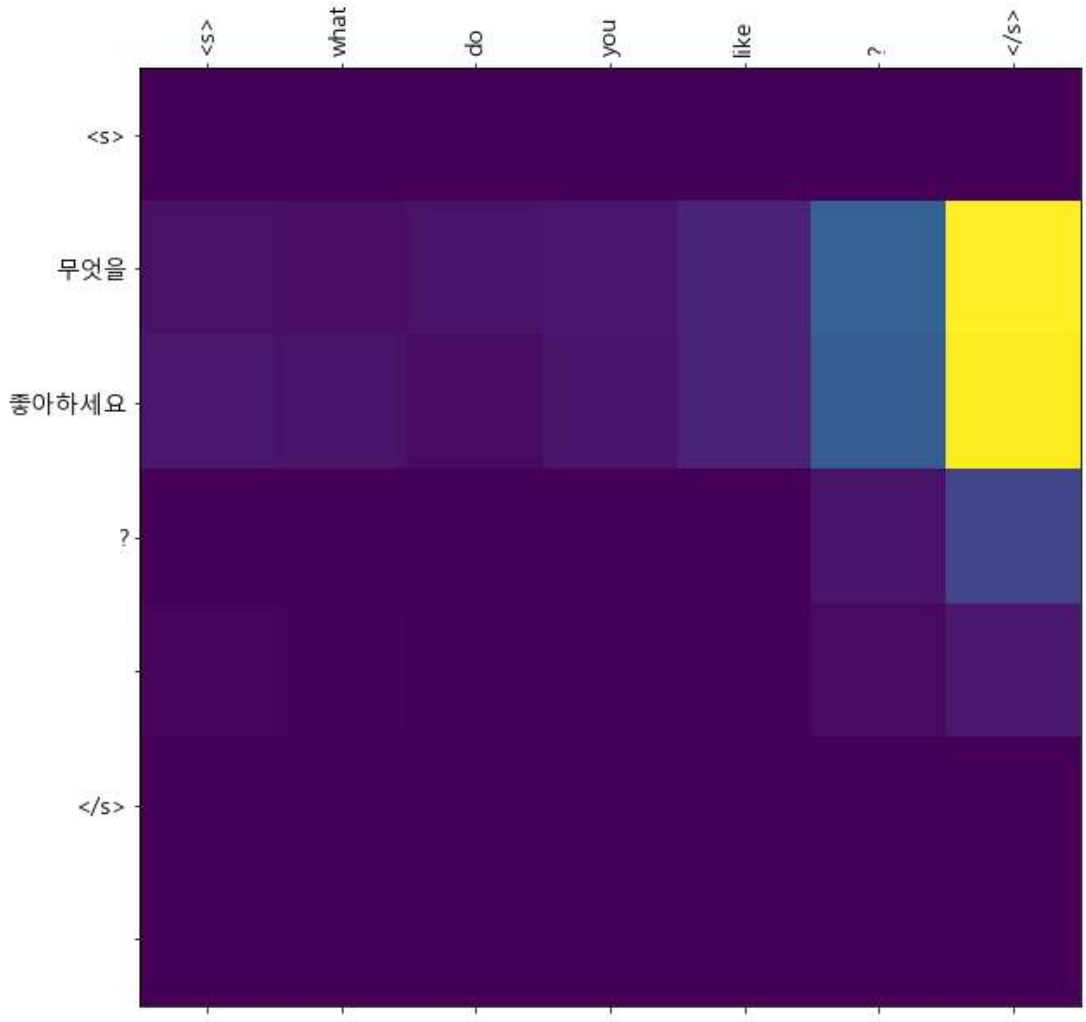
```
Input: <s> did you buy cat food ? </s>
Predicted translation: <s> 고양이 사료 샀어 ?  </s>
```



<복원 테스트>

```
1 translate(u'what do you like ?')
```

```
Input: <s> what do you like ? </s>
Predicted translation: <s> 무엇을 좋아하세요 ?  </s>
```

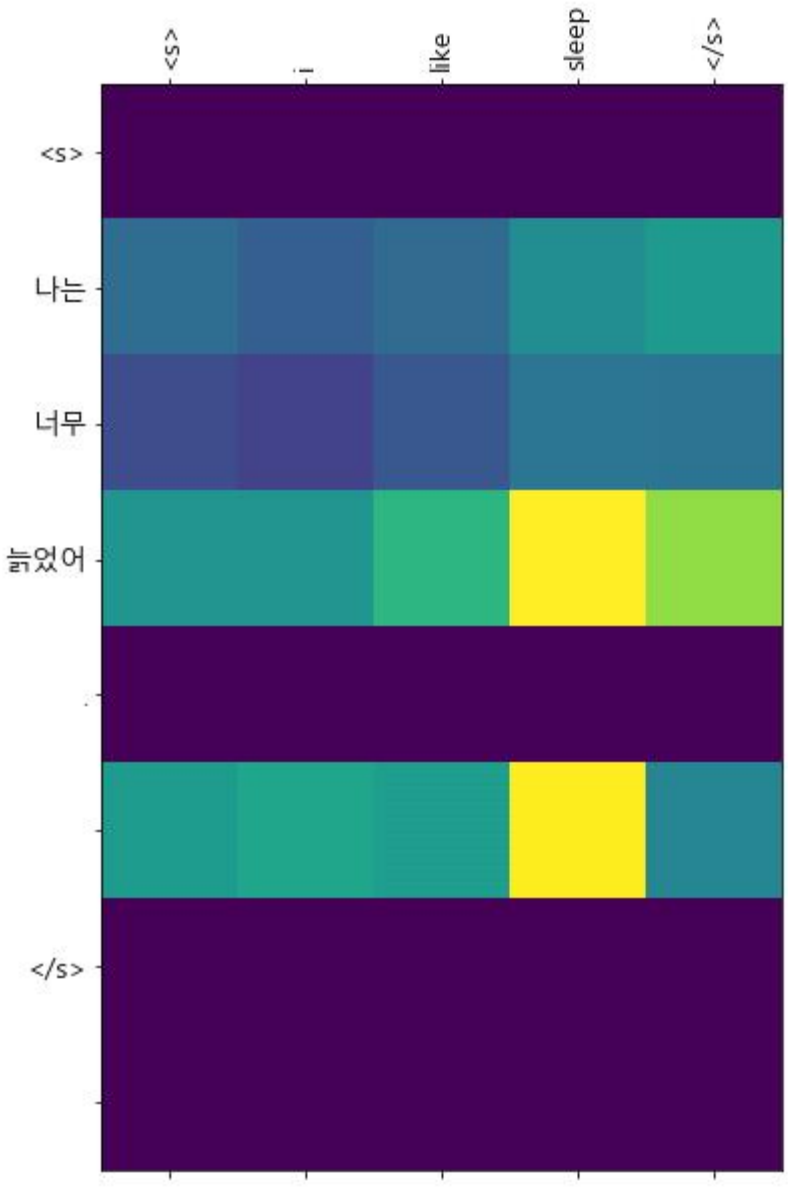


<복원 테스트>

5-3. 번역 오류

```
1 translate(u' i like sleep')
```

```
Input: <s> i like sleep </s>
Predicted translation: <s> 나는 너무 늙었어 . </s>
```



```
1 translate(u' do spiders scare you ? ')
```

```
Input: <s> do spiders scare you ? </s>
Predicted translation: <s> 노래하는 거 좋아해 ?  </s>
```