텍스트 생성 - 한글->영어 기계 번역

In [ ]:

```
# 사용 모듈 import
import os
import pandas as pd
import numpy as np
from pprint import pprint

from tensorflow import keras
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dense, SimpleRNN, Input
from tensorflow.keras.utils import to_categorical

from IPython.display import SVG
from tensorflow.keras.utils import model_to_dot
```

In [ ]:

```
# 데이터 읽기
d_path = './data/KEnglish_Text_Corpus_sample/'
files = os.listdir(d_path)
print(files)
```

In [ ]:

```
# 데이터 읽기 및 확인
xlsx = pd.read_excel(d_path+files[0])
print(xlsx)
```

In [ ]:

```
# 데이터 확인
print(xlsx['한국어'])
```

In [ ]:

```
# 데이터 확인
print(xlsx['영어'])
```

In [ ]:

```
# 사용할 데이터만 추출
kor_sentences = xlsx['한국어'].values
print(len(kor_sentences))
pprint(kor_sentences[:5])
kor_sentences = kor_sentences[:100]
```

In [ ]:

```
# 사용할 데이터만 추출
eng_sentences = xlsx['영어'].values
print(len(eng_sentences))
pprint(eng_sentences[:5])
eng_sentences = eng_sentences[:100]
```

In [ ]:

```
# 한글 words_set 생성
kor_words_set = set()
kor_max_len = 0
for ix, sentence in enumerate(kor_sentences):
    words = [word.strip() for word in sentence.strip().split(' ')]
    if len(words) > kor_max_len:
        kor_max_len = len(words)
    while '' in words:
        words.remove('')
    kor_words_set.update(words)
```

In [ ]:

```
# 한글 words_set 확인
kor_words_set = list(sorted(kor_words_set))
kor_words_set.insert(0, '')
print('#kor words: ', len(kor_words_set))
print('max seq length: ', kor_max_len)
```

In [ ]:

```
# 영어 words_set 생성
eng_words_set = set()
eng_max_len = 0
for ix, sentence in enumerate(eng_sentences):
    words = [word.strip() for word in sentence.strip().split(' ')]
    if len(words) > eng_max_len:
        eng_max_len = len(words)
    while '' in words:
        words.remove('')
    eng_words_set.update(words)
eng_words_set.update(['<start>', '<eos>'])
```

In [ ]:

```
# 영어 words_set 확인
eng_words_set = list(sorted(eng_words_set))
eng_words_set.insert(0, '')
print('#end words', len(eng_words_set))
print('max seq length: ', eng_max_len)
```

In [ ]:

```
# 한국어 문장 정수 인코딩 및 패딩 함수 작성
def kor_encoding(sentence):
    words = [word.strip() for word in sentence.strip().split(' ')]
    while '' in words:
        words.remove('')

    words = [kor_words_set.index(word) for word in words]
    encode_sentence = words
    for _ in range(kor_max_len-len(encode_sentence)):
        encode_sentence.insert(0, 0)
    return np.asarray(encode_sentence).astype('float')
```

In [ ]:

```
# 한국어 인코딩 및 데이터 확인
encode_sentences = []
for sentence in kor_sentences:
    encode_sentences.append(kor_encoding(sentence))
encode_sentences = np.asarray(encode_sentences).astype('float')
print(encode_sentences[:5])
```

In [ ]:

```
# 영어 문장 Special Symbol 포함 정수 인코딩 및 패딩
decode_sentences = []
for sentence in eng_sentences:
    words = [word.strip() for word in sentence.strip().split(' ')]
    while '' in words:
        words.remove('')

    words = [eng_words_set.index(word) for word in words]
    decode_sentence = words
    for _ in range(eng_max_len-len(decode_sentence)):
        decode_sentence.insert(0, 0)
    decode_sentence[0]=eng_words_set.index('<start>')
    decode_sentence.append(eng_words_set.index('<eos>'))
    decode_sentences.append(decode_sentence)
decode_sentences = np.asarray(decode_sentences).astype('float')
```

In [ ]:

```
# 데이터 확인
print(decode_sentences[:5])
```

In [ ]:

```
# 모든 데이터 One hot 인코딩
encoder_train = to_categorical(encode_sentences)
decoder_train = to_categorical(decode_sentences[:, :-1])
decoder_target = to_categorical(decode_sentences[:, 1:])
```

In [ ]:

```
# 데이터 확인
print(encoder_train.shape)
print(decoder_target.shape)
print(decoder_train.shape)
print(encoder_train[:2])
print(decoder_train[:2])
print(decoder_target[:2])
```

In [ ]:

```
# Loss 저장을 위한 클래스 생성
class LossHistory(keras.callbacks.Callback):
    def init(self):
        self.losses = []

    def on_epoch_end(self, batch, logs={}):
        self.losses.append(logs.get('loss'))
```

In [ ]:

```
# RNN cell 개수, 및 단어 개수 설정
num_units = 128
kor_num_token = len(kor_words_set)
eng_num_token = len(eng_words_set)
```

In [ ]:

```
# 인코더 레이어 생성 및 연결
enc_inputs = Input(shape=(None, kor_num_token), name='encoder_input')
enc = SimpleRNN(num_units, return_sequences=True, return_state=True, name='encoder')
enc_outputs, enc_state = enc(enc_inputs)
```

In [ ]:

```
# 디코더 레이어 생성 및 연결
dec_inputs = Input(shape=(None, eng_num_token), name='decoder_input')
dec = SimpleRNN(num_units, return_sequences=True, return_state=True, name='decoder')
dec_outputs, _ = dec(dec_inputs, initial_state=enc_state)
dec_dense = Dense(eng_num_token, activation='softmax', name='decoder_output')
dec_outputs = dec_dense(dec_outputs)
```

In [ ]:

```
# 모델 생성
model = Model([enc_inputs, dec_inputs], dec_outputs)
```

In [ ]:

```
# 모델 컴파일
model.compile(loss='categorical_crossentropy', optimizer='rmsprop', metrics=['accuracy'])
```

In [ ]:

```
# 모델 학습
model.fit([encoder_train, decoder_train], decoder_target, batch_size=8, epochs=100)
```

In [ ]:

```
# 모델 확인
SVG(model_to_dot(model, show_shapes=True, dpi=65).create(prog='dot', format='svg'))
## 이 모델에는 입력으로 항상 인코더 인풋과 디코더 인풋이 함께 들어가야 하기때문에 학습에만
## 사용하며 실제 사용은 불가, 하지만 이미 학습은 진행되었기 때문에 weights는 학습 되었음.
## 따라서 학습된 레이어를 사용해 모델 재구성 필요.
```

In [ ]:

```
# 인코더 모델 생성
encoder_model = Model(enc_inputs, enc_state)
```

In [ ]:

```
# 인코더 모델 시각화
SVG(model_to_dot(encoder_model, show_shapes=True, dpi=65).create(prog='dot', format='svg'))
```

In [ ]:

```
# 디코더 모델 생성
decoder_state_input = Input(shape=(num_units,), name='decoder_state_input')
dec_outputs, dec_state = dec(dec_inputs, initial_state=decoder_state_input)
dec_outputs = dec_dense(dec_outputs)

decoder_model = Model(
    [dec_inputs] + [decoder_state_input],
    [dec_outputs] + [dec_state])
```

In [ ]:

```
# 디코더 모델 시각화
SVG(model_to_dot(decoder_model, show_shapes=True, dpi=65).create(prog='dot', format='svg'))
```

In [ ]:

```
# 모델로 한글을 영어로 번역하는 함수 작성
def decode_sequence(input_seq):
    state = encoder_model.predict(input_seq)

    target_seq = np.zeros((1, 1, eng_num_token))
    target_seq[0, 0, eng_words_set.index('<start>')] = 1

    stop_condition = False
    decoded_sentence = []
    while not stop_condition:
        output_token, state = decoder_model.predict([target_seq]+[state])

        sampled_token_index = np.argmax(output_token[0, -1, :])
        sampled_word = eng_words_set[sampled_token_index]
        if sampled_token_index != 0:
            decoded_sentence.append(sampled_word)

        if sampled_word == '<eos>':
            stop_condition = True

        target_seq = np.zeros((1, 1, eng_num_token))
        target_seq[0, 0, sampled_token_index] = 1

    return decoded_sentence[:-1]
```

In [ ]:

```
# 이름만 translate
def translate(input_seq):
    decode_sequences = decode_sequence(input_seq)
    return decode_sequences
```

In [ ]:

```
# 테스트용 인코더 입력 데이터 생성 및 확인
enc_sent = [kor_words_set[np.argmax(word_vec)]
            for word_vec in encoder_train[0] if np.argmax(word_vec) != 0]
print(len(enc_sent))
print(enc_sent)
```

In [ ]:

```
# 번역 및 결과 확인
sentence = translate(encoder_train[0].reshape(1, kor_max_len, -1))
print(len(sentence))
print(sentence)
```

In [ ]:

```
# 한글 -> 영어 결과 비교
print(' '.join(enc_sent))
print(' '.join(sentence))
```

In [ ]:

```
print(len(kor_words_set))
print(kor_words_set)
```

In [ ]:

```
kor_sent = '혹시 한국 전통술에 막걸리를 먹어야 되나요?'
enc_kor_sent = kor_encoding(kor_sent)
print(enc_kor_sent)
```

In [ ]:

```
sentence = translate(to_categorical(enc_kor_sent, num_classes=len(kor_words_set)).reshape(1, kor
_max_len, -1))
print(len(sentence))
print(' '.join(sentence))
```

In [ ]:

```

```