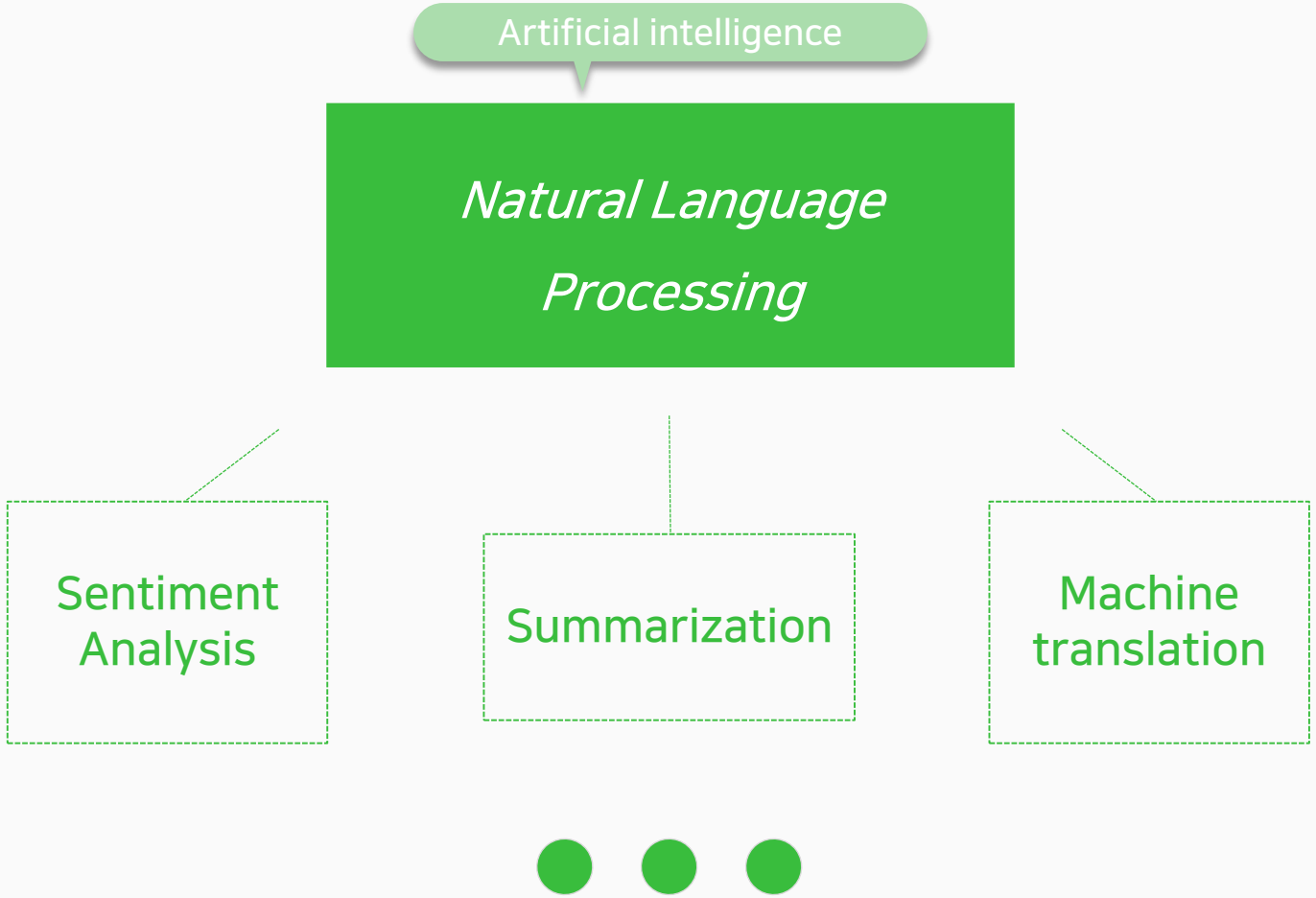
FINAL PROJECT

객체지향언어와실습

김예원, 이상민, 추예진

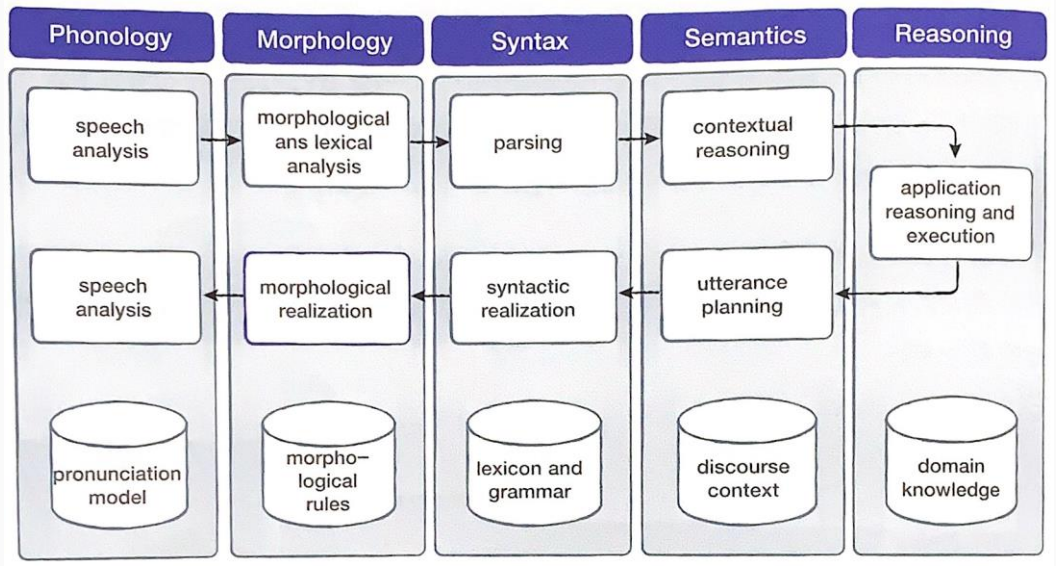
PROLOGUE


Artificial intelligence
Natural Language Processing
Sentiment Analysis
Summarization
Machine translation

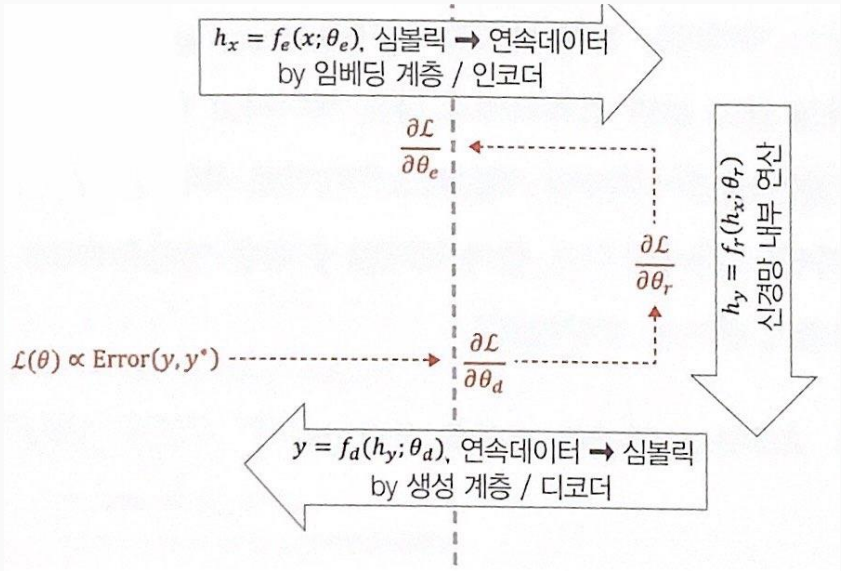
PROLOGUE

CHECK POINT

Paradigm Shift in Natural Language Processing



자연어 처리를 위한 딥러닝의 도입



딥러닝 자연어 처리 과정

PROLOGUE

CHECK POINT

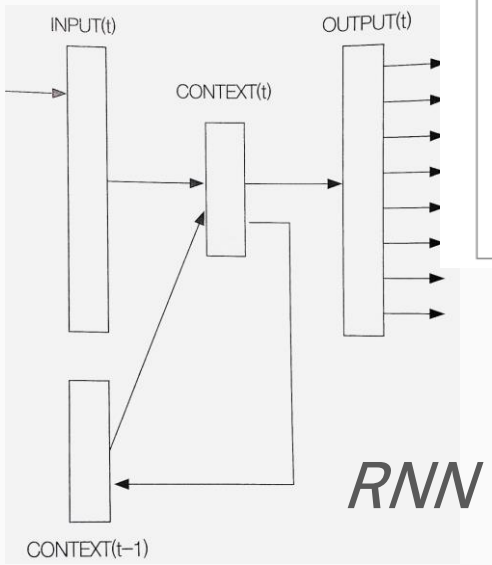
Why is NLP difficult?

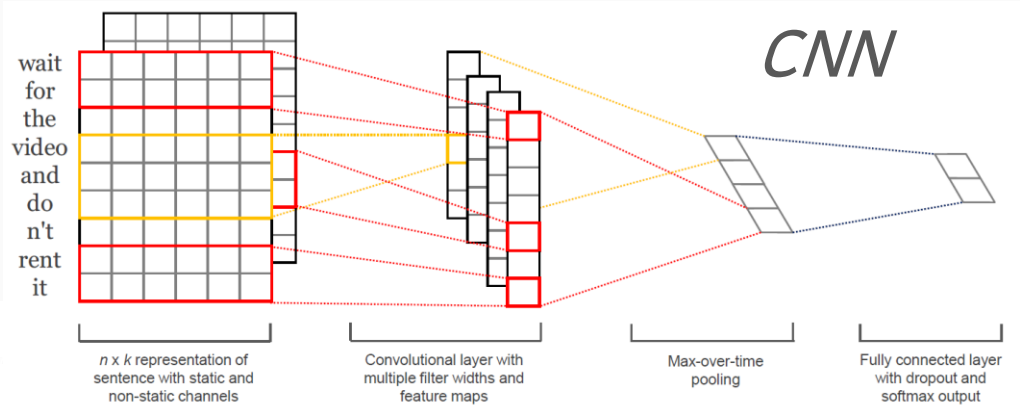
ambiguity	various expressions	discontinuous data
문장의 중의성, 문장 내 정보부족	다양한 문장의 표현방식, 비슷한 의미의 단어들	차원의 저주, 노이즈와 정규화

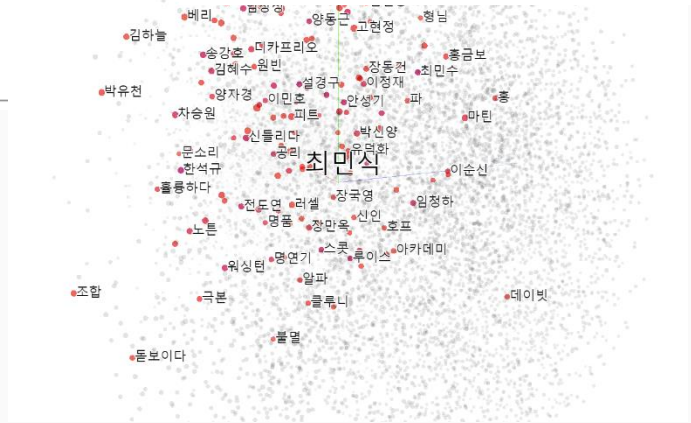
PROLOGUE

CHECK POINT

Process of Natural Language Processing by Deep Learning





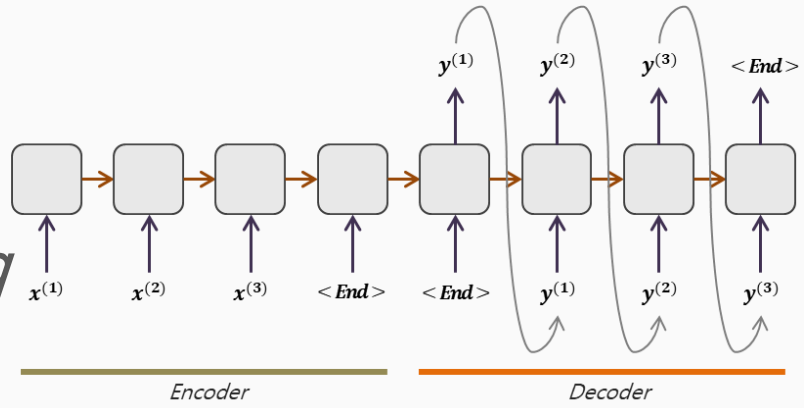


PROLOGUE

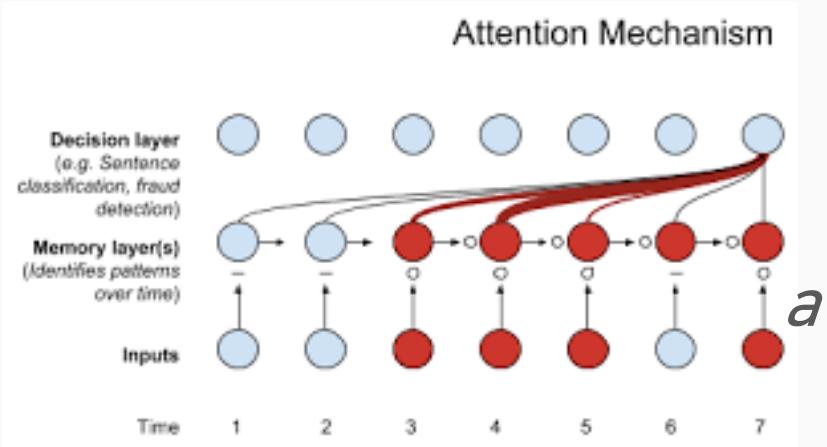
CHECK POINT

Process of Natural Language Processing by Deep Learning

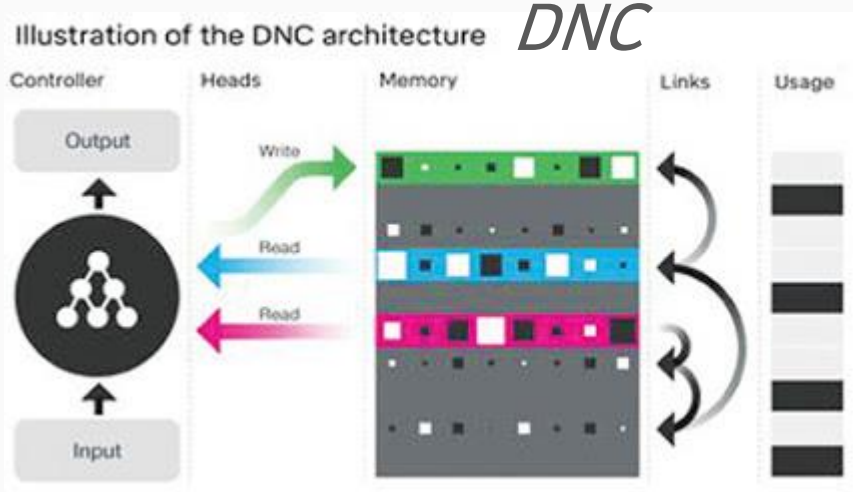
seq2seq



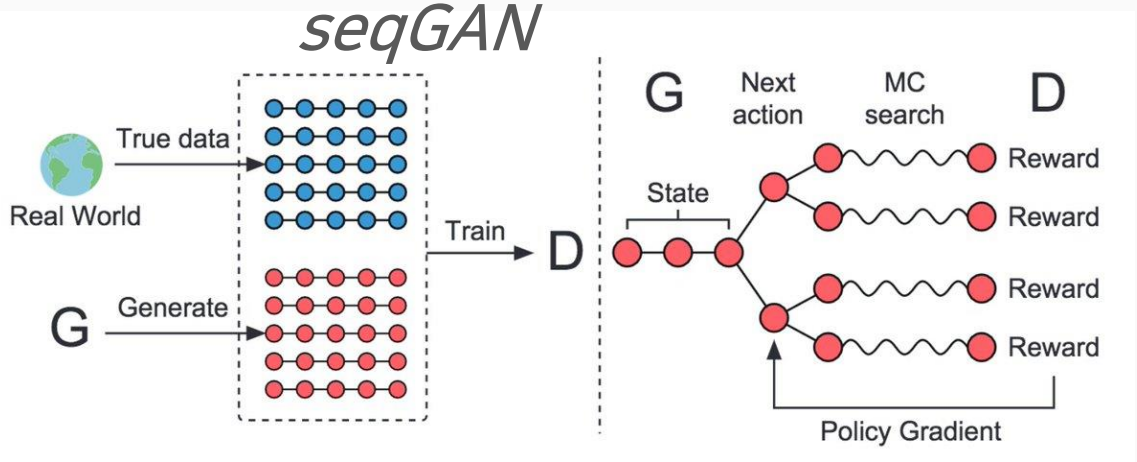
attention



DNC



seqGAN



PROLOGUE

TensorFlow

TensorFlow > Learn > TensorFlow Core > Tutorials

☆☆☆☆☆

Basic text classification

Text Classification

감성 분석, 스팸 메일 탐지, 사용자 의도 분류, 주제 분류, 카테고리 분류 · · ·

PROJECT

데이터셋 만들기

[규칙 / 어휘 기반 감성분석]

```
import requests
from bs4 import BeautifulSoup
import re
import pandas as pd

positive = []
negative = []
pos_neg = []

pos = open("C:/anaconda3/envs/tensorflow_env/tensorflow_prac/positive_words_self.txt", 'r', encoding='UTF-8')
lines_p = pos.readlines()

for line in lines_p:
    p_line = line.replace('\n', '')
    positive.append(p_line)
    pos_neg.append(p_line)

pos.close()

neg = open("C:/anaconda3/envs/tensorflow_env/tensorflow_prac/negative_words_self.txt", '
lines_n = neg.readlines()

for line in lines_n:
    n_line = line.replace('\n', '')
    negative.append(n_line)
```

학습데이터:
네이버에서 애플, 삼성 관련 기사들의 제목을 크롤링해서 데이터셋 생성
총 6,176개의 데이터

어휘사전:
기사에서 자주 등장하는 긍정적인 단어와 부정적인 단어 리스트 생성

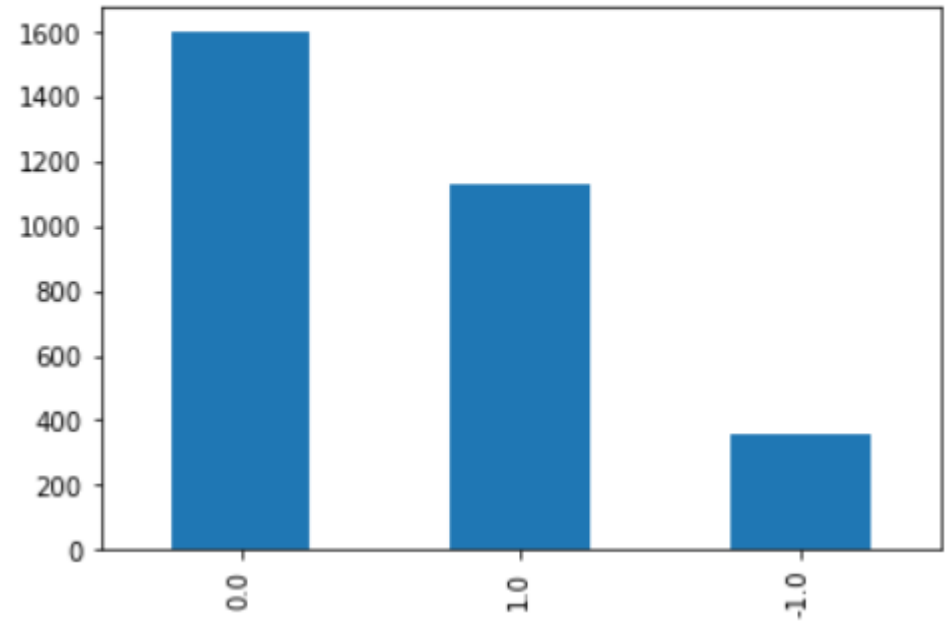
PROJECT

 # 데이터 분석

[규칙 / 어휘 기반 감성분석]

```
train_data['label'].value_counts().plot(kind='bar')
```

```
<AxesSubplot:>
```




```
print(train_data.grou
```

	label	count
0	-1.0	356
1	0.0	1600
2	1.0	1133

PROJECT

[규칙 / 어휘 기반 감성분석]

데이터 전처리

```
stopwords = ['의', '가', '이', '은', '들', '는', '좀', '잘', '걍
```

```
import konlpy
from konlpy.tag import Okt
okt = Okt()
X_train = []
for sentence in train_data['title']:
    temp_X = []
    temp_X = okt.morphs(sentence, stem=True)
    temp_X = [word for word in temp_X if not word in stopwords]
    X_train.append(temp_X)
X_test = []
for sentence in test_data['title']:
    temp_X = []
    temp_X = okt.morphs(sentence, stem=True)
    temp_X = [word for word in temp_X if not word in stopwords]
    X_test.append(temp_X)
```

```
from keras.preprocessing.text import Tokenizer
max_words = 35000
tokenizer = Tokenizer(num_words = max_words)
tokenizer.fit_on_texts(X_train)
X_train = tokenizer.texts_to_sequences(X_train)
X_test = tokenizer.texts_to_sequences(X_test)
```

어휘에 기반하고 있기 때문에 특히 전처리가 중요

Okt 형태소 분석기 활용
불용어 제거

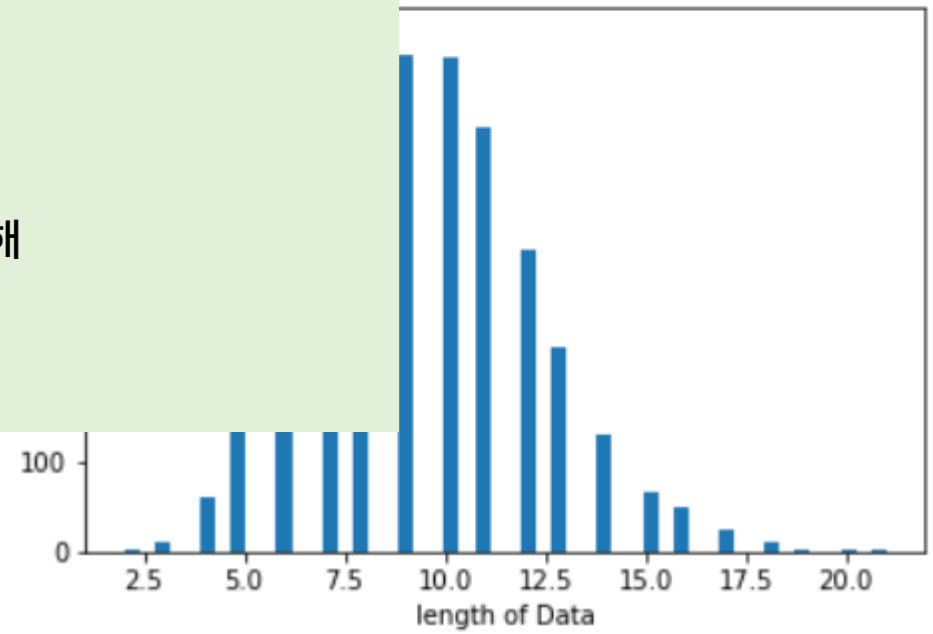
정수인코딩

PROJECT

 # 데이터 전처리

```
print("제목 최대 길이 : ", max(len(l) for l in X_train))
print("제목 평균 길이 : ", sum(map(len, X_train))/ len(X_train))
plt.hist([len(s) for s in X_train], bins=50)
plt.xlabel('length of Data')
plt.ylabel('number of Data')
plt.show()
```

```
제목의 최대 길이 :  21
제목의 평균 길이 :  9.3985
```



데이터 분포 확인

라벨값 1, -1, 0 에 대해
one-hot encoding

[규칙 / 어휘 기반 감성분석]

```
import numpy as np

y_train = []
y_test = []

for i in range(len(train_data['label'])):
    if train_data['label'].iloc[i] == 1:
        y_train.append([0, 0, 1])
    elif train_data['label'].iloc[i] == 0:
        y_train.append([0, 1, 0])
    elif train_data['label'].iloc[i] == -1:
        y_train.append([1, 0, 0])

for i in range(len(test_data['label'])):
    if test_data['label'].iloc[i] == 1:
        y_test.append([0, 0, 1])
    elif test_data['label'].iloc[i] == 0:
        y_test.append([0, 1, 0])
    elif test_data['label'].iloc[i] == -1:
        y_test.append([1, 0, 0])

y_train = np.array(y_train)
y_test = np.array(y_test)
```

PROJECT

모델 만들기

[규칙 / 어휘 기반 감성분석]

패딩 과정
데이터 길이 통일

정확도: 84.28%

```
from keras.layers import Embedding, Dense, LSTM
from keras.models import Sequential
from keras.preprocessing.sequence import pad_sequences

max_len = 20
X_train = pad_sequences(X_train, maxlen=max_len)
X_test = pad_sequences(X_test, maxlen=max_len)
```

```
model = Sequential()
model.add(Embedding(max_words, 100))
model.add(LSTM(128))
model.add(Dense(3, activation='softmax'))

model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
history = model.fit(X_train, y_train, epochs=10, batch_size=10, validation_split=0.1)
```

```
360/360 [==============================]
Epoch 5/10
360/360 [==============================]
Epoch 6/10
360/360 [==============================]
Epoch 7/10
360/360 [==============================]
Epoch 8/10
360/360 [==============================]
Epoch 9/10
360/360 [==============================]
Epoch 10/10
360/360 [==============================]
```

```
print("accuracy:{:.2f}%".format(model.eva
```

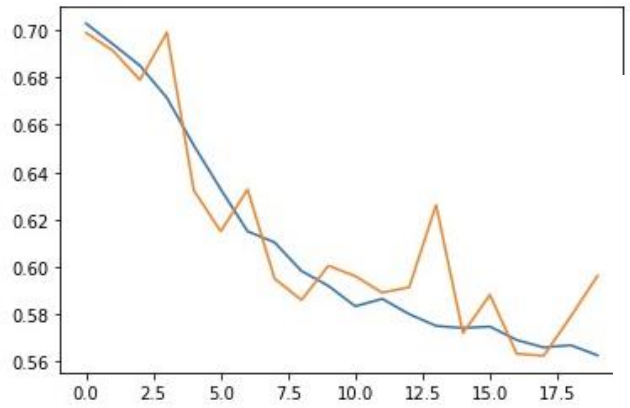
```
125/125 [==============================]
accuracy:84.28%
```

PROJECT

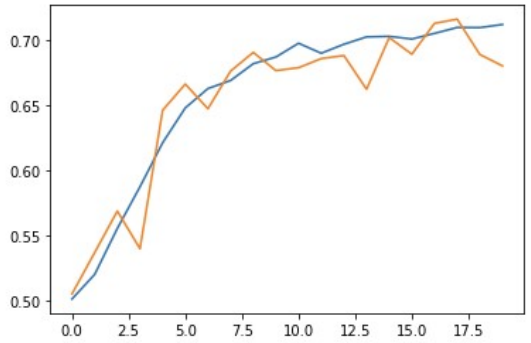
 # SimpleRNN으로 순환신경망 만들기

[순환신경망을 이용한 텍스트 분류]

```
[25]  plt.plot(history.history['loss'])
      plt.plot(history.history['val_loss'])
      plt.show()
```




```
[26]  plt.plot(history.history['accuracy'])
      plt.plot(history.history['val_accuracy'])
      plt.show()
```




```
[27]
      loss, accuracy = model.evaluate(x_val_onehot, y_val, verbose=0)
      print(accuracy)
```

```
0.680400013923645
```

데이터셋: IMDB
25000개의 train data 중 5000개를 test data 사용

to_categorical()함수를 사용해 원-핫 인코딩
손실 함수는 binary_crossentropy로 지정

20번의 에포크 동안 조금씩 과대적합되는 현상이 나타남
test set에 대한 정확도는 약 68%

PROJECT

 # 임베딩층으로 모델 성능 향상

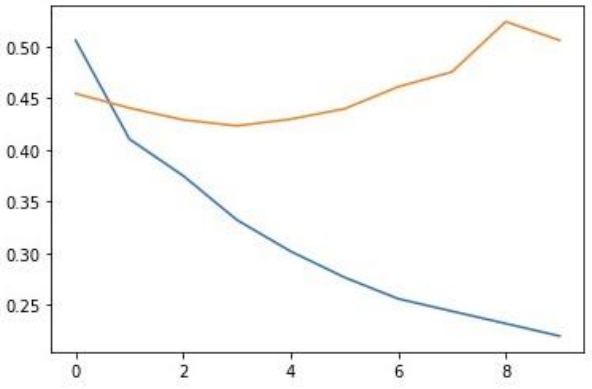
[순환신경망을 이용한 텍스트 분류]

파일 수정 보기 삽입 런타임 도구 도움말

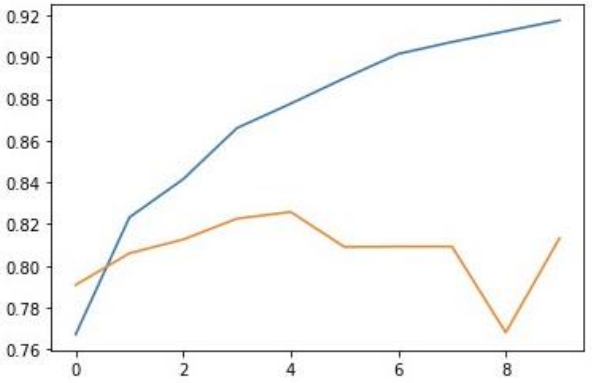
+ 코드 + 텍스트

```
625/625 [==============================] - 11s 17ms/step - loss: 0.2
```

```
[33] plt.plot(history.history['loss'])
     plt.plot(history.history['val_loss'])
     plt.show()
```




```
[34] plt.plot(history.history['accuracy'])
     plt.plot(history.history['val_accuracy'])
     plt.show()
```



단어 임베딩은 모델을 훈련하면서 같이 훈련되므로 훈련이 진행될수록 단어의 연관 관계를 더 정확하게 찾을 수 있음

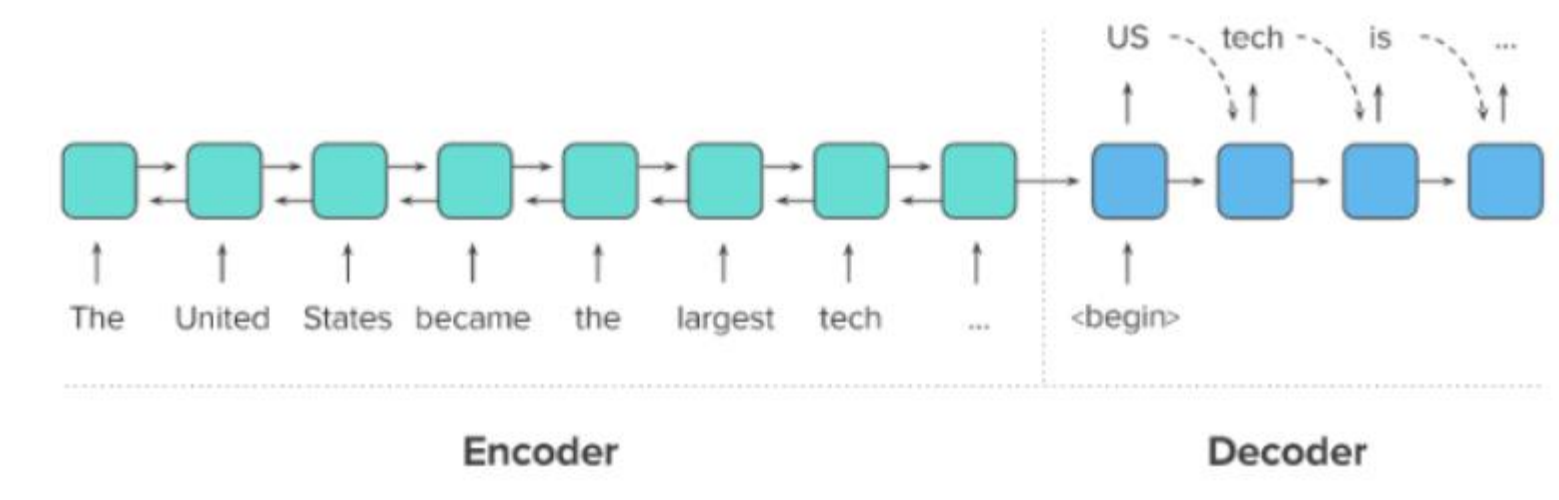
이전 모델보다 적은 셀 개수에서도 더 높은 성능을 보여줌
정확도는 약 81%

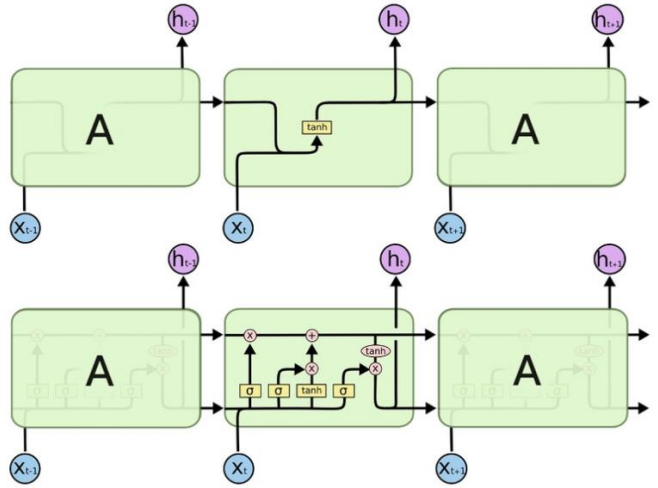
```
loss, accuracy = model_ebd.evaluate(x_val_seq, y_val, verbose=0)
print(accuracy)
```

```
0.8131999969482422
```

PROJECT

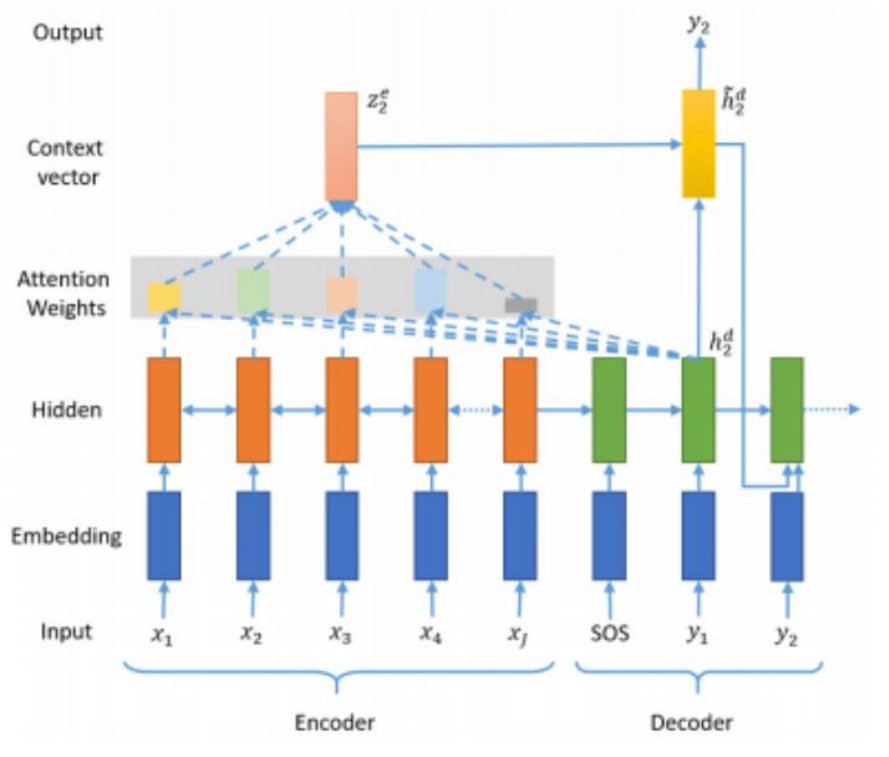
 # seq2seq

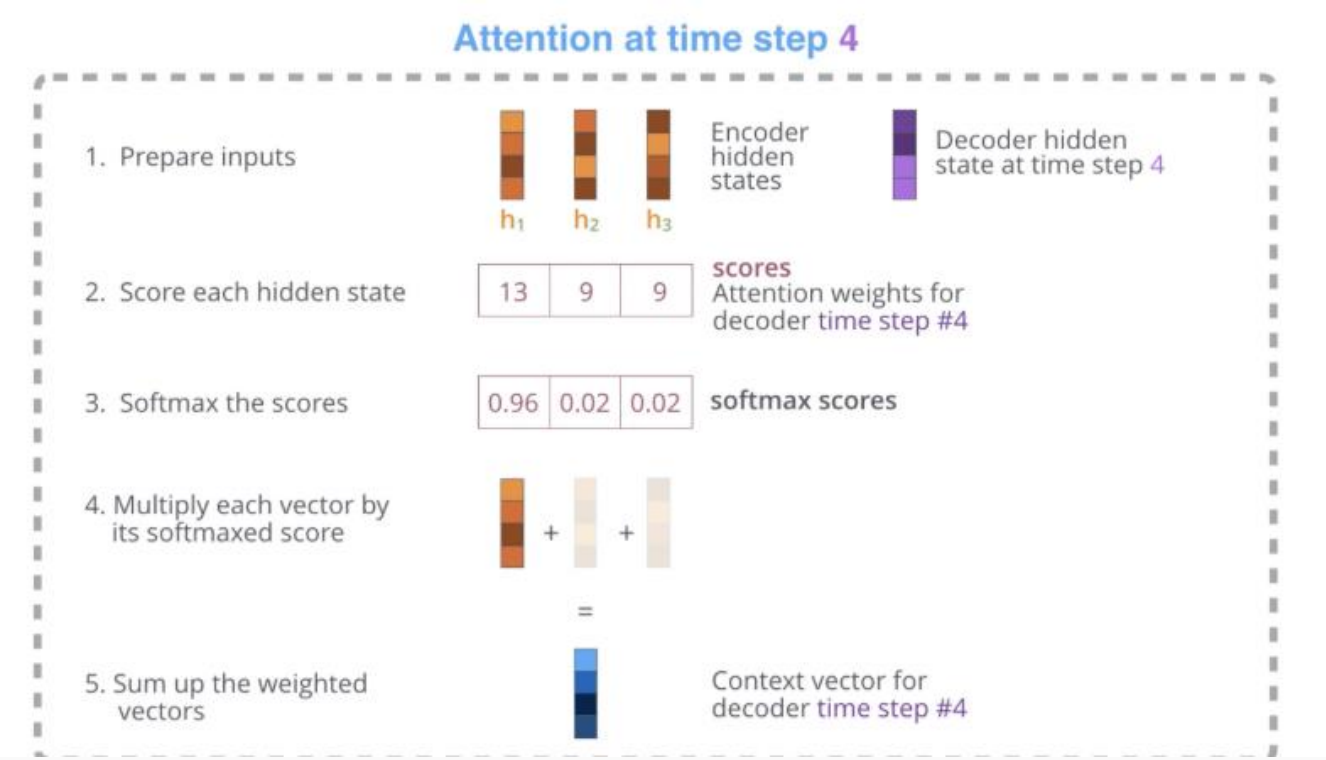




PROJECT

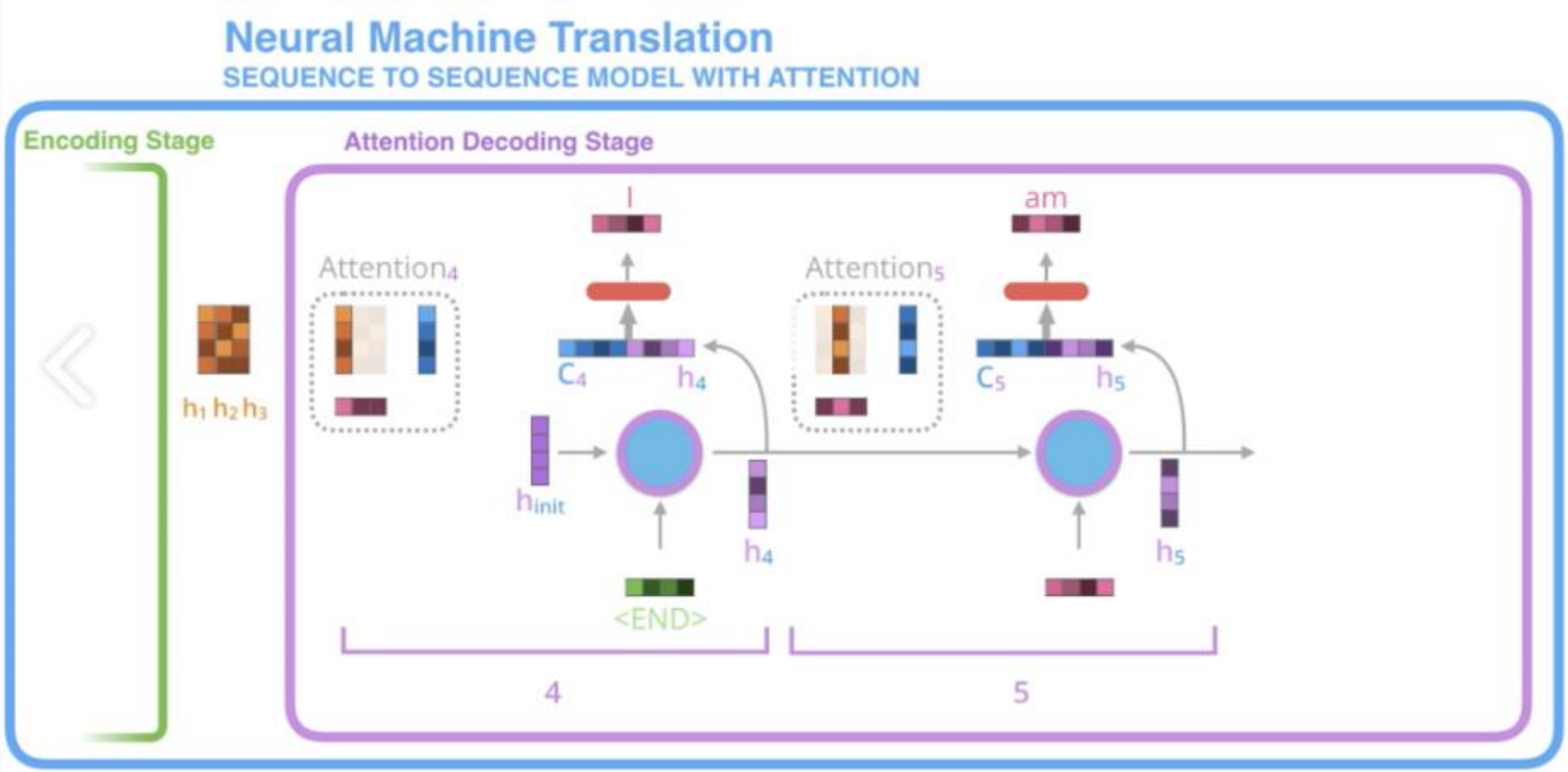
 # attention mechanism





PROJECT

 # seq2seq + attention



PROJECT

데이터

[Abstractive summarization 방식의 텍스트 요약]

데이터 준비 과정
데이터셋: kaggle에서 제공하는 아마존 리뷰 데이터셋

NLTK 라이브러리 이용

In [1]:
```
import nltk
nltk.download("stopwords")
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:₩Users₩yj200₩AppData₩Roaming₩nltk_data...
[nltk_data]   Unzipping corpora₩stopwords.zip.
```

Out[1]: True

In [2]:
```
import numpy as np
import pandas as pd
import os
import re
import matplotlib.pyplot as plt
from nltk.corpus import stopwords
from bs4 import BeautifulSoup
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
import urllib.request
```

In [3]:
```
file_path = os.path.dirname(os.path.abspath('__file__')) + r'/Reviews.csv'
data = pd.read_csv(file_path, nrows=100000)
print("전체 샘플 수 :", len(data))
```

전체 샘플 수 : 100000

In [4]:
```
data.head()
```

Out[4]:

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessI
0	1	B001E4KFG0	A3SGXH7AUHU8GW	delmartian		1

In [5]:
```
data = data[['Text', 'Summary']]
data.head()
```

Out[5]:

	Text	Summary
0	I have bought several of the Vitality canned d...	Good Quality Dog Food
1	Product arrived labeled as Jumbo Salted Peanut...	Not as Advertised
2	This is a confection that has been around a fe...	"Delight" says it all
3	If you are looking for the secret ingredient i...	Cough Medicine
4	Great taffy at a great price. There was a wid...	Great taffy

In [6]:
```
data.sample(15)
```

		Summary
		Best Food for the Price
		good stuff
		I like oatmeal
		Kirkland Chocolates
		meal for those hectic days
		r water?..pick one,please!
81568	I would just like to say that this company pro...	read the comment :D
41369	this is the best cat food ever the cats love i...	purrrrrrrrrrrrrfect
41847	Delicious aroma and good quality. One of the	Delicious aroma

PROJECT

 # 데이터전처리

[Abstractive summarization 방식의 텍스트 요약]

중복된 데이터 제거
텍스트 정규화
NLTK를 이용해 불용어 제거
(Text 전처리에만 사용하고 Summary 전처리 할 때는 호출x
요약문은 불용어들이 남아있어야 좀 더 자연스러운 문장일거라 생각)

```
# 데이터 전처리 함수
def preprocess_sentence(sentence, remove_stopwords = True):
    sentence = sentence.lower() # 텍스트 소문자화
    sentence = BeautifulSoup(sentence, 'lxml').text # html 태그 제거
    sentence = re.sub(r'₩([^)]*₩)', '', sentence) # 괄호로 닫힌 문자열 제거
    sentence = re.sub('"','', sentence) # 쌍따옴표 제거
    sentence = ' '.join([contractions[t] if t in contractions else t for t in sentence.split(" ")]) #
    sentence = re.sub(r"'s₩b","",sentence) # 소유격 제거
    sentence = re.sub("[^a-zA-Z]", " ", sentence) # 영어 외 문자 공백으로 변환
    sentence = re.sub('[m]{2,}', 'mm', sentence) # m이 3개 이상이면 2개로 변경

    # 불용어 제거
    if remove_stopwords:
        tokens = ' '.join(word for word in sentence.split() if not word in stopwords.words('english')
    else:
        tokens = ' '.join(word for word in sentence.split() if len(word) > 1)

    return tokens
```

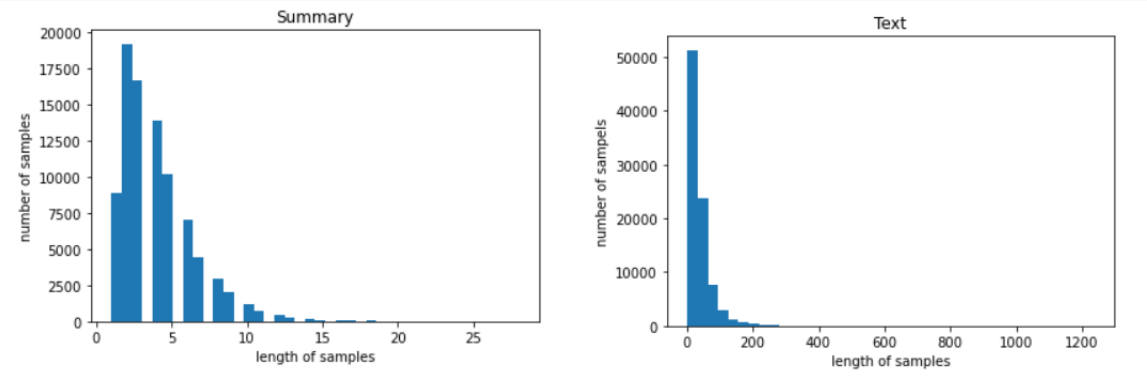
```
clean_text = []

for s in data['Text']:
    clean_text.append(preprocess_sentence(s))
```

```
clean_text[:5]
```

PROJECT

데이터전처리

[Abstractive summarization 방식의 텍스트 요약]



데이터 정제 과정 후 생겨난 empty sample NULL값으로 변환하여 제거

Text, Summary의 길이 파악하고 적절한 길이 설정

```
In [24]: import matplotlib.pyplot as plt

text_len = [len(s.split()) for s in data['Text']]
summary_len = [len(s.split()) for s in data['Summary']]

print("텍스트의 최소 길이 : ", np.min(text_len))
print("텍스트의 최대 길이 : ", np.max(text_len))
print("텍스트의 평균 길이 : ", np.mean(text_len))
print("요약의 최소 길이 : ", np.min(summary_len))
print("요약의 최대 길이 : ", np.max(summary_len))
print("요약의 평균 길이 : ", np.mean(summary_len))
```

```
텍스트의 최소 길이 :  2
텍스트의 최대 길이 :  1235
텍스트의 평균 길이 :  38.792428272310566
요약의 최소 길이 :  1
요약의 최대 길이 :  28
요약의 평균 길이 :  4.010729443721352
```

```
In [25]: plt.subplot(1, 2, 1)
plt.boxplot(summary_len)
plt.title("Summary")

plt.subplot(1, 2, 2)
plt.boxplot(text_len)
plt.title("Text")
plt.tight_layout()

plt.show()
```

PROJECT

데이터전처리

[Abstractive summarization 방식의 텍스트 요약]

요약데이터에 시작 토큰과 종료 토큰 추가

데이터 샘플을 잘 섞어 8:2 비율로 훈련셋, 테스트셋 분리

테스트 데이터 수: 13163개
훈련 데이터 수: 52655개

```
data['decoder_input'] = data['Summary'].apply(lambda x : 'sostoken ' + x)
data['decoder_target'] = data['Summary'].apply(lambda x : x + ' eostoken')
data.head()
```

	Text	Summary	decoder_input	decoder_target
0	bought several vitality canned dog food produc...	good quality dog food	sostoken good quality dog food	good quality dog food eostoken
1	product arrived labeled jumbo salted peanuts p...	not as advertised	sostoken not as advertised	not as advertised eostoken
2	confection around centuries light pillowy citr...	delight says it all	sostoken delight says it all	delight says it all eostoken
3	looking secret ingredient robitussin believe f...	cough medicine	sostoken cough medicine	cough medicine eostoken
4	great taffy great price wide assortment yummy ...	great taffy	sostoken great taffy	great taffy eostoken

In [33]:
```
encoder_input = np.array(data['Text']) # 인코더의 입력
decoder_input = np.array(data['decoder_input']) # 디코더의 입력
decoder_target = np.array(data['decoder_target']) # 디코더의 레이블
```

In [34]:
```
indices = np.arange(encoder_input.shape[0])
np.random.shuffle(indices)
print(indices)
```

PROJECT

데이터전처리

[Abstractive summarization 방식의 텍스트 요약]

등장 빈도가 6회 이하인 단어들은 정수 인코딩 과정에서 제외,
훈련 데이터에서도 제거

정수인코딩:
텍스트 시퀀스를 정수 시퀀스로 변환

패딩하기

In [40]:
```
src_vocab = 8000
src_tokenizer = Tokenizer(num_words = src_vocab)
src_tokenizer.fit_on_texts(encoder_input_train)
```

In [41]:
```
encoder_input_train = src_tokenizer.texts_to_sequences(encoder_input_train)
encoder_input_test = src_tokenizer.texts_to_sequences(encoder_input_test)

print(encoder_input_train[:3])
```

```
[[78, 44, 5, 4782, 85, 580, 24, 18, 1061, 315, 54, 251, 188, 147, 428, 766, 5, 3024, 9, 181, 15], [334, 61, 62, 113, 48
94, 21, 76, 329, 515], [209, 6, 165, 11, 513, 1447, 140, 2885, 81, 362, 5688, 3591, 628, 165, 11, 26, 2, 539, 130, 7, 1
37, 558, 1609, 1357, 16, 1820, 1146, 329, 1039]]
```

:
```
tar_tokenizer = Tokenizer()
tar_tokenizer.fit_on_texts(decoder_input_train)
```

:
```
threshold = 6
total_cnt = len(tar_tokenizer.word_index)
rare_cnt = 0
total_freq = 0
rare_freq = 0

for key, value in tar_tokenizer.word_counts.items():
    total_freq = total_freq + value

    if(value < threshold):
        rare_cnt = rare_cnt + 1
        rare_freq = rare_freq + value
```

PROJECT

모델 설계

[Abstractive summarization 방식의 텍스트 요약]

In [47]:
```
from tensorflow.keras.layers import Input, LSTM, Embedding, Dense, Concatenate
from tensorflow.keras.models import Model
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint

embedding_dim = 128
hidden_size = 256

# 인코더
encoder_inputs = Input(shape = (text_max_len, ))

# 인코더의 임베딩 층
enc_emb = Embedding(src_vocab, embedding_dim)(encoder_inputs)

# 인코더의 LSTM 1
encoder_lstm1 = LSTM(hidden_size, return_sequences=True, return_state=True, dropout=0.4, recurrent_dropout=0.4)
encoder_output1, state_h1, state_c1 = encoder_lstm1(enc_emb)

# 인코더의 LSTM 2
encoder_lstm2 = LSTM(hidden_size, return_sequences=True, return_state=True, dropout=0.4, recurrent_dropout=0.4)
encoder_output2, state_h2, state_c2 = encoder_lstm2(encoder_output1)

# 인코더의 LSTM 3
encoder_lstm3 = LSTM(hidden_size, return_sequences=True, return_state=True, dropout=0.4, recurrent_dropout=0.4)
encoder_outputs, state_h, state_c = encoder_lstm3(encoder_output2)
```

Hidden state: LSTM에서 얼만큼의 수용력을 가질지

인코더 설계:
인코더의 LSTM은 총 3개의 층으로 구성

PROJECT

모델 설계

[Abstractive summarization 방식의 텍스트 요약]

In [48]:
```
# 디코더
decoder_inputs = Input(shape=(None,))

# 디코더의 임베딩 층
dec_emb_layer = Embedding(tar_vocab, embedding_dim)
dec_emb = dec_emb_layer(decoder_inputs)

# 디코더의 lstm
decoder_lstm = LSTM(hidden_size, return_sequences = True, return_state = True, dropout=0.4, recurrent_dropout = 0.2)
decoder_outputs, _, _ = decoder_lstm(dec_emb, initial_state = [state_h, state_c])
```

In [49]:
```
decoder_softmax_layer = Dense(tar_vocab, activation='softmax')
decoder_softmax_outputs = decoder_softmax_layer(decoder_outputs)

model = Model([encoder_inputs, decoder_inputs], decoder_softmax_outputs)
model.summary()
```

```
Model: "functional_1"
```

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 50)]	0	
embedding (Embedding)	(None, 50, 128)	1024000	input_1[0][0]
lstm (LSTM)	[(None, 50, 256), (N	394240	embedding[0][0]
input_2 (InputLayer)	[(None, None)]	0	
lstm_1 (LSTM)	[(None, 50, 256), (N	525312	lstm[0][0]
embedding_1 (Embedding)	(None, None, 128)	256000	input_2[0][0]
lstm_2 (LSTM)	[(None, 50, 256), (N	525312	lstm_1[0][0]
lstm_3 (LSTM)	[(None, None, 256),	394240	embedding_1[0][0]

디코더 설계:
인코더와 거의 동일

PROJECT

모델 설계

[Abstractive summarization 방식의 텍스트 요약]

In [51]:
```
attn_layer = AttentionLayer(name = 'attention_layer')

attn_out, attn_states = attn_layer([encoder_outputs, decoder_outputs])

decoder_concat_input = Concatenate(axis=-1, name='concat_layer')([decoder_outputs, attn_out])

# 디코더의 출력층
decoder_softmax_layer = Dense(tar_vocab, activation = 'softmax')
decoder_softmax_outputs = decoder_softmax_layer(decoder_concat_input)

model = Model([encoder_inputs, decoder_inputs], decoder_softmax_outputs)
model.summary()
```

```
Model: "functional_3"
```

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 50)]	0	
embedding (Embedding)	(None, 50, 128)	1024000	input_1[0][0]
lstm (LSTM)	[(None, 50, 256), (N	394240	embedding[0][0]
input_2 (InputLayer)	[(None, None)]	0	
lstm_1 (LSTM)	[(None, 50, 256), (N	525312	lstm[0][0]
embedding_1 (Embedding)	(None, None, 128)	256000	input_2[0][0]
lstm_2 (LSTM)	[(None, 50, 256), (N	525312	lstm_1[0][0]
lstm_3 (LSTM)	[(None, None, 256),	394240	embedding_1[0][0] lstm_2[0][1] lstm_2[0][2]
attention_layer (AttentionLayer	((None, None, 256),	131328	lstm_2[0][0]

디코더의 출력층

여기까지는 기본적인 seq2seq이지만,
출력층 설계에서 어텐션 매커니즘을 더하여 성능 향상
깃허브에 공개되어 있는 어텐션 함수 사용

PROJECT

모델 훈련

[Abstractive summarization 방식의 텍스트 요약]

검증 데이터의 손실이 줄어들지 않고 2회 이상 증가하면
조기종료하도록 설정했는데,
생각보다 학습시간이 너무 오래 걸려서 epoch 횟수 조절

```
In [*]: model.compile(optimizer = 'rmsprop', loss='sparse_categorical_crossentropy')
es = EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=2)
history = model.fit(
    x = [encoder_input_train, decoder_input_train],
    y = decoder_target_train,
    validation_data = ([encoder_input_test, decoder_input_test], decoder_target_test),
    batch_size = 256,
    callbacks = [es],
    epochs = 12
```

```
Epoch 1/12
201/201 [==============================] - 2957s 15s/step - loss: 2.7048 - val_loss: 2.4286
Epoch 2/12
201/201 [==============================] - 3285s 16s/step - loss: 2.3941 - val_loss: 2.2944
Epoch 3/12
201/201 [==============================] - 3387s 17s/step - loss: 2.2436 - val_loss: 2.1556
Epoch 4/12
201/201 [==============================] - 3375s 17s/step - loss: 2.1200 - val_loss: 2.0629
Epoch 5/12
102/201 [==========>...............] - ETA: 26:05 - loss: 2.0468
```

PROJECT

[Abstractive summarization 방식의 텍스트 요약]

#모델 구현

```
encoder_model = Model(inputs = encoder_inputs, outputs = [encoder_outputs, state_h, state_c])

decoder_state_input_h = Input(shape = (hidden_size, ))
decoder_state_input_c = Input(shape = (hidden_size, ))

dec_emb2 = dec_emb_layer(decoder_inputs)

decoder_outputs2, state_h2, state_c2 = decoder_lstm(dec_emb2, initial_state=[decoder_state_input_h, decoder_state
```

```
decoder_hidden_state_input = Input(shape = (text_max_len, hidden_size))
attn_out_inf, attn_states_inf = attn_layer([decoder_hidden_state_input, decoder_outputs2])
decoder_inf_concat = Concatenate(axis=-1, name='concat')([decoder_outputs2, attn_out_inf])

decoder_outputs2 = decoder_softmax_layer(decoder_inf_concat)

decoder_model = Model(
    [decoder_inputs] + [decoder_hidden_state_input, decoder_state_input_h, decoder_state_input_c],
    [decoder_outputs2] + [state_h2, state_c2]
```

```
def decode_sequence(input_seq):
    e_out, e_h, e_c = encoder_model.predict(input_seq)

    target_seq = np.zeros((1, 1))
    target_seq[0, 0] = tar_word_to_index['sostoken']

    stop_condition = False
    decoded_sentence = ""

    while not stop_condition:
        output_tokens, h, c = decoder_model.predict([target_seq] + [e_out
        sampled_token_index = np.argmax(output_tokens[0, -1, :])
        sampled_token = tar_index_to_word[sampled_token_index]

        if sampled_token != 'eostoken':
            decoded_sentence += ' ' + sampled_token

        if sampled_token == 'eostoken' or len(decoded_sentence.split()) :
            stop_condition = True

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

        e_h, e_c = h, c

    return decoded_sentence
```

Seq2seq는 훈련할 때와 실제 동작할 때의 방식이 다름

훈련 단계:
디코더의 입력부에 정답이 되는 문장 전체를 한번에 넣고 디코더의 출력과 한번에 비교 가능
인퍼런스 단계: 정답 문장x
만들어야 하는 문장의 길이 만큼 디코더가 반복 구조로 동작

모델 검증

[Abstractive summarization 방식의 텍스트 요약]

테스트 데이터 50개에 대해 실제 예약과 예측된 요약 비교

```
for i in range(50, 100):
    print("원문 : ", seq2text(encoder_input_test[i]))
    print("실제 요약 : ", seq2summary(decoder_input_test[i]))
    print("예측 요약 : ", decode_sequence(encoder_input_test[i].reshape(1, text_max_len)))
    print("₩n")
```

```
원문 :  great heavy chewer long lasting dogs love second one purchased seems one come
실제 요약 :  big chewer loves it
예측 요약 :   great product


원문 :  loved quick shipping terrific gift
실제 요약 :  taste fresh and so delish
예측 요약 :   not the best


원문 :  great gum chew hours still tastes great bad widely available used
실제 요약 :  minty sweet twist rocks
예측 요약 :   great gum


원문 :  first discovered pasta nyc restaurant first time pasta gluten free diet tried mom dish someone knows real thing tastes like absolutely amazed pasta gluten free actually preferred regular pasta give heavy feeling eating great basically recipe eating gluten free order case skip gluten free pastas try promise amazed
실제 요약 :  best gluten free pasta period
예측 요약 :   great pasta


원문 :  fee would recommend everyone excellent gift coffee drinker
실제 요약 :  yumm coffee
예측 요약 :   great coffee


원문 :  creamer fine creamers exploded box problem break open left whole box covered sticky go hand wash putting staff clients simply irritating
실제 요약 :  milk
예측 요약 :   not what expected


원문 :  first time tried brand love tried vita coco brands ok taste fact tasted flat stale brand fresh delicious enjoying much also tried brand taste similar would buy zico zico little cheaper great deal
실제 요약 :  taste fresh and so delish
예측 요약 :   not the best


원문 :  sure company thinking packaged half mine arrived leaking make matters worse ones opened tasted horrible way sweet ordered brands much happier packaging taste value never reorder
실제 요약 :  horrible packaging and taste
예측 요약 :   not bad


원문 :  product sweet also use nectar find sweeter looking something take place granular sugar think right stick nectar trying get away sweet low splenda
실제 요약 :  natural sweetner
예측 요약 :   good stuff


원문 :  family always loved black hard find area reason started using old fashioned bagged pop corn using covered skillet stove like microwave type try tell anyone family something different watch reactions popcorn huge popped kernals like regular yellow corn lot flavor
실제 요약 :  old style popcorn taste
예측 요약 :   great for on the go
```

PROJECT

 # 느낀점

[Abstractive summarization 방식의 텍스트 요약]

조금 어색한 요약도 보이지만 대부분 잘 맞는 요약
어떤 요약의 경우 원문에 없던 단어를 사용해서 요약을 하기도 함

Beam search, pre-trained word embedding, Transformer 등
좀 더 성능을 향상시킬 여러 개선 방안들이 존재