텍스트 요약
Text Summarization

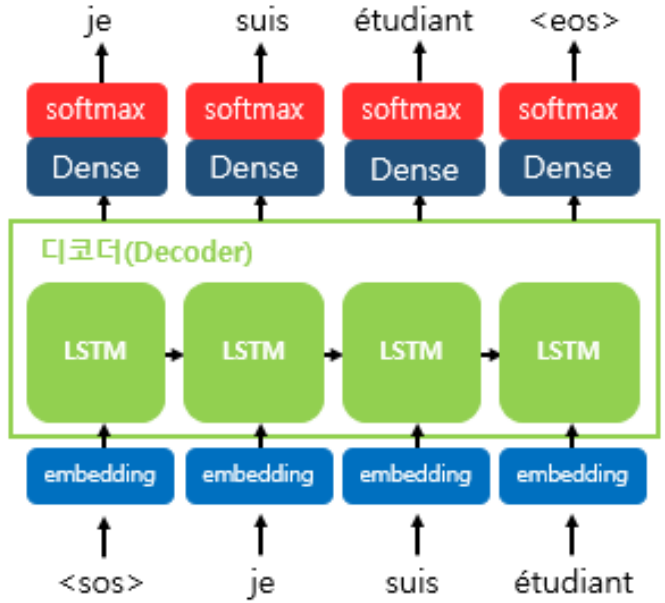
집현전 초급 10조
- 김동영 나건주 민선아 -

목차

들어가기에 앞서 …

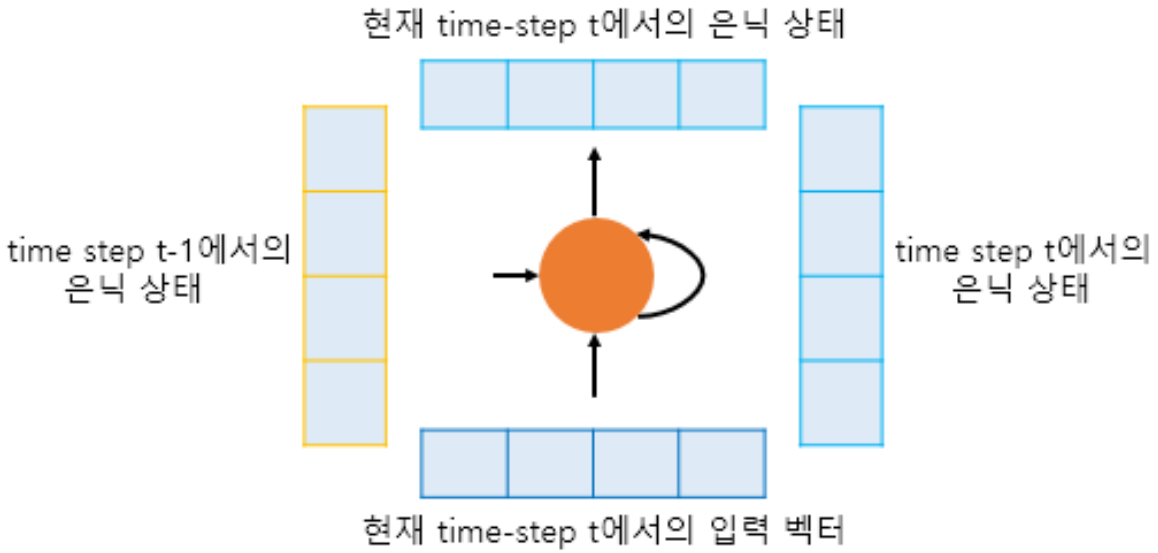
Sequence-to-Sequence

- seq2seq는 크게 인코더와 디코더라는 두 개의 모듈로 구성
- 두 개의 RNN 아키텍처
- RNN 셀의 입력(t 시점): t-1에서의 은닉 상태와 t에서의 입력 벡터
- RNN 셀의 입력(t+1 시점): 현재 t에서의 은닉 상태



컨텍스트 벡터

: 현재 시점 t에서의 은닉 상태는 과거 시점의 동일한
RNN 셀에서의 모든 은닉 상태의 값들의 영향을
누적해서 받아온 값
-> 입력 문장의 모든 단어 토큰들의 정보를 요약해서 담고있다



1. 어텐션을 이용한 텍스트 요약

1) 추출적 요약(extractive summarization)

https://summariz3.herokuapp.com/

- **중요한 핵심 문장 또는 단어구를 몇 개 뽑아서 이들로 구성된 요약문**
- 요약문의 문장이나 단어구들은 **전부 원문에 존재하던 문장들**

단점: 모델의 언어 표현 능력이 제한된다

1. 어텐션을 이용한 텍스트 요약

2) 추상적 요약(abstractive summarization)

- 원문에 없던 문장이라도 핵심 문맥을 반영한 새로운 문장을 생성해서 원문을 요약하는 방법
- 마치 사람이 요약하는 것 같은 방식
- 추출적 요약보다는 난이도가 높다

모델

- 주로 인공 신경망을 사용하며 대표적인 모델로 seq2seq 사용
- seq2seq와 같은 인공 신경망들은 기본적으로 지도 학습
- '원문' 뿐만 아니라 '실제 요약문'이라는 레이블 데이터 필요

단점; 데이터를 구성하는 것 자체가 하나의 부담

2. 실습

- 데이터셋

STANFORD NETWORK ANALYSIS PROJECT · UPDATED 5 YEARS AGO

1891 | New Notebook | Download (254 MB)

Amazon Fine Food Reviews

Analyze ~500,000 food reviews from Amazon

Data | Code (686) | Discussion (17) | Metadata

About Dataset

Context

This dataset consists of reviews of fine foods from amazon. The data span a period of more than 10 years, including all ~500,000 reviews up to October 2012. Reviews include product and user information, ratings, and a plain text review. It also includes reviews from all other Amazon categories.

Contents

- Reviews.csv: Pulled from the corresponding SQLite table named Reviews in database.sqlite
- database.sqlite: Contains the table 'Reviews'

Usability
7.94

License
CC0: Public Domain

Expected update frequency
Not specified

3. 요약 모델 설계 및 훈련

-> seq2seq + attention

- 필요한 도구들을 임포트합니다.

```
1 from tensorflow.keras.layers import Input, LSTM, Embedding, Dense, Concatenate
2 from tensorflow.keras.models import Model
3 from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint
```

3. 요약 모델 설계 및 훈련

- 인코더 설계: LSTM 층을 3개 쌓습니다.

```
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_state=True, return_sequences=True, dropout=0.4, recurrent_dropout=0.4)
encoder_outputs, state_h, state_c= encoder_lstm3(encoder_output2)

```

3. 요약 모델 설계 및 훈련

- 디코더 설계: 초기 상태(initial_state)를 인코더의 상태로 주어야 함

```
# 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])
```

- LSTM이 리턴하는 은닉 상태와 셀 상태인 state_h와 state_c를 버림

3. 요약 모델 설계 및 훈련

- 디코더의 출력층 설계

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

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

```
Model: "model"
__________________________________________________________________________________________________
 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),    394240      ['embedding[0][0]']
                                 (None, 256),
                                 (None, 256)]

 input_2 (InputLayer)           [(None, None)]       0           []

 lstm_1 (LSTM)                  [(None, 50, 256),    525312      ['lstm[0][0]']
                                 (None, 256),
                                 (None, 256)]

 embedding_1 (Embedding)        (None, None, 128)    256000      ['input_2[0][0]']

 lstm_2 (LSTM)                  [(None, 50, 256),    525312      ['lstm_1[0][0]']
                                 (None, 256),
                                 (None, 256)]

 lstm_3 (LSTM)                  [(None, None, 256),  394240      ['embedding_1[0][0]',
                                 (None, 256),                     'lstm_2[0][1]',
                                 (None, 256)]                     'lstm_2[0][2]']

 dense (Dense)                  (None, None, 2000)   514000      ['lstm_3[0][0]']

==================================================================================================
Total params: 3,633,104
Trainable params: 3,633,104
Non-trainable params: 0
__________________________________________________________________________________________________
```

총 3,633,104개의 매개변수를 가진
seq2seq 모델 설계

3. 요약 모델 설계 및 훈련

- 어텐션 메커니즘을 사용할 예정

->바다나우 어텐션

->깃허브에 작성된 어텐션 사용

```
# 바다나우 어텐션 메커니즘이 결합된 새로운 출력층을 설계
urllib.request.urlretrieve("https://raw.githubusercontent.com/ukairia777/tensorflow-nlp-tutorial/main/20.%20Text%20Summarization%20with%20Attention/attention.py",
                           filename="attention.py")
from attention import AttentionLayer
```

3. 요약 모델 설계 및 훈련

- 어텐션 메커니즘이 결합된 새로운 출력층 설계

- 소프트맥스 함수를 적용하여, 모든 값을 합하면 1이 되는 확률 분포를 얻어낸다

```
# 어텐션 층(어텐션 함수)
attn_layer = AttentionLayer(name='attention_layer')
attn_out, attn_states = attn_layer([encoder_outputs, decoder_outputs])

# 어텐션의 결과와 디코더의 hidden state들을 연결
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)

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

3. 요약 모델 설계 및 훈련

총 4,276,432개의 파라미터를
가진 모델 설계

```
Model: "model_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),    394240      ['embedding[0][0]']
                                 (None, 256),
                                 (None, 256)]

 input_2 (InputLayer)           [(None, None)]       0           []

 lstm_1 (LSTM)                  [(None, 50, 256),    525312      ['lstm[0][0]']
                                 (None, 256),
                                 (None, 256)]

 embedding_1 (Embedding)        (None, None, 128)    256000      ['input_2[0][0]']

 lstm_2 (LSTM)                  [(None, 50, 256),    525312      ['lstm_1[0][0]']
                                 (None, 256),
                                 (None, 256)]

 lstm_3 (LSTM)                  [(None, None, 256),  394240      ['embedding_1[0][0]',
                                 (None, 256),                     'lstm_2[0][1]',
                                 (None, 256)]                     'lstm_2[0][2]']

 attention_layer (AttentionLaye  ((None, None, 256),  131328     ['lstm_2[0][0]',
 r)                              (None, None, 50))                'lstm_3[0][0]']

 concat_layer (Concatenate)     (None, None, 512)    0           ['lstm_3[0][0]',
                                                                  'attention_layer[0][0]']

 dense_1 (Dense)                (None, None, 2000)   1026000     ['concat_layer[0][0]']

==================================================================================================
Total params: 4,276,432
Trainable params: 4,276,432
Non-trainable params: 0
__________________________________________________________________________________________________
```

3. 요약 모델 설계 및 훈련

- 모델 컴파일

```
model.compile(optimizer='rmsprop', loss='sparse_categorical_crossentropy')
```

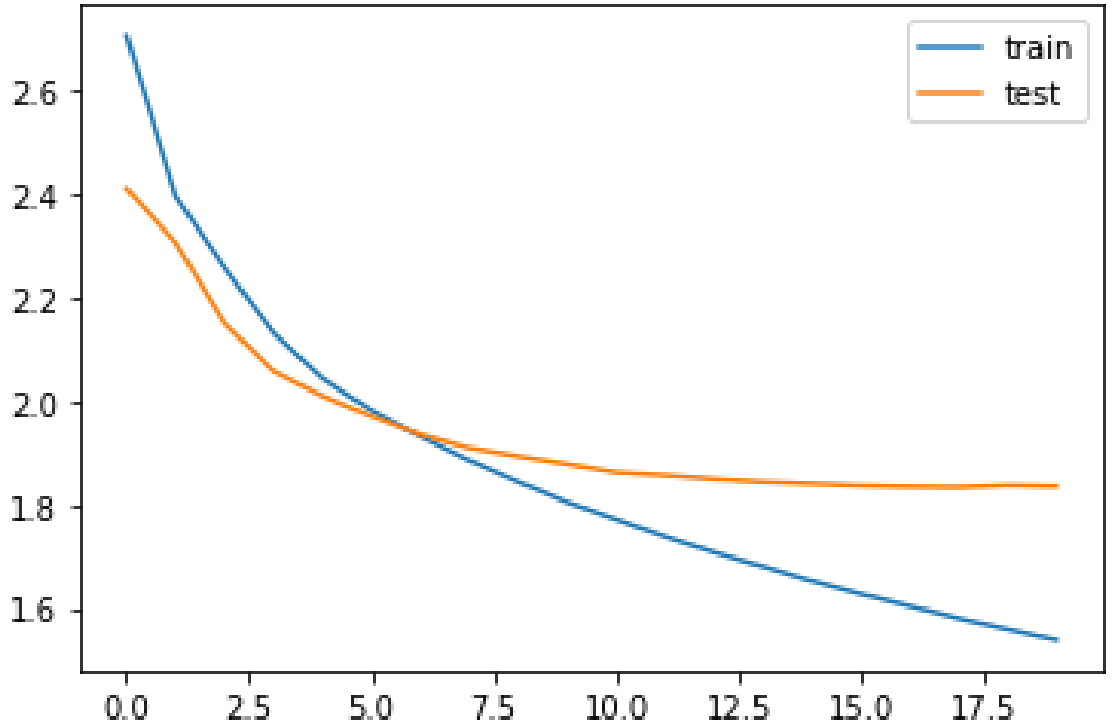
- EarlyStopping 조건 설정 및 모델 학습

```
# 조기 종료 조건 설정
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 = 50)
```

3. 요약 모델 설계 및 훈련

- 학습 과정에서 훈련 데이터와 테스트 데이터의 loss값 시각화

```
1 plt.plot(history.history['loss'], label='train')
2 plt.plot(history.history['val_loss'], label='test')
3 plt.legend()
4 plt.show()
```



4. seq2seq + attention으로 요약 모델 테스트

- 테스트를 위해 필요한 3개의 사전 생성

```
1 src_index_to_word = src_tokenizer.index_word # 원문 단어 집합에서 정수 -> 단어를 얻음
2 tar_word_to_index = tar_tokenizer.word_index # 요약 단어 집합에서 단어 -> 정수를 얻음
3 tar_index_to_word = tar_tokenizer.index_word # 요약 단어 집합에서 정수 -> 단어를 얻음
```

- src_tokenizer.word_index에 단어 고유의 정수 저장됨

{1: 'like',
2: 'good',
3: 'great',
4: 'taste',
5: 'product',}

4. seq2seq + attention으로 요약 모델 테스트

- 새로운 seq2seq 모델 설계
- 인코더 정의

-> seq2seq는 훈련 단계와 테스트 단계의 동작이 다름

-> 테스트 단계의 모델 별도로 설계

```
1 # 인코더 설계
2 encoder_model = Model(inputs=encoder_inputs, outputs=[encoder_outputs, state_h, state_c])
```

- **LSTM은 은닉 상태와 셀 상태라는 두 가지 상태를 가진다**

- state_h: 은닉상태
- state_c: t시점의 셀 상태

4. seq2seq + attention으로 요약 모델 테스트

- 테스트 단계의 디코더 설계

```
# 이전 시점의 상태들을 저장하는 텐서
decoder_state_input_h = Input(shape=(hidden_size,))
decoder_state_input_c = Input(shape=(hidden_size,))

dec_emb2 = dec_emb_layer(decoder_inputs)
# 문장의 다음 단어를 예측하기 위해서 초기 상태(initial_state)를 이전 시점의 상태로 사용. 이는 뒤의 함수 decode_sequence()에 구현
# 훈련 과정에서와 달리 LSTM의 리턴하는 은닉 상태와 셀 상태인 state_h와 state_c를 버리지 않음.
decoder_outputs2, state_h2, state_c2 = decoder_lstm(dec_emb2, initial_state=[decoder_state_input_h, decoder_state_input_c])
```

- 문장의 다음 단어를 예측하기 위해서 초기 상태(initial_state)를 이전 시점의 상태로 사용.
- 훈련 단계와 달리, LSTM의 리턴하는 은닉 상태와 셀 상태인 state_h와 state_c를 버리지 않음.

4. seq2seq + attention으로 요약 모델 테스트

- 테스트 단계의 디코더 설계

```
# 어텐션 함수
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])

# 디코더의 출력층
decoder_outputs2 = decoder_softmax_layer(decoder_inf_concat)

# 최종 디코더 모델
decoder_model = Model(
    [decoder_inputs] + [decoder_hidden_state_input,decoder_state_input_h, decoder_state_input_c],
    [decoder_outputs2] + [state_h2, state_c2])
```

4. seq2seq + attention으로 요약 모델 테스트

- 테스트를 위해 사용되는 함수 decode_sequence를 설계

```python
def decode_sequence(input_seq):
    # 입력으로부터 인코더의 상태를 얻음
    e_out, e_h, e_c = encoder_model.predict(input_seq)

     # <SOS>에 해당하는 토큰 생성
    target_seq = np.zeros((1,1))
    target_seq[0, 0] = tar_word_to_index['sostoken']

    stop_condition = False
    decoded_sentence = ''
    while not stop_condition: # stop_condition이 True가 될 때까지 루프 반복

        output_tokens, h, c = decoder_model.predict([target_seq] + [e_out, e_h, e_c])
        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

        #  <eos>에 도달하거나 최대 길이를 넘으면 중단.
        if (sampled_token == 'eostoken'  or len(decoded_sentence.split()) >= (summary_max_len-1)):
            stop_condition = True

        # 길이가 1인 타겟 시퀀스를 업데이트
        target_seq = np.zeros((1,1))
        target_seq[0, 0] = sampled_token_index

        # 상태를 업데이트 합니다.
        e_h, e_c = h, c

    return decoded_sentence
```

4. seq2seq + attention으로 요약 모델 테스트

- **정수 시퀀스를 텍스트 시퀀스로 만드는 함수 설계**

-> 테스트 단계에서 원문과 실제 요약문, 예측 요약문을 비교하기 위함

```python
# 원문의 정수 시퀀스를 텍스트 시퀀스로 변환
def seq2text(input_seq):
    sentence=''
    for i in input_seq:
        if(i!=0):
            sentence = sentence + src_index_to_word[i]+' '
    return sentence

# 요약문의 정수 시퀀스를 텍스트 시퀀스로 변환
def seq2summary(input_seq):
    sentence=''
    for i in input_seq:
        if((i!=0 and i!=tar_word_to_index['sostoken']) and i!=tar_word_to_index['eostoken']):
            sentence = sentence + tar_index_to_word[i] + ' '
    return sentence
```

4. seq2seq + attention으로 요약 모델 테스트

- 테스트 샘플 중 500번부터 1000번까지 테스트

```
for i in range(500, 1000):
    print("원문 : ",seq2text(encoder_input_test[i]))
    print("실제 요약문 :",seq2summary(decoder_input_test[i]))
    print("예측 요약문 :",decode_sequence(encoder_input_test[i].reshape(1, text_max_len)))
    print("₩n")
```

4. seq2seq + attention으로 요약 모델 테스트

- 결과 출력

```
원문 :  really like bars taste texture really good however consistency bars somewhat sticky learned eat wipe hands wash
실제 요약문 : good alternative for gluten free snack
예측 요약문 :  not bad


원문 :  tried costco could find anymore finally found amazon loves help breath teeth
실제 요약문 : hard to find she loves them
예측 요약문 :  great product
```

원문 : cookies taste great kids love feel good giving healthier snack reason give star becuase bag comes servings makes little less convenient use traveling extra bowl baggie problem easily solved definitely buy

실제 요약문 : kids love them

예측 요약문 : great for snacks

원문 : tea works calms makes sleep better may work everyone thank

실제 요약문 : relaxing

예측 요약문 : great tea

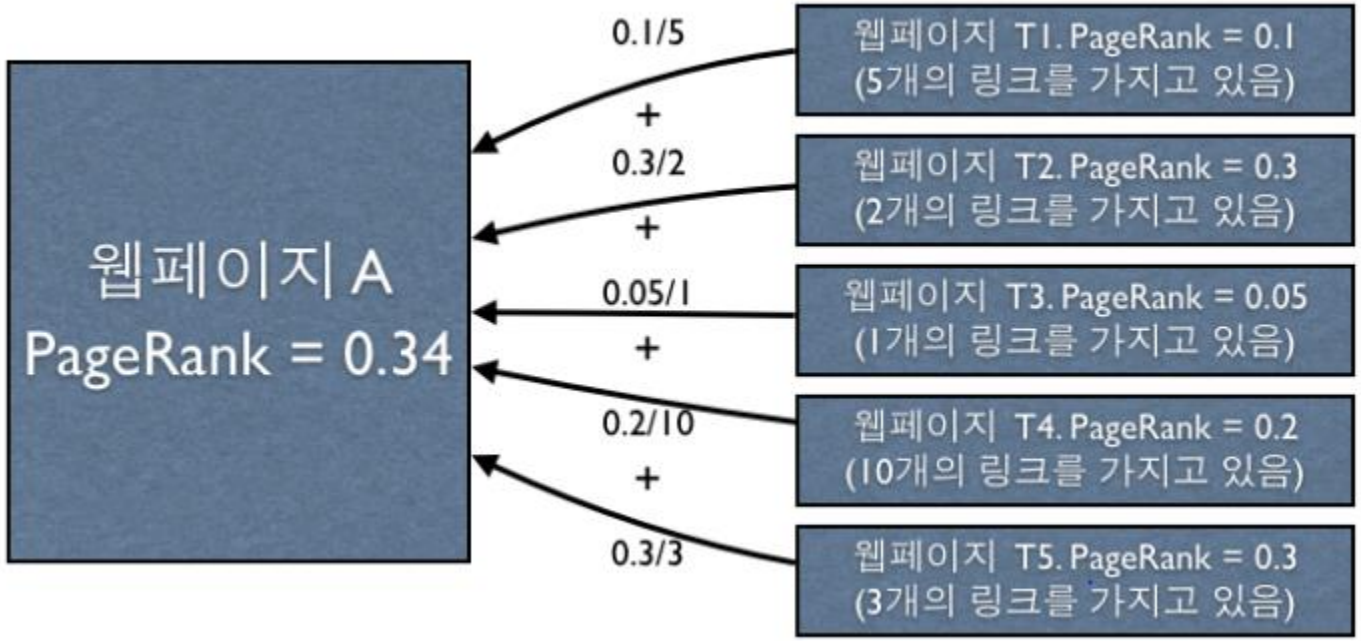
문장 임베딩 기반 텍스트 랭크

텍스트 랭크

- 페이지 랭크 기반으로 한 텍스트 추출적요약
- 특정 단어가 다른 단어와 얼마만큼의 관계를 맺고 있는 지를 계산하여 각 단어의 중요도를 선정하는 알고리즘

페이지 랭크

$$PR(A) = (1-d)/N + d\,(PR(T1)/C(T1) + \ldots + PR(Tn)/C(Tn))$$



문장 임베딩 기반 텍스트 랭크

- 각 문장을 벡터화 된 단어들을 평균내서 벡터화
- 벡터화된 문장을 기준으로 하여 중요도 평가(텍스트 랭크)
- 중요도 순으로 원하는 개수의 문장으로 해당 글 요약문으로 설정

사전학습된 glove 단어 벡터 사용

```
import numpy as np
import gensim
from urllib.request import urlretrieve, urlopen
import gzip
import zipfile

urlretrieve("http://nlp.stanford.edu/data/glove.6B.zip", filename="glove.6B.zip")
zf = zipfile.ZipFile('glove.6B.zip')
zf.extractall()
zf.close()
```

```
glove_dict = dict()
f = open('glove.6B.100d.txt', encoding="utf8") # 100차원의 GloVe 벡터를 사용

for line in f:
    word_vector = line.split()
    word = word_vector[0]
    word_vector_arr = np.asarray(word_vector[1:], dtype='float32') # 100개의 값을 가지는 array로 변환
    glove_dict[word] = word_vector_arr
f.close()
```

glove 단어벡터 예시

```
glove_dict['dog']
```

```
array([-1.1752e-01,  9.7272e-01, -2.9021e-01,  2.5914e-01, -4.2644e-01,
       -1.4736e-01,  6.2162e-01, -1.2626e-04, -1.5885e+00,  3.0567e-01,
        1.0951e+00, -8.7477e-01, -8.0877e-01,  3.7008e-01,  9.8138e-01,
       -8.7040e-01, -6.9067e-01, -1.0754e-01,  2.3355e-02,  3.2394e-01,
        1.0950e-01, -2.8412e-01,  1.1034e+00,  5.8392e-01,  2.8271e-01,
        7.4784e-01,  8.0530e-01,  1.0216e-01, -1.5370e-02,  1.2624e-01,
        2.2516e-01,  6.6591e-01, -5.9523e-01,  5.1642e-01,  3.9173e-01,
        6.7354e-01,  7.1769e-01,  5.9060e-01, -1.2889e-01,  5.2605e-01,
        1.9100e-01, -1.1437e-01, -3.9725e-01, -8.9456e-01,  8.1271e-02,
        6.3898e-01, -5.8567e-01,  2.3063e-01, -1.1845e-01, -4.9358e-01,
        1.7417e-01,  6.0707e-01,  1.9180e-01, -4.6645e-01, -1.1239e+00,
       -2.2339e+00, -9.5387e-01, -3.0962e-01,  1.5652e+00,  1.0899e+00,
       -3.6337e-01, -8.0285e-01, -6.3040e-01,  3.6323e-01,  6.9253e-01,
        3.2473e-01,  1.0351e+00,  6.3678e-01,  6.3749e-01,  1.8845e-01,
       -5.8410e-01, -3.3264e-02,  5.0673e-02,  8.6712e-03,  5.9612e-01,
       -6.4531e-01, -3.8498e-01, -1.9530e-01, -4.1893e-01, -2.9927e-01,
       -5.3108e-01, -7.0176e-01, -2.1998e-01, -1.2016e+00, -1.8840e-01,
        8.6845e-01, -3.5293e-01, -6.1548e-01,  1.7660e-01,  9.9418e-01,
       -5.5700e-01,  7.3609e-01,  3.7205e-01,  6.4293e-02, -1.2736e-01,
        3.1447e-01,  2.8212e-01, -5.0598e-01, -2.8476e-01, -7.0045e-01],
      dtype=float32)
```

문장 벡터화

- 입력된 문장의 단어벡터 들을 합한 후 문장길이만큼 나눔

```
embedding_dim = 100
zero_vector = np.zeros(embedding_dim)

# 단어 벡터의 평균으로부터 문장 벡터를 얻는다.
def calculate_sentence_vector(sentence):
  if len(sentence) != 0:
    return sum([glove_dict.get(word, zero_vector)
                for word in sentence])/len(sentence)
  else:
    return zero_vector
```

문장 벡터화 예시

```
eng_sent = ['i', 'am', 'a', 'student']
sentence_vector = calculate_sentence_vector(eng_sent)
print(sentence_vector)
```

```
[ 0.08464026  0.31403124 -0.06504749 -0.20965815 -0.49992     0.6679375
  0.11522973  0.3484675  -0.02958     0.27833998  0.3011375   0.16529718
  0.26805526 -0.01858751 -0.0573175  -0.3731975  -0.110595   -0.06706676
 -0.465712    0.41784653 -0.007707    0.45923424 -0.07589749 -0.23346025
  0.31245476  0.02771001 -0.03991751 -0.5542325   0.39761627  0.2776575
 -0.449035    1.0627425   0.2929855   0.1062125  -0.04474699  0.04396251
 -0.3561775   0.36051202  0.46784502 -0.19585249 -0.44674248 -0.21592925
  0.42044201 -0.4068225  -0.17416498 -0.34669     0.40123424 -0.191755
 -0.121475   -0.94957    -0.28199875 -0.33948424  0.1917175   0.78151
 -0.5357438  -2.5141      0.19787325  0.14861248  1.622025    0.3003325
  0.1468165   0.7729645  -0.81665254 -0.19985661  0.7653775   0.03366375
  0.4760825   0.51388496 -0.08364251  0.17495     0.0282825  -0.2972625
  0.12549752 -0.33907974  0.0770525   0.3684415   0.3105415  -0.215221
 -0.84261    -0.45005     0.42610922 -0.25839123 -0.32755527 -0.28227624
 -1.286835   -0.1366575  -0.02663499 -0.3336375  -0.287886   -0.6108775
  0.17296     0.10157625  0.28758746  0.41235426 -0.44530374  0.09792
 -0.0446425  -0.45058003  0.4103145   0.21527499]
```

텍스트 요약 실습-데이터 불러오기

- 테니스 관련 기사 사용

```
urlretrieve("https://raw.githubusercontent.com/prateekjoshi565/textrank_text_summarization/master/tennis_articles_v4.csv",
            filename="tennis_articles_v4.csv")
data = pd.read_csv("tennis_articles_v4.csv")
data.head()
```

	article_id	article_text	source
0	1	Maria Sharapova has basically no friends as te...	https://www.tennisworldusa.org/tennis/news/Mar...
1	2	BASEL, Switzerland (AP), Roger Federer advance...	http://www.tennis.com/pro-game/2018/10/copil-s...
2	3	Roger Federer has revealed that organisers of ...	https://scroll.in/field/899938/tennis-roger-fe...
3	4	Kei Nishikori will try to end his long losing ...	http://www.tennis.com/pro-game/2018/10/nishiko...
4	5	Federer, 37, first broke through on tour over ...	https://www.express.co.uk/sport/tennis/1036101...

문장 단위로 글 나누기

- nltk.tokenize의 sent_tokenize사용

```
nltk.download('punkt')


from nltk.tokenize import sent_tokenize
data = data[['article_text']]
data['sentences'] = data['article_text'].apply(sent_tokenize)
data
```

단어 토큰화 및 전처리

```
from nltk.tokenize import word_tokenize
nltk.download('stopwords')
stop_words = stopwords.words('english')


# 토큰화 함수
def tokenization(sentences):
    return [word_tokenize(sentence) for sentence in sentences]

# 전처리 함수
def preprocess_sentence(sentence):
  # 영어를 제외한 숫자, 특수 문자 등은 전부 제거. 모든 알파벳은 소문자화
  sentence = [re.sub(r'[^a-zA-z₩s]', '', word).lower() for word in sentence]
  # 불용어가 아니면서 단어가 실제로 존재해야 한다.
  return [word for word in sentence if word not in stop_words and word]

# 위 전처리 함수를 모든 문장에 대해서 수행. 이 함수를 호출하면 모든 행에 대해서 수행.
def preprocess_sentences(sentences):
    return [preprocess_sentence(sentence) for sentence in sentences]
```

문장 벡터화

- 각 기사의 문장들을 단어 벡터들의 평균값으로 벡터화

```
embedding_dim = 100
zero_vector = np.zeros(embedding_dim)

# 단어 벡터의 평균으로부터 문장 벡터를 얻는다.
def calculate_sentence_vector(sentence):
  if len(sentence) != 0:
    return sum([glove_dict.get(word, zero_vector)
                for word in sentence])/len(sentence)
  else:
    return zero_vector

# 각 문장에 대해서 문장 벡터를 반환
def sentences_to_vectors(sentences):
    return [calculate_sentence_vector(sentence)
            for sentence in sentences]
```

문장 간의 유사도 계산

- 각 기사내의 문장 벡터들을 코사인 유사도 검사를 통해 유사도 행렬 생성

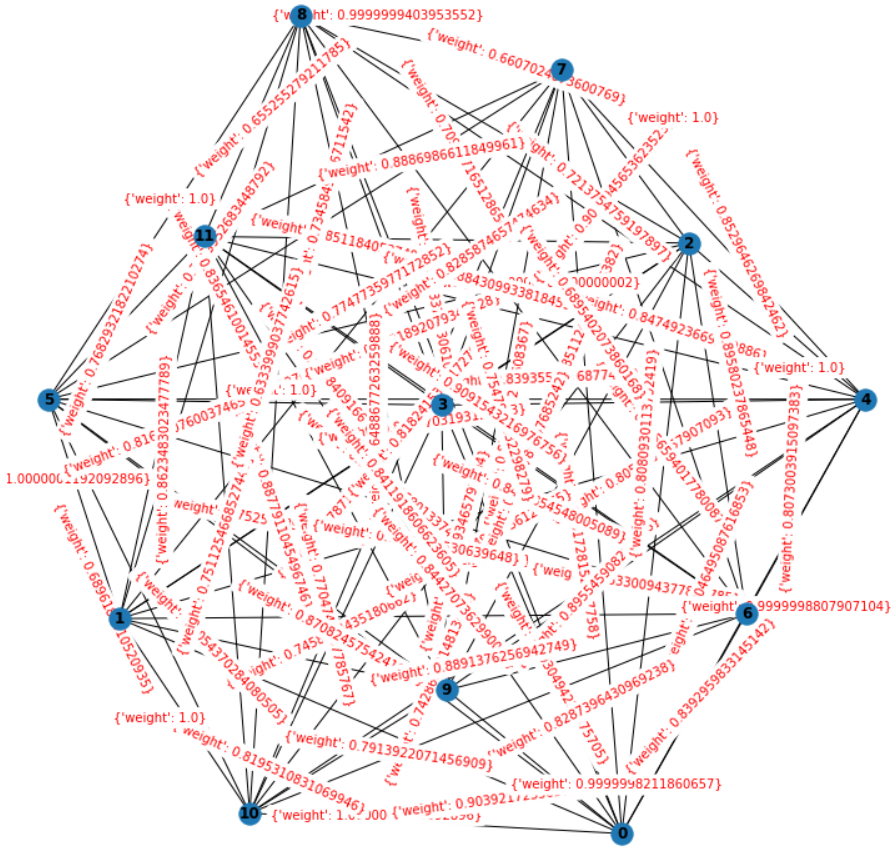
```
def similarity_matrix(sentence_embedding):
  sim_mat = np.zeros([len(sentence_embedding), len(sentence_embedding)])
  for i in range(len(sentence_embedding)):
      for j in range(len(sentence_embedding)):
        sim_mat[i][j] = cosine_similarity(sentence_embedding[i].reshape(1, embedding_dim),
                                          sentence_embedding[j].reshape(1, embedding_dim))[0,0]
  return sim_mat
```

```
data['SimMatrix'] = data['SentenceEmbedding'].apply(similarity_matrix)
```

```
print('두번째 샘플의 문장 개수 :',len(data['tokenized_sentences'][1]))
print('두번째 샘플의 문장 벡터가 모인 문장 행렬의 크기(shape) :',np.shape(data['SentenceEmbedding'][1]))
print('두번째 샘플의 유사도 행렬의 크기(shape) :',data['SimMatrix'][1].shape)
```

```
두번째 샘플의 문장 개수 : 12
두번째 샘플의 문장 벡터가 모인 문장 행렬의 크기(shape) : (12, 100)
두번째 샘플의 유사도 행렬의 크기(shape) : (12, 12)
```

유사도 행렬 그래프 시각화(nx)



```python
def draw_graphs(sim_matrix):
  nx_graph = nx.from_numpy_array(sim_matrix)
  plt.figure(figsize=(10, 10))
  pos = nx.spring_layout(nx_graph)
  nx.draw(nx_graph, with_labels=True, font_weight='bold')
  nx.draw_networkx_edge_labels(nx_graph,pos,font_color='red')
  plt.show()
```

문장간 유사도 기반 페이지랭크 계산

```python
def calculate_score(sim_matrix):
    nx_graph = nx.from_numpy_array(sim_matrix)
    scores = nx.pagerank(nx_graph)
    return scores

data['score'] = data['SimMatrix'].apply(calculate_score)
```

```
data['score'][1]
```

```
{0: 0.08315094474060455,
 1: 0.08498611405296501,
 2: 0.08555019786198463,
 3: 0.08383717299575927,
 4: 0.08137940307911188,
 5: 0.08439285067975581,
 6: 0.08507725735628792,
 7: 0.08092839280412682,
 8: 0.07454046000848007,
 9: 0.08535836572027003,
 10: 0.0849824249168908,
 11: 0.08581641578375629}
```

기사내 문장 순위 기반 요약 문장 선정

```
def ranked_sentences(sentences, scores, n=3):
    top_scores = sorted(((scores[i],s)
                         for i,s in enumerate(sentences)),
                                reverse=True)
    top_n_sentences = [sentence
                       for score,sentence in top_scores[:n]]
    return " ".join(top_n_sentences)

data['summary'] = data.apply(lambda x:
                             ranked_sentences(x.sentences,
                             x.score), axis=1)
```

결과

1 번 문서

원문 : Maria Sharapova has basically no friends as tennis players on the WTA Tour. The Russian player has no problems in openly speaking about it and in a recent interview she said: 'I don't really hide any feelings too much. I think everyone knows this is my job here. When I'm on the courts or when I'm on the court playing, I'm a competitor and I want to beat every single person whether they're in the locker room or across the net.So I'm not the one to strike up a conversation about the weather and know that in the next few minutes I have to go and try to win a tennis match. I'm a pretty competitive girl. I say my hellos, but I'm not sending any players flowers as well. Uhm, I'm not really friendly or close to many players. I have not a lot of friends away from the courts.' When she said she is not really close to a lot of players, is that something strategic that she is doing? Is it different on the men's tour than the women's tour? 'No, not at all. I think just because you're in the same sport doesn't mean that you have to be friends with everyone just because you're categorized, you're a tennis player, so you're going to get along with tennis players. I think every person has different interests. I have friends that have completely different jobs and interests, and I've met them in very different parts of my life. I think everyone just thinks because we're tennis players we should be the greatest of friends. But ultimately tennis is just a very small part of what we do. There are so many other things that we're interested in, that we do.'

요약 : I think just because you're in the same sport doesn't mean that you have to be friends with everyone just because you're categorized, you're a tennis player, so you're going to get along with tennis players. When I'm on the courts or when I'm on the court playing, I'm a competitor and I want to beat every single person whether they're in the locker room or across the net.So I'm not the one to strike up a conversation about the weather and know that in the next few minutes I have to go and try to win a tennis match. I think everyone just thinks because we're tennis players we should be the greatest of friends.

어텐션을 이용한 텍스트 요약 (한국어)

데이터셋

WikiLingua: A Multilingual Abstractive Summarization Dataset

UPDATE:
We have created new Train/Test splits for all 17 languages that can be downloaded here. These splits were created to ensure that there is no (document, summary) pair overlap across any of the 17 languages so that they can be safely used for multilingual evaluations.

This repo contains dataset introduced in the following paper:

WikiLingua: A New Benchmark Dataset for Multilingual Abstractive Summarization

Download the dataset using this link.

Please refer to this Collab notebook to see how to align articles in other languages with the parallel English articles.

Korean	12,189

데이터셋


```
{'부정적인 생각 제거하기': {'summary': '부정적인 생각은 즉시 멈추기. 자신의 말에 신경쓰기. 사용하는 언어에서 과도하게 부정적인 단어 빼기. 부정을 긍정으로 바꾸기.',
  'document': '일단 부정적인 생각이 들었다는 점을 인식했다면, 곧바로 자신에게 긍정적인 말을 해서 이를 상쇄시키도록 하자. "아침이 너무 힘들어"라고 말하는 대신 "아침은 힘들지만, 하루를 보내면서 점점 더 나아지겠지"라고 말해보자. 항상 긍정적인 사고방식을 유지하는 것이 중요하다. 이 방법을 실생활에 적용시키는 것이 힘들다면 다음처럼 생각을 바꿔본다: 당신이 친한 친구에게 말하지 않을 내용을 자신에게도 말하지 않는다. 항상 긍정적으로 생각하는 것이 습관화될 때까지 계속하라. 절대, 무조건과 같은 말을 자주 사용하는가? 그러면 고쳐라. 일반적으로 "난 절대 이 일을 끝내지 못할 거야", "난 항상 일을 망쳐"와 같은 절대적인 문장, 단어는 과장되는 경우가 많다. 게다가 과장시킨 문장을 고치려면 근본부터 바꿔야 하니 수고도 많이 든다. 타인에게 언성을 높이는 것도 여기에 포함된다는 사실을 알고 있자. 당신이 스스로에게 말하는 것 역시 포함된다. 말뿐만 아니라 생각까지 말이다. "끔찍해", "재앙", "망했어" 등의 과장된 단어는 단순한 불편함이나 짜증에 적용되어서는 안 될 말이다. 말하기 전에 먼저 한 차례 생각을 해보고 더 건강하고 긍정적인 단어로 바꿔보는 것이 긍정적인 사고방식을 유지하는 데 도움이 될 것이다. 앞서 언급한 단어나 문장을 사용했다면, 말은 어쩔 수 없으니 생각으로라도 바꿔보도록 한다. "끔찍함"은 "불운" 또는 "의외의 결과" 등으로 바꿀 수 있으며, "재앙"은 "불편함"이나 "도전"으로 바꾸면 된다. 어떤 상황은 매우 좋을 수도 있으며 매우 나쁠 수도 있다. 하지만 나쁜 상황에서 긍정적인 요소를 찾아내는 것이 아무래도 자신에게 부담을 덜 주는 방법이 아니겠는가? 이제부터라도 부정적인 생각이 들면, 그 즉시 멈추고 긍정적인 요소를 찾으려 노력해보자. 한 가지 예로 갑자기 컴퓨터가 멈춰서 내부 부품을 바꿔야 한다면, 무조건 불평하는 것보다는 다음처럼 생각할 수 있을 것이다. "그래도 이 기회에 컴퓨터 부품에 대해 더 잘 알고 부품을 교체하는 기술을 배웠으니 이득을 본 거나 마찬가지네."',
  'english_section_name': 'Eliminating Negative Thinking',
  'english_url': 'https://www.wikihow.com/Control-Negative-Thoughts'},
 '전문가의 도움 받기': {'summary': '상담사나 치료사의 도움 받기. 상담 약속 잡기. 상담사에게 자신의 부정적인 생각에 대해 설명하기. 필요하면 추가 약속 잡기.',
  'document': '부정적인 생각을 스스로 조절하지 못한다고 느끼면 전문가의 도움을 받는 것이 가장 현명할 것이다. 상담은 당신을 큰 폭으로 도와줄 수 있으며, 긍정적으로 생각하는 요령까지 배울 수도 있을 것이다. 특히 인지 행동 치료를 전문으로 하는 상담사를 찾아보는 것이 좋다. 당신의 사고방식을 긍정적으로 바꿔줄 것이다. 믿을 수 있는 상담사를 찾기 위해 친구나 이전에 상담을 받아봤던 사람에게 물어본다. 아니면 의사에게 추천을 받아본다. 당신의 마음을 진찰받는다고 생각하라. 상담받는 것이 불편하다면 의무적으로 하는 것이 아니니 잠시만 상담을 받아도 된다. 아니면 한 번만 상담받고 정기적으로 가지 않아도 되니 시도해보는 것이 나쁘지는 않을 것이다. 열린 사고를 가지고 약속을 잡도록 하자. 상담사가 당신을 도와줄 것이라는 희망을 가져라. 큰 도움이 되지 않는다면, 당신을 더 편안하게 만들어 줄 수 있는 상담사를 찾아보라. 상담한 내용은 절대로 비밀이 보장되니 마음 속에 있는 것을 정직하게 털어놓는 것이 좋다. 당신이 상담사에게 진실되게 말할 수록 상담사가 당신에게 더 효과적인 해결책을 제시할 수 있을 것이다. 부정적인 생각이 당신에게 어떤 감정을 들게 하는지도 설명한다. 당신이 얼마나 자주 이를 경험하고 평소에 어떻게 부정적인 생각을 해소하는지도 설명하도록 한다. 만약 상담사가 편안하게 느껴졌다면 추가로 한두 번 정도 약속을 잡도록 한다. 아마 당신의 부정적인 사고방식을 완전히 고치기까지 어느 정도 시간이 걸릴 수도 있을 것이다. 상담사가 도움이 되지 않았다고 해서 실망하지 마라. 단순히 다른 상담사를 찾으면 될 일이다.',
  'english_section_name': 'Seeking Outside Counsel',
  'english_url': 'https://www.wikihow.com/Control-Negative-Thoughts'},
 '긍정적인 하루 보내기': {'summary': '긍정적인 요소 5가지를 생각하면서 하루 시작하기. 하루 즐기기. 건강한 습관 기르기. 주변 환경 조절하기. 저녁에는 긴장을 풀고 스트레스 해소하기.',
  'document': '꼭 숭고하거나 야망에 가득찬 생각이 아니어도 좋다. 단순히 맛있는 커피나 좋아하는 음악을 듣는 것 등도 충분히 긍정적인 요소가 될 수 있다. 이런 것들을 생각하고 입 밖으로 꺼내 말하는 것은 하루를 긍정적으로 시작하게 도와준다. 또한 일과 중에 긍정적인 기운이 자리잡아 부정적인 생각이 잘 떠오르지 않게 되기도 한다. 긍정적인 문장을 말로 꺼내는 것이 불필요하게 느껴질 수도 있다. 하지만 연구 결과에 따르면 말로 긍정적인 요소를 말하는 것이 실제로 이를 믿도록 뇌를 설득한다고 한다. 따라서 긍정적인 생각을 직접 말로 하면 생각만 할 때보다 더 큰 행복감을 느낄 수 있을 것이다. 물론 바쁠 수도 있다. 하지만 일상 속의 작은 것들이 당신의 기분을 좋게 만들어 부정적인 습관이 머물지 않게 도와줄 수 있다. 먼저 상황이나 상대방의 말을 너무 심각하게 받아들이지 않는 것에서 시작한다. 긴장을 풀고 미소를 짓고 웃어라. 상대방, 또는 주변 환경과 소통하는 시간을 가지고, 긍정적이고 당신을 지지하는 사람들과 시간을 보내라. 스트레스를 받는다는 기분이 든다면, 잠시 휴식을 취하고 스트레스 원인을 생각하는 대신 다른 즐거운 것들을 생각해본다. 부정적인 생각과 스트레스는 서로를 강화해 악순환을 만든다. 부정적인 생각은 스트레스를 형성하고, 건강하지 않은 생활 습관은 문제를 일으키기 마련이다. 건강하고 영양소가 풍부한 식사를 하고 규칙적인 운동과 충분한 수면 시간을 가지도록 하자. 매일 운동하는 것은 부정적인 생각을 떨쳐내는 데 큰 도움이 될 수 있다. 담배는 피지 않는 것이 좋다. 마찬가지로 술이나 약물에 의존하는 것은 몸에 부담을 주니 하지 않도록 한다. 당신은 자기 생각을 조절할 수 있다. 불행하다는 생각이 들면 주변 환경을 바꿔보라. 음악을 틀어 긴장을 풀고 너무 덥거나 춥지 않게 옷을 갖춰 입은 다음, 조명을 원하는 대로 조절해 무력감과 스트레스를 이겨낼 수 있는 기운을 얻어보도록 하자. 환경에 변화를 주었으면, 자신의 기분을 바꾼 것에 대해 스스로 축하한다. 활발하게 사고방식을 바꾸려 노력하는 것은 부정적인 생각이 자리잡을 수 없게 한다. 조용하고 편안한 장소를 찾아 몸의 긴장을 풀어준다. 하루를 되돌아보며 일과 속에서 경험했던 긍정적인 요소 중 5가지를 골라본다. 생각나는 것들을 일기에 적고 큰 소리로 읽어본다. 당신이 감사하는 것을 목록으로 만들어도 좋다. 이를 통해 일상 속에서 긍정적인 요소를 찾는 연습을 할 수 있다.',
  'english_section_name': 'Making a Positive Day',
  'english_url': 'https://www.wikihow.com/Control-Negative-Thoughts'}}
```

불용어

Stopwords Korean (KO)

build passing

The most comprehensive collection of stopwords for the korean language.

A multiple language collection is also available.

Usage

The collection comes in a JSON format and a text format. You are free to use this collection any way you like. It is only currently published on npm and bower.

```
$ npm install stopwords-ko
```

```
$ bower install stopwords-ko
```

```
len(stopwords)
```

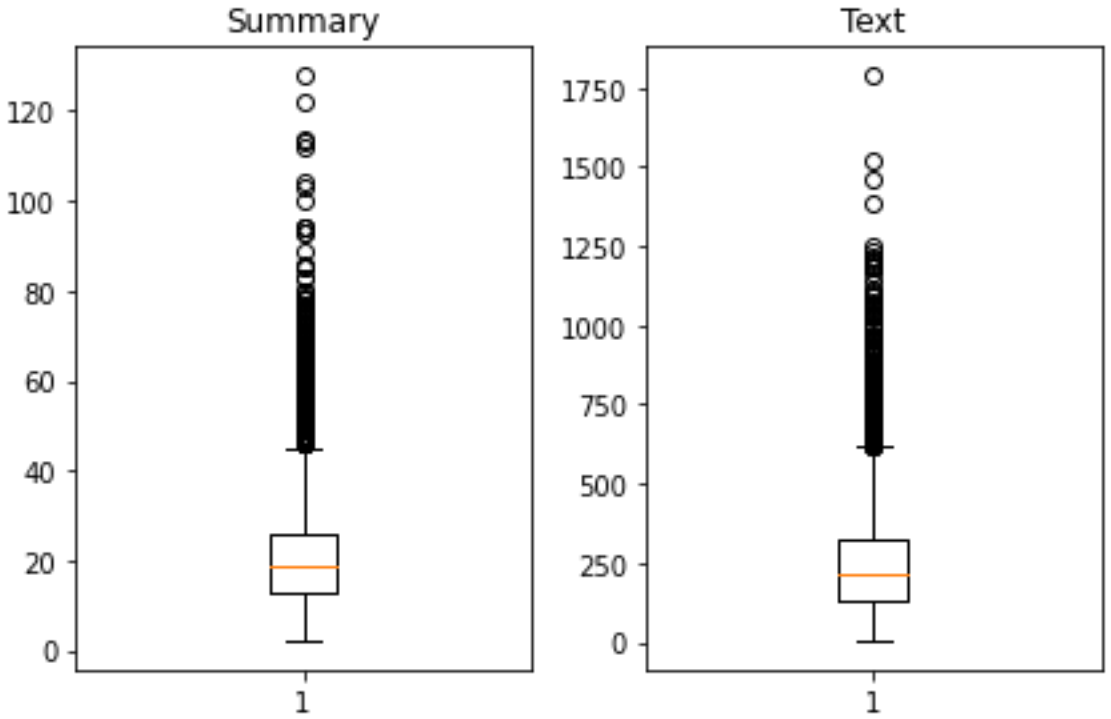
```
595
```

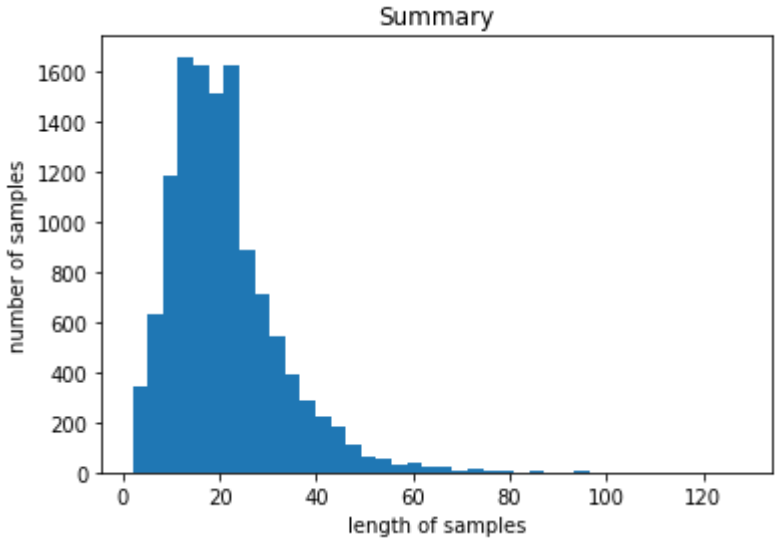
전처리

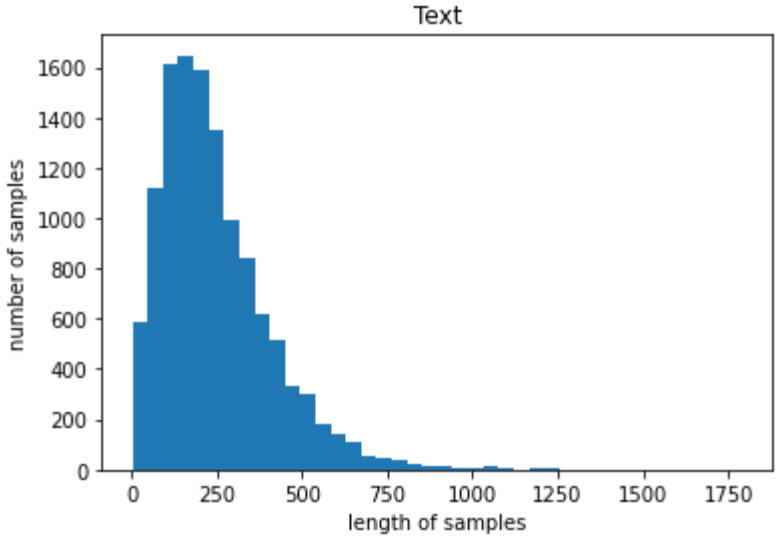
```
def preprocess_sentence(sentence, remove_stopwords = True):
    sentence = BeautifulSoup(sentence, "lxml").text # <br />, <a href = ...> 등의 html 태그 제거
    sentence = re.sub(r'₩([^)]*₩)', '', sentence) # 괄호로 닫힌 문자열  제거 Ex) my husband (and myself) for => my husband for
    sentence = re.sub('"','', sentence) # 쌍따옴표 " 제거
    sentence = re.sub(r"'s₩b","",sentence) # 소유격 제거. Ex) roland's -> roland

    # 불용어 제거 (Text)
    if remove_stopwords:
        tokens = ' '.join(word for word in sentence.split() if not word in stopwords if len(word) > 1)
    # 불용어 미제거 (Summary)
    else:
        tokens = ' '.join(word for word in sentence.split() if len(word) > 1)
    return tokens
```

데이터 분포


Summary
Text
120
100
80
60
40
20
0
1750
1500
1250
1000
750
500
250
0
1
1


Summary
number of samples
length of samples
0
200
400
600
800
1000
1200
1400
1600
0
20
40
60
80
100
120


Text
number of samples
length of samples
0
200
400
600
800
1000
1200
1400
1600
0
250
500
750
1000
1250
1500
1750

데이터 분포

```
[ ]  below_threshold_len(text_max_len, data['document'])

     전체 샘플 중 길이가 350 이하인 샘플의 비율: 0.7871169172623449

[ ]  below_threshold_len(summary_max_len, data['summary'])

     전체 샘플 중 길이가 50 이하인 샘플의 비율: 0.978637745460521

[ ]  data = data[data['document'].apply(lambda x: len(x.split()) <= text_max_len)]
     data = data[data['summary'].apply(lambda x: len(x.split()) <= summary_max_len)]
     print('전체 샘플수 :',(len(data)))

     전체 샘플수 : 9429
```

희귀 단어

```
threshold = 7
total_cnt = len(src_tokenizer.word_index) # 단어의 수
rare_cnt = 0 # 등장 빈도수가 threshold보다 작은 단어의 개수를 카운트
total_freq = 0 # 훈련 데이터의 전체 단어 빈도수 총 합
rare_freq = 0 # 등장 빈도수가 threshold보다 작은 단어의 등장 빈도수의 총 합

# 단어와 빈도수의 쌍(pair)을 key와 value로 받는다.
for key, value in src_tokenizer.word_counts.items():
    total_freq = total_freq + value

    # 단어의 등장 빈도수가 threshold보다 작으면
    if(value < threshold):
        rare_cnt = rare_cnt + 1
        rare_freq = rare_freq + value

print('단어 집합(vocabulary)의 크기 :',total_cnt)
print('등장 빈도가 %s번 이하인 희귀 단어의 수: %s'%(threshold - 1, rare_cnt))
print('단어 집합에서 희귀 단어를 제외시킬 경우의 단어 집합의 크기 %s'%(total_cnt - rare_cnt))
print("단어 집합에서 희귀 단어의 비율:", (rare_cnt / total_cnt)*100)
print("전체 등장 빈도에서 희귀 단어 등장 빈도 비율:", (rare_freq / total_freq)*100)
```

```
단어 집합(vocabulary)의 크기 : 162473
등장 빈도가 6번 이하인 희귀 단어의 수: 137548
단어 집합에서 희귀 단어를 제외시킬 경우의 단어 집합의 크기 24925
단어 집합에서 희귀 단어의 비율: 84.65898949363894
전체 등장 빈도에서 희귀 단어 등장 빈도 비율: 17.498095977292845
```

```
[ ] threshold = 6
    total_cnt = len(tar_tokenizer.word_index) # 단어의 수
    rare_cnt = 0 # 등장 빈도수가 threshold보다 작은 단어의 개수를 카운트
    total_freq = 0 # 훈련 데이터의 전체 단어 빈도수 총 합
    rare_freq = 0 # 등장 빈도수가 threshold보다 작은 단어의 등장 빈도수의 총 합

    # 단어와 빈도수의 쌍(pair)을 key와 value로 받는다.
    for key, value in tar_tokenizer.word_counts.items():
        total_freq = total_freq + value

        # 단어의 등장 빈도수가 threshold보다 작으면
        if(value < threshold):
            rare_cnt = rare_cnt + 1
            rare_freq = rare_freq + value

    print('단어 집합(vocabulary)의 크기 :',total_cnt)
    print('등장 빈도가 %s번 이하인 희귀 단어의 수: %s'%(threshold - 1, rare_cnt))
    print('단어 집합에서 희귀 단어를 제외시킬 경우의 단어 집합의 크기 %s'%(total_cnt - rare_cnt))
    print("단어 집합에서 희귀 단어의 비율:", (rare_cnt / total_cnt)*100)
    print("전체 등장 빈도에서 희귀 단어 등장 빈도 비율:", (rare_freq / total_freq)*100)
```

```
단어 집합(vocabulary)의 크기 : 33343
등장 빈도가 5번 이하인 희귀 단어의 수: 28922
단어 집합에서 희귀 단어를 제외시킬 경우의 단어 집합의 크기 4421
단어 집합에서 희귀 단어의 비율: 86.74084515490507
전체 등장 빈도에서 희귀 단어 등장 빈도 비율: 29.063062950750187
```

모델

```
Model: "model_1"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 input_1 (InputLayer)           [(None, 350)]        0           []

 embedding (Embedding)          (None, 350, 128)     3200000     ['input_1[0][0]']

 lstm (LSTM)                    [(None, 350, 256),   394240      ['embedding[0][0]']
                                 (None, 256),
                                 (None, 256)]

 input_2 (InputLayer)           [(None, None)]       0           []

 lstm_1 (LSTM)                  [(None, 350, 256),   525312      ['lstm[0][0]']
                                 (None, 256),
                                 (None, 256)]

 embedding_1 (Embedding)        (None, None, 128)    640000      ['input_2[0][0]']

 lstm_2 (LSTM)                  [(None, 350, 256),   525312      ['lstm_1[0][0]']
                                 (None, 256),
                                 (None, 256)]

 lstm_3 (LSTM)                  [(None, None, 256),  394240      ['embedding_1[0][0]',
                                 (None, 256),                     'lstm_2[0][1]',
                                 (None, 256)]                     'lstm_2[0][2]']

 attention_layer (AttentionLaye  ((None, None, 256),  131328     ['lstm_2[0][0]',
 r)                              (None, None, 350))               'lstm_3[0][0]']

 concat_layer (Concatenate)     (None, None, 512)    0           ['lstm_3[0][0]',
                                                                  'attention_layer[0][0]']

 dense_1 (Dense)                (None, None, 5000)   2565000     ['concat_layer[0][0]']

==================================================================================================
Total params: 8,375,432
Trainable params: 8,375,432
Non-trainable params: 0
__________________________________________________________________________________________________
```

결과

https://github.com/keonju2/nlp_study/blob/426ba1547bb71bcdf302f6c70e5baa6d594b8c63/attention.ipynb - 실습 코드

```
원문 :  이것은 화면의 녹색 배경에 있는 흰색의 전화 모양의 아이콘이다 탭한다 이는 화면의 오른쪽 아래에서 찾을 있다 음성 메시지를 탭하여 전화를 걸면 나오는 음성 안내를 본인의 음성 메시지를 설정을 있다 버튼을 누른다 이는 해당 페이지 중간에 나타날 것이다 옵션을
실제 요약문 : 아이폰의 전화 앱을 탭한다 음성 메시지 지금 설정 비밀번호를 입력한다 완료 암호를 다시 입력하십시오 완료 설정을 누른다 녹음 음성 메시지를 저장
예측 요약문 :  페이스북 엽니다 엽니다 클릭하십시오 클릭하십시오 클릭하십시오 클릭하십시오 클릭하십시오 클릭하십시오 클릭하십시오 클릭하십시오


원문 :  치료제를 바르기 30분 전에 일반 진통제를 한두 또는 이부프로펜은 치료의 통증을 좋은 사마귀를 것이 견딜 없을 정도로 않지만 진통제를 먹으면 최대한 치료를 진행할 있다 자가 냉동 요법 세트는 대부분의 약국에서 구입할 있으며 가격은 보통 정도이다 집에서 치료
실제 요약문 : 통증이 의사 처방전 없이 있는 진통제를 복용한다 사용 사마귀를 바른다 부분에 반창고를 살짝 붙인다 않는 경우 2 후에 다시
예측 요약문 :  증상 알기 알기 증상 확인하기 증상 알아보기


원문 :  먼저 버터를 녹여 동일한 온도가 되게 한다 버터를 녹이는 방법은 다양하다 가지 소개하자면 다음과 같다 가스레인지로 버터를 녹인다 프라이팬에 버터를 담아 약불로 가열한다 버터는 온도가 녹는다 온도는 날이 더울 실내 온도와 비슷하다 버터를 계속 3 불에서 내
실제 요약문 : 버터 녹이기 크림 활용하기
예측 요약문 :  자기 알기


원문 :  절대 심각한 화상을 집에서 하지 않도록 한다 전문가의 치료를 받아야 한다 부르거나 응급실을 찾는다 절대 심각한 화상을 스스로 집에서 하지 않는다 다음은 의학적 119가 전까지 취할 있는 사전 가능하다면 부상이나 피부 손상을 막을 있는 모든 조치를 취한다 뜨거
실제 요약문 : 즉시 119에 연락한다 안전하게 환자를 뜨거운 화상 부위를 덮는다 화학적 있는 모든 씻어낸다 입은 부위를 심장 보다 높게 둔다 즉시 도움을 청한다
예측 요약문 :  증상 알아보기 증상 알아보기 증상 알아보기


원문 :  브라우저의 주소 바에 www com 입력하고 키보드에서 enter를 누른다 메신저에 로그인이 되지 않았다면 이메일 비밀번호를 입력해서 로그인한다 브라우저 창의 왼쪽 면에 모든 그룹 개인 대화 목록이 나타난다 여기에서 삭제하고 싶은 그룹을 찾은 클릭해보자 그룹 이
실제 요약문 : 인터넷 브라우저에서 메신저 열기 좌측 그룹을 클릭하기 정보 아이콘 클릭하기 그룹 구성원 옆에 있는 3개의 아이콘을 누르기 드롭다운 메뉴에서 제거하기 누르기 제거 다른 그룹 모두 제거하기 기어 아이콘 클릭하기 드롭다운 메뉴에서 삭제 클릭하기 삭제
예측 요약문 :  페이스북 열기 클릭하기 클릭하기 클릭하기 클릭하기 클릭하기 클릭하기 클릭하기 클릭하기 클릭하기 클릭하기 클릭하기


원문 :  색소 넣는 것은 슬라임을 더욱 재밌게 만들 있다 혼합물이 알맞은 농도를 있게 한다 옅에 밀폐 용기에 넣어 어둡고 서늘한 장소에 보관하도록 하자
실제 요약문 : 원하는 색의 식품 넣기 1시간 동안 기다리기 밀폐 용기에 보관하기
예측 요약문 :  오븐을 예열하기 끓는 물을 넣는다


원문 :  구내염이 생기면 통증과 인해 쉽지 않다 입을 때 이를 닦을 때마다 구내염 증상이 있는 부위를 물리적으로 밖에 없기 때문이다 이로 인한 고통을 지속적으로 느끼는 상황이라면 자극하지 않는 방법을 것이 좋을 것이다 먼저 음식을 먹을 때는 구내염이 없는 하자 혀도
실제 요약문 : 않기 주의하기 성분이 들어간 구강 관련 제품 피하기 강한 들어간 음식 피하기 음식 피하기 담배 피하기
예측 요약문 :  매일 운동하기 전에 운동하기 운동하기


원문 :  흰색으로 적혀있는 질은 푸른색 응용 프로그램을 연다 페이지 좌측상단에 있다 맥은 과정을 생략한다 달력을 만들고자 하는 입력하고 누른다 이렇게 하면 달력 위에 달이 상단에 있는 푸른색 있다 삽입 도구 모음이 아래에 열린다 도구 모음의 섹션에 있다 마우스를 오
실제 요약문 : 워드를 연다 문서를 클릭한다 입력한다 클릭한다 클릭한다 만든다 입력한다 크게 만든다 숫자를 입력한다 정보를 입력한다 다른 만든다 저장한다
예측 요약문 :  파일 열기 파일 열기 파일 선택하기 파일 메뉴 클릭하기 클릭하기 클릭하기 파일 선택하기


원문 :  잎사귀 끝이 갈색으로 변하는 것은 경우가 많지만 물이 적은 것 많은 문제를 일으킬 있다 싱크대에 식물을 잡은 다음 줄기를 잡고 흔들어서 식물과 뿌리를 화분에서 빼낸다 물이 적은 많은 있다 흙이 붙어 있지 않고 물을 너무 적게 주는 것이다 물이 떨어지거나 뿌리
실제 요약문 : 식물을 빼서 뿌리 검사하기 물을 과하게 식물을 다시 넣고 주기 계획을 짜기 물이 적은 물을 주기 특히 경우 주변의 습도 높이기
예측 요약문 :  말하기 대해 대해 물어보기 대해 물어보기 물어보기


원문 :  개를 점점 종류의 개가 가장 좋아하는지 알아보세요 개들은 배를 문질러 주는 것을 좋아하고 개들은 다리 마사지를 해주는 것을 좋아합니다 개들은 다리 가까이 가면 수도 있습니다 개의 부위를 가장 좋아하는지 알아보세요 꼬리를 흔들거나 근육이 당신이 쓰다음는 몸
실제 요약문 : 개가 가장 좋아하는 부위를 알아보세요 개의 배를 것은 항상 개를 어떻게 하는지 개에게 마사지를 해주세요 새끼 발을 새끼 주변을 마사지 해주세요
```