Sentiment Analysis

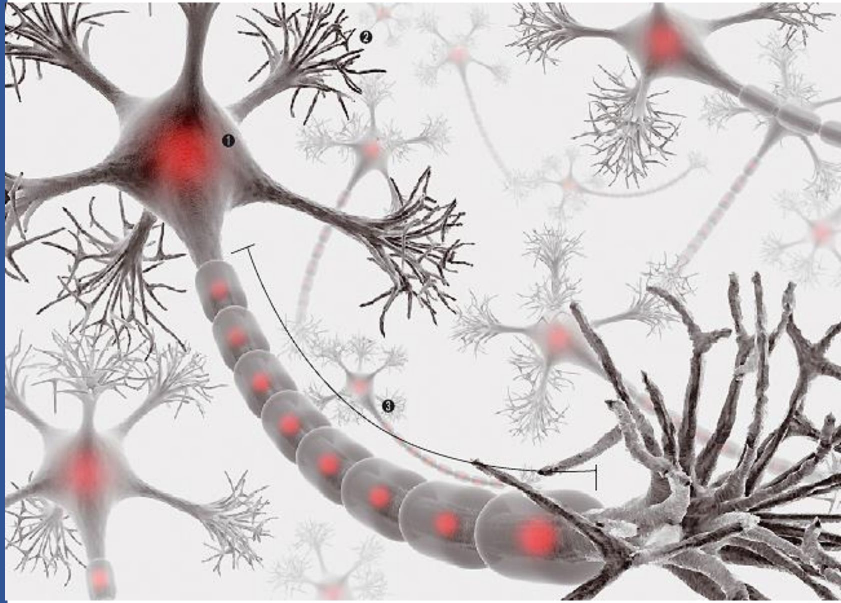
학습 목표

- 감정 분석을 하는 RNN 신경망 모델을 만들어 본다.

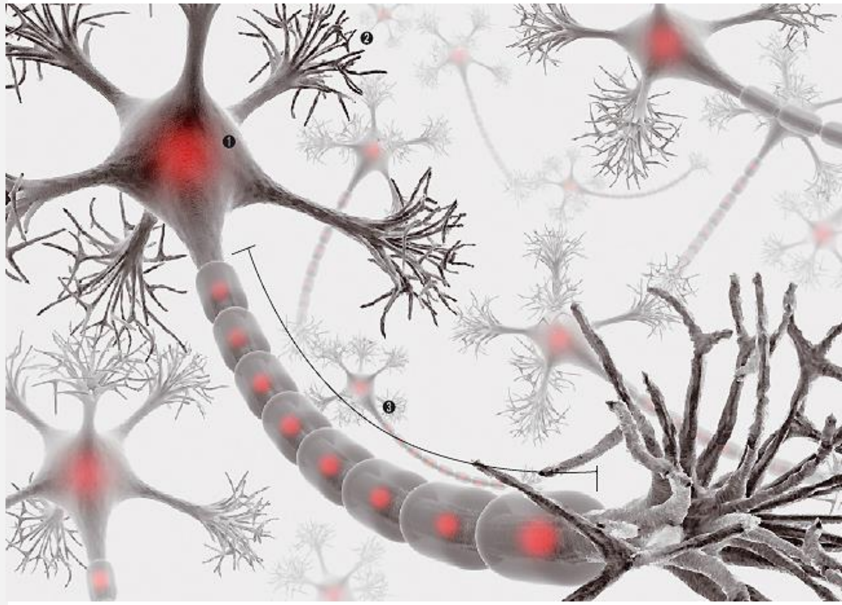
주요 내용

- 1. 문제 정의
- 2. 데이터 준비
- 3. 모델 정의 및 훈련, 검증

https://wikidocs.net/24586



1 문제 정의



문제

영화에 대한 리뷰를 이용해서 Sentiment Analysis를 해보자!

<u>Sentiment Analysis</u>

영화 리뷰가 긍정인지 부정인지 예측

Positive(1) or Negative(0)

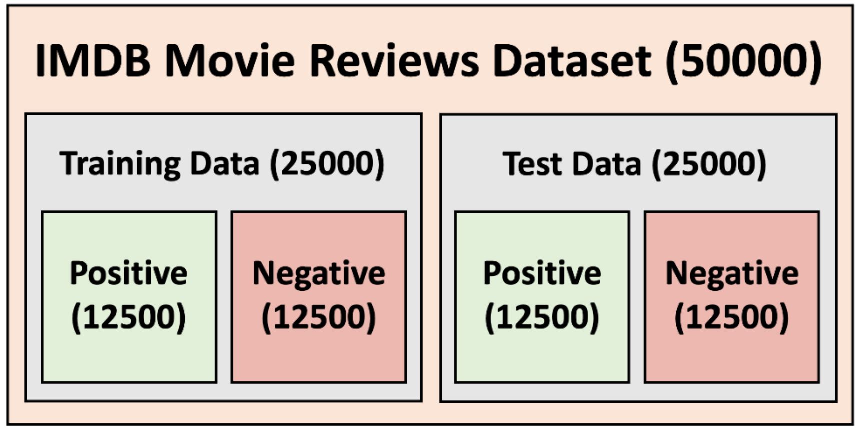
RNN

Word Embedding + LSTM

“i watched the movie in a preview and i really loved it the cast is excellent and the plot is sometimes absolutely hilarious another highlight of the movie is definitely the music which hopefully will be released soon i recommend it to everyone who likes the british humour and especially to all musicians go and see it's great.. ”

영화 리뷰

Large Movie Review Dataset

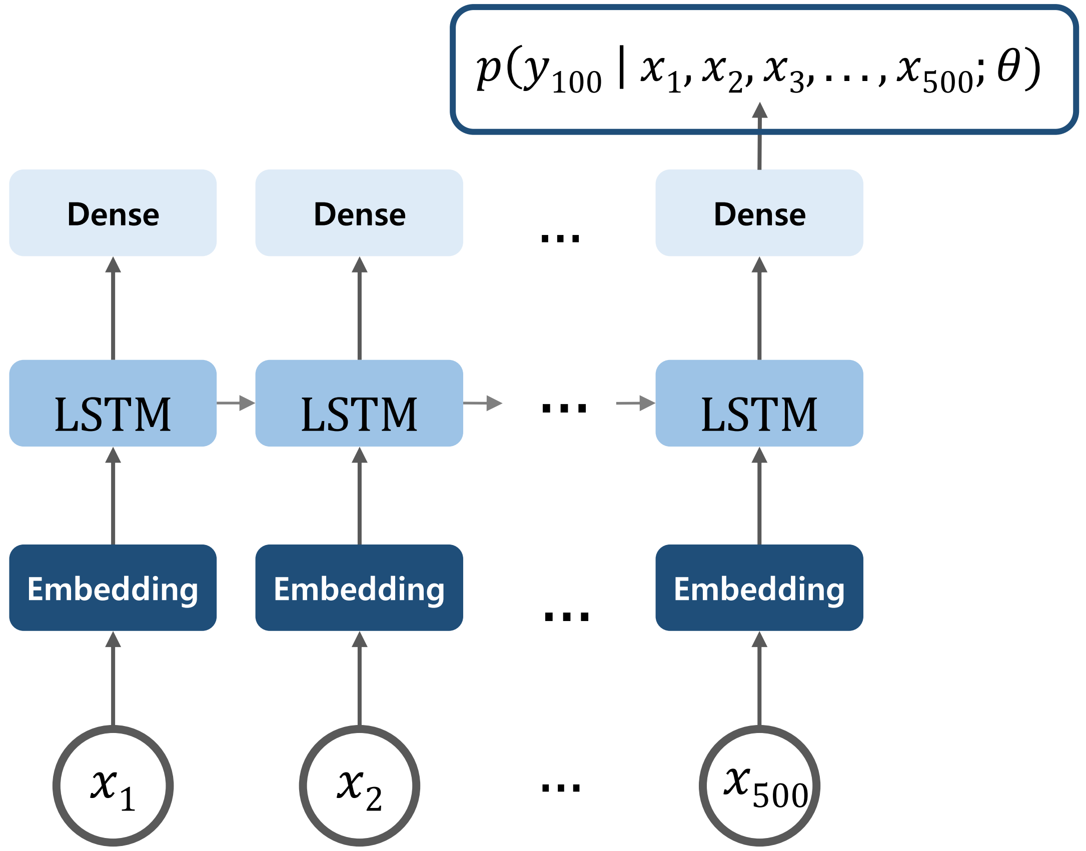


- 영화 사이트 IMDB의 리뷰 데이터
- 리뷰에 대한 텍스트와 해당 리뷰가 긍정인 경우 1을 부정인 경우 0으로 표시한 레이블로 구성
- 총 50000 샘플
- Training Set 25,000, Test Set 25,000

https://ai.stanford.edu/~amaas/data/sentiment/

Hint : Network 구성

마지막 time step만 출력 (*return_sequences=False*)



(batch size, time steps)
(64, 500)

입력 길이를 500으로 맞춰서 입력 (Padding 사용)

Binary Classifier

# of clsss = 1
(batch size, clsss)
(None, 1)

rnn units = 120
(batch size, rnn units)
(None, 120)

embedding dimension = 120
(batch size, time steps, embedding dimension)
(None, None, 120)

참고 Representing text as numbers

One-hot encodings

차원 : 전체 Vocabulary 개수

cat	1	0	0	0	0
mat	0	1	0	0	0
on	0	0	1	0	0
set	0	0	0	1	0
the	0	0	0	0	1

- Sparse Integer Vector
- 단어 개수가 많아질수록 차원이 높아짐

Unique number encodings

임의로 배정

cat	0
mat	1
on	2
set	3
the	4

- Dense Integer Scalar
- 단어 간 연관성이나 유사성을 표현하기 어려움
- 신경망 가중치 계산에 문제 발생

Word Embeddings

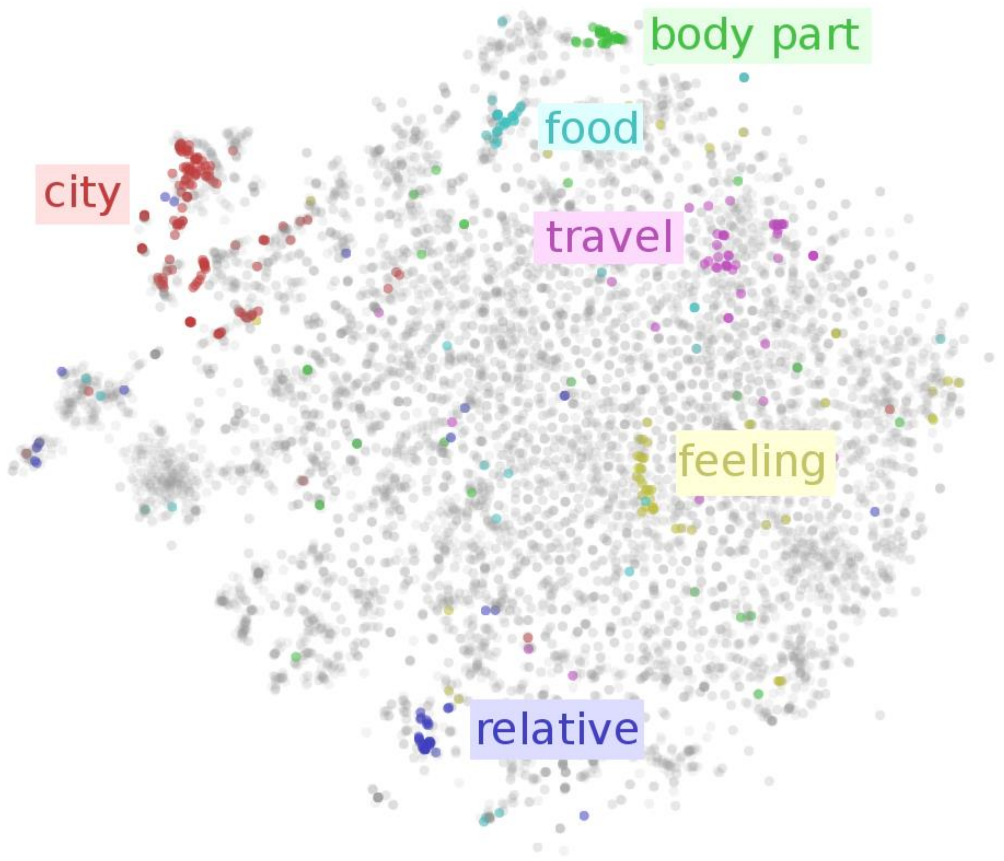
학습을 통해 정해짐

	embedding
cat	(0.3, 6.12, 2.22, 1.2)
mat	(1.9, 0.73, 0.8, 2.11)
on	(8.1, 2.37, 4.9, 2.7)
set	(0.1, 8.33, 4.01, 9.39)
the	(6.26, 3.92, 8.28, 3.27)

- Dense Float Vector
- Embedding은 학습을 통해 정해짐
- Discrete Space를 Continuous Space로 변경
- 단어 간의 연관성이 존재

참고 Word Embedding

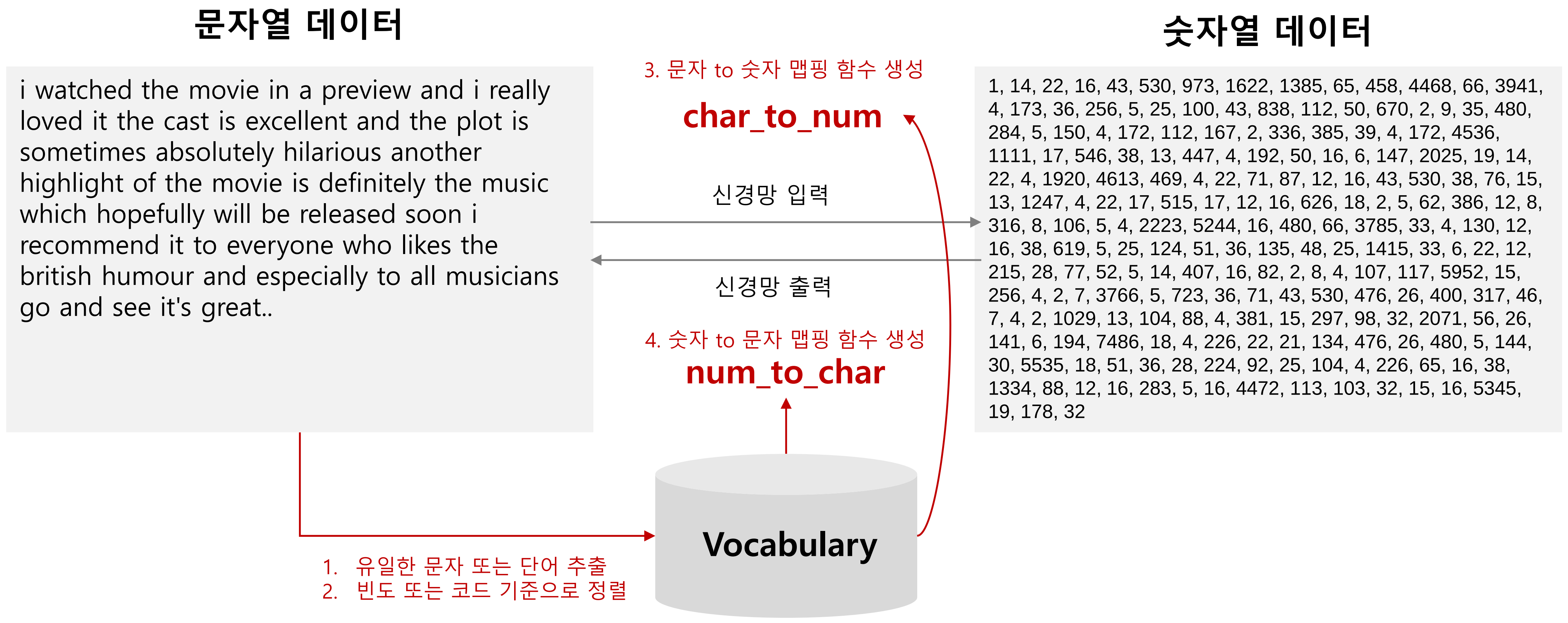
Word Embedding Visualization



- Latent Space 상에서의 각 단어의 위치를 학습을 통해 구함
- 비슷한 단어는 가까이 위치
- 최근 Language Model에서는 Context에 따라 같은 단어라도 다른 위치에 존재하도록 embedding함

https://ruder.io/word-embeddings-1/

참고 문자열과 숫자열 변환



문자열 데이터

i watched the movie in a preview and i really loved it the cast is excellent and the plot is sometimes absolutely hilarious another highlight of the movie is definitely the music which hopefully will be released soon i recommend it to everyone who likes the british humour and especially to all musicians go and see it's great..

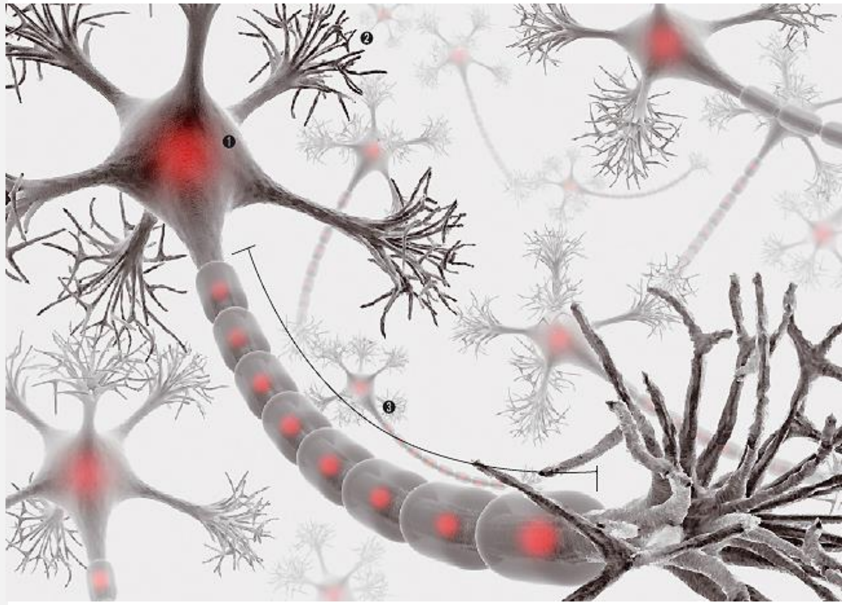
숫자열 데이터

1, 14, 22, 16, 43, 530, 973, 1622, 1385, 65, 458, 4468, 66, 3941, 4, 173, 36, 256, 5, 25, 100, 43, 838, 112, 50, 670, 2, 9, 35, 480, 284, 5, 150, 4, 172, 112, 167, 2, 336, 385, 39, 4, 172, 4536, 1111, 17, 546, 38, 13, 447, 4, 192, 50, 16, 6, 147, 2025, 19, 14, 22, 4, 1920, 4613, 469, 4, 22, 71, 87, 12, 16, 43, 530, 38, 76, 15, 13, 1247, 4, 22, 17, 515, 17, 12, 16, 626, 18, 2, 5, 62, 386, 12, 8, 316, 8, 106, 5, 4, 2223, 5244, 16, 480, 66, 3785, 33, 4, 130, 12, 16, 38, 619, 5, 25, 124, 51, 36, 135, 48, 25, 1415, 33, 6, 22, 12, 215, 28, 77, 52, 5, 14, 407, 16, 82, 2, 8, 4, 107, 117, 5952, 15, 256, 4, 2, 7, 3766, 5, 723, 36, 71, 43, 530, 476, 26, 400, 317, 46, 7, 4, 2, 1029, 13, 104, 88, 4, 381, 15, 297, 98, 32, 2071, 56, 26, 141, 6, 194, 7486, 18, 4, 226, 22, 21, 134, 476, 26, 480, 5, 144, 30, 5535, 18, 51, 36, 28, 224, 92, 25, 104, 4, 226, 65, 16, 38, 1334, 88, 12, 16, 283, 5, 16, 4472, 113, 103, 32, 15, 16, 5345, 19, 178, 32

3. 문자 to 숫자 맵핑 함수 생성
char_to_num
신경망 입력
신경망 출력
4. 숫자 to 문자 맵핑 함수 생성
num_to_char
Vocabulary
1. 유일한 문자 또는 단어 추출
2. 빈도 또는 코드 기준으로 정렬

- 문자열을 신경망의 입력으로 사용하려면 숫자열로 변환해야 함
- 반대로 신경망에서 예측된 숫자열은 다시 문자열로 변환해야 함

2 데이터 준비



텐서플로와 다른 라이브러리 임포트

```
from __future__ import absolute_import, division, print_function, unicode_literals

from tensorflow.keras.datasets import imdb

import tensorflow as tf
import matplotlib.pyplot as plt
import numpy as np
```

데이터셋 다운로드

```
(x_train, y_train), (x_test, y_test) = imdb.load_data(num_words=10000)
```

num_words: max number of words to include. Words are ranked by how often they occur (in the training set) and only the most frequent words are kept

```
print('훈련용 리뷰 개수 : {}'.format(len(x_train)))
print('테스트용 리뷰 개수 : {}'.format(len(x_test)))
num_classes = max(y_train) + 1
print('카테고리 : {}'.format(num_classes))
```

```
훈련용 리뷰 개수 : 25000
테스트용 리뷰 개수 : 25000
카테고리 : 2
```

참고 tf.keras.datasets.imdb.load_data

```
tf.keras.datasets.imdb.load_data(
    path='imdb.npz',
    num_words=None,
    skip_top=0,
    maxlen=None,
    seed=113,
    start_char=1,
    oov_char=2,
    index_from=3,
    **kwargs
)
```

- **path**: where to cache the data (relative to ~/.keras/dataset).
- **num_words**: max number of words to include. Words are ranked by how often they occur (in the training set) and only the most frequent words are kept
- **skip_top**: skip the top N most frequently occurring words (which may not be informative).
- **maxlen**: sequences longer than this will be filtered out.
- **seed**: random seed for sample shuffling.
- **start_char**: The start of a sequence will be marked with this character. Set to 1 because 0 is usually the padding character.
- **oov_char**: words that were cut out because of the num_words or skip_top limit will be replaced with this character.
- **index_from**: index actual words with this index and higher.

데이터셋 확인

숫자로 인코딩 상태 확인

```
print(x_train[0])
print(y_train[0])
```

```
[1, 14, 22, 16, 43, 530, 973, 1622, 1385, 65, 458, 4468, 66, 3941, 4, 173, 36, 256, 5, 25, 100, 43, 838, 112, 50,
670, 2, 9, 35, 480, 284, 5, 150, 4, 172, 112, 167, 2, 336, 385, 39, 4, 172, 4536, 1111, 17, 546, 38, 13, 447, 4,
192, 50, 16, 6, 147, 2025, 19, 14, 22, 4, 1920, 4613, 469, 4, 22, 71, 87, 12, 16, 43, 530, 38, 76, 15, 13, 1247, 4,
22, 17, 515, 17, 12, 16, 626, 18, 2, 5, 62, 386, 12, 8, 316, 8, 106, 5, 4, 2223, 5244, 16, 480, 66, 3785, 33, 4, 130,
12, 16, 38, 619, 5, 25, 124, 51, 36, 135, 48, 25, 1415, 33, 6, 22, 12, 215, 28, 77, 52, 5, 14, 407, 16, 82, 2, 8, 4,
107, 117, 5952, 15, 256, 4, 2, 7, 3766, 5, 723, 36, 71, 43, 530, 476, 26, 400, 317, 46, 7, 4, 2, 1029, 13, 104, 88,
4, 381, 15, 297, 98, 32, 2071, 56, 26, 141, 6, 194, 7486, 18, 4, 226, 22, 21, 134, 476, 26, 480, 5, 144, 30, 5535,
18, 51, 36, 28, 224, 92, 25, 104, 4, 226, 65, 16, 38, 1334, 88, 12, 16, 283, 5, 16, 4472, 113, 103, 32, 15, 16,
5345, 19, 178, 32]
1
```

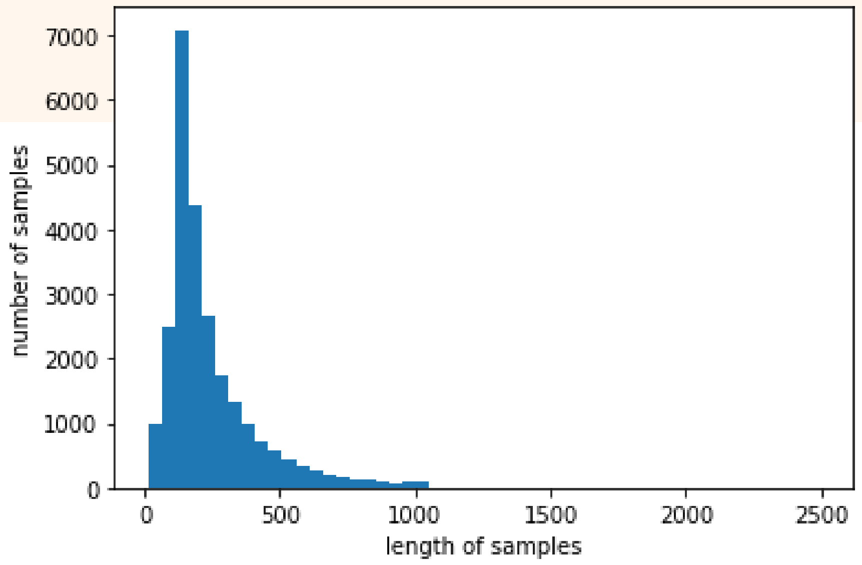
데이터셋 통계정보

리뷰 길이 통계 정보 확인

```
print('리뷰의 최대 길이 : {}'.format(max(len(l) for l in x_train)))
print('리뷰의 평균 길이 : {}'.format(sum(map(len, x_train))/len(x_train)))

plt.hist([len(s) for s in x_train], bins=50)
plt.xlabel('length of samples')
plt.ylabel('number of samples')
plt.show()
```

```
리뷰의 최대 길이 : 2494
리뷰의 평균 길이 : 238.71364
```



데이터셋 통계정보

레이블 별 개수

```
unique_elements, counts_elements = np.unique(y_train, return_counts=True)
print("각 레이블에 대한 빈도수:")
print(np.asarray((unique_elements, counts_elements)))
```

```
각 레이블에 대한 빈도수:
[[ 0     1     ]
[12500 12500]]
```

데이터셋 통계정보

단어에서 인덱스 맵핑, 인덱스에서 단어 맵핑 함수 정의

```
word_to_index = imdb.get_word_index()
index_to_word={}
for key, value in word_to_index.items():
    index_to_word[value] = key
```

단어 빈도 순위 확인

```
print('빈도수 상위 1번 단어 : {}'.format(index_to_word[1]))
print('빈도수 상위 3941번 단어 : {}'.format(index_to_word[3941]))
```

```
빈도수 상위 1번 단어 : the
빈도수 상위 3941번 단어 : journalist
```

데이터셋 통계정보

숫자열 데이터를 문자열로 디코딩

```
print(' '.join([index_to_word[X] for X in x_train[0]]))
```

```
the as you with out themselves powerful lets loves their becomes reaching had journalist of lot from anyone to
have after out atmosphere never more room and it so heart shows to years of every never going and help
moments or of every chest visual movie except her was several of enough more with is now current film as you
of mine potentially unfortunately of you than him that with out themselves her get for was camp of you movie
sometimes movie that with scary but and to story wonderful that in seeing in character to of 70s musicians with
heart had shadows they of here that with her serious to have does when from why what have critics they is you
that isn't one will very to as itself with other and in of seen over landed for anyone of and br show's to whether
from than out themselves history he name half some br of and odd was two most of mean for 1 any an boat she
he should is thought frog but of script you not while history he heart to real at barrel but when from one bit then
have two of script their with her nobody most that with wasn't to with armed acting watch an for with heartfelt film
want an
```

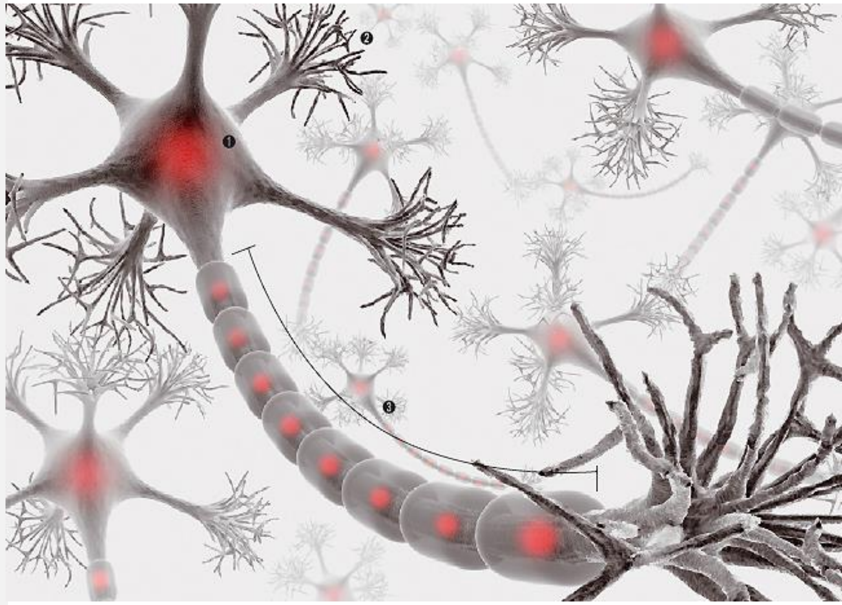
훈련을 위한 데이터 선택

```
TRAIN_VOCABULARY=5000
(x_train, y_train), (x_test, y_test) = imdb.load_data(num_words=TRAIN_VOCABULARY)
```

```
max_time_steps=500
x_train = tf.keras.preprocessing.sequence.pad_sequences(x_train, maxlen=max_time_steps)
x_test = tf.keras.preprocessing.sequence.pad_sequences(x_test, maxlen=max_time_steps)
```

Sequence가 maxlen 보다 크면 truncate

3 모델 정의 및 훈련, 검증



모델 정의 (문제)

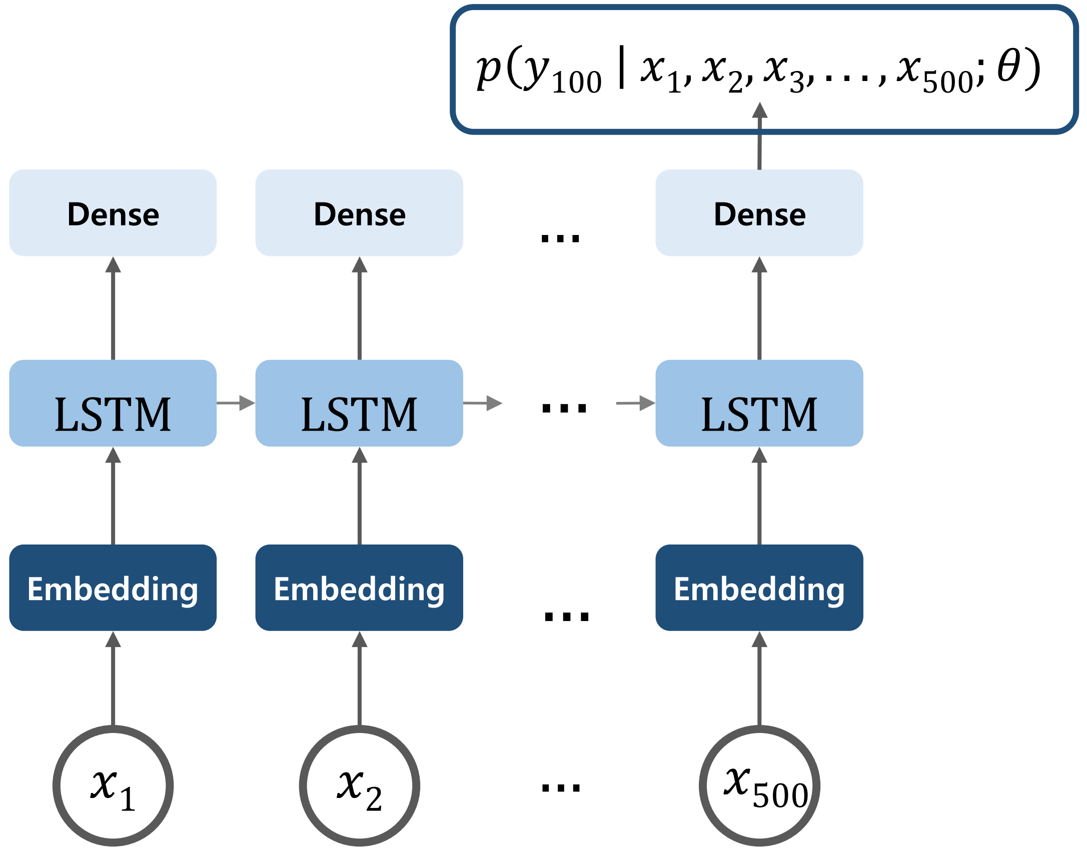
모델 정의 (Hint : tf.keras.layers.Embedding, tf.keras.layers.LSTM, tf.keras.layers.Dense 사용)

```
def build_model(vocab_size, embedding_dim, rnn_units, batch_size):
  # your code
  return model
```

주의 : input_shape이나 input_length는 명시하지 않아도 됨
Dense Output Layer에서 sigmoid activation을 사용하시오.

Hint : Network 구성

마지막 time step만 출력 (*return_sequences=False*)



Binary Classifier

# of clsss = 1
(batch size, clsss)
(None, 1)

rnn units = 120
(batch size, rnn units)
(None, 120)

embedding dimension = 120
(batch size, time steps, embedding dimension)
(None, None, 120)

(batch size, time steps)
(64, 500)

입력 길이를 500으로 맞춰서 입력 (Padding 사용)

참고 tf.keras.layers.Embedding

```
tf.keras.layers.Embedding(
    input_dim, output_dim, embeddings_initializer='uniform',
    embeddings_regularizer=None, activity_regularizer=None,
    embeddings_constraint=None, mask_zero=False, input_length=None, **kwargs
)
```

nput_dim	int > 0. Size of the vocabulary, i.e. maximum integer index + 1. 입력 숫자가 이 값 이상이 될 수 없음
output_dim	int >= 0. Dimension of the dense embedding.

https://www.tensorflow.org/api_docs/python/tf/keras/layers/Embedding

참고 tf.keras.layers.LSTM

```
tf.keras.layers.LSTM(
    units, activation='tanh', recurrent_activation='sigmoid', use_bias=True,
    kernel_initializer='glorot_uniform', recurrent_initializer='orthogonal',
    bias_initializer='zeros', unit_forget_bias=True, kernel_regularizer=None,
    recurrent_regularizer=None, bias_regularizer=None, activity_regularizer=None,
    kernel_constraint=None, recurrent_constraint=None, bias_constraint=None,
    dropout=0.0, recurrent_dropout=0.0, implementation=2, return_sequences=False,
    return_state=False, go_backwards=False, stateful=False, time_major=False,
    unroll=False, **kwargs
)
```

- **units**: Positive integer, dimensionality of the output space.
- **return_sequences**: Boolean. Whether to return the last output. in the output sequence, or the full sequence. Default: False.
- **stateful**: Boolean (default False). If True, the last state for each sample at index i in a batch will be used as initial state for the sample of index i in the following batch.

https://www.tensorflow.org/api_docs/python/tf/keras/layers/LSTM

모델 생성

모델 생성

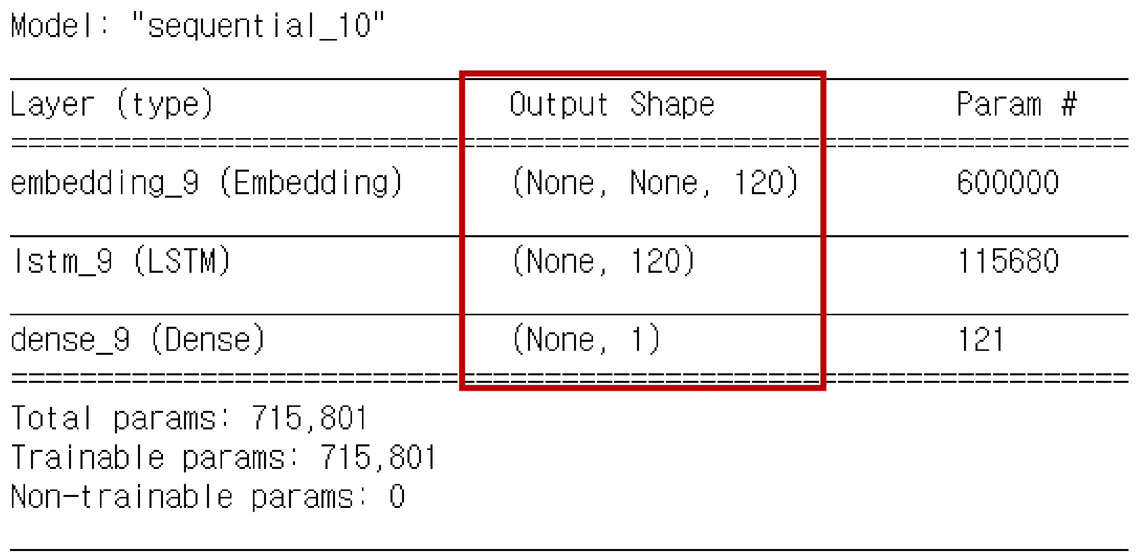
```
embedding_dim = 120
rnn_units = 120
BATCH_SIZE=64

model = build_model(
  vocab_size = TRAIN_VOCABULARY,
  embedding_dim=embedding_dim,
  rnn_units=rnn_units,
  batch_size=BATCH_SIZE)
```

모델 구조 확인

모델 구조 확인

```
model.summary()
```



```
Model: "sequential_10"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_9 (Embedding)      (None, None, 120)         600000
_________________________________________________________________
lstm_9 (LSTM)                (None, 120)               115680
_________________________________________________________________
dense_9 (Dense)              (None, 1)                 121
=================================================================
Total params: 715,801
Trainable params: 715,801
Non-trainable params: 0
_________________________________________________________________
```

학습 방식 설정

모델 컴파일

```
model.compile(loss=tf.keras.losses.BinaryCrossentropy(from_logits=False),
              optimizer=tf.keras.optimizers.Adam(1e-4),
              metrics=['accuracy'])
```

모델 훈련

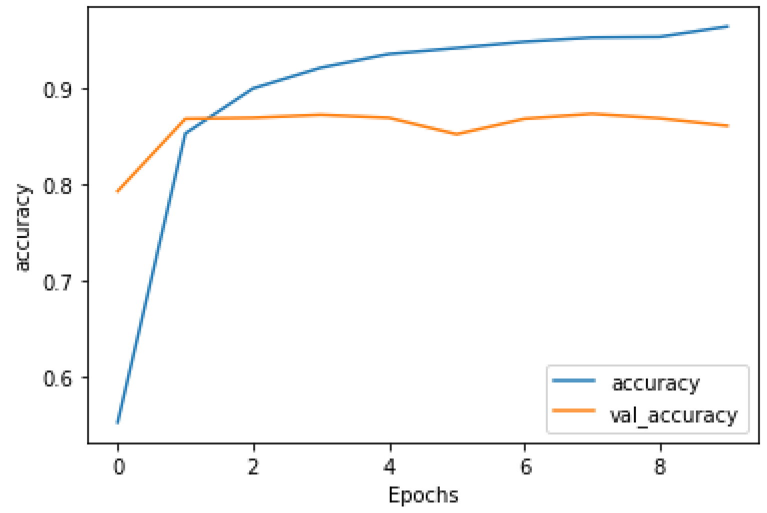
```
EPOCHS = 5
history = model.fit(x_train, y_traint, epochs= EPOCHS, validation_split=0.1)
```

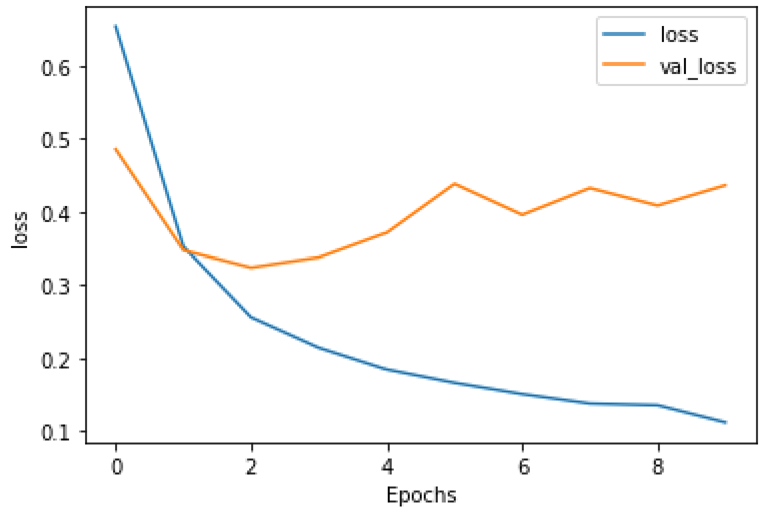
```
Train on 22500 samples, validate on 2500 samples
Epoch 1/5
22500/22500 [==============================] - 24s 1ms/sample - loss: 0.1825 - accuracy: 0.9340 - val_loss: 0.1938 - val_accuracy: 0.9292
Epoch 2/5
22500/22500 [==============================] - 25s 1ms/sample - loss: 0.1707 - accuracy: 0.9394 - val_loss: 0.2139 - val_accuracy: 0.9240
Epoch 3/5
22500/22500 [==============================] - 24s 1ms/sample - loss: 0.1616 - accuracy: 0.9441 - val_loss: 0.2441 - val_accuracy: 0.9096
Epoch 4/5
22500/22500 [==============================] - 24s 1ms/sample - loss: 0.1523 - accuracy: 0.9438 - val_loss: 0.2183 - val_accuracy: 0.9180
Epoch 5/5
22500/22500 [==============================] - 24s 1ms/sample - loss: 0.1417 - accuracy: 0.9506 - val_loss: 0.2411 - val_accuracy: 0.9092
```

모델 훈련

```
def plot_graphs(history, metric):
  plt.plot(history.history[metric])
  plt.plot(history.history['val_'+metric], '')
  plt.xlabel("Epochs")
  plt.ylabel(metric)

plot_graphs(history, ‘loss')
plot_graphs(history, 'accuracy')
```





모델 평가

```
test_loss, test_acc = model.evaluate(x_test, y_test)

print('Test Loss: {}'.format(test_loss))
print('Test Accuracy: {}'.format(test_acc))
```

```
391/Unknown - 18s 45ms/step - loss: 0.4529 - accuracy: 0.8517
Test Loss: 0.4529471364243866
Test Accuracy: 0.8516799807548523
```

모델 테스트 (문제)

샘플 리뷰에 대한 결과를 예측하시오. (Hint x_pred를 (1, None) 크기의 입력으로 만드시오.)

```
def sample_predict(pred_text):
 list_of_words = tf.keras.preprocessing.text.text_to_word_sequence(
                    pred_text,
                    filters='!"#$%&()*+,-./:;<=>?@[\\]^_`{|}~\t\n',
                    lower=True,
                    split=' '
                )
  # list_of_words를 모델의 입력인 x_pred으로 변환
  # your code
  predictions = model.predict(x_pred)
  return (predictions)
```

- word_to_index를 사용해서 문자열을 숫자열로 변환
- tf.expand_dims 사용해서 1차원에서 2차원 Tensor로 변환 (1, None)

```
# predict on a sample
sample_pred_text = ('The movie was cool. The animation and the graphics '
                    'were out of this world. I would recommend this movie.')
predictions = sample_predict(sample_pred_text, pad=False)
print(predictions)
```

```
[[0.7321655]]
```

참고 tf.expand_dims

```
tf.expand_dims(
    input, axis, name=None
)
```

- **Input** : A Tensor.
- **Axis** : Integer specifying the dimension index at which to expand the shape of input. Given an input of D dimensions, axis must be in range [-(D+1), D] (inclusive).
- **Name** : Optional string. The name of the output Tensor.

https://www.tensorflow.org/api_docs/python/tf/expand_dims

Thank you!