Natural Language Processing

for

Review Classification Model Implementation

강상진 | 이준희 | 원영오

KAIST College of Business

2022-06-17

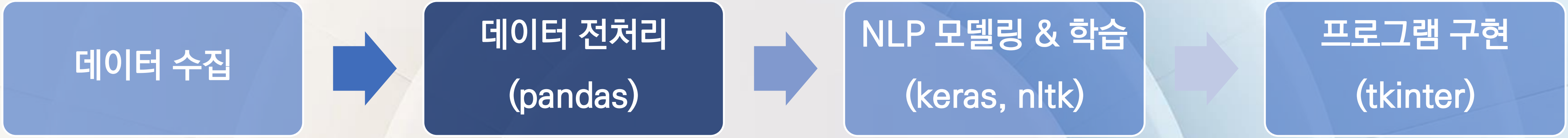
Introduction

- 소비자 온라인 리뷰는 소비자의 의사 결정 과정에 필수적인 부분이 되었습니다.
- 최근 연구에 따르면 온라인 리뷰는 93% 소비자들의 구매 결정에 영향을 미치며 (Kaimingk 2019),
- 91%의 소비자가 온라인 리뷰를 개인적인 추천만큼 신뢰하는 것으로 나타났습니다 (Igniyte 2019).
- 또한 온라인 리뷰는 비즈니스에 실체적인 경제적 영향을 미치는 것으로 나타났습니다 (Moe and Trusov 2011).

Positive?

Negative?

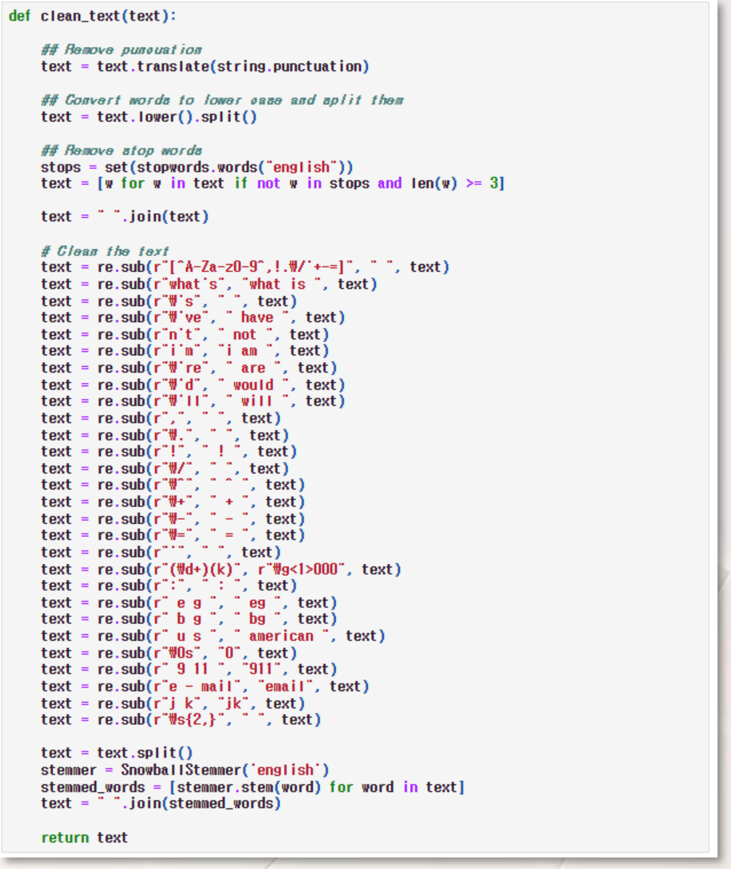
Procedure


데이터 수집

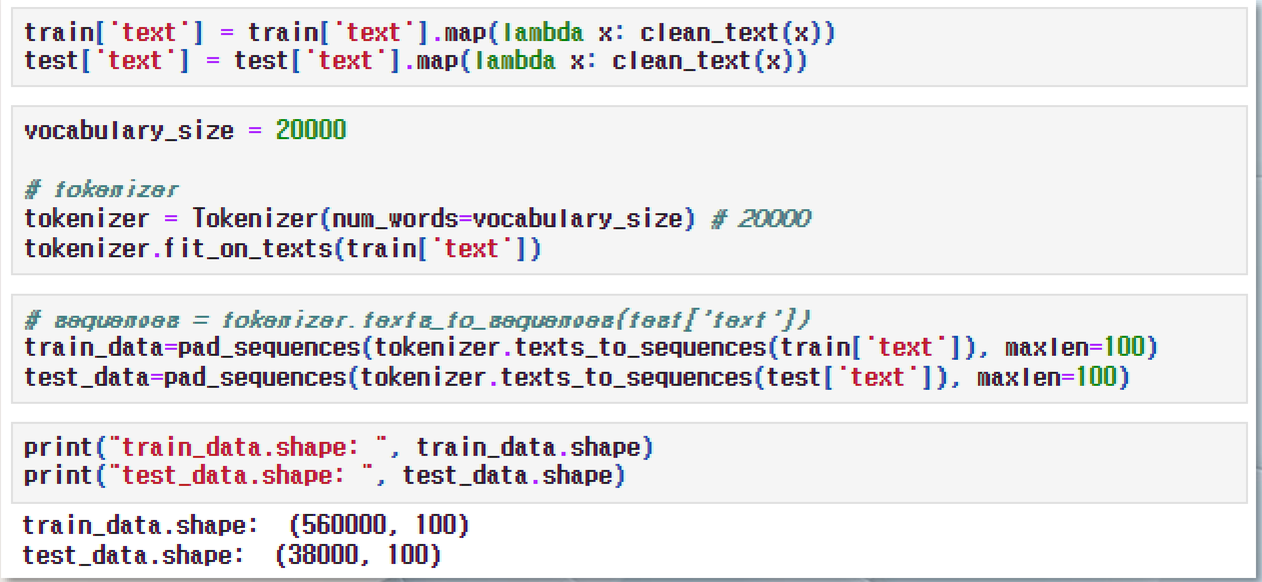
데이터 전처리
(pandas)

NLP 모델링 & 학습
(keras, nltk)

프로그램 구현
(tkinter)



```
def clean_text(text):

    ## Remove punctuation
    text = text.translate(string.punctuation)

    ## Convert words to lower case and split them
    text = text.lower().split()

    ## Remove stop words
    stops = set(stopwords.words("english"))
    text = [w for w in text if not w in stops and len(w) >= 3]

    text = " ".join(text)

    # Clean the text
    text = re.sub(r"[^A-Za-z0-9^,!.₩/'+-=]", " ", text)
    text = re.sub(r"what's", "what is ", text)
    text = re.sub(r"₩'s", " ", text)
    text = re.sub(r"₩'ve", " have ", text)
    text = re.sub(r"n't", " not ", text)
    text = re.sub(r"i'm", "i am ", text)
    text = re.sub(r"₩'re", " are ", text)
    text = re.sub(r"₩'d", " would ", text)
    text = re.sub(r"₩'ll", " will ", text)
    text = re.sub(r",", " ", text)
    text = re.sub(r"₩.", " ", text)
    text = re.sub(r"!", " ! ", text)
    text = re.sub(r"₩/", " ", text)
    text = re.sub(r"₩^", " ^ ", text)
    text = re.sub(r"₩+", " + ", text)
    text = re.sub(r"₩-", " - ", text)
    text = re.sub(r"₩=", " = ", text)
    text = re.sub(r"'", " ", text)
    text = re.sub(r"(₩d+)(k)", r"₩g<1>000", text)
    text = re.sub(r":", " : ", text)
    text = re.sub(r" e g ", " eg ", text)
    text = re.sub(r" b g ", " bg ", text)
    text = re.sub(r" u s ", " american ", text)
    text = re.sub(r"₩0s", "0", text)
    text = re.sub(r" 9 11 ", "911", text)
    text = re.sub(r"e - mail", "email", text)
    text = re.sub(r"j k", "jk", text)
    text = re.sub(r"₩s{2,}", " ", text)

    text = text.split()
    stemmer = SnowballStemmer('english')
    stemmed_words = [stemmer.stem(word) for word in text]
    text = " ".join(stemmed_words)

    return text
```



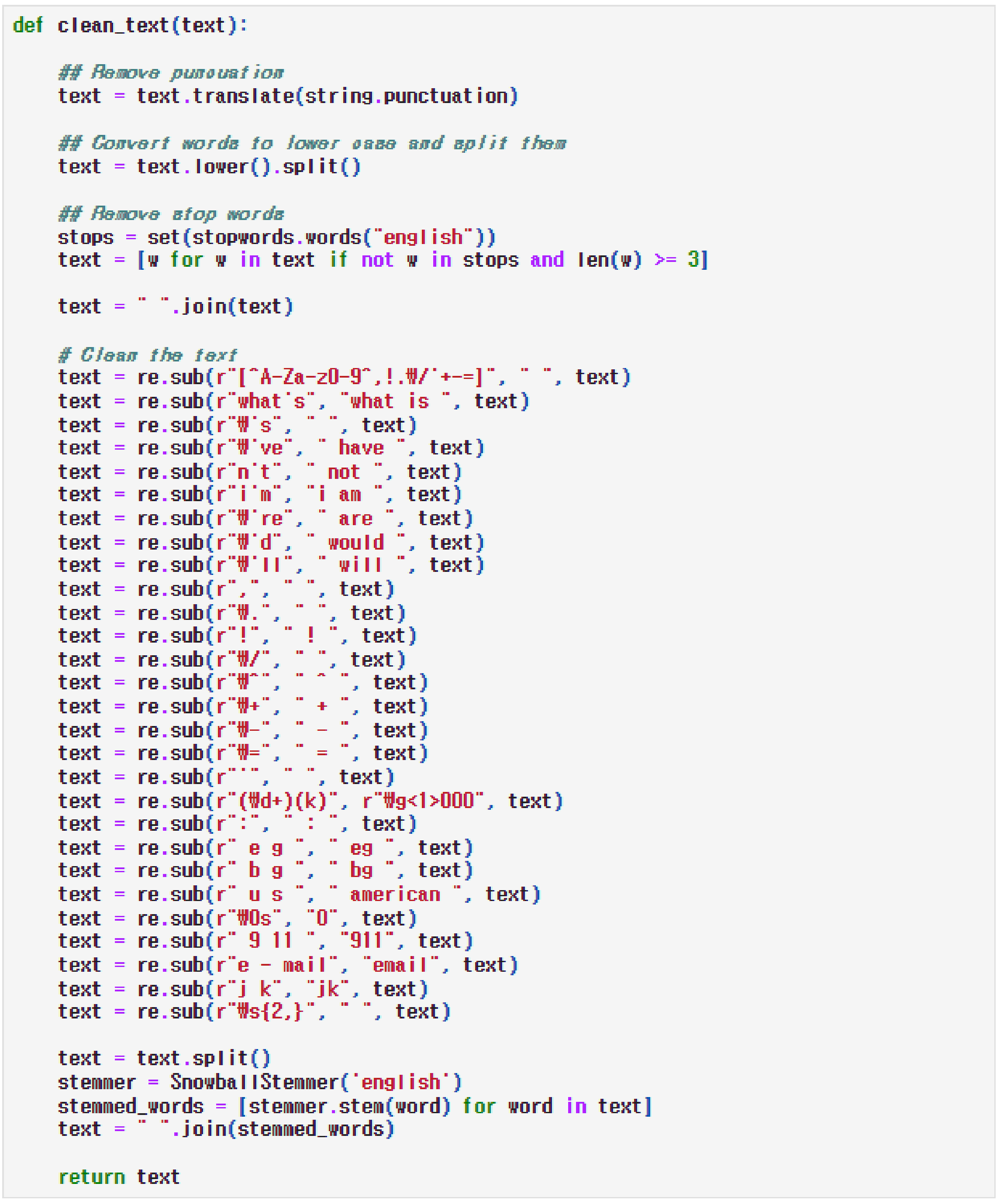
```
train['text'] = train['text'].map(lambda x: clean_text(x))
test['text'] = test['text'].map(lambda x: clean_text(x))
```

```
vocabulary_size = 20000

# tokenizer
tokenizer = Tokenizer(num_words=vocabulary_size) # 20000
tokenizer.fit_on_texts(train['text'])
```

```
# sequences = tokenizer.texts_to_sequences(test['text'])
train_data=pad_sequences(tokenizer.texts_to_sequences(train['text']), maxlen=100)
test_data=pad_sequences(tokenizer.texts_to_sequences(test['text']), maxlen=100)
```

```
print("train_data.shape: ", train_data.shape)
print("test_data.shape: ", test_data.shape)
```

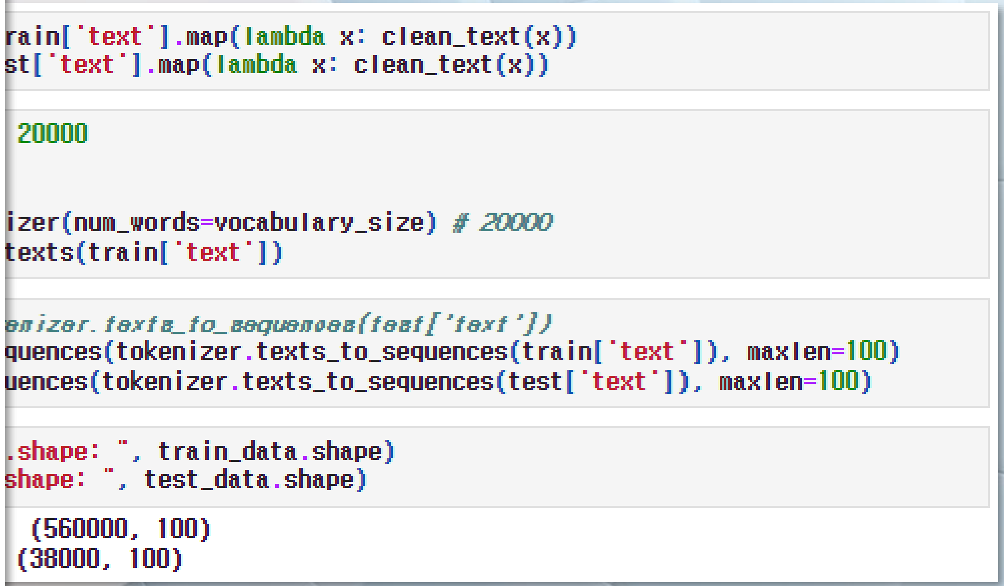
```
train_data.shape:  (560000, 100)
test_data.shape:  (38000, 100)
```



```
def clean_text(text):

    ## Remove punctuation
    text = text.translate(string.punctuation)

    ## Convert words to lower case and split them
    text = text.lower().split()

    ## Remove stop words
    stops = set(stopwords.words("english"))
    text = [w for w in text if not w in stops and len(w) >= 3]

    text = " ".join(text)

    # Clean the text
    text = re.sub(r"[^A-Za-z0-9^,!.₩/'+-=]", " ", text)
    text = re.sub(r"what's", "what is ", text)
    text = re.sub(r"₩'s", " ", text)
    text = re.sub(r"₩'ve", " have ", text)
    text = re.sub(r"n't", " not ", text)
    text = re.sub(r"i'm", "i am ", text)
    text = re.sub(r"₩'re", " are ", text)
    text = re.sub(r"₩'d", " would ", text)
    text = re.sub(r"₩'ll", " will ", text)
    text = re.sub(r",", " ", text)
    text = re.sub(r"₩.", " ", text)
    text = re.sub(r"!", " ! ", text)
    text = re.sub(r"₩/", " ", text)
    text = re.sub(r"₩^", " ^ ", text)
    text = re.sub(r"₩+", " + ", text)
    text = re.sub(r"₩-", " - ", text)
    text = re.sub(r"₩=", " = ", text)
    text = re.sub(r"'", " ", text)
    text = re.sub(r"(₩d+)(k)", r"₩g<1>000", text)
    text = re.sub(r":", " : ", text)
    text = re.sub(r" e g ", " eg ", text)
    text = re.sub(r" b g ", " bg ", text)
    text = re.sub(r" u s ", " american ", text)
    text = re.sub(r"₩0s", "0", text)
    text = re.sub(r" 9 11 ", "911", text)
    text = re.sub(r"e - mail", "email", text)
    text = re.sub(r"j k", "jk", text)
    text = re.sub(r"₩s{2,}", " ", text)

    text = text.split()
    stemmer = SnowballStemmer('english')
    stemmed_words = [stemmer.stem(word) for word in text]
    text = " ".join(stemmed_words)

    return text
```

NLP 모델링 & 학습
(keras, nltk)

프로그램 구현
(tkinter)



```
rain['text'].map(lambda x: clean_text(x))
st['text'].map(lambda x: clean_text(x))
```

```
 20000
```

```
izer(num_words=vocabulary_size) # 20000
texts(train['text'])
```

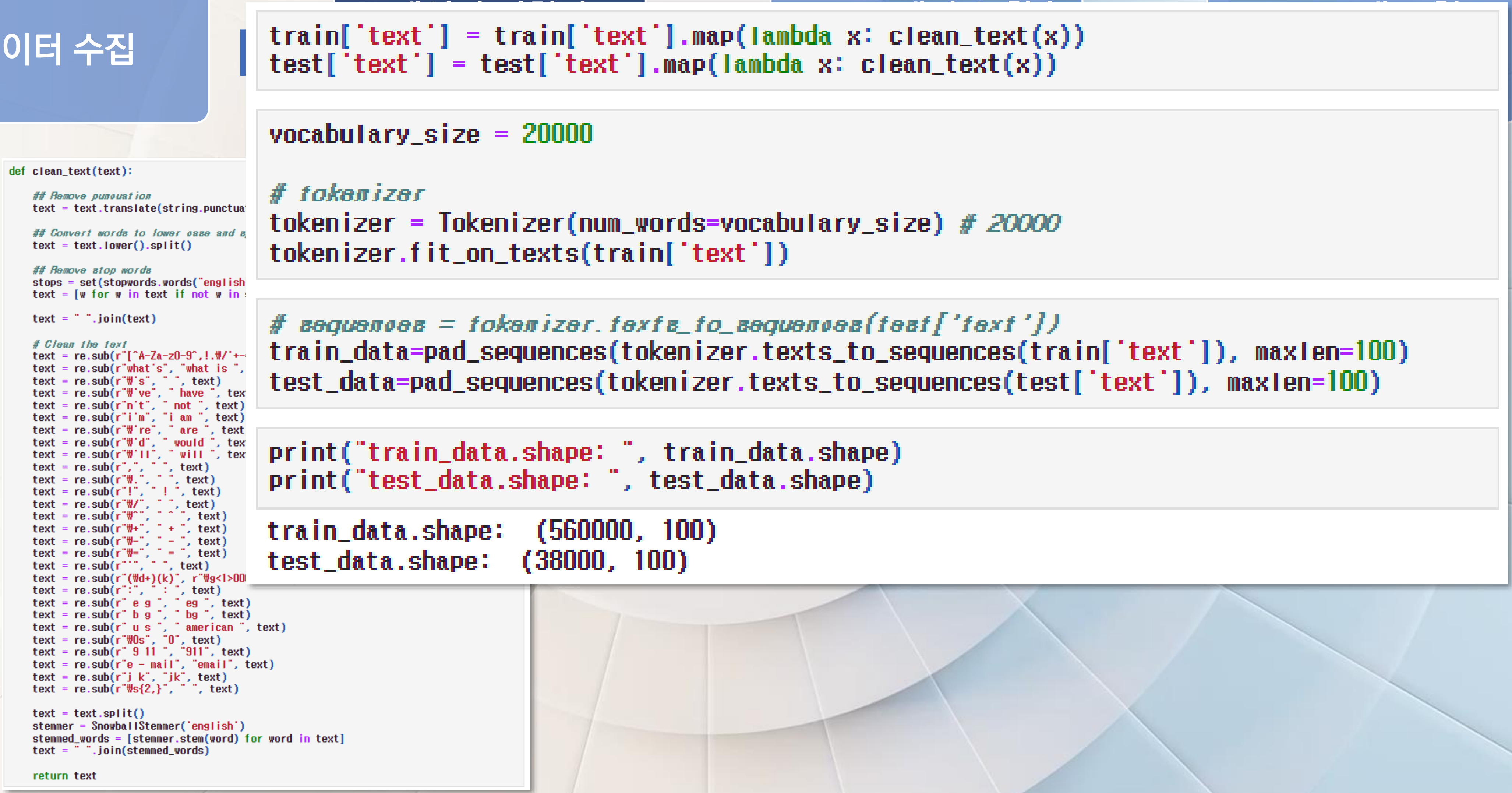
```
enizer.texts_to_sequences(test['text'])
quences(tokenizer.texts_to_sequences(train['text']), maxlen=100)
uences(tokenizer.texts_to_sequences(test['text']), maxlen=100)
```

```
.shape: ", train_data.shape)
shape: ", test_data.shape)
```

```
  (560000, 100)
 (38000, 100)
```

Procedure

데이터 수집



```
def clean_text(text):

    ## Remove puncuation
    text = text.translate(string.punctua

    ## Convert words to lower case and s
    text = text.lower().split()

    ## Remove stop words
    stops = set(stopwords.words("english
    text = [w for w in text if not w in :

    text = " ".join(text)

    # Clean the text
    text = re.sub(r"[^A-Za-z0-9^,!.₩/'+-
    text = re.sub(r"what's", "what is ",
    text = re.sub(r"₩'s", " ", text)
    text = re.sub(r"₩'ve", " have ", tex
    text = re.sub(r"n't", " not ", text)
    text = re.sub(r"i'm", "i am ", text)
    text = re.sub(r"₩'re", " are ", text
    text = re.sub(r"₩'d", " would ", tex
    text = re.sub(r"₩'ll", " will ", tex
    text = re.sub(r",", " ", text)
    text = re.sub(r"₩.", " ", text)
    text = re.sub(r"!", " ! ", text)
    text = re.sub(r"₩/", " ", text)
    text = re.sub(r"₩^", " ^ ", text)
    text = re.sub(r"₩+", " + ", text)
    text = re.sub(r"₩-", " - ", text)
    text = re.sub(r"₩=", " = ", text)
    text = re.sub(r"'", " ", text)
    text = re.sub(r"(₩d+)(k)", r"₩g<1>00
    text = re.sub(r":", " : ", text)
    text = re.sub(r" e g ", " eg ", text)
    text = re.sub(r" b g ", " bg ", text)
    text = re.sub(r" u s ", " american ", text)
    text = re.sub(r"₩0s", "0", text)
    text = re.sub(r" 9 11 ", "911", text)
    text = re.sub(r"e - mail", "email", text)
    text = re.sub(r"j k", "jk", text)
    text = re.sub(r"₩s{2,}", " ", text)

    text = text.split()
    stemmer = SnowballStemmer('english')
    stemmed_words = [stemmer.stem(word) for word in text]
    text = " ".join(stemmed_words)

    return text
```

```
train['text'] = train['text'].map(lambda x: clean_text(x))
test['text'] = test['text'].map(lambda x: clean_text(x))
```

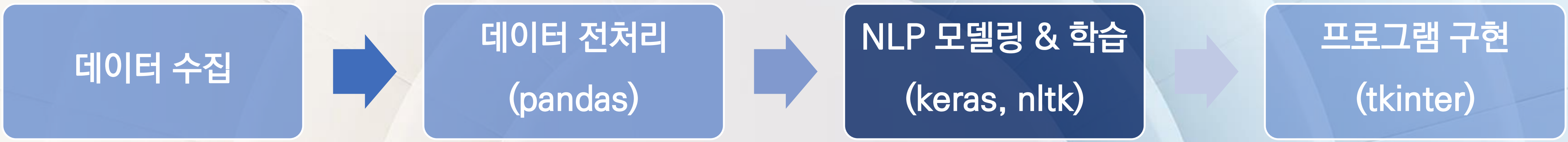
```
vocabulary_size = 20000

# tokenizer
tokenizer = Tokenizer(num_words=vocabulary_size) # 20000
tokenizer.fit_on_texts(train['text'])
```

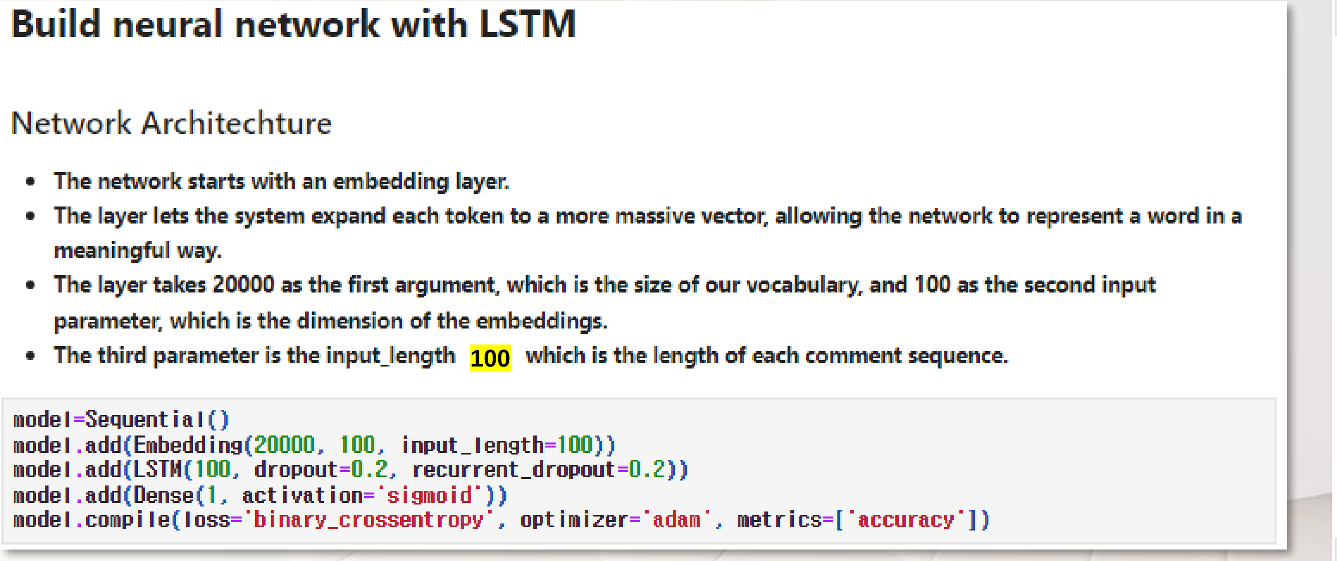
```
# sequences = tokenizer.texts_to_sequences(test['text'])
train_data=pad_sequences(tokenizer.texts_to_sequences(train['text']), maxlen=100)
test_data=pad_sequences(tokenizer.texts_to_sequences(test['text']), maxlen=100)
```

```
print("train_data.shape: ", train_data.shape)
print("test_data.shape: ", test_data.shape)
```

```
train_data.shape:  (560000, 100)
test_data.shape:  (38000, 100)
```

Procedure



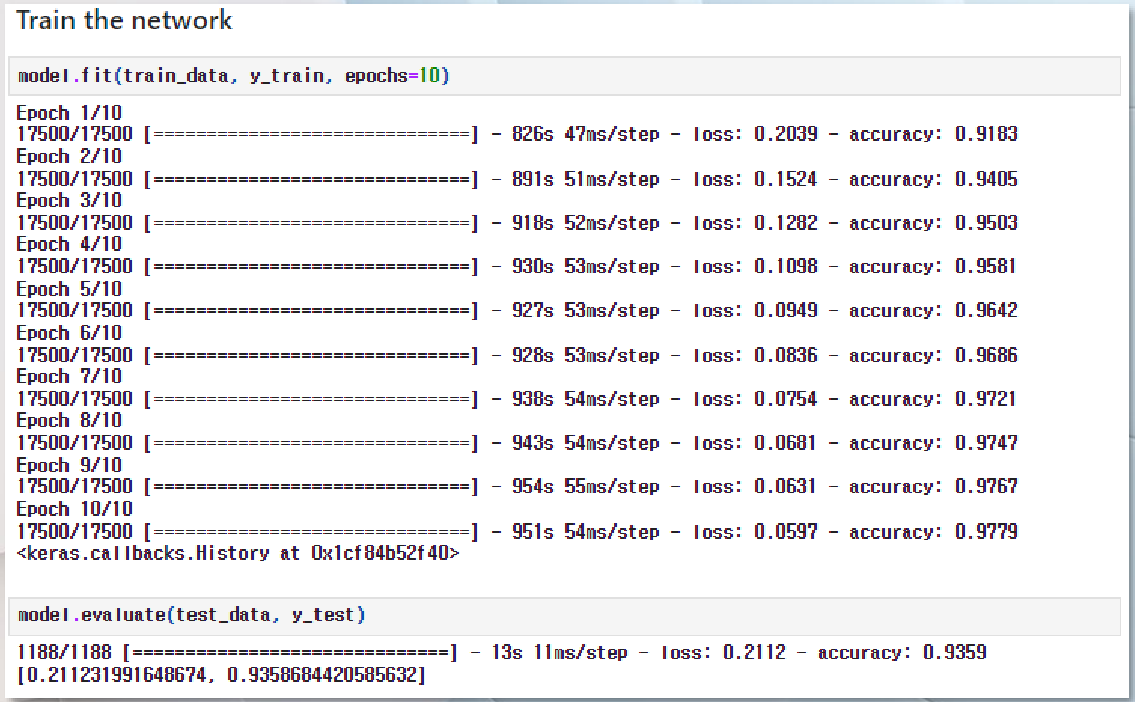


Build neural network with LSTM

Network Architechture

- The network starts with an embedding layer.
- The layer lets the system expand each token to a more massive vector, allowing the network to represent a word in a meaningful way.
- The layer takes 20000 as the first argument, which is the size of our vocabulary, and 100 as the second input parameter, which is the dimension of the embeddings.
- The third parameter is the input_length **100** which is the length of each comment sequence.

```
model=Sequential()
model.add(Embedding(20000, 100, input_length=100))
model.add(LSTM(100, dropout=0.2, recurrent_dropout=0.2))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
```



Train the network

```
model.fit(train_data, y_train, epochs=10)
```

```
Epoch 1/10
17500/17500 [==============================] - 826s 47ms/step - loss: 0.2039 - accuracy: 0.9183
Epoch 2/10
17500/17500 [==============================] - 891s 51ms/step - loss: 0.1524 - accuracy: 0.9405
Epoch 3/10
17500/17500 [==============================] - 918s 52ms/step - loss: 0.1282 - accuracy: 0.9503
Epoch 4/10
17500/17500 [==============================] - 930s 53ms/step - loss: 0.1098 - accuracy: 0.9581
Epoch 5/10
17500/17500 [==============================] - 927s 53ms/step - loss: 0.0949 - accuracy: 0.9642
Epoch 6/10
17500/17500 [==============================] - 928s 53ms/step - loss: 0.0836 - accuracy: 0.9686
Epoch 7/10
17500/17500 [==============================] - 938s 54ms/step - loss: 0.0754 - accuracy: 0.9721
Epoch 8/10
17500/17500 [==============================] - 943s 54ms/step - loss: 0.0681 - accuracy: 0.9747
Epoch 9/10
17500/17500 [==============================] - 954s 55ms/step - loss: 0.0631 - accuracy: 0.9767
Epoch 10/10
17500/17500 [==============================] - 951s 54ms/step - loss: 0.0597 - accuracy: 0.9779
<keras.callbacks.History at 0x1cf84b52f40>
```

```
model.evaluate(test_data, y_test)
```

```
1188/1188 [==============================] - 13s 11ms/step - loss: 0.2112 - accuracy: 0.9359
[0.211231991648674, 0.9358684420585632]
```

Procedure

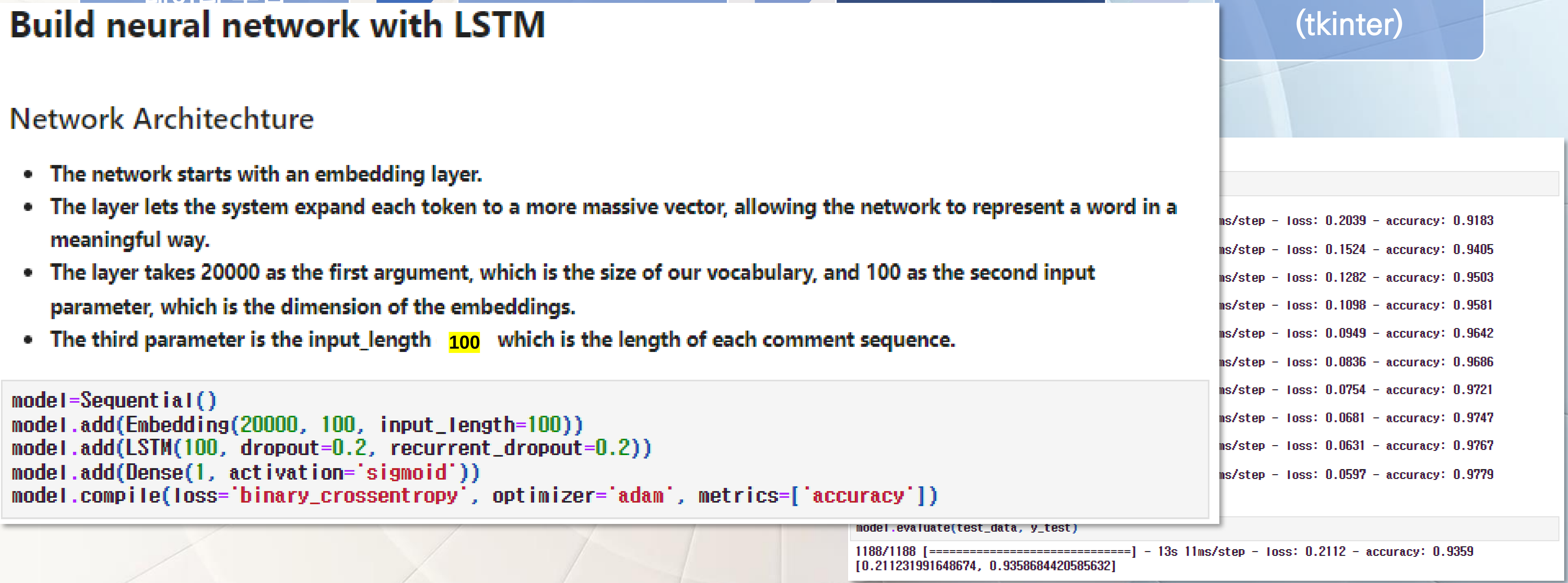
데이터 수집

데이터 전처리

NLP 모델링 & 학습

프로그램 구현

(tkinter)



Build neural network with LSTM

Network Architechture

- The network starts with an embedding layer.
- The layer lets the system expand each token to a more massive vector, allowing the network to represent a word in a meaningful way.
- The layer takes 20000 as the first argument, which is the size of our vocabulary, and 100 as the second input parameter, which is the dimension of the embeddings.
- The third parameter is the input_length **100** which is the length of each comment sequence.

```
model=Sequential()
model.add(Embedding(20000, 100, input_length=100))
model.add(LSTM(100, dropout=0.2, recurrent_dropout=0.2))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
```

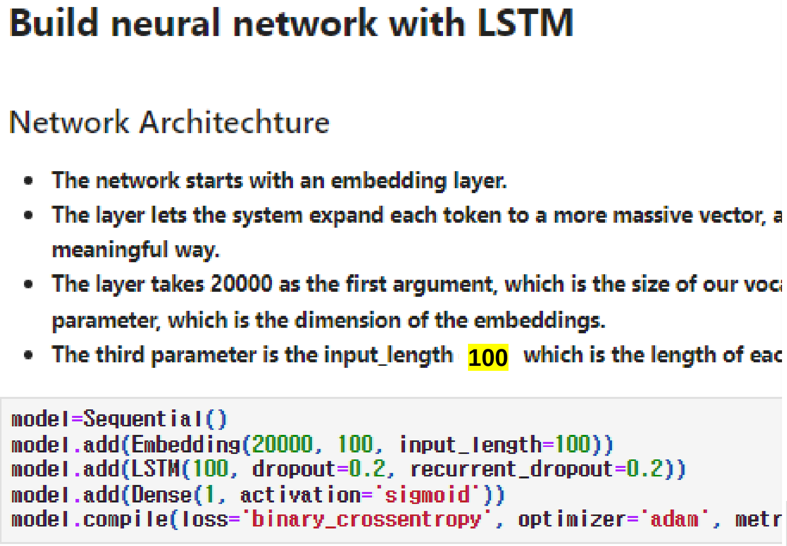
```
ms/step - loss: 0.2039 - accuracy: 0.9183
ms/step - loss: 0.1524 - accuracy: 0.9405
ms/step - loss: 0.1282 - accuracy: 0.9503
ms/step - loss: 0.1098 - accuracy: 0.9581
ms/step - loss: 0.0949 - accuracy: 0.9642
ms/step - loss: 0.0836 - accuracy: 0.9686
ms/step - loss: 0.0754 - accuracy: 0.9721
ms/step - loss: 0.0681 - accuracy: 0.9747
ms/step - loss: 0.0631 - accuracy: 0.9767
ms/step - loss: 0.0597 - accuracy: 0.9779
```

```
model.evaluate(test_data, y_test)
```

```
1188/1188 [==============================] - 13s 11ms/step - loss: 0.2112 - accuracy: 0.9359
[0.211231991648674, 0.9358684420585632]
```

Procedure

데이터 수집 → 데이터 전처리 → NLP 모델링 & 학습 → 프로그램 구현

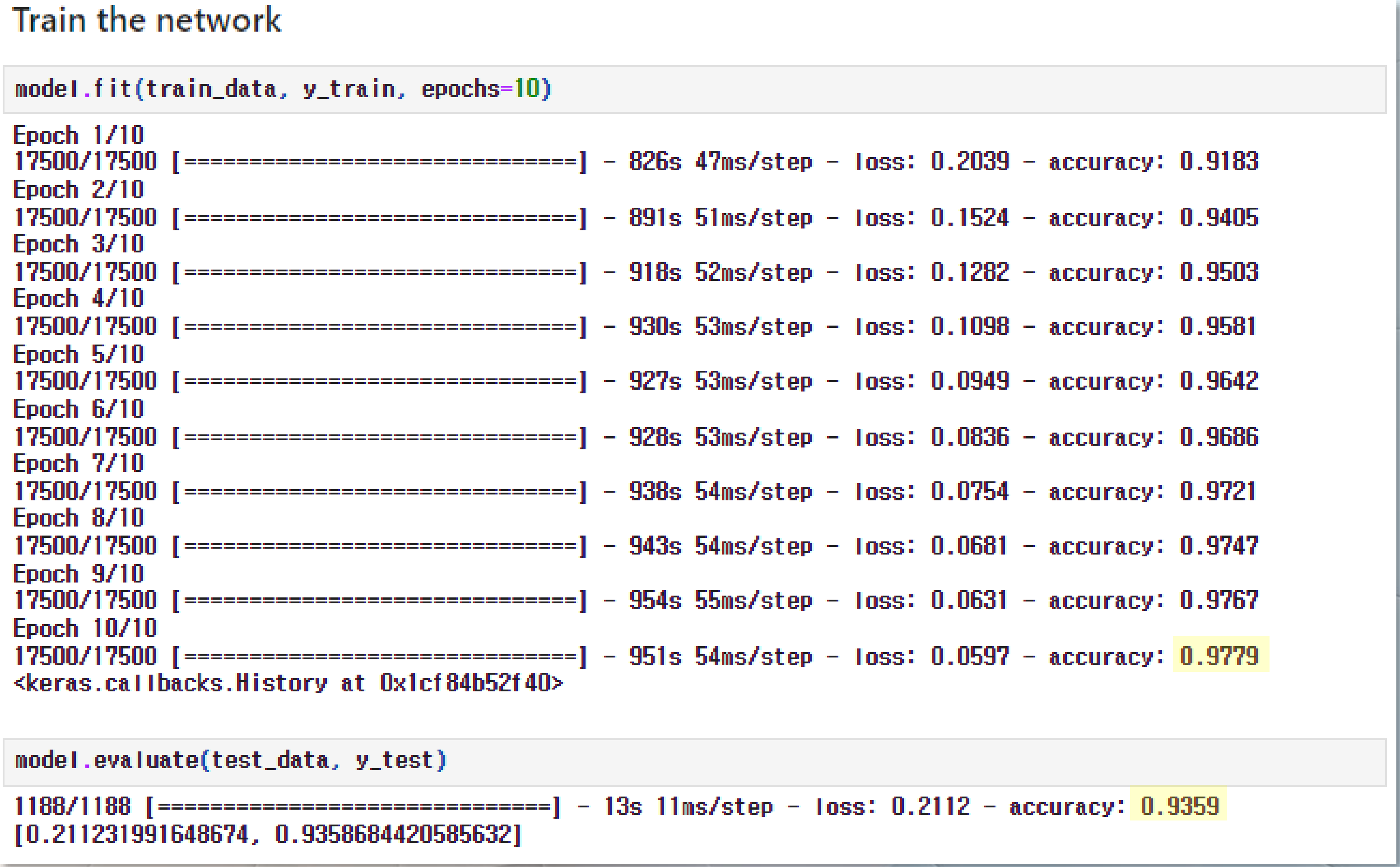


Build neural network with LSTM

Network Architechture

- The network starts with an embedding layer.
- The layer lets the system expand each token to a more massive vector, a meaningful way.
- The layer takes 20000 as the first argument, which is the size of our voca parameter, which is the dimension of the embeddings.
- The third parameter is the input_length **100** which is the length of eac

```
model=Sequential()
model.add(Embedding(20000, 100, input_length=100))
model.add(LSTM(100, dropout=0.2, recurrent_dropout=0.2))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='adam', metr
```



Train the network

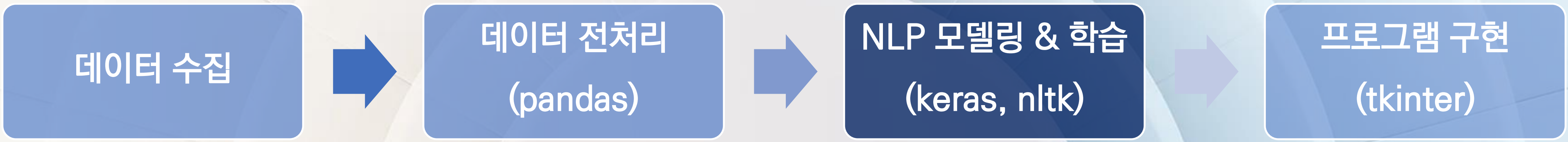
```
model.fit(train_data, y_train, epochs=10)
```

```
Epoch 1/10
17500/17500 [==============================] - 826s 47ms/step - loss: 0.2039 - accuracy: 0.9183
Epoch 2/10
17500/17500 [==============================] - 891s 51ms/step - loss: 0.1524 - accuracy: 0.9405
Epoch 3/10
17500/17500 [==============================] - 918s 52ms/step - loss: 0.1282 - accuracy: 0.9503
Epoch 4/10
17500/17500 [==============================] - 930s 53ms/step - loss: 0.1098 - accuracy: 0.9581
Epoch 5/10
17500/17500 [==============================] - 927s 53ms/step - loss: 0.0949 - accuracy: 0.9642
Epoch 6/10
17500/17500 [==============================] - 928s 53ms/step - loss: 0.0836 - accuracy: 0.9686
Epoch 7/10
17500/17500 [==============================] - 938s 54ms/step - loss: 0.0754 - accuracy: 0.9721
Epoch 8/10
17500/17500 [==============================] - 943s 54ms/step - loss: 0.0681 - accuracy: 0.9747
Epoch 9/10
17500/17500 [==============================] - 954s 55ms/step - loss: 0.0631 - accuracy: 0.9767
Epoch 10/10
17500/17500 [==============================] - 951s 54ms/step - loss: 0.0597 - accuracy: 0.9779
<keras.callbacks.History at 0x1cf84b52f40>
```

```
model.evaluate(test_data, y_test)
```

```
1188/1188 [==============================] - 13s 11ms/step - loss: 0.2112 - accuracy: 0.9359
[0.211231991648674, 0.9358684420585632]
```

Procedure

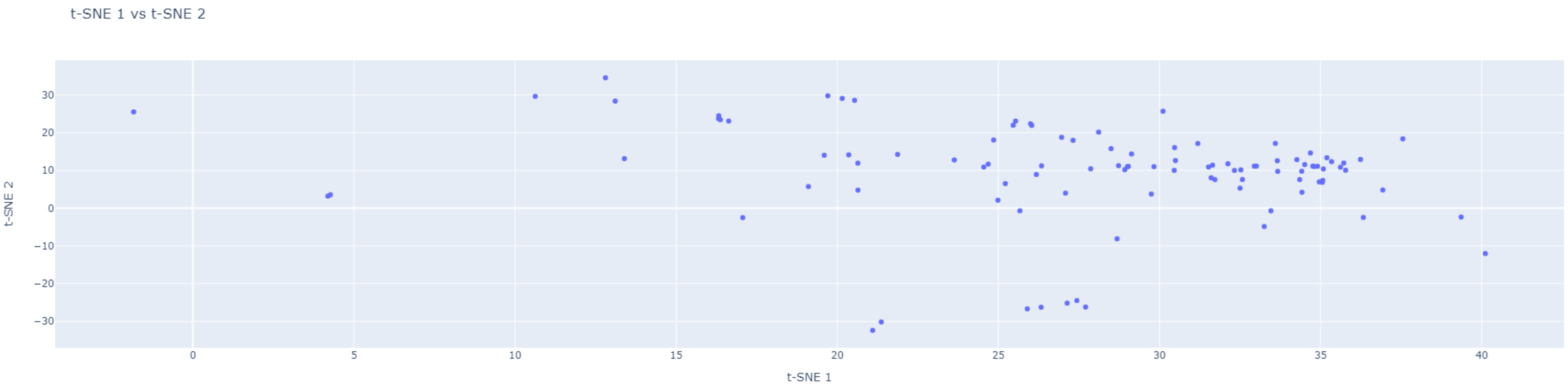


데이터 수집

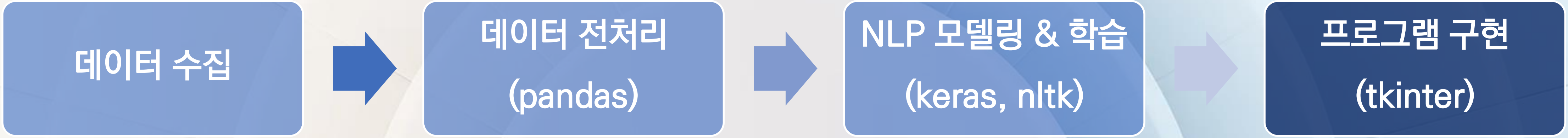
데이터 전처리 (pandas)

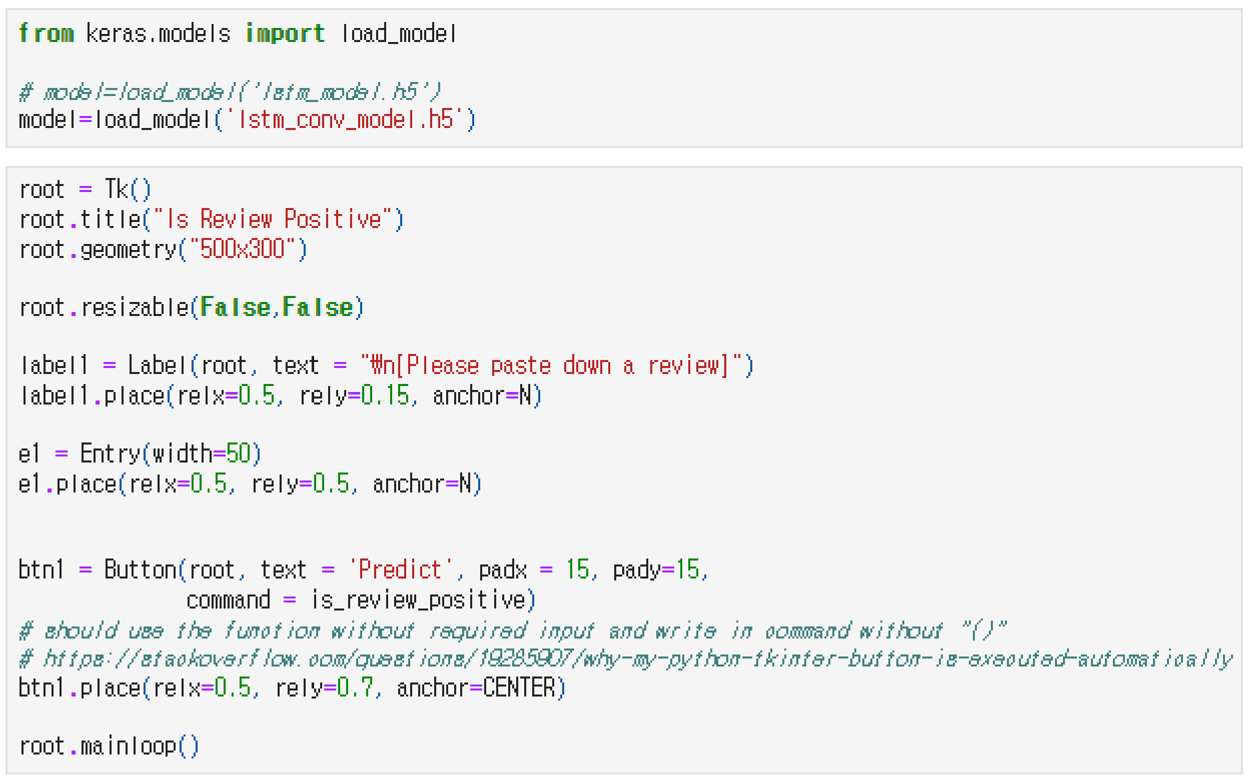
NLP 모델링 & 학습 (keras, nltk)

프로그램 구현 (tkinter)



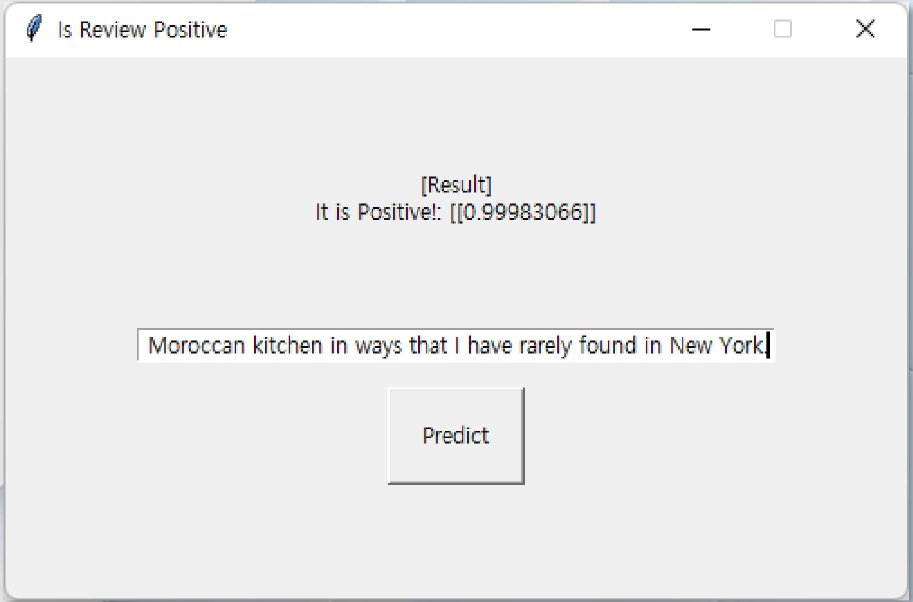
Procedure





```
from keras.models import load_model

# model=load_model('lstm_model.h5')
model=load_model('lstm_conv_model.h5')
```

```
root = Tk()
root.title("Is Review Positive")
root.geometry("500x300")

root.resizable(False,False)

label1 = Label(root, text = "₩n[Please paste down a review]")
label1.place(relx=0.5, rely=0.15, anchor=N)

e1 = Entry(width=50)
e1.place(relx=0.5, rely=0.5, anchor=N)


btn1 = Button(root, text = 'Predict', padx = 15, pady=15,
              command = is_review_positive)
# should use the function without required input and write in command without "()"
# https://stackoverflow.com/questions/19285907/why-my-python-tkinter-button-is-executed-automatically
btn1.place(relx=0.5, rely=0.7, anchor=CENTER)

root.mainloop()
```

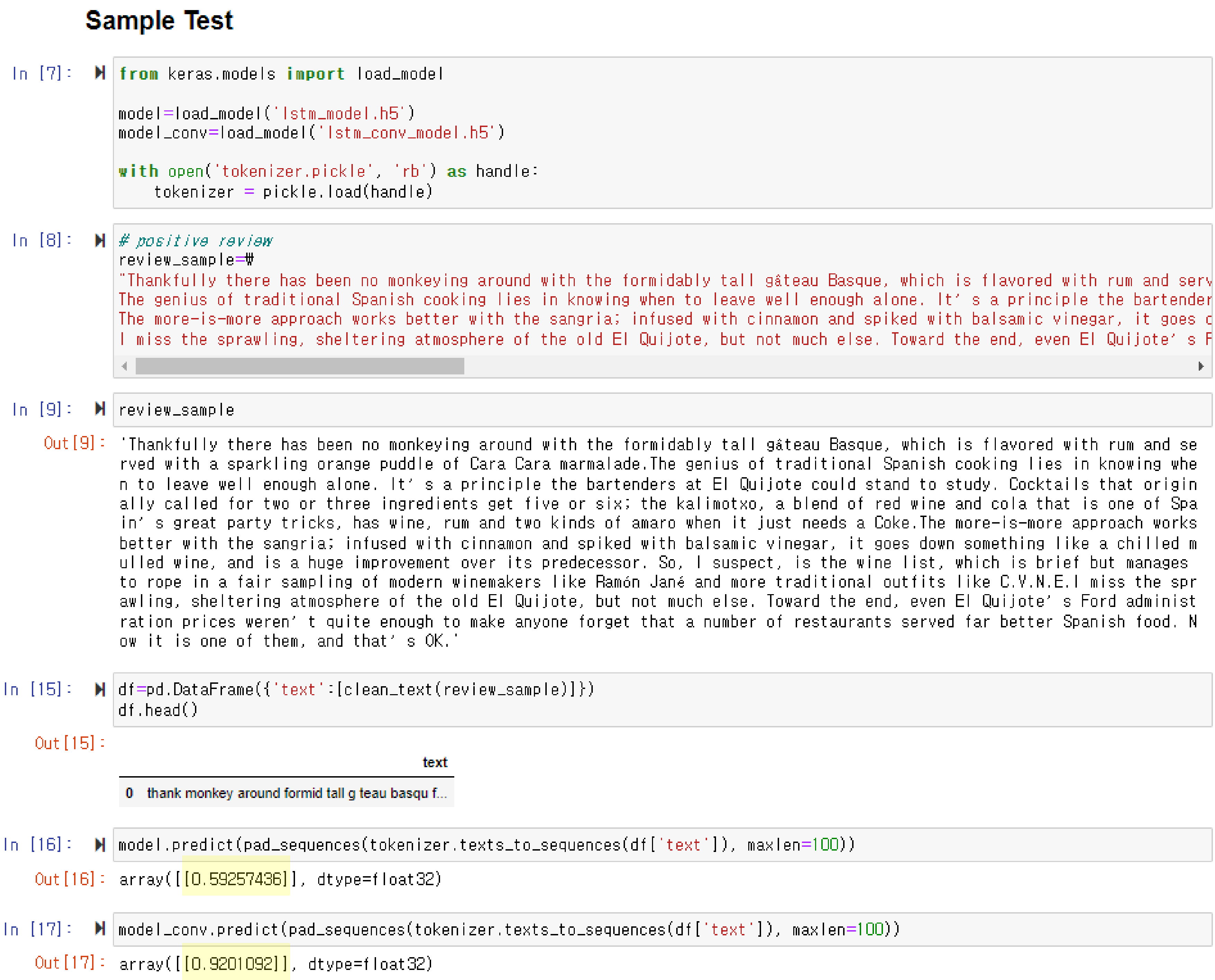


```
[Result]
It is Positive!: [[0.99983066]]
```

Moroccan kitchen in ways that I have rarely found in New York.

Demo

Lessons



Sample Test

```
In [7]:
from keras.models import load_model

model=load_model('lstm_model.h5')
model_conv=load_model('lstm_conv_model.h5')

with open('tokenizer.pickle', 'rb') as handle:
    tokenizer = pickle.load(handle)
```

```
In [8]:
# positive review
review_sample=₩
"Thankfully there has been no monkeying around with the formidably tall gâteau Basque, which is flavored with rum and serv
The genius of traditional Spanish cooking lies in knowing when to leave well enough alone. It’ s a principle the bartender
The more-is-more approach works better with the sangria; infused with cinnamon and spiked with balsamic vinegar, it goes c
I miss the sprawling, sheltering atmosphere of the old El Quijote, but not much else. Toward the end, even El Quijote’ s F
```

```
In [9]:
review_sample
```

```
Out[9]: 'Thankfully there has been no monkeying around with the formidably tall gâteau Basque, which is flavored with rum and se
rved with a sparkling orange puddle of Cara Cara marmalade.The genius of traditional Spanish cooking lies in knowing whe
n to leave well enough alone. It’ s a principle the bartenders at El Quijote could stand to study. Cocktails that origin
ally called for two or three ingredients get five or six; the kalimotxo, a blend of red wine and cola that is one of Spa
in’ s great party tricks, has wine, rum and two kinds of amaro when it just needs a Coke.The more-is-more approach works
better with the sangria; infused with cinnamon and spiked with balsamic vinegar, it goes down something like a chilled m
ulled wine, and is a huge improvement over its predecessor. So, I suspect, is the wine list, which is brief but manages
to rope in a fair sampling of modern winemakers like Ramón Jané and more traditional outfits like C.V.N.E.I miss the spr
awling, sheltering atmosphere of the old El Quijote, but not much else. Toward the end, even El Quijote’ s Ford administ
ration prices weren’ t quite enough to make anyone forget that a number of restaurants served far better Spanish food. N
ow it is one of them, and that’ s OK.'
```

```
In [15]:
df=pd.DataFrame({'text':[clean_text(review_sample)]})
df.head()
```

Out[15]:

	text
0	thank monkey around formid tall g teau basqu f...

```
In [16]:
model.predict(pad_sequences(tokenizer.texts_to_sequences(df['text']), maxlen=100))
```

```
Out[16]: array([[0.59257436]], dtype=float32)
```

```
In [17]:
model_conv.predict(pad_sequences(tokenizer.texts_to_sequences(df['text']), maxlen=100))
```

```
Out[17]: array([[0.9201092]], dtype=float32)
```

Price?

Instagram?

Taste?

Time?

Date?

Home–made?