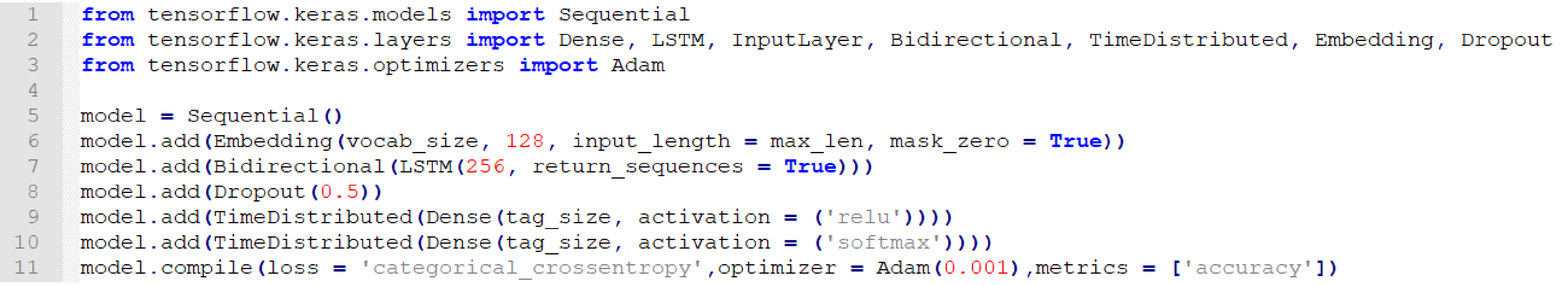
Model Description

Advanced AI(CSE750200)

Term project

2019310290 Sangman Jung

LSTMmodel.py



```python
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM, InputLayer, Bidirectional, TimeDistributed, Embedding, Dropout
from tensorflow.keras.optimizers import Adam

model = Sequential()
model.add(Embedding(vocab_size, 128, input_length = max_len, mask_zero = True))
model.add(Bidirectional(LSTM(256, return_sequences = True)))
model.add(Dropout(0.5))
model.add(TimeDistributed(Dense(tag_size, activation = ('relu'))))
model.add(TimeDistributed(Dense(tag_size, activation = ('softmax'))))
model.compile(loss = 'categorical_crossentropy',optimizer = Adam(0.001),metrics = ['accuracy'])
```

Library : Keras

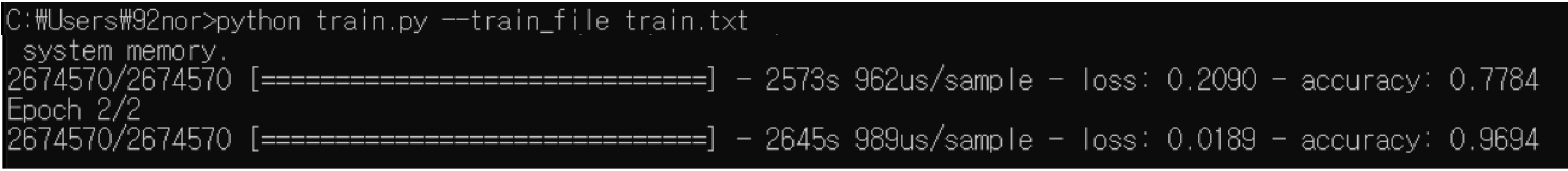
Hypothesis model : Bidirectional LSTM

Activation : dence = Relu , output = Softmax

Loss function : Categorical cross entropy

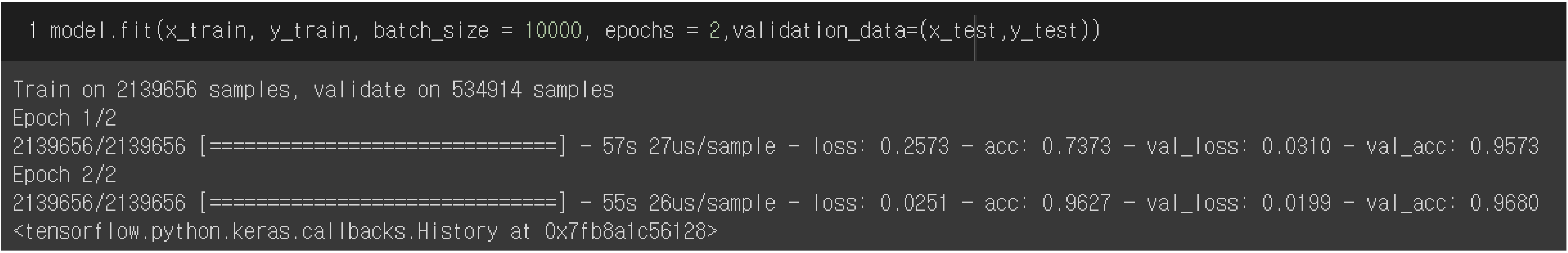
Optimizer : Adam, lr = 0.001

Model accuracy



```
C:₩Users₩92nor>python train.py --train_file train.txt
 system memory.
2674570/2674570 [==============================] - 2573s 962us/sample - loss: 0.2090 - accuracy: 0.7784
Epoch 2/2
2674570/2674570 [==============================] - 2645s 989us/sample - loss: 0.0189 - accuracy: 0.9694
```

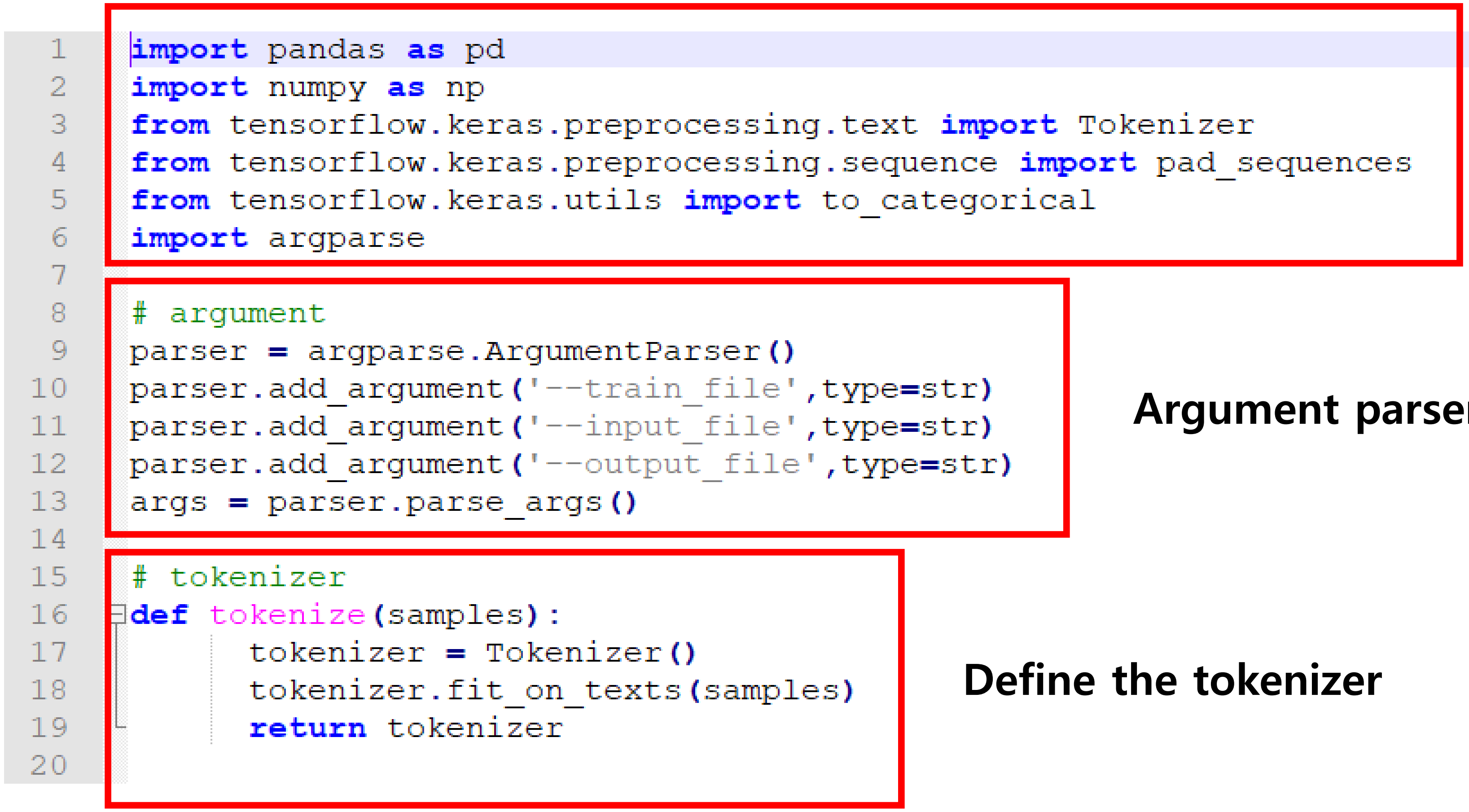
My laptop (only CPU) / training accuracy : 0.9694, batch size = 10000, epochs = 2



```
model.fit(x_train, y_train, batch_size = 10000, epochs = 2,validation_data=(x_test,y_test))

Train on 2139656 samples, validate on 534914 samples
Epoch 1/2
2139656/2139656 [==============================] - 57s 27us/sample - loss: 0.2573 - acc: 0.7373 - val_loss: 0.0310 - val_acc: 0.9573
Epoch 2/2
2139656/2139656 [==============================] - 55s 26us/sample - loss: 0.0251 - acc: 0.9627 - val_loss: 0.0199 - val_acc: 0.9680
<tensorflow.python.keras.callbacks.History at 0x7fb8a1c56128>
```

Google CoLab (GPU) / accuracy of (train, valid) : (0.9627,0.9680), used train.txt only (train_test_split in sklearn), batch size and epochs is the same as above.

train.py (1)



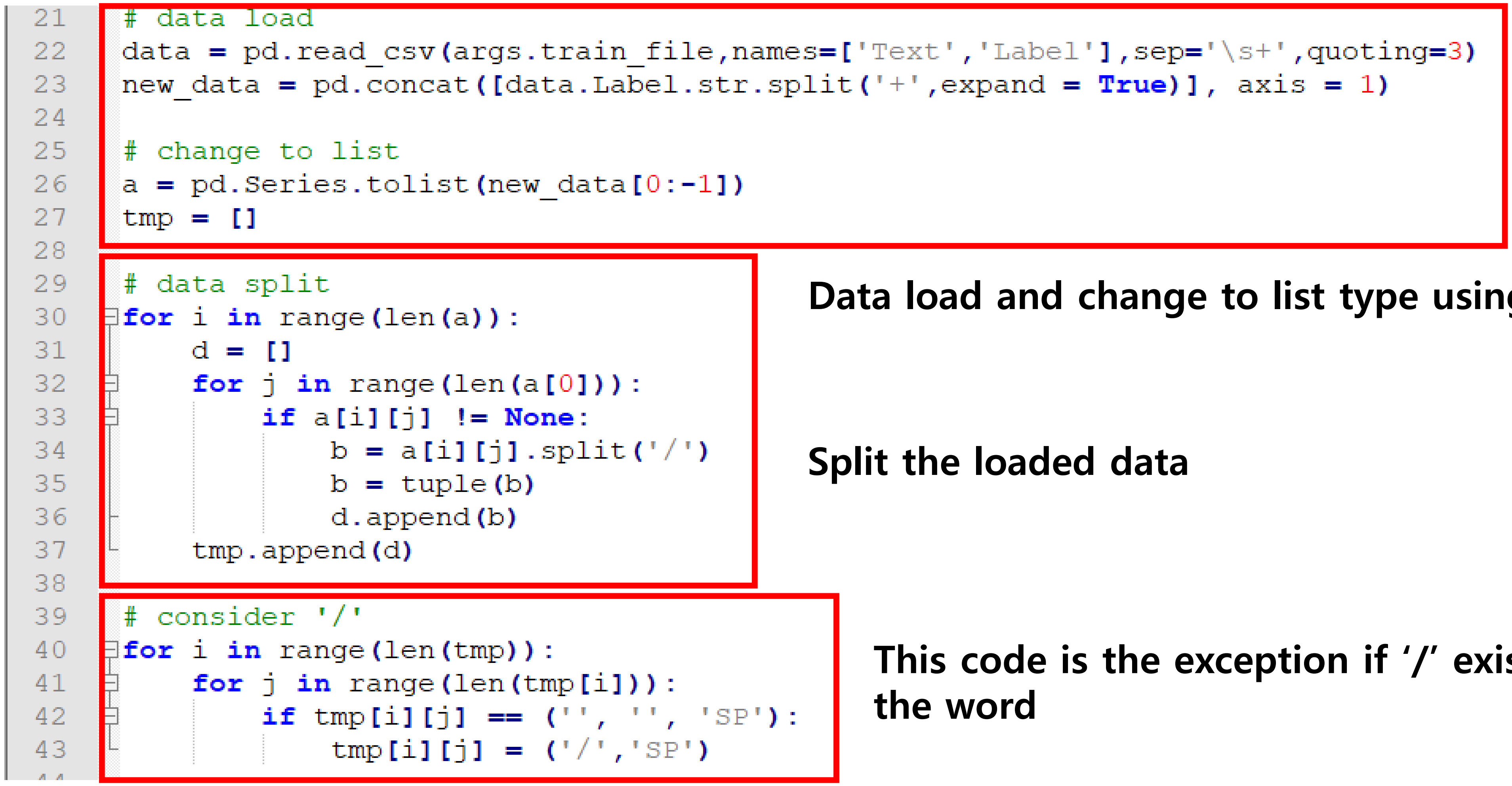
```
import pandas as pd
import numpy as np
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.utils import to_categorical
import argparse

# argument
parser = argparse.ArgumentParser()
parser.add_argument('--train_file',type=str)
parser.add_argument('--input_file',type=str)
parser.add_argument('--output_file',type=str)
args = parser.parse_args()

# tokenizer
def tokenize(samples):
    tokenizer = Tokenizer()
    tokenizer.fit_on_texts(samples)
    return tokenizer

```

Library import

Argument parser in terminal

Define the tokenizer

train.py (2)



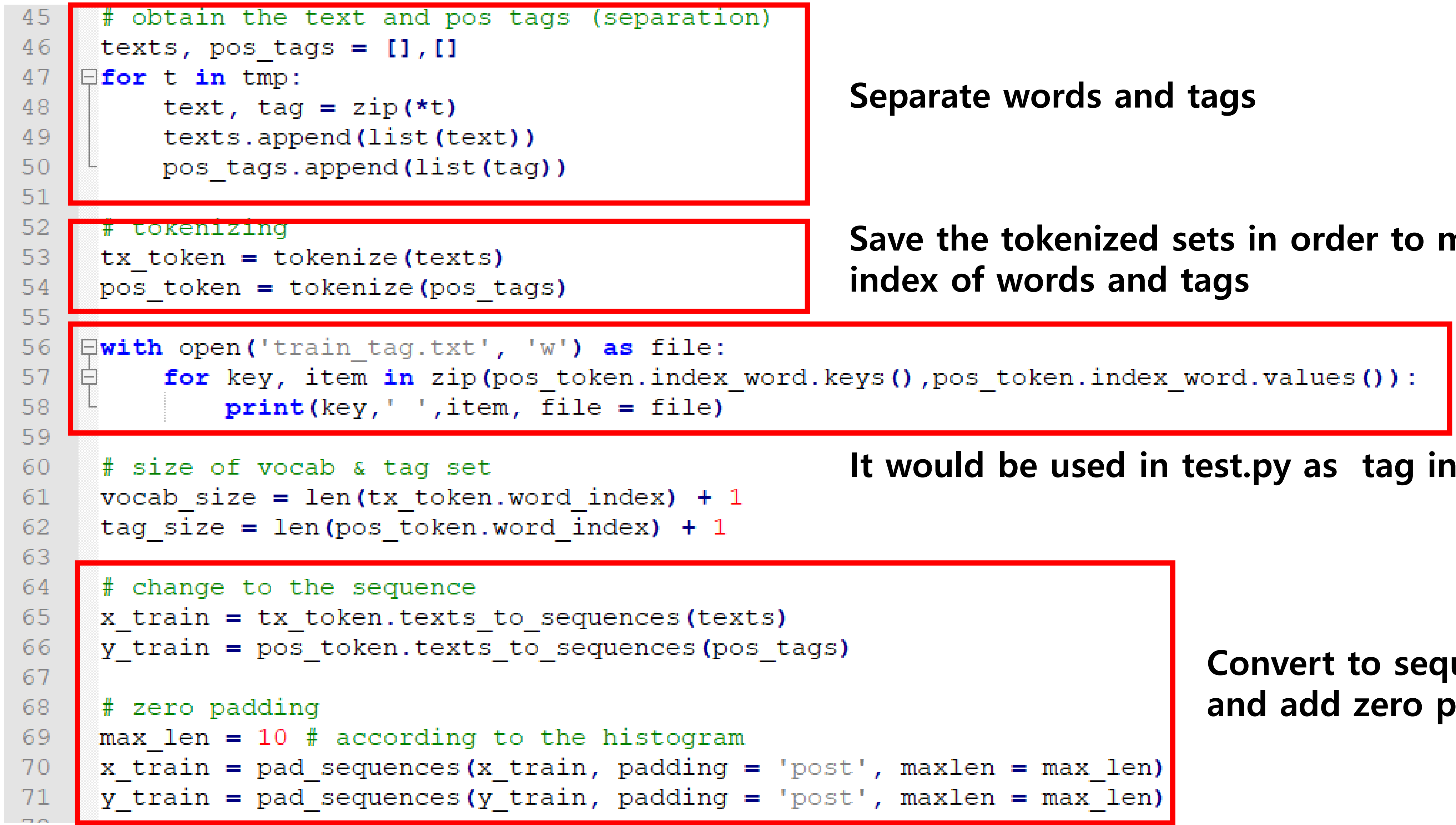
```
# data load
data = pd.read_csv(args.train_file,names=['Text','Label'],sep='\s+',quoting=3)
new_data = pd.concat([data.Label.str.split('+',expand = True)], axis = 1)

# change to list
a = pd.Series.tolist(new_data[0:-1])
tmp = []

# data split
for i in range(len(a)):
    d = []
    for j in range(len(a[0])):
        if a[i][j] != None:
            b = a[i][j].split('/')
            b = tuple(b)
            d.append(b)
    tmp.append(d)

# consider '/'
for i in range(len(tmp)):
    for j in range(len(tmp[i])):
        if tmp[i][j] == ('', '', 'SP'):
            tmp[i][j] = ('/','SP')
```

Data load and change to list type using Pandas

Split the loaded data

This code is the exception if '/' exists in the word

train.py (3)



```python
# obtain the text and pos tags (separation)
texts, pos_tags = [],[]
for t in tmp:
    text, tag = zip(*t)
    texts.append(list(text))
    pos_tags.append(list(tag))

# tokenizing
tx_token = tokenize(texts)
pos_token = tokenize(pos_tags)

with open('train_tag.txt', 'w') as file:
    for key, item in zip(pos_token.index_word.keys(),pos_token.index_word.values()):
        print(key,' ',item, file = file)

# size of vocab & tag set
vocab_size = len(tx_token.word_index) + 1
tag_size = len(pos_token.word_index) + 1

# change to the sequence
x_train = tx_token.texts_to_sequences(texts)
y_train = pos_token.texts_to_sequences(pos_tags)

# zero padding
max_len = 10 # according to the histogram
x_train = pad_sequences(x_train, padding = 'post', maxlen = max_len)
y_train = pad_sequences(y_train, padding = 'post', maxlen = max_len)
```

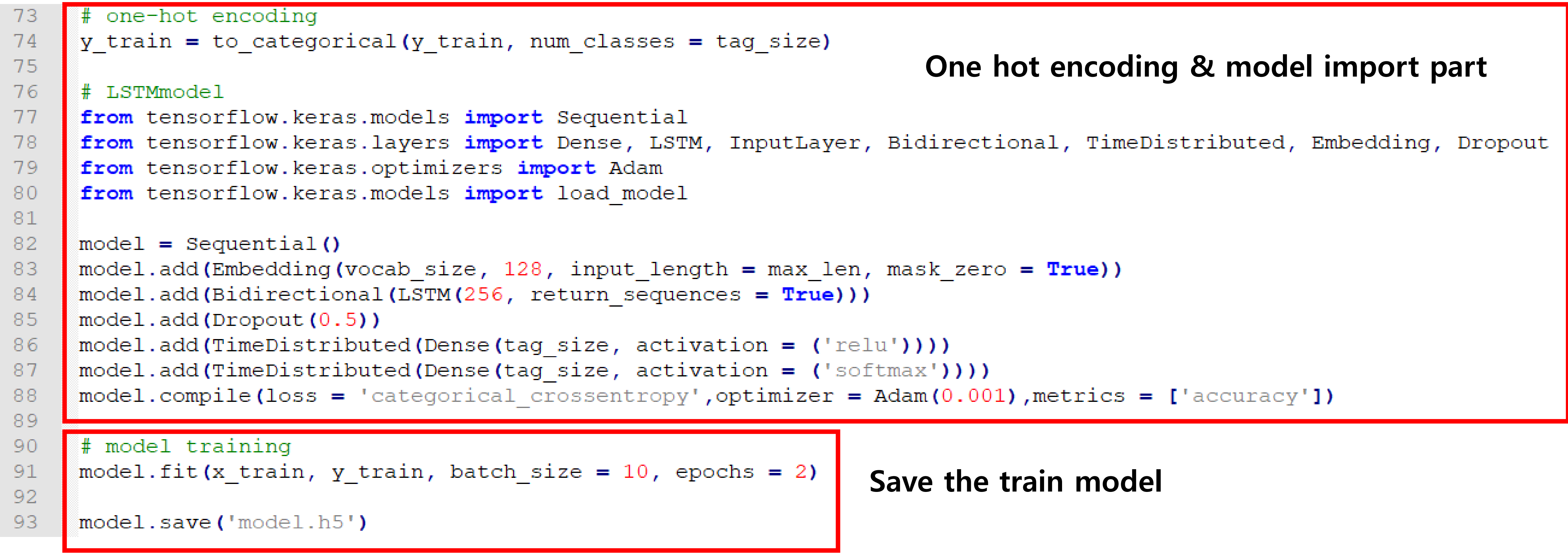
Separate words and tags

Save the tokenized sets in order to make the index of words and tags

It would be used in test.py as tag index

Convert to sequence data and add zero padding

train.py (4)

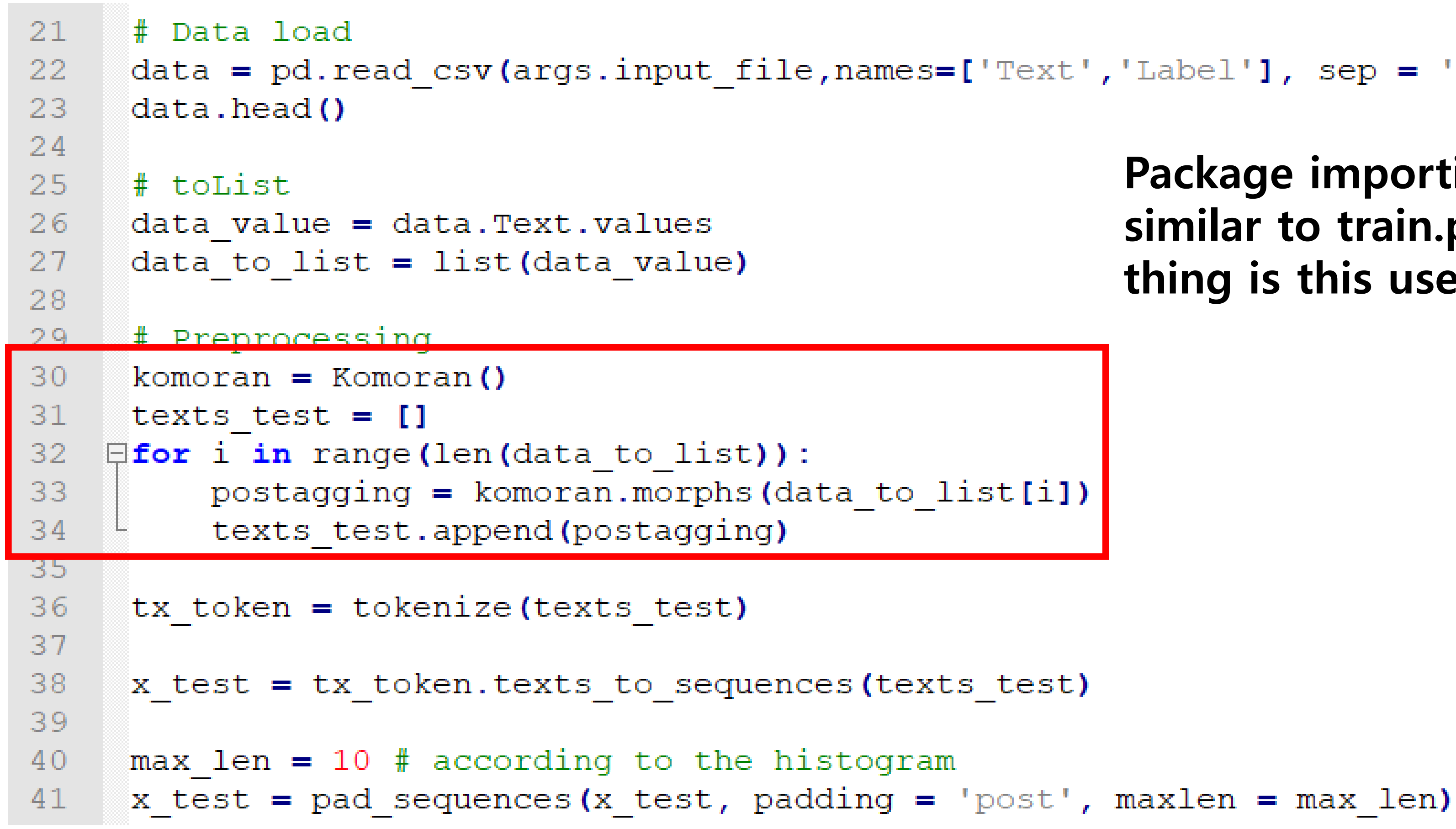


```
# one-hot encoding
y_train = to_categorical(y_train, num_classes = tag_size)

# LSTMmodel
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM, InputLayer, Bidirectional, TimeDistributed, Embedding, Dropout
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.models import load_model

model = Sequential()
model.add(Embedding(vocab_size, 128, input_length = max_len, mask_zero = True))
model.add(Bidirectional(LSTM(256, return_sequences = True)))
model.add(Dropout(0.5))
model.add(TimeDistributed(Dense(tag_size, activation = ('relu'))))
model.add(TimeDistributed(Dense(tag_size, activation = ('softmax'))))
model.compile(loss = 'categorical_crossentropy',optimizer = Adam(0.001),metrics = ['accuracy'])

# model training
model.fit(x_train, y_train, batch_size = 10, epochs = 2)

model.save('model.h5')
```

One hot encoding & model import part

Save the train model

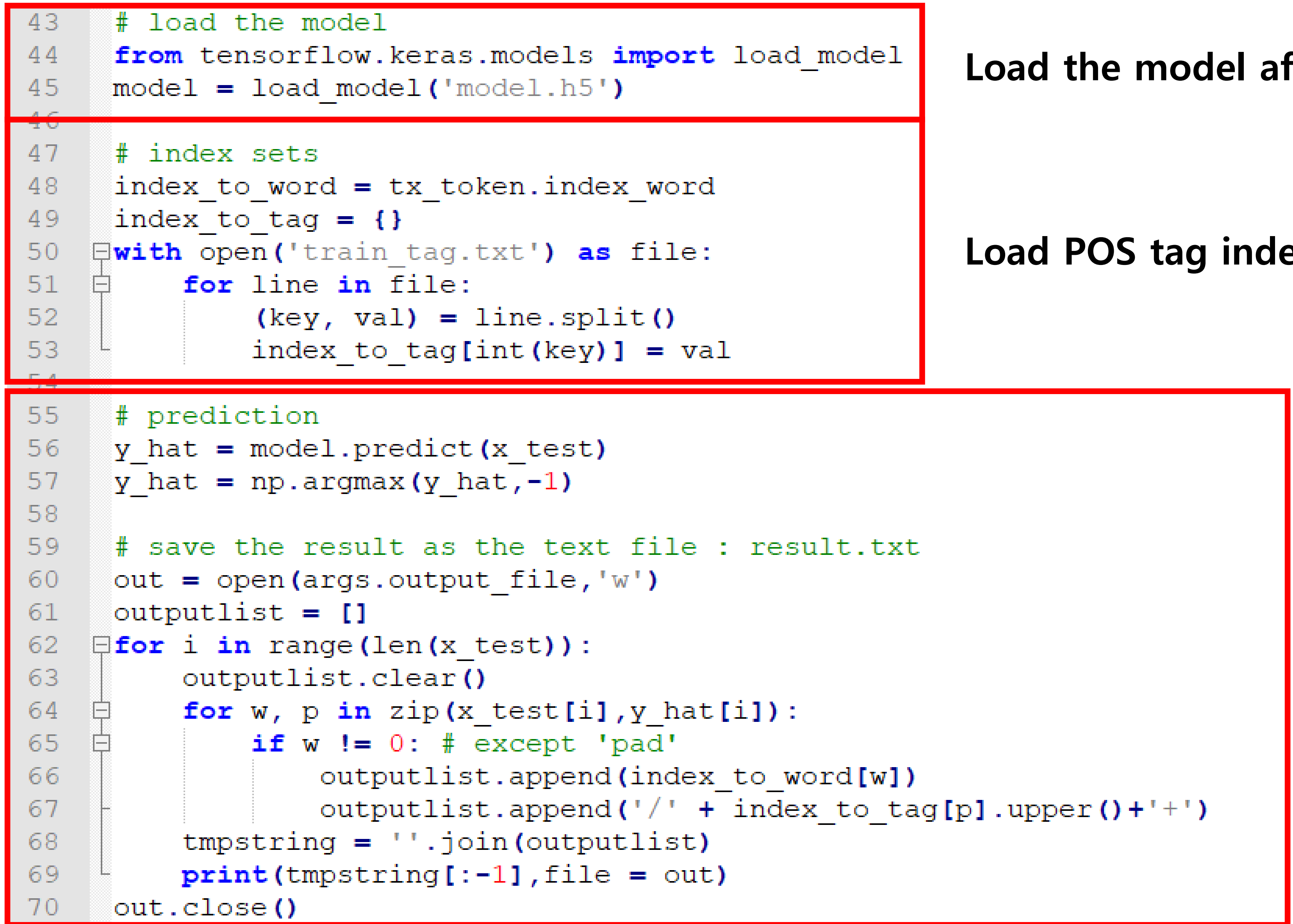
test.py (1)



```python
# Data load
data = pd.read_csv(args.input_file,names=['Text','Label'], sep = '\s+',quoting = 3)
data.head()

# toList
data_value = data.Text.values
data_to_list = list(data_value)

# Preprocessing
komoran = Komoran()
texts_test = []
for i in range(len(data_to_list)):
    postagging = komoran.morphs(data_to_list[i])
    texts_test.append(postagging)

tx_token = tokenize(texts_test)

x_test = tx_token.texts_to_sequences(texts_test)

max_len = 10 # according to the histogram
x_test = pad_sequences(x_test, padding = 'post', maxlen = max_len)
```

Package importing and data loading is similar to train.py, but the different thing is this used konlpy for tokenizing

test.py (2)



```python
# load the model
from tensorflow.keras.models import load_model
model = load_model('model.h5')
```

Load the model after excute train.py

```python

# index sets
index_to_word = tx_token.index_word
index_to_tag = {}
with open('train_tag.txt') as file:
    for line in file:
        (key, val) = line.split()
        index_to_tag[int(key)] = val
```

Load POS tag index set in train.py

```python

# prediction
y_hat = model.predict(x_test)
y_hat = np.argmax(y_hat,-1)

# save the result as the text file : result.txt
out = open(args.output_file,'w')
outputlist = []
for i in range(len(x_test)):
    outputlist.clear()
    for w, p in zip(x_test[i],y_hat[i]):
        if w != 0: # except 'pad'
            outputlist.append(index_to_word[w])
            outputlist.append('/' + index_to_tag[p].upper()+'+')
    tmpstring = ''.join(outputlist)
    print(tmpstring[:-1],file = out)
out.close()
```

Model prediction part

Save as .txt file named by result.txt