DL Final Competition

0853420_林若瑜

簡單敘述實作執行方式之 README 或報告：

一開始先把資料讀進去

把 keyword 和 title 合併訓練

```
train_title = list(train_df['title'])
train_keyword = list(train_df['keyword'])
test_title = list(test_df['title'])
test_keyword = list(test_df['keyword'])
```

```
train_title_keyword = []
test_title_keyword = []
for x,y in zip(train_title,train_keyword):
    train_title_keyword.append(str(x)+str(y))
for x,y in zip(test_title,test_keyword):
    test_title_keyword.append(str(x)+str(y))
```

用 jieba 分割中文字

```
train_df['title_keyword'] = train_title_keyword
test_df['title_keyword'] = test_title_keyword
```

```
train_df['title_keyword(token)'] = train_df.title_keyword.apply(jieba.lcut)
```

```
test_df['title_keyword(token)'] = test_df.title_keyword.apply(jieba.lcut)
```

```
# train_df
```

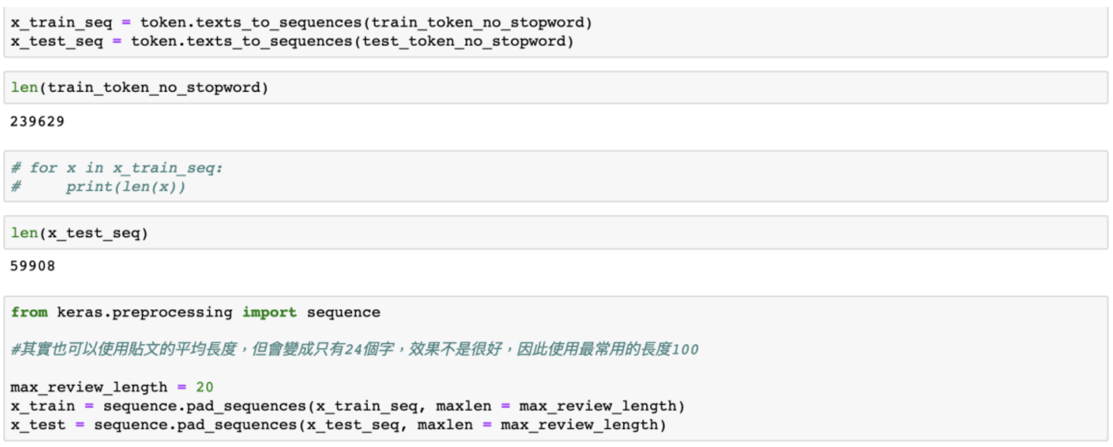
```
train_token = list(train_df['title_keyword(token)'])
test_token = list(test_df['title_keyword(token)'])
```

去除一些 stop word

```
# Read chinese stopword file

all_doc = []
f = open('cn_stopwords.txt','r',encoding="utf-8")
doc = f.read().splitlines()
all_doc.append(doc)
stop_word_all = all_doc[0]
```

```
train_token_no_stopword = []
test_token_no_stopword = []

stop_word = ['。','？','nan','！','，',',','“','”','.','..','...','!','!!','!!!','2','3','4','5','6','7','8','9',':','"',
                ',', '.', ':', ';', '?', '(', ')', '[', ']', '&', '!', '*', '@', '#', '%',' ：']

for x in train_token:
    temp = []
    for y in x:
        if y not in stop_word and y not in stop_word_all:
            temp.append(y)
    train_token_no_stopword.append(temp)

for x in test_token:
    temp = []
    for y in x:

        if y not in stop_word and y not in stop_word_all:
            temp.append(y)
    test_token_no_stopword.append(temp)
```

把它轉成可訓練的資料形式



```
x_train_seq = token.texts_to_sequences(train_token_no_stopword)
x_test_seq = token.texts_to_sequences(test_token_no_stopword)
```

```
len(train_token_no_stopword)
```

```
239629
```

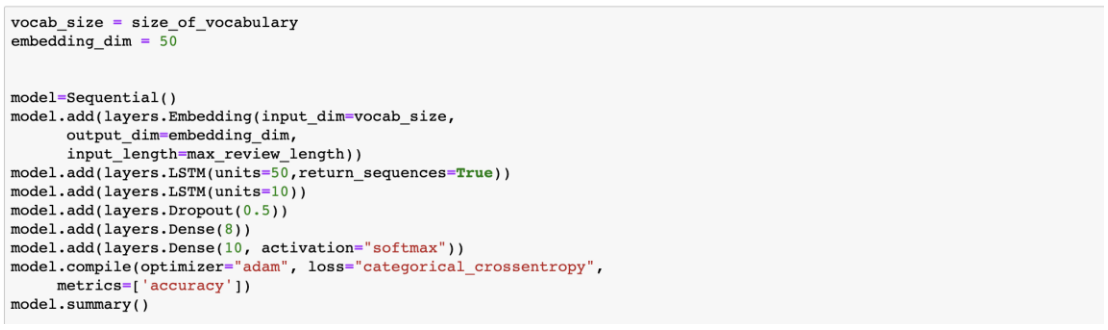
```
# for x in x_train_seq:
#     print(len(x))
```

```
len(x_test_seq)
```

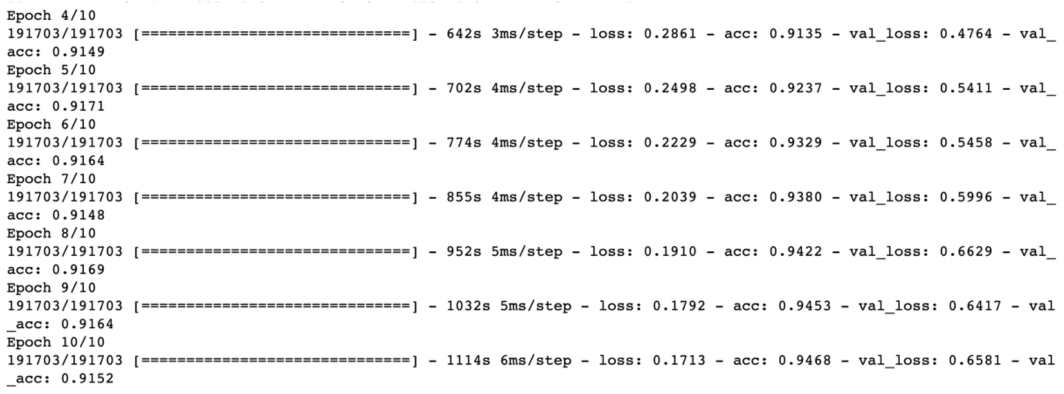
```
59908
```

```
from keras.preprocessing import sequence

#其實也可以使用貼文的平均長度，但會變成只有24個字，效果不是很好，因此使用最常用的長度100

max_review_length = 20
x_train = sequence.pad_sequences(x_train_seq, maxlen = max_review_length)
x_test = sequence.pad_sequences(x_test_seq, maxlen = max_review_length)
```

建立 lstm 模型訓練



```
vocab_size = size_of_vocabulary
embedding_dim = 50


model=Sequential()
model.add(layers.Embedding(input_dim=vocab_size,
      output_dim=embedding_dim,
      input_length=max_review_length))
model.add(layers.LSTM(units=50,return_sequences=True))
model.add(layers.LSTM(units=10))
model.add(layers.Dropout(0.5))
model.add(layers.Dense(8))
model.add(layers.Dense(10, activation="softmax"))
model.compile(optimizer="adam", loss="categorical_crossentropy",
     metrics=['accuracy'])
model.summary()
```



```
Epoch 4/10
191703/191703 [==============================] - 642s 3ms/step - loss: 0.2861 - acc: 0.9135 - val_loss: 0.4764 - val_
acc: 0.9149
Epoch 5/10
191703/191703 [==============================] - 702s 4ms/step - loss: 0.2498 - acc: 0.9237 - val_loss: 0.5411 - val_
acc: 0.9171
Epoch 6/10
191703/191703 [==============================] - 774s 4ms/step - loss: 0.2229 - acc: 0.9329 - val_loss: 0.5458 - val_
acc: 0.9164
Epoch 7/10
191703/191703 [==============================] - 855s 4ms/step - loss: 0.2039 - acc: 0.9380 - val_loss: 0.5996 - val_
acc: 0.9148
Epoch 8/10
191703/191703 [==============================] - 952s 5ms/step - loss: 0.1910 - acc: 0.9422 - val_loss: 0.6629 - val_
acc: 0.9169
Epoch 9/10
191703/191703 [==============================] - 1032s 5ms/step - loss: 0.1792 - acc: 0.9453 - val_loss: 0.6417 - val
_acc: 0.9164
Epoch 10/10
191703/191703 [==============================] - 1114s 6ms/step - loss: 0.1713 - acc: 0.9468 - val_loss: 0.6581 - val
_acc: 0.9152
```

最後把 test 的預測存出一個 excel 檔