Sentiment Analysis on Movie Reviews

李謦伊

My ID: Ching-I LEE

1.Data preprocessing

Put forward Phrase and Sentiment .

And convert Sentiment to five categories , Sentiment_0 , Sentiment_1, Sentiment_2 , Sentiment_3 , Sentiment_4 , respectively.

Let these five categories as y_train.

```
In [19]: train_phrase=train['Phrase']   #將需要讀取的內容提取出來
         train=pd.get_dummies(train,columns=[ "Sentiment" ])
         y_train=train.iloc[:,3:8]
         y_train
```

Out[19]:

	Sentiment_0	Sentiment_1	Sentiment_2	Sentiment_3	Sentiment_4
0	0	1	0	0	0
1	0	0	1	0	0
2	0	0	1	0	0
3	0	0	1	0	0
4	0	0	1	0	0
5	0	0	1	0	0
6	0	0	1	0	0
7	0	0	1	0	0
8	0	0	1	0	0
9	0	0	1	0	0

Then I build token. (num_words=20000)

```
from keras.datasets import imdb
from keras.preprocessing import sequence        #用於資料預處理
from keras.preprocessing.text import Tokenizer
token = Tokenizer(num_words=20000)  #使用Tokenizer建立token，20000字的字典
token.fit_on_texts(train_phrase)     #讀取所有訓練資料
```

Convert characters to numbers.

```
#將每一篇文章的文字轉換一連串的數字，只有在字典中的文字會轉換為數字
x_train_seq = token.texts_to_sequences(train_phrase)
```

Limited the maximum number of characters is 30. If the number length is greater than 30, discard the number over 30. By contrast, add 0 to the front.

Next , let x_train denote these numbers.

```
x_train = sequence.pad_sequences(x_train_seq, maxlen=30)
```

2.How did you design your model?

First , I used embedding to convert a numeric list to a vector list. Convert each number to a 32-dimensional vector.

Second , I added dropout layer , in order to avoid overfitting.

Third ,I used LSTM and established LSTM layer of 32 neurons.

Fourth , I established hidden layer and output layer.

Among of them, the number of neurons in the hidden layer I set to 256 , and "relu" as an activation function. The number of neurons in the output layer I set to 5 , because the output is five. And "sigmoid" as an activation function.

Finally, I used "binary_crossentropy" as loss function. After I train the model.

```
from keras.models import Sequential
from keras.layers.core import Dense, Dropout, Activation,Flatten
from keras.layers.embeddings import Embedding
from keras.layers.recurrent import LSTM


model = Sequential()
model.add(Embedding(output_dim=32,input_dim=20000, input_length=30))
model.add(Dropout(0.5))

model.add(LSTM(32))

model.add(Dense(units=256,activation='relu' ))
model.add(Dropout(0.5))

model.add(Dense(units=5,activation='sigmoid' ))

model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

train_history =model.fit(x_train, y_train,batch_size=300, epochs=35 ,verbose=2 ,validation_split=0.2)
```

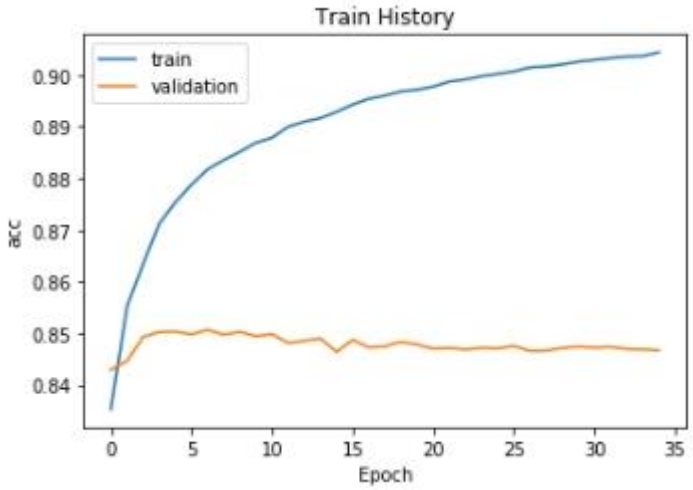
```
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_8 (Embedding)      (None, 30, 32)            640000
_________________________________________________________________
dropout_15 (Dropout)         (None, 30, 32)            0
_________________________________________________________________
lstm_8 (LSTM)                (None, 32)                8320
_________________________________________________________________
dense_15 (Dense)             (None, 256)               8448
_________________________________________________________________
dropout_16 (Dropout)         (None, 256)               0
_________________________________________________________________
dense_16 (Dense)             (None, 5)                 1285
=================================================================
Total params: 658,053
Trainable params: 658,053
Non-trainable params: 0
```

3. What hyper parameters you choose and why?

- I added dropout layer, and chose 0.5 as dropout rate. Let each training iteration randomly give up 50% of the neurons to avoid Overfitting.
- I used LSTM , because language processing has timing. Used to solve the problem of gradient vanishing or exploding when training with RNN.
- My batch size chose 300 , because data is large , batch size should be set a little more. But if the batch size is set too much, it will not work well. After testing , it is the best that choosing 300 batch size.
- As batch size increases, the number of epochs required to achieve the same accuracy is increasing. After testing , it is the best that choosing 35 epochs.

4. Chart of the training progress (if you train a NN model)



5. The screenshot of submission in Kaggle with my ID and score