Hw2 – 109403021

Text Generation 連結: 連結

StockRNN 連結: 連結

1. Text Generation

先利用提供的爬蟲程式下載了金庸的笑傲江湖來做為此次更改使用的資料集，然後載入資料集

```
url = "https://www.bookscool.com/%E9%87%91%E5%BA%B8%E3%80%8A%E7%AC%91%E5%82%B2%E6%B1%9F%E6%B9%96%E3%80%8B.php/0.xhtml"
download_one_book(url)
```

```
儲存位置: ./books/金庸《笑傲江湖》.txt
100%|██████████| 41/41 [00:31<00:00,  1.28it/s]
```

```
!gdown --id 1E4YxL1ApsfwOpTjxBvf9C2sYbcGG7oVx --output "./金庸《笑傲江湖》.txt"

# !wget -O Eileen_Legendary.txt "http://140.115.82.54/NN/Recurrent/Eileen_Legendar
```

```
/usr/local/lib/python3.7/dist-packages/gdown/cli.py:131: FutureWarning: Option `--id` wa
  category=FutureWarning,
Downloading...
From: https://drive.google.com/uc?id=1E4YxL1ApsfwOpTjxBvf9C2sYbcGG7oVx
To: /content/金庸《笑傲江湖》.txt
100% 2.95M/2.95M [00:00<00:00, 248MB/s]
```

```python
init_seq = "任盈盈"
init_seq_ind = [word_2_index[w] for w in init_seq]
input = init_seq_ind[-seq_len:]

generateWords(input,500)
```

Python

```
任盈盈死了，你怎地如此胡塗？」岳靈珊道：「我……我怎能是除了辟邪劍譜？」林平之道：「我怎知道？」林平之道：「我在恆山上出手，你殺我而已。」
令狐冲「嗯」了一聲，心想：「原來是一個小小婆娘打扮的，有甚麼不好？」岳靈珊道：「是，是。」林平之道：「不錯，我自宮之後，仍和你師娘人數雖大，說過的話，卻和田伯光一定喜歡了喉嚨，便要走上山去。」
令狐冲和盈盈又感欣點嘴角，不知去向。



令狐冲和盈盈交個朋友求死，大家分明以後，必定要一場不勝。」說著挺劍直向他左肩。令狐冲見過要倒地的好看，已然熟了，也沒加傷的傷口，便道：「好，我送你去見師娘。」
令狐冲默然不語。
這日令狐冲已經得過今日，他一直要自己拿了去，自己自然只有自殺盈盈，隨即也切他不得其事，心下也就不想，但從前十餘人也是十分聰明機境，其實本事雖在頗深，心想：「這位佛門高僧不通世務，當真得深自可。」
耳聽得定靜師太從南安客店中出來。
※※※
令狐冲躬身從馬匹走了開去，一張椅便給噴了出去。
令狐冲忽覺右臂上劇痛，右臂小腹中登時露出了一隻青布長袍，只嚇得手中發出半截聲音，跟著砰拍之聲大作。
令狐冲心想：「青城派那姓余的小子無冤無仇，為甚麼要拘留任小姐？你是
```

最後我使用”任盈盈”做為開頭產生出以上的文章，雖然一些文法如上下引號的使用有一些錯誤，但是可以看出仍有一些字詞的正確對應如各角色的姓名、林平之知道自己自宮過。

2. StockRNN

我選擇了 2317:FOXCONN 這家公司來做訓練、預測

```
#####################################
#                                   #
#  自己決定三家公司哪家公司(填數字)#
#  挑選一家公司來做訓練、預測       #
#                                   #
#                                   #
#####################################
companys = {"2330":"TSMC","2454":"MEDIATEK","2317":"FOXCONN"}
company_id = 2317
```

```
company = train_df[train_df["company"] == company_id]
company.index = company["date"]
company.head()
```

	company	date	price
date			
2012/1/2	2317	2012/1/2	81.4
2012/1/3	2317	2012/1/3	82.7
2012/1/4	2317	2012/1/4	83.2
2012/1/5	2317	2012/1/5	83.2
2012/1/6	2317	2012/1/6	83.9

將 seq_len 修改為 10，因為覺得增加觀察和預測的天數能較不擬合

```
# todo 決定Seq長度
seq_len = 10

def split_input_target(seq):
    input_txt = tf.expand_dims(seq[:-1],-1)
    target_txt = tf.expand_dims(seq[1:],-1)
    return input_txt,target_txt
```

來回訓練調整 batch size 為 25

```
# Batch size
# todo 決定Batch大小
BATCH_SIZE = 25

BUFFER_SIZE = training_len

train_ds = train_ds.shuffle(BUFFER_SIZE).batch(BATCH_SIZE)
val_ds = val_ds.batch(BATCH_SIZE)
```

建立模型。參考 TextGeneration 的模型最終疊了 3 層 LSTM，並加上 dropout 防止過度擬合，但效果其實不顯著

```
#  todo
input_shape  =  (seq_len,  1)
output_shape  =  [BATCH_SIZE,  seq_len  ,1  ]

keras.backend.clear_session()
model  =  tf.keras.Sequential(
        [
            layers.LSTM(units  =  4096,input_shape=input_shape,return_sequences=True),
            layers.LSTM(units  =  2048,input_shape=input_shape,return_sequences=True),
            layers.LSTM(units  =  1024,input_shape=input_shape,return_sequences=True),
            layers.Dropout(0.3),
            layers.Dense(1),
        ]
)

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 10, 4096)          67141632

 lstm_1 (LSTM)               (None, 10, 2048)          50339840

 lstm_2 (LSTM)               (None, 10, 1024)          12587008

 dropout (Dropout)           (None, 10, 1024)          0

 dense (Dense)               (None, 10, 1)             1025

=================================================================
Total params: 130,069,505
Trainable params: 130,069,505
Non-trainable params: 0
```

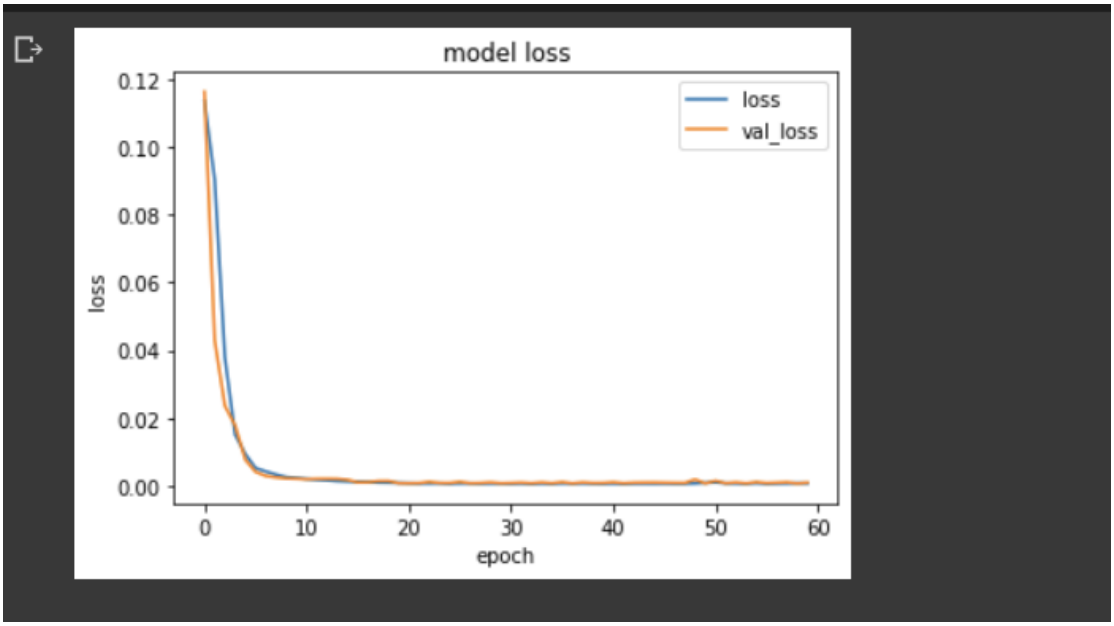
訓練了 200 次。相比上次的作業，這次使用 mse 計算 loss，而 optimizer 依然使用 adam。model.fit 中 shuffle 設為 true 打亂資料集

```
#  todo
epochs  =  200

#  todo
#  model.compile
model.compile(loss='mean_squared_error',  optimizer='adam')

#  todo
#  model.fit
history  =  model.fit(
        train_ds,
        batch_size=BATCH_SIZE,
        epochs  =  epochs,
        validation_data=val_ds,
        shuffle=True
)
```

結果只訓練了不到十次就收斂到頗小的 loss

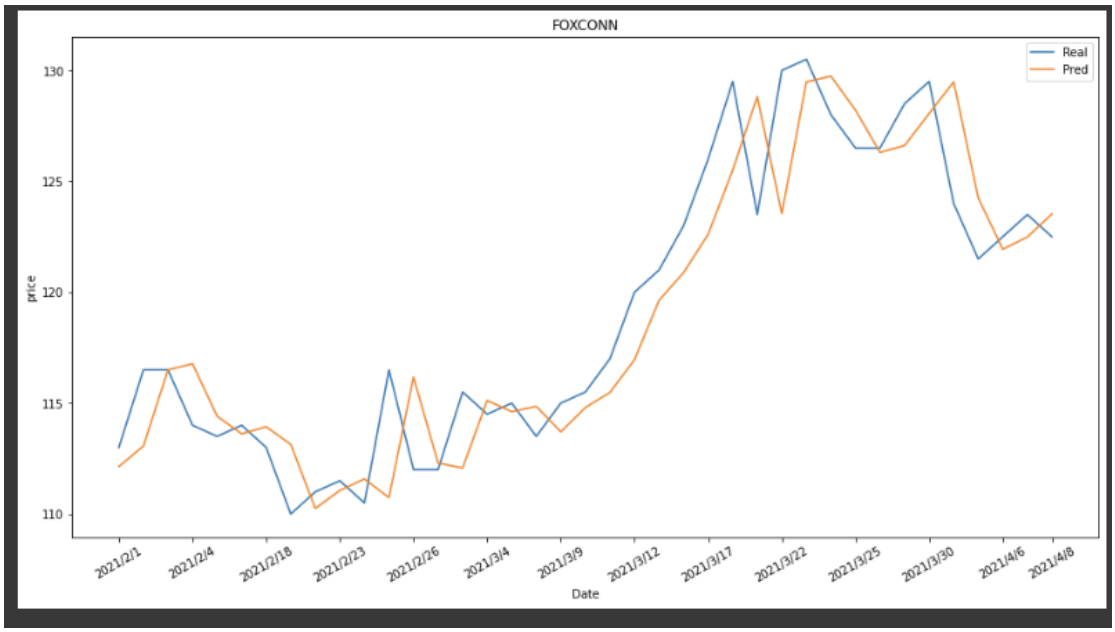


根據提示先將 input 的維度轉換為 next_input，再取出最後一個 output 值加入到 pred_prices 列表中，再將 real_price 轉為小數點形式並加入 input 列表，最後為了維持 input 的長度去掉了第一個值

```
for  index,row  in  test_company.iterrows():
    #  我已經幫你把real_price和date放入list
    real_price  =  row["price"]
    real_prices.append(real_price)
    date  =  row["date"]
    dates.append(date)

    #  參考步驟  (在for迴圈中)  :
    #  1.  把input轉**維度**成一個新變數next_input丟入model
    next_input  =  tf.expand_dims(input,axis=-1)
    next_input  =  tf.expand_dims(next_input,axis=0)
    #  2.  獲得的pred取出seq中最後一個時間點的output數值  (提示:  [0,-1,0])
    predicts  =  model(next_input)
    predicts  =  predicts[0,-1,0]
    #  3.  把pred丟入pred_prices中
    #  4.  把real_price經過transform_one轉換成小數點並加在input後面
    pred_prices.append(float(predicts))
    real_price  =  scalar.transform_one(real_price)
    input.append(real_price)
    #  5.  保持input長度為seq_len長  (也就是把最前面的數字踢掉)
    input  =  input[1:]
    ##########
    #  todo  #
    ##########
```

預測和真實的結果如下，走勢大致相同但預測的會有一些延遲感



3. 心得

這次的作業相較上次的確難度有所降低，但我覺得比上次有趣很多，文字生產器看到能產生自己喜歡的小說風格之文章，就是前後語意毫無邏輯甚至有點搞笑。由此我認為這種文字生產器基本上應該很難應用在小說這種帶有大量對話和劇情邏輯的地方，畢竟 RNN 只是藉由前一個字延續預測、產生下去，不可能顧及到文句前後的邏輯。所以這種技術應該較於適合應用在一些抽象、藝術，前後文較無強烈邏輯連結的文章，諸如新詩之類的。

股票走勢預測的作業因為我還對股市沒有接觸所以較無深刻感受，雖然預測出來的走勢和真實的十分相近，但由於是參考前面的資料做預測而有難以避免的延遲效果。而且我認為這種單純由之前的數據來預測的方式仍然不夠現實，畢竟真實的情況要考慮的層面還多得很多。不過如果能將其他層面的影響也化為參數考慮進來，可能真的就可以製造出精準度很高的股票預測器。

RNN 這種「記憶」模式的預測感覺還有很多可以實現的應用，十分有可能讓我未來想要預測或是呈現某種序列時，嘗試使用 RNN 實作、實現，因此在思考層面上這次作業激發了我不少。