NCU AI & ML HW2 RNN

資管二 A 109403019 鄒翔宇

目錄

A. HW COLAB LINK

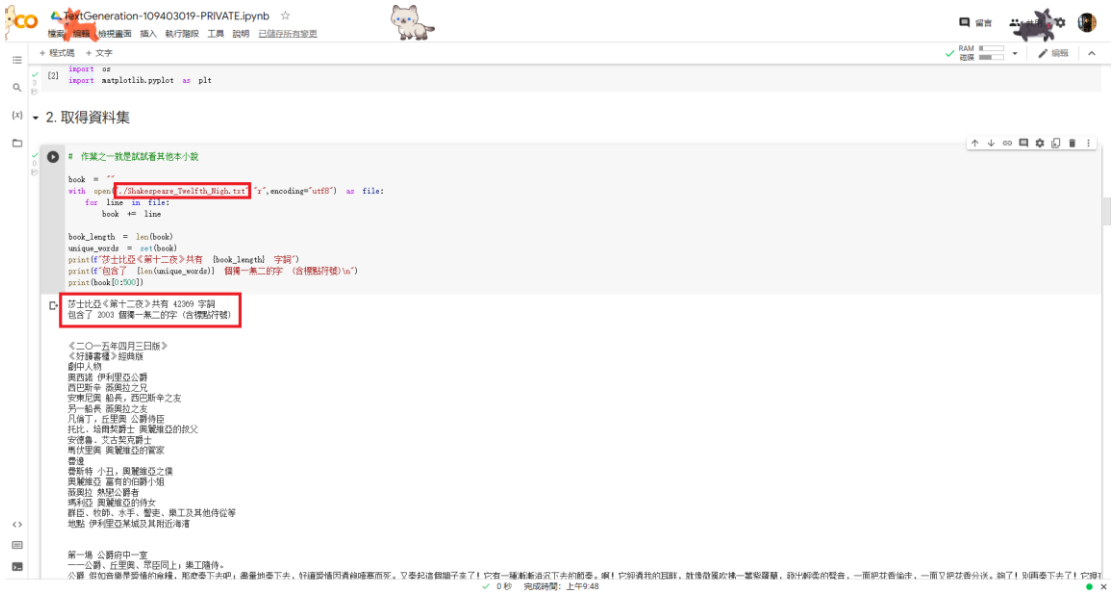
TextGeneration: click me

StockRNN: click me

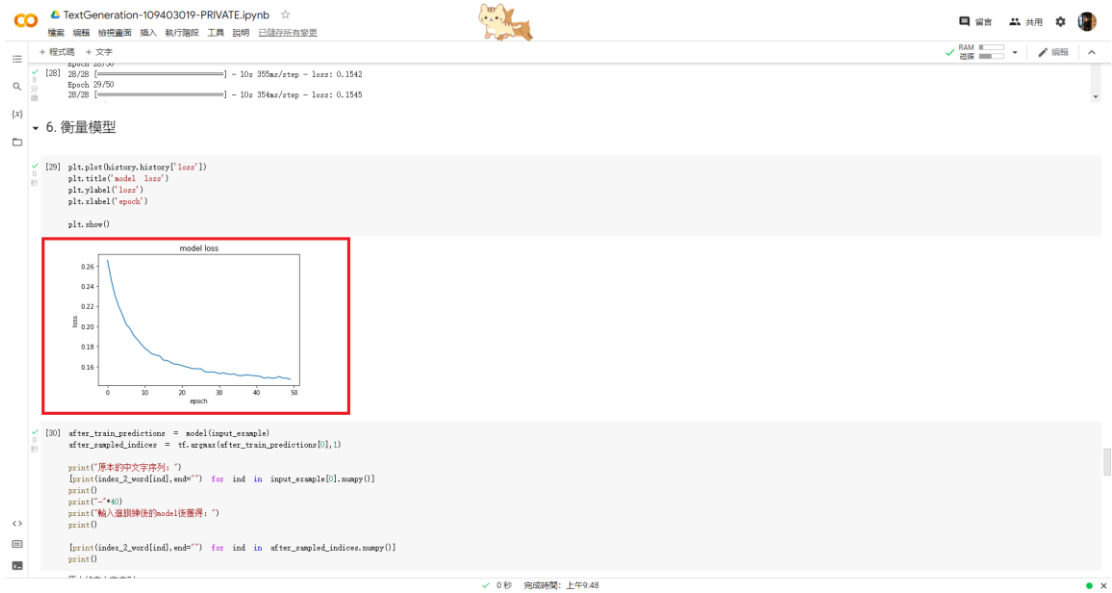
B. DEMO SCREENSHOTS

a. TextGeneration

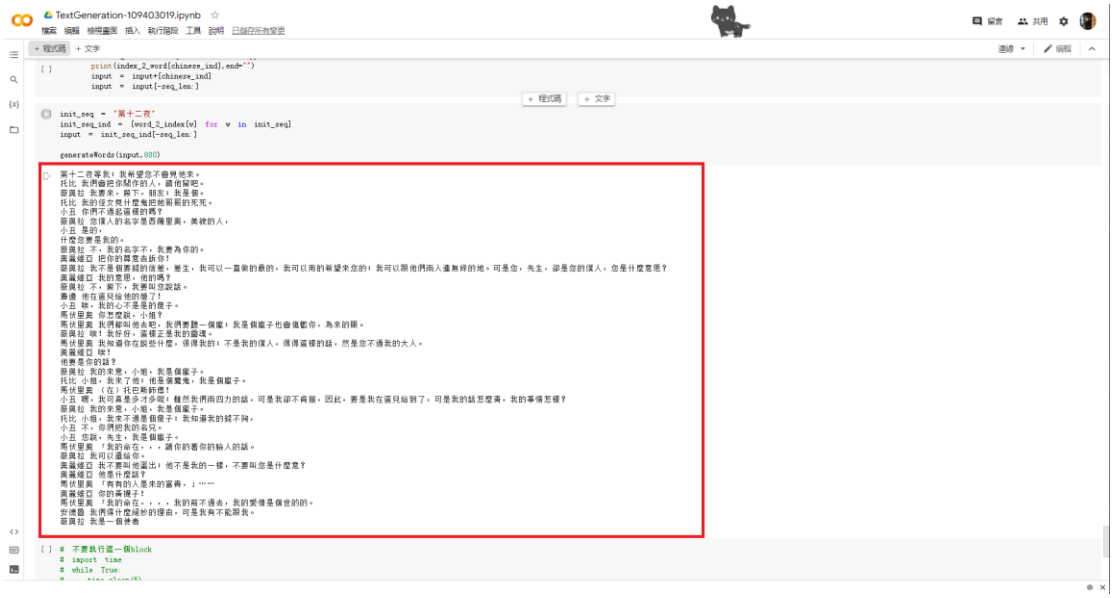
評分標準 1: 更換訓練書籍 - 莎士比亞《第十二夜》



Model Loss

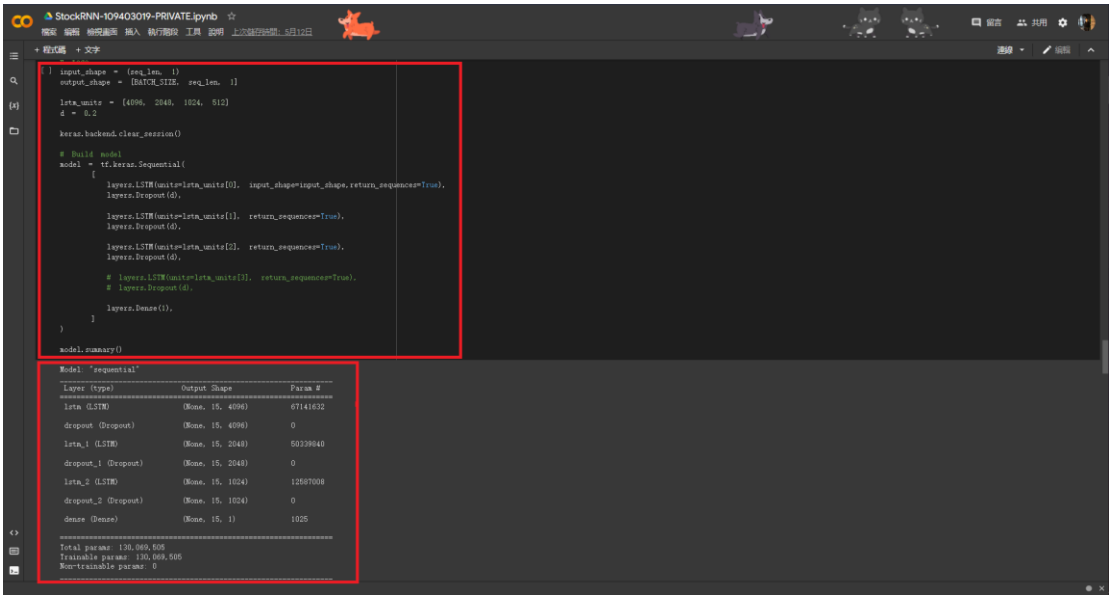


產生文本結果 - 莎士比亞《第十二夜》

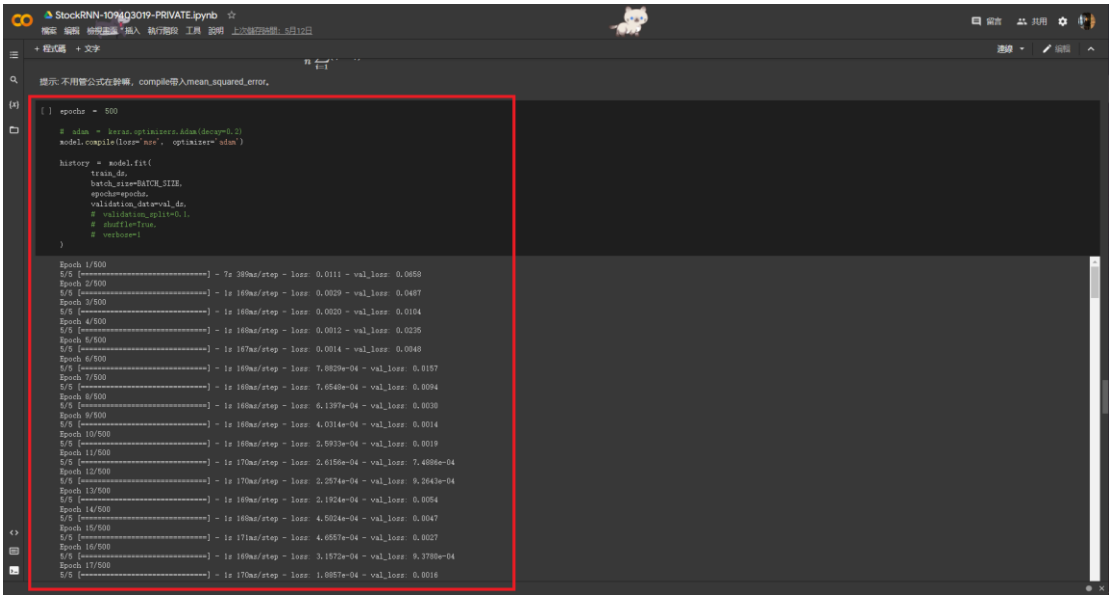


b. StockRNN

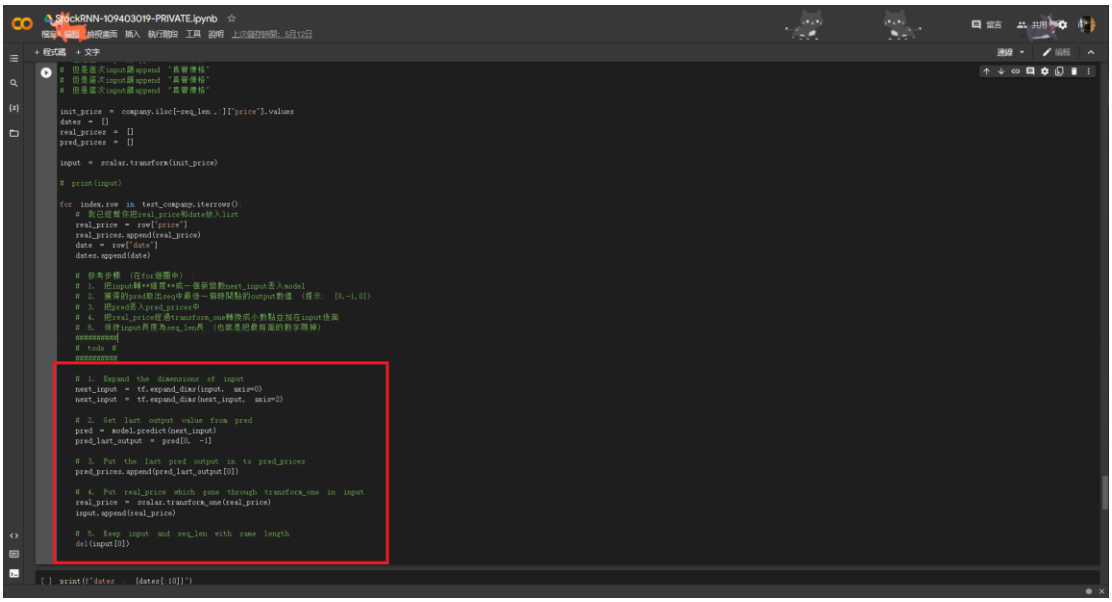
評分標準 2: 完成建立模型



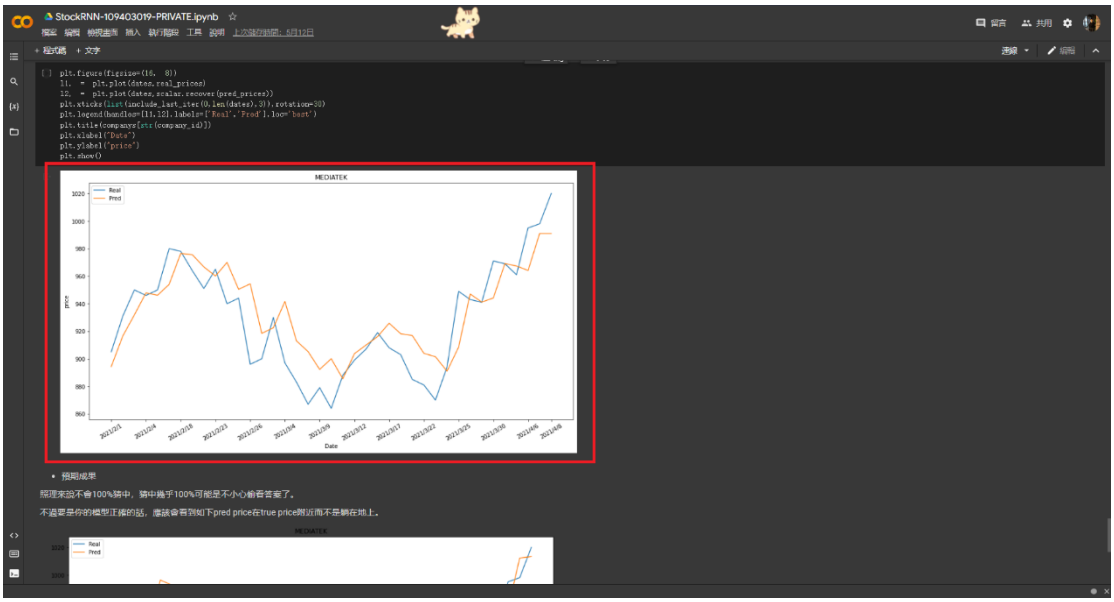
訓練過程



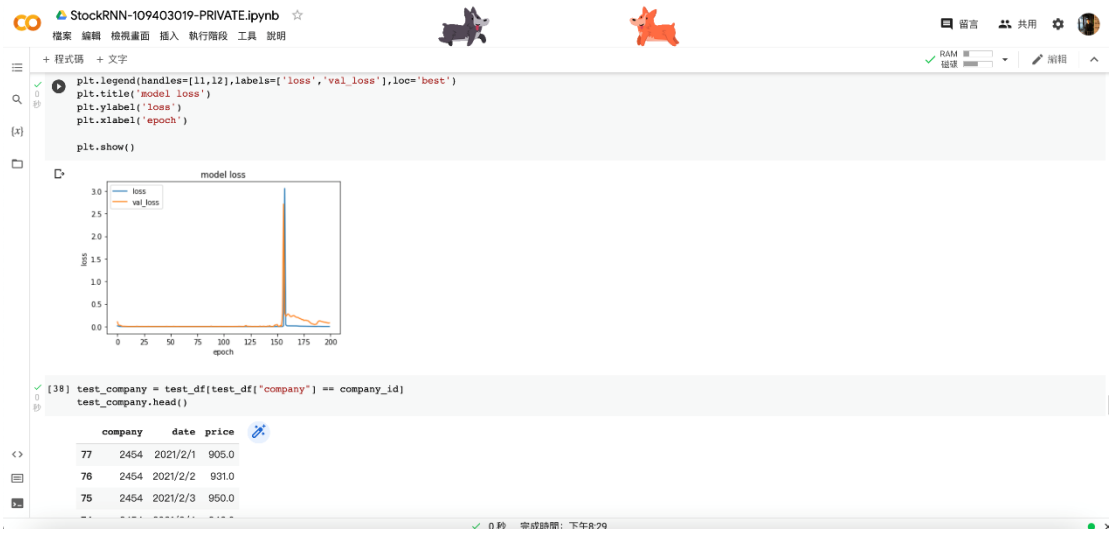
評分標準 3: Predict Funtion



Predict Result



補：愛情來的太突然 (Gradient Explode ?)



C. 撰寫過程

先將整個檔案看過再開始撰寫，基本上接照著助教的 todo 步驟撰寫。

a. TextGeneration

1. 導入 Package: 無修改。
2. 取得資料集:

   將文件改為從 Bookscool 爬蟲抓下來的書，書籍為莎士比亞的《第十二夜》。
3. 資料前處理: 將一些書籍相關文字修改，其餘無修改。
4. 建立模型: 無修改。
5. 制定訓練計畫並訓練:

   epochs 設為 50，其餘無修改。
6. 衡量模型: 無修改。
7. 做預測:

   init_seq 改為“第十二夜”，並將輸出長度改為 800，其餘無修改。

b. StockRNN

1. 導入 Package: 無修改。
2. 取得資料集:

   選擇訓練、預測的公司為 2454 (MEDIATEK)，其餘無修改。
3. 資料前處理:

   將 seq_len 設為 15，因為覺得選擇天數長一點有助於訓練以及預測，測試 3、5、15，最後 15 的結果最好。其餘無修改。
4. 建立模型:

   輸入維度為 (seq_len, 1)，輸出維度應為 [batch_size, seq_len, 1]。使用 keras 的 Sequential 建立模型，原先是疊四層 LSTM，但測試過後發先三層效果較為優秀，因此最終疊了三層 LSTM，並在每一層都 dropout(0.2)，不過實際上 dropout 的影響並不顯著。
5. 制定訓練計畫並訓練:

   epochs 設為 500，loss 基本上 50 後都差不多(沒遇到 gradient explode 的狀況下)，不過預測結果仍是 epochs 設較多次預測出的結果比較好，因此設 500。model.compile() 帶入 mse，optimizer 如往常般使用 adam。接著就使用 model.fit() 開始訓練並記錄 history。

6. 評估模型: 無修改。
7. 做預測:

首先，使用 tf.expand_dims() 去擴張 input 成三維的 next_input。第二步，要取得最後一個時間點的 output 數值，使用 model.predict() 取得預測值並擷取最後一個。第三步，將第二步驟取得的結果加到 pred_prices 的陣列後。第四步，將 real_price 轉成小數點並加到 input 陣列後。第五步，為了保持輸入以及 seq_len 的長度一致，刪去 input 最前端的值。其餘無修改。

D. 作業心得

Text Generation 的部分其實更動的不多，就只有更換訓練集，不過我認為可以應用的層面很廣。我最想運用的場景是，大家不乏都有很喜歡的作家或是歌手，但人生在世總會面臨死亡，有的作品無法得以延續並完成，甚至是產出更多作品。雖然偉大的作品正因即使作家或是歌手逝去仍不朽長存於大家的心中，但身為粉絲總是希望能看到作品的後續甚至更多新作品的產出，此時便能應用 text generation 訓練該創作者過去的或是尚未完成的作品，並做預測，以完成未完成的創作或是產出更多的作品。即使沒辦法百分之百跟原作者一模一樣，但或許能撫平大家心中的可惜，甚至激發更多的藝術靈感。

相信不少人都對賺錢很有興趣，我也是如此，說到賺錢勢必會與股票作連結，因此我對於 StockRNN 這項作業有頗大的興致，但其實這類型的預測其實只能參考或是對我來說更像是娛樂，畢竟偉大的巴菲特也說過:「用過去的資料去預測未來的股票，就像看著後照鏡往前開車。」

而 StockRNN 最讓我不解或是困難的有三點。第一，常常 model loss 的曲線很漂亮的收斂到 x 軸，但是預測結果卻不進人意，都只有實際股價的八成，這個到目前原因仍不得而知，第二，LSTM 模型的訓練非常的快速，因為跟上次作業 (CNN) 比較起來速度真的差太多，因此上網查詢，得到的結果都是說 RNN 模型訓練較為困難及耗時，最後推測是這次作業的 sequence 沒有很大，因此感受不明顯。

第三，同樣的模型，同樣 hyperparameters 的狀態下，model loss 有時正常收斂到 x 軸，有時卻震盪的很嚴重，甚至出現上面補充的截圖(但兩者狀況應該不同)。看完李宏毅教授的教學影片後找到答案，因為 RNN 的

Error Surface 是非常崎嶇的，有些地方非常平坦，有些地方非常陡峭，如果跳到非常陡峭的位置，gradient descent 變動， learning rate 卻沒跟著一起變，便會使的 loss 大爆噴，這部分可以透過 clipping (設一個門檻值，若超過門檻值就以門檻值計算) 去做處理。而 LSTM 雖然可以處理 gradient vanishing 卻不能處理 gradient explode，因此仍有些地方非常崎嶇，也導致了 loss 的震盪。

模型建立及訓練的部份仍是有趣的，因為可以透過修改參數得到不同結果，只要得到好的結果心情就會很好，而且這次訓練的速度非常的快。模型內疊了三層 LSTM，因為四層表現沒有比較好所以改回三層。dropout 的部分上網查詢幾乎都說會 dropout 並且值設為 0.2 會有最好的表現，不過自己實際跑的結果有沒有 dropout 感受不大。而訓練的部分因為訓練得很快，也沒有被鎖，所以 epochs 設到 500。同樣的的參數及模型下試了兩三次雖然 model loss 一直震盪，本來心都有點涼了，想說是不是得再重跑，結果發現預測的結果還算漂亮，因此本次的作業就告一段落了。日後待補足更多金融及 RNN 的知識，勢必回來增進模型！

E. REFERENCES

1. RNN 模型与 NLP 应用(6/9)：Text Generation (自动文本生成)
2. tf.keras.layers.LSTM
3. Keras 实战：基于 LSTM 的股价预测方法
4. https://medium.com/@fredericklee_73485/股票 lstm 分析-b3014c0f7321
5. ML Lecture 21-1: Recurrent Neural Network (Part I)
6. ML Lecture 21-2: Recurrent Neural Network (Part II)
7. rnn 为什么训练速度慢？