機器學習專論

作業 1

Abstract classification 論文分類

隊伍名稱: SDML_HSW

組員:

資工所 徐瑞亨 R08944023

資工所 沈柏瑋 R08922016

電機丙 王韋鈞 R08921112

一、 實驗目的

設計一個演算法，透過語意與文本分析的技術，創建出一個藉由分析論文摘要、標籤、等等特徵值，便能得知該篇論文所屬分類的系統。藉此幫助研究者省下統整論文類別的時間，減少分類論文所需的成本。

二、 使用設備、環境:

作業系統: Ubuntu 16.06
顯示卡: Nvidia GTX 1080 TI
使用語言: Python 3.6
使用套件: Pytorch with Cuda 9.2 Cudnn9.2

三、 實驗方法

1. 資料預處理:

我們選擇以 Abstract 與 Categories 兩個特徵來進行分類。

對於 Abstract，資料取出之後先為其斷詞，同時去除預設字典中沒有的文字或符號。接著為了能夠平行化進行模型的訓練，我們統一將斷詞後的文長補齊到 256，超過設定的文長則直接去除。最後再依預設字典中的 ID 編號對句子中的每個詞作替換。因為有作補齊的動作，在轉換的同時也會產生一個相對應的 Mask 的向量，用來代表每個文詞是否為 padding。

對於 Categories，我們事先為訓練資料中有出現的數值做了編號，之後以 one-hot-encoding 方式將特徵轉換成向量。

對於訓練資料，我們將資料以 4:1 的比例隨機切割成了 training set 與 validation set 進行訓練與評估。

2. 損失函數:

我們選擇了以 binary cross entropy with sigmoid 作為我們的 loss function，詳細的算法如從 pytorch 官方文本 [1]擷取之圖一所示。對於同個 batch 中所得到的 loss 值則做平均。

$$\ell(x, y) = L = \{l_1, \dots, l_N\}^\top, \quad l_n = -w_n \left[y_n \cdot \log \sigma(x_n) + (1 - y_n) \cdot \log(1 - \sigma(x_n)) \right]$$

圖一: BCEWithLogitsLoss，其中x 為預測值，y 為 label 值

3. 模型架構一: Bert + Linear Classifier

為了將文字丟進模型進行訓練，我們需要對句子進行 Word Embedding 的動作。由於 Bert 所產生的 Embedding 的向量會隨著模型訓練的時候一起更新，我們預期使用 Bert 的效果會不錯，所以我們選擇了

HuggingFace [2] 團隊所開源的 Pytorch 版本已預訓練好的 Bert 來做為我們 Embedding 的方法。再將句子做完 Embedding 之後，我們在後面接上了一個現行分類器來進行 Fine-tune 訓練，架構如下圖三所示。

起初我們將分類問題縮減到最簡單的單一 label 問題，便能得到 0.62 的成績，這使我們確定了使用 Bert 是可行的。隨後我們便把同個模型套用到了 Multi-label 的問題上來，準確度來到了 0.65。就在此時我們發現了個問題，無論我們如何增加線性分類器的深度或參數，準確率都沒有太大的變化。這令我們思考起是否有比 Bert 更好的 embedding 方式。以下附

圖二為我們在此架構下得到最高分的模型在訓練時的 F1-score 與 loss 圖。

4. 模型架構二: Roberta + Linear Classifier

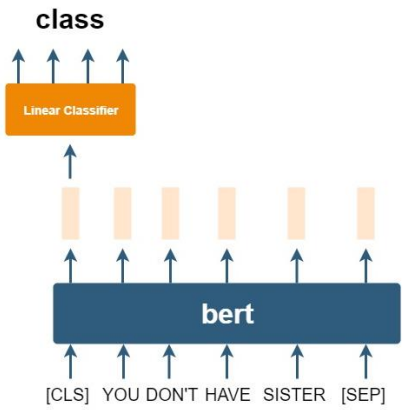
Roberta 使用了更多的參數以及更大的訓練資料來訓練，同時優化了 Bert 在訓練時所使用的所使用的訓練方式，在各大排行榜更取得了超越 Bert 的成績。因此在架構中我們嘗試了將架構一中的 Bert 改成了同樣為 HuggingFace [2]所預訓練好的 Roberta 模型來做為我們 Embedding 的方法，架構如下圖五所示。在線性分類器的參數都相同的情況下，Roberta 使我們的準確率有效的提升，並且使最高的準確率來到了 0.66，以下圖四:架構二 F1-Score 與 loss 圖為此模型在訓連時的 F1-Score 與 loss 圖。

然而在此架構中依然存在一個問題，無論我們如何修改線性分類器的參數，準確率依舊沒有太大的改變，因此我們便思考，或許我們給分類器的資訊太過簡單了，也就是說線性分類器無法單靠 Roberta 所得出的資訊便能將文章有效的進行分類。

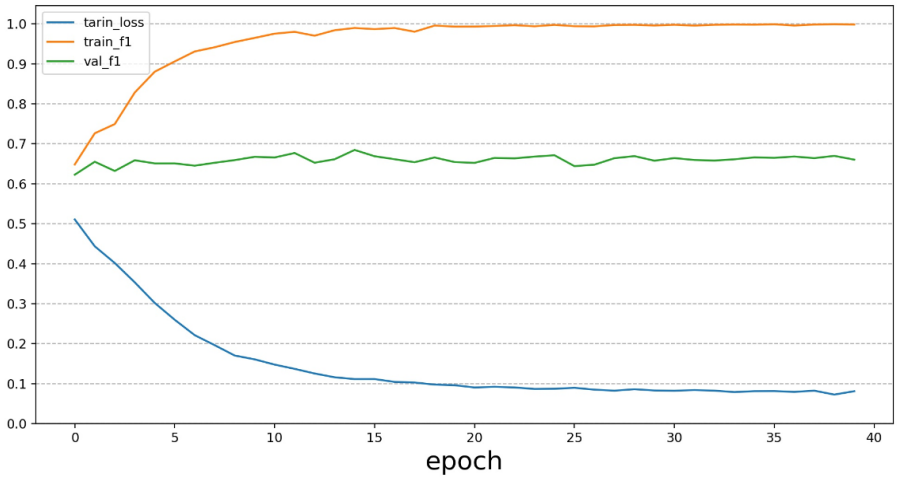
5. 模型架構三: Roberta + Rnn + Linear Classifier

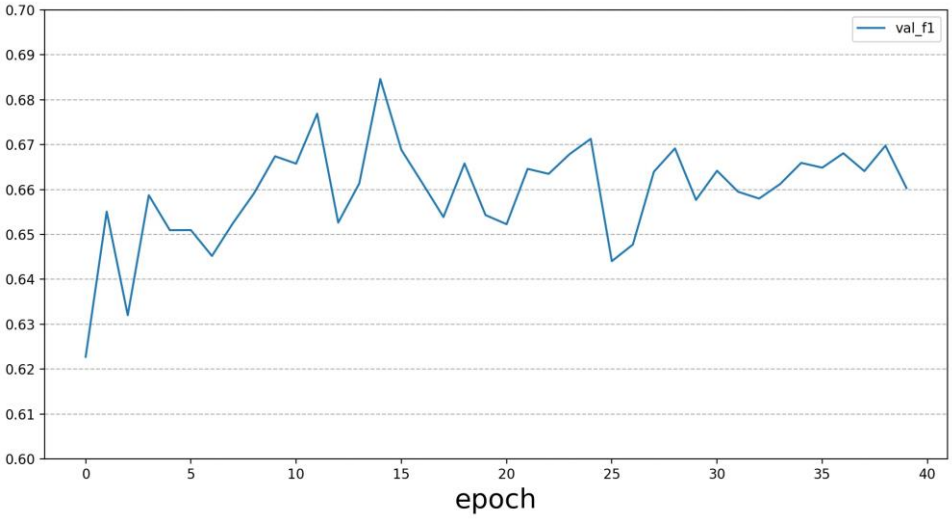
在架構二中，我們確定了 Roberta 在 Embedding 上有著不錯的效果。然而在架構二中，我們僅僅只是取出單一 Embedding 後的向量來進行分類，卻沒有考慮的文章中，文詞間上下文的關係，因此我們在模型加入了 Rnn 的機制，希望透過使用 Rnn 我們可以讓模型理解到文章中上下文的的訊息，藉此增加給予分類器的資訊，並進一步地使得整體模型準確度上升。在此架構三中，我們將 Roberta 的所 Embedding 完的文章全部接入到一層雙向的 Lstm 中，並將 RNN 取得上下的資訊傳給分類器進行分類，架構圖如下圖七所示。

加入 Rnn 之後，成功的使模型整體的準確度再度上升並且來到了 0.68 的成績，下圖六為此成績的模型訓練 F1-Score 與 loss 圖。此架構驗證了透過 Rnn 取得上下文資訊對於準確率提升是有用的。

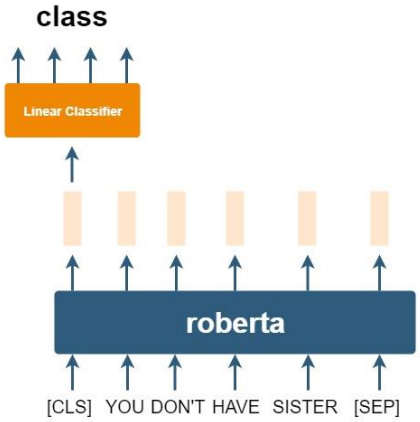


圖二: Bert + Linear Classifier

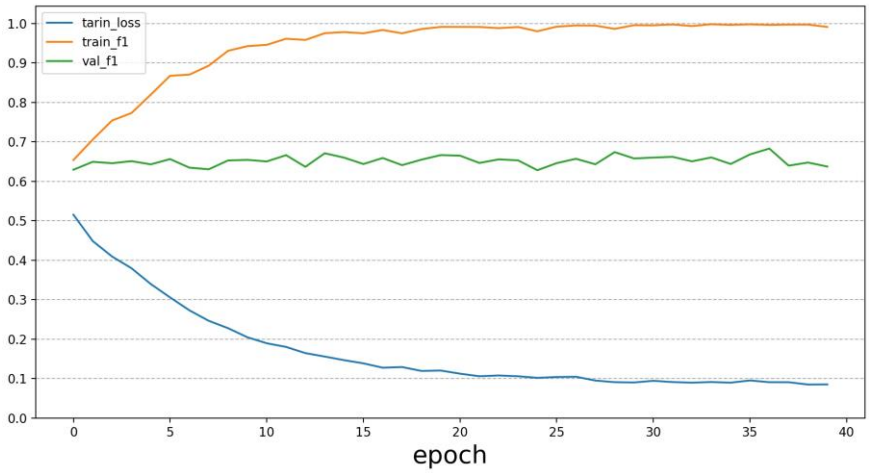


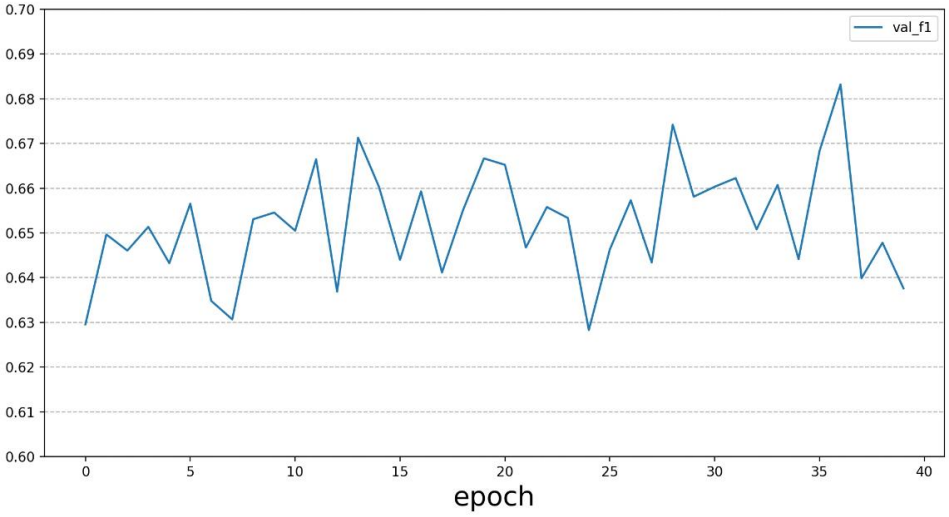


圖三:架構一 F1-Score 與 loss 圖

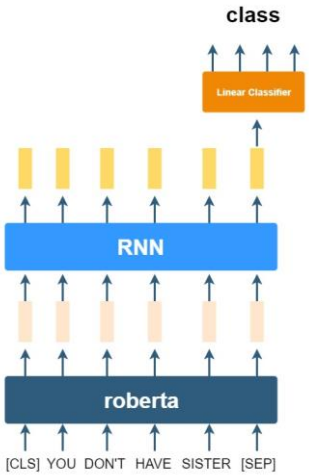


圖四: Roberta + Linear Classifier

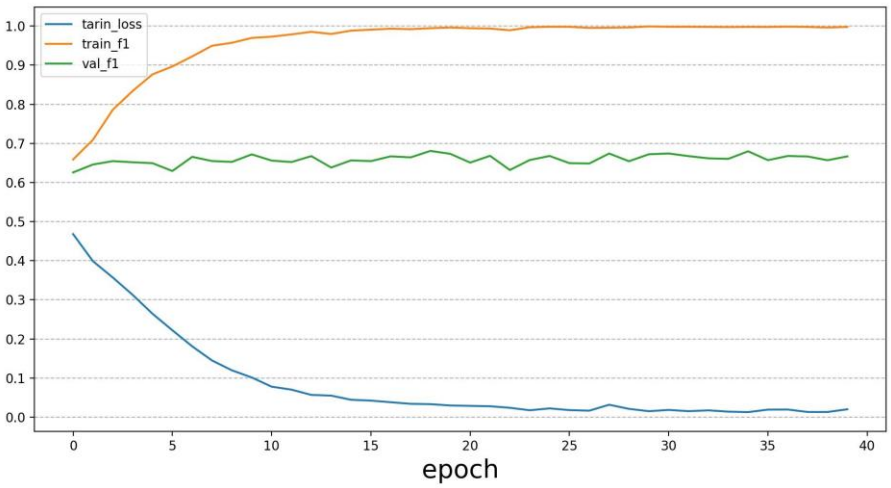


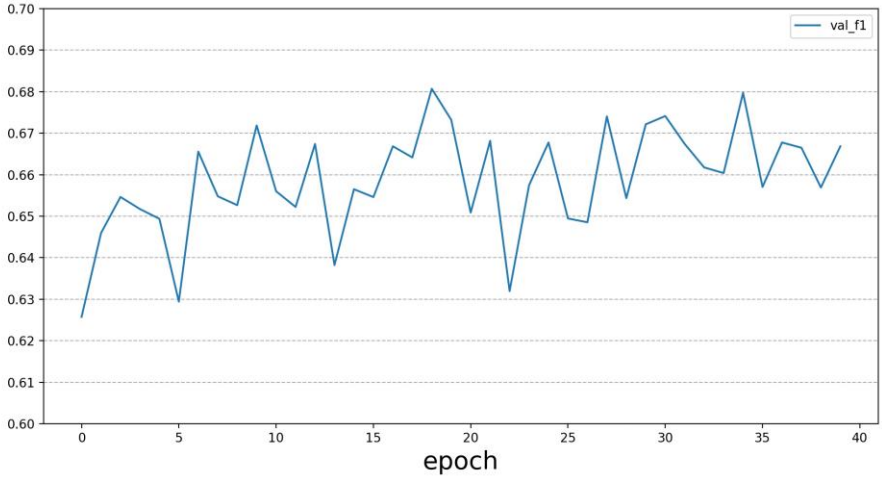


圖五:架構二 F1-Score 與 loss 圖



圖六: Roberta + Rnn + Linear Classifier





圖七: 架構三 F1-Score 與 loss 圖

6. 模型架構四: 架構三 with Attention

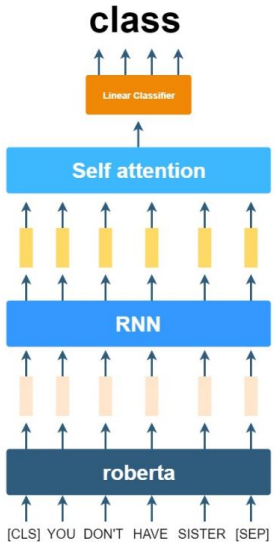
在架構三訓練時我們發現了個問題，雖說 Rnn 會取得上下文的資訊，但卻會集中於取得前後幾個字的資訊。也就是說當 Rnn 讀取到文章結尾的時候所得出的資訊會比較受到文章結尾的語句影響。再換句話說，雖然 Rnn 看過了文章開頭，但到結尾的時後卻忘了。從人類分類文章的角度來看，這顯然不是我們所期望看到的，因此我們為模型加入了 Attention 的機制，架構圖如下圖九所示。希望透過對句子做 self attention [3]我們可以找出句子中文字間相互依賴的關係，並更進一步的取得更加能夠代表整篇文章的特徵，來增加分類器的準確度。

此架構使我們的準確度來到了 0.696，模型訓練 F1-Score 與 loss 過程如圖八所示。此架構使我們確定了找出文句間相互關係與注意力的大小對於整體模型分類是有用正面幫助的。

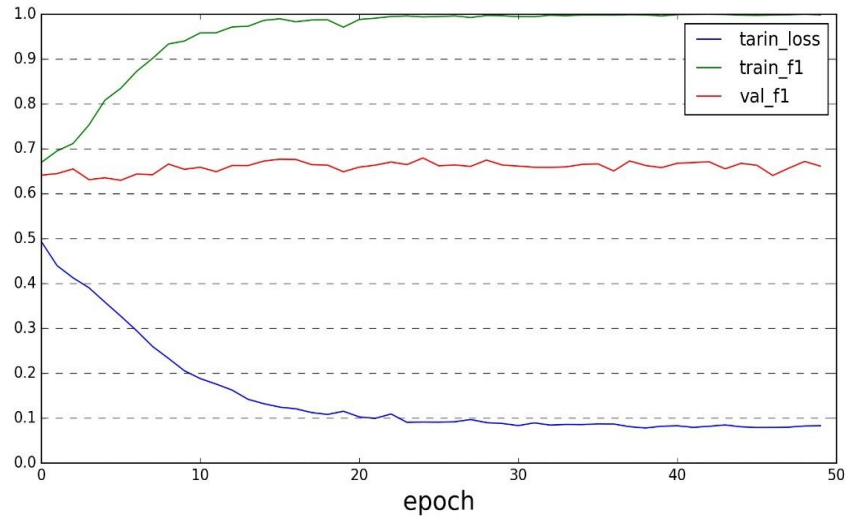
7. 模型架構五: 架構三 with Categories

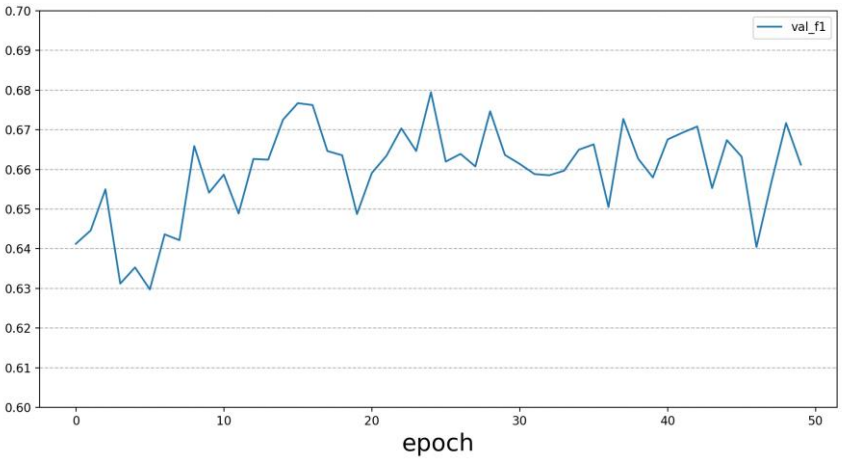
在架構三的時候，我們發現單純的增加減少 Rnn 的層數無法使準確度上升，因此我們推估可能單純從文本分析的方向下手無法找出個好的分類法，因此我們加入了 Categories 特徵進入模型當中。在對其做完 one-hot-encoding 之後將所獲得的向量與原本的 Abstract 一起丟入模型進行訓練。期望透過增加不一樣類型的特徵，因而讓分類器的效能可以再次的提升。模型架構如下圖十一所示，訓練如下圖十所示。

在新增特徵進去訓練之後，模型的準確度首次突破到了 0.7。這使我們確定了增加新特徵近來是有幫助分類問題的。

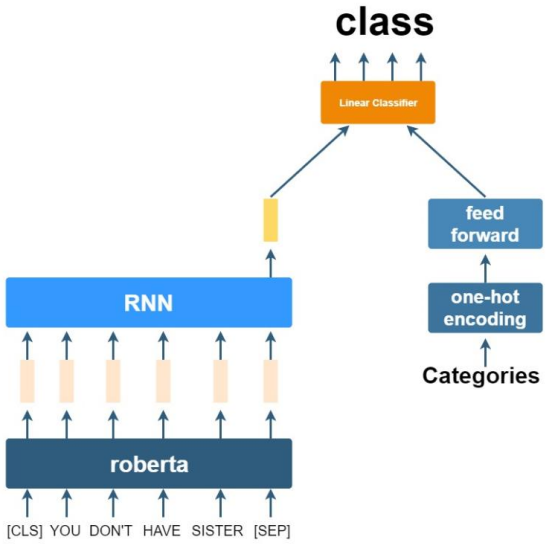


圖八:架構三 + Attention

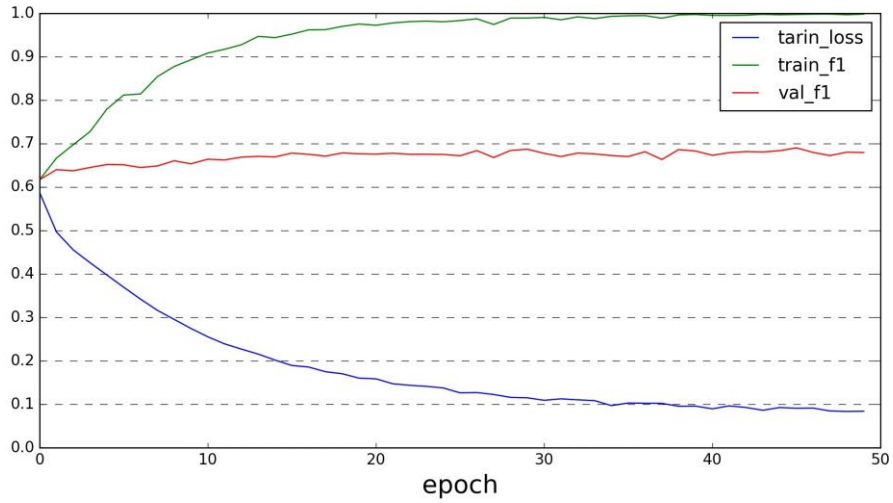


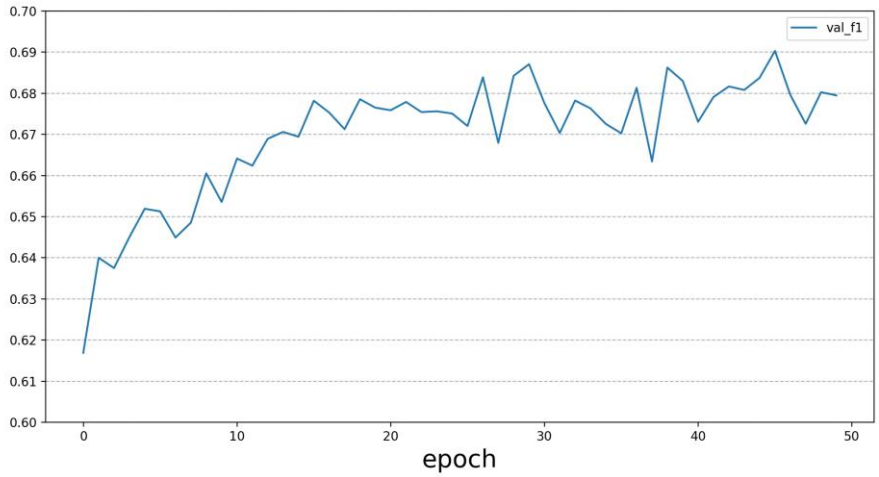


圖九: 架構四 F1-Score 與 loss 圖



圖十: 使用 Categories 架構圖



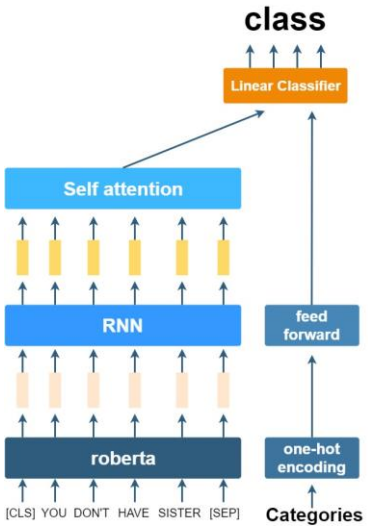


圖十一:模型五 F1-Score 與 loss 圖

8. 模型架構六: Rnn + Attention + Categories

在經過架構四與架構五的實驗之後，我們分別確定了兩者的可行性。因此我們我們覺得若能將架構四與架構五合併起來的話，整體的效能肯定可以在近一步的上升，架構如圖十二所示。

然而此架構無論我們怎麼修改參數，其所獲得的準確度皆無提升，平均下來的準確度甚至較架構四與架構五還來的低。

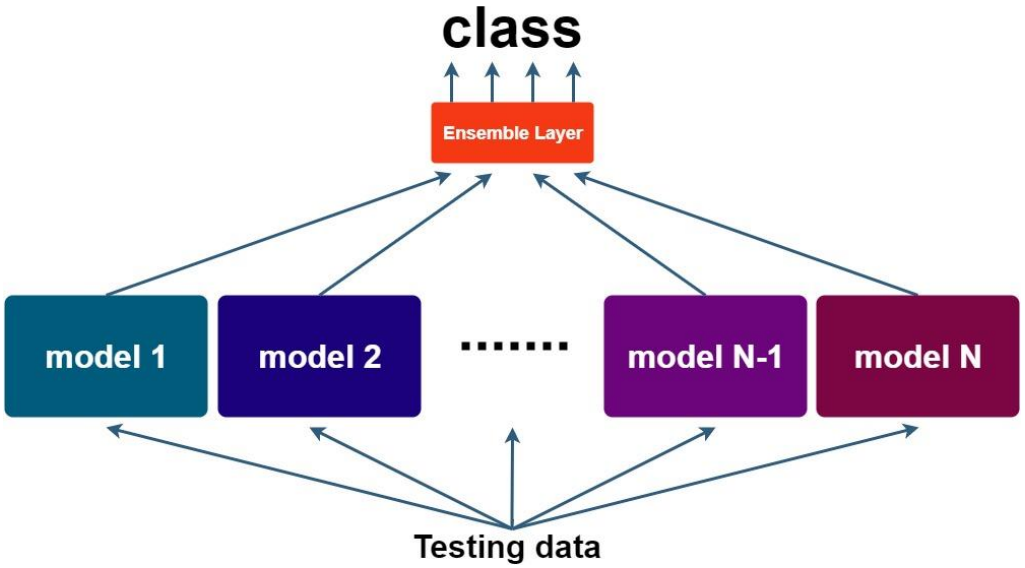


圖十二: 模型六架構圖

9. 模型架構七: Model Ensemble

最後我們選擇對以訓練好的幾個模型進行 ensemble，期望透過多個 model 的投票，使得最後的準確度能夠在現有的基礎上再一步的上升，架構如圖十三所示。

首先我們先將模型架構三到架構五分別訓練七至十個模型，並分別取其準確率前一至二名出來加以保存。隨後分別將測試資料餵入這些模型之中並將這些模型所得到的答案做加總統計，最後統計時若某類別的數值為正數，我們便將判斷屬於該類別。最後在我們的嘗試下，使用一個架構三、一個架構四、兩個架構五可以得到我們最高的準確度 0.71。



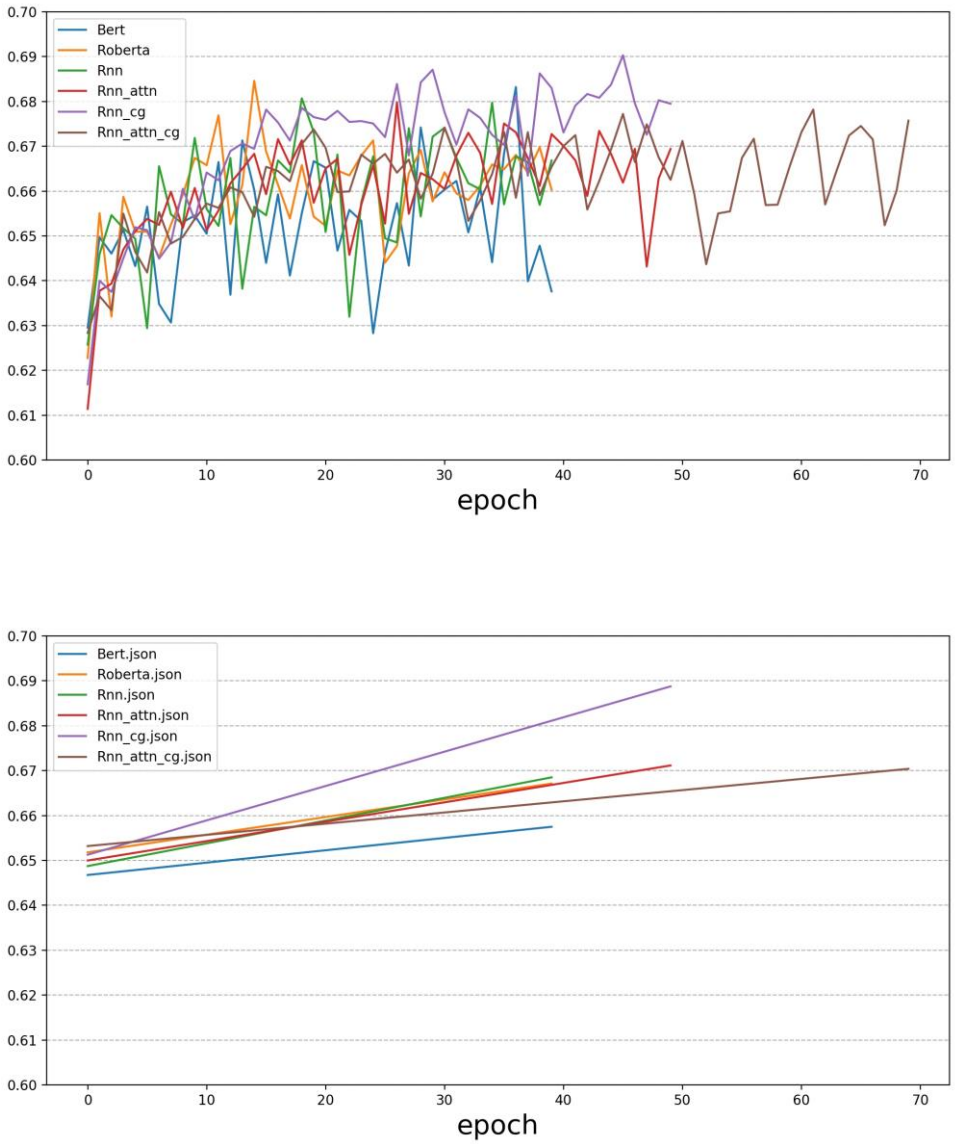
圖十三: ensemble 架構圖

四、 結果分析:

我們將上述實驗所得到的 val-f1-score 畫在一起，如下圖十四所示。我們可以發現隨著模型的演進，整體的準確度有上升的趨勢。從圖中可以很明顯的可以發現 Category 對於解決此問題有著很大的幫助。最後一點則可以發現，若使用將 Category 特徵與 Attention 機制兩者一起使用，整體的訓練準確度反而下降了許多。目前仍無法解釋此原因為何，後續研究可往此方向前進。

五、 超參數推薦:

Learning late: 0.000008~0.00002
Rnn cell : LSTM with hidden dimension 384
Attention Layer hidden dimension: 128
Classifier dimension: 128



圖十四：f1 score 總覽

六、 參考文獻

[1] “Pytorch document,” Pytorch, [線上]. Available: https://pytorch.org/docs/stable/index.html.

[2] “Transformers: State-of-the-art Natural Language Processing for TensorFlow 2.0 and PyTorch,” huggingface, [線上]. Available: https://github.com/huggingface/transformers.

[3] “attention-is-all-you-need-pytorch,” [線上]. Available: https://github.com/jadore801120/attention-is-all-you-need-pytorch.