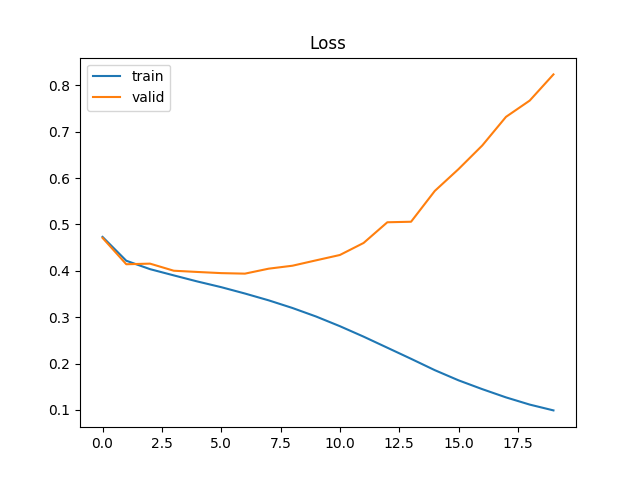
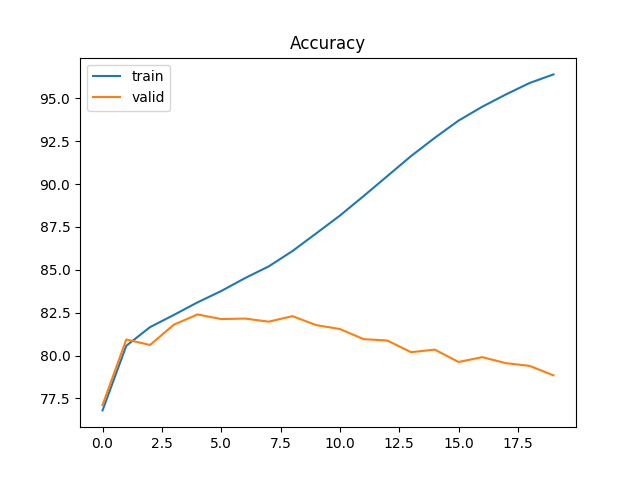
學號：r07921001 系級：電機所二 姓名：李尚倫

1. (1%) 請說明你實作的RNN的模型架構、word embedding 方法、訓練過程(learning curve)和準確率為何？ (盡量是過public strong baseline的model)

由於這次的模型實在不是很好train，暴調一波參數還是沒什麼太大的進步，勉勉強強過strong baseline，測試過的方式有：(粗體為最終選用的)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Categories | Setting | Selected | Tried | Discussion |
| Network structure | Different RNN model | **GRU** | LSTM | GRU是簡化後的RNN模型，比LSTM的參數更少，因此會相對較好收斂，實做起來的結果略好於LSTM |
| Different RNN hidden layer dimension | **200** | 250  150  100 | 一層的neuron太少的話，參數量就會少， model的涵蓋的函式範圍就會較小，但太多參數也會導致不好收斂找到極值，測試下來200剛好 (基於word vector 250的情況下) |
| Different RNN layer | **2 layers** | 1 layer  /more than 3 layers | 兩層的效果剛好，太深太淺效果都不會比較好 |
| Different fully connected layer | **Shallow**  **(1ayer)** | Deeper  (2~3layer) | 實驗下來的結果有點像是前面RNN出來後就大致底定了，深淺的FC對結果沒有太大幫助，因此選擇淺的，較易收斂 |
| Different dropout rate | **50%** | 0%or 90% | 有dropout可以避免overfitting training set, 50%剛剛好，太多無義，反而會影響收斂 |
| Optimizer | Different optimizer | **Adam with lr=0.0002** | Adadelta with lr=1 | 理論上Adadelta在某些情況下收斂的速度非常快效果很好，不過在這個case上兩者速度差不多，結果也差不多，adam略好一點 |
| Training parameters | Different batch size | **Small (16/32)** | Large (128~1024) | Batch size會影響到算gradient時是一次多少筆data一起算average的gradient，在noisy data的時候小的batch size通常有助於幫助收斂，但運算速度較慢，因為GPU算一次矩陣運算的時間不會因為矩陣的size變慢或快太多，本次作業的句子分析就屬於比較noisy的data。 |
|  | Different sentence segment | **Full** | Random sample sub-sentence | 有鑒於這次即使加了dropout還是很容易overfit training set，因此想仿照CV的做法，做data argumentation 但如果取一個size的window左右隨機滑動擷取input，很容易結到全部都是padding，效果不好，還是都固定從頭開始取，能讓padding的影響最小 |
| Word embedding | Different sentence len | **32** | 20 or 39 | 全部句子中最長的長度是39，而32幾乎涵蓋了大多數的句子長，但又不致於會補太多padding |
|  | Different padding method | **Add <PAD>** | Repeat the last word | 原本想說與其加一個無意義的<PAD>項不如試試重複字尾看看有沒有效，結果是沒用，會誤導語意，效果更差。 |
|  | Different vector dimension | **250** | 200 or 300 | 根據gensim的word2vec通常轉換成200~300  dimension之間的word vector效果會最好 |
|  | Different embedding method | **Skip-gram** | CBOW | 兩種不同的Perdition based的word vector context，經實驗sg較好 |
|  | Different window size | **5** | 3 or 7 | Window size也是適中即可，5剛好。 |
| Semi-supervied  (self-training) | Different threshold | **10/90** | 20/80  50/50 | 使用self-training 的 hard label，把根據labeled training data訓練好的模型拿來預測unlabel，取分數>0.9的label為1，<0.1的label為0。 |

總的來說流程是，使用上述架構和參數先訓練了一個based on labeled training data (190000/200000 as train, rest as valid)，挑epoch5為第一個model (acc=82.4):



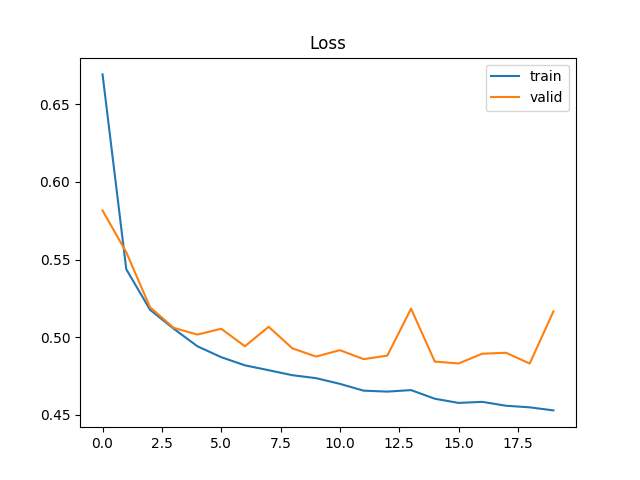
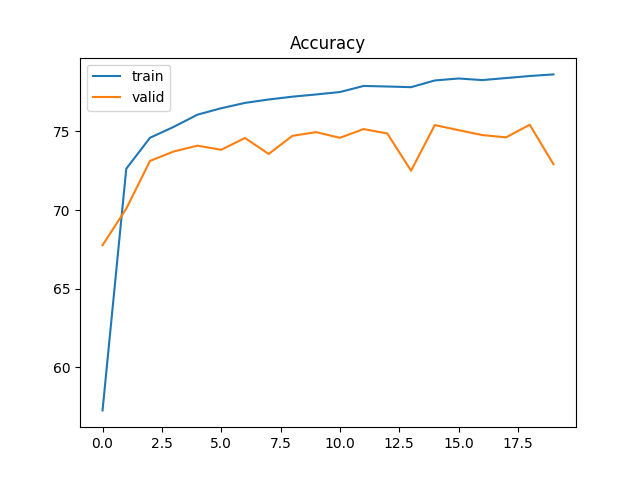
然後再根據這個model，其他條件ㄧ樣把lr從0.0002降到0.00005再train，挑epoch2為第二個model (acc=82.43) ，然後再拿第二個model來做self-learning得到93510比labeled data，併入，再基於第二個model再train，1~190000&200000~293510 as train，190000~200000 as valid，挑epoch1為第三個model (acc=82.380)，即為最後的best model acc > strong baseline。

|  |  |  |  |
| --- | --- | --- | --- |
| Model | Valid accuracy | Kaggle public acc |  |
| Model 1 (ep5) | 82.400 | 82.165 | Train on labeled  lr = 0.0002 |
| Model 2 (ep2) | 82.430 | 82.254 | Train on labeled  lr = 0.00005 |
| Model 3 (ep1)  (best model) | 82.380 | 82.321 | Train on labeled+new data lr = 0.00005 |

1. (2%) 請比較BOW+DNN與RNN兩種不同model對於"today is a good day, but it is hot"與"today is hot, but it is a good day"這兩句的分數(過softmax後的數值)，並討論造成差異的原因。

|  |  |  |  |
| --- | --- | --- | --- |
| Model | Structure & Parameters | today is a good day, but it is hot | today is hot, but it is a good day |
| RNN based best model  (Acc=82.38 on valid) | parameter total:6686601, trainable:512601 | 0.2460 | 0.8226 |
| BOW+DNN based model  (Acc=75.42 on valid) |  | 0.7088 | 0.7000 |

兩個model一個來自第一題的best model另外一個為其他參數皆不變，只抽換掉RNN的層改為三層的fully connected layer（故總共四層），在labeled training set上training 20個epoch後的結果如下:



其實還不會太差，我們選epoch 作為最終的model，然後預測上面表格的句子，由結果可知，雖然此model也有不錯的accuracy 但是因為其使用的方法是DNN，只是胡亂把句子串起來，整串丟進去做fully connected算weight，有點像是一次同時看全部然後下結論，而RNN based的卻一個一個word餵進RNN，並有memory cell彼此前後影響的，有點像是按照順序從頭到尾看過後才下結論，因此會對文字出現的順序有鑑別度。像上面的實驗就證明了這個現象，先出現good再出現but，就會被RNN判斷為negative，而先有but再有good則會是positive，而在DNN眼中，因為所有文字都是獨立的，可能dataset中有關於good的都會是positive的，讓他把good看得比but和hot重要，最終分數傾向positive，但由於but和hot的扣分，也沒有像RNN的positive分數那麼高(0.7 v.s 0.8)。不過這個實驗也說明了只抓關鍵字不看語法還是對於這個正反面語意判斷很有用的，準確度還是有一定的水準。

1. (1%) 請敘述你如何 improve performance（preprocess、embedding、架構等等），並解釋為何這些做法可以使模型進步，並列出準確率與improve前的差異。（semi supervised的部分請在下題回答）

由第一題我們給的實驗結果可以看出，這次作業主要的問題是model很容易overfitting training data，5個epoch之後valid的loss就開始變大，acc變小，因此除了第一題提到的那些調整來幫助model更易收斂外，如何減少overfit的改動其實才是對accuracy improve最多的部分 ，包含dropout和semi-supervised learning。而關於如何improve performance的做法都在第一題的第一個表格中皆有提到，而最後best model採用的都是對accuracy 有improve的方法，第一題的第二個表格和下方表格則可看出improve前後量化的差異，詳參照第一題。

|  |  |  |
| --- | --- | --- |
| Model | Valid accuracy | Kaggle public acc |
| LSTM / 1layer / 150hidden | 82.000 | 82.010 |
| GRU / 2layer / 150hidden | 82.225 | 82.022 |
| GRU / 2layer / 200hidden (model2) | 82.430 | 82.254 |

1. (2%) 請描述你的semi-supervised方法是如何標記label，並比較有無semi-supervised training對準確率的影響並試著探討原因（因為 semi-supervise learning 在 labeled training data 數量較少時，比較能夠發揮作用，所以在實作本題時，建議把有 label 的training data從 20 萬筆減少到 2 萬筆以下，在這樣的實驗設定下，比較容易觀察到semi-supervise learning所帶來的幫助）。

在這次作業中我們有使用到semi-supervised的方法來得到最終的best model，詳見第一題的描述與表格，具體的作法我們是使用self-learning 中的hard label，先訓練好一個不錯的model (model2)，用這個model來幫unlabeled data標記，分數超過0.9或低於0.1則直接label為1或0，然後再把這些新的labeled data併入training data，基於之前那個還不錯的model (model2) 的參數上再下去訓練，雖然因為hard label導致distribution更兩極化和model之前看過的資料不同，因此valid loss會上升，valid acc也會下降， 但加入新的data能有效解決overfit於training data的問題，因此我們可以在kaggle public上看到acc有明顯的improve。

而若是在data更懸殊的情況，結果如下：(batch size=1024, 其他同第一題)

|  |  |  |  |
| --- | --- | --- | --- |
|  | Training data | Valid data | Valid accuracy |
| 1. Model train  w/o self-learn data | 0~20000  labeled data | 190000~200000  dabeled data | 76.084 |
| 2. Model train  w/ self-learn data | 0~20000  labeled data +  200000~283408 new labeled data | 190000~200000  dabeled data | 67.197 |
| 3. Model train  w/ self-learn data  based on 1’s pretrain | 0~20000  labeled data +  200000~283408 new labeled data | 190000~200000  dabeled data | 69.648 |

Training Data的增幅達500% (20000 -> 103408)，但self-training 的方法在validation上還是沒有很理想，可能是data亮的差距還不夠大，也可能是這樣的方法在這個問題上效果不好，可能要採取其他clustering的方式來幫unlabeled data做更好的分類和labeling。

|  |  |  |
| --- | --- | --- |
| 1. Model train  w/o self-learn data |  |  |
| 2. Model train  w/ self-learn data |  |  |
| 3. Model train  w/ self-learn data  based on 1’s pretrain |  |  |