Machine Learning Final Project - conversations in TV shows

NTU_b05902031_命理師協會旗艦店 b05902031 謝議霆　b05902008 王行健　b05901033 莊永松

Introduction & Motivation : 1%

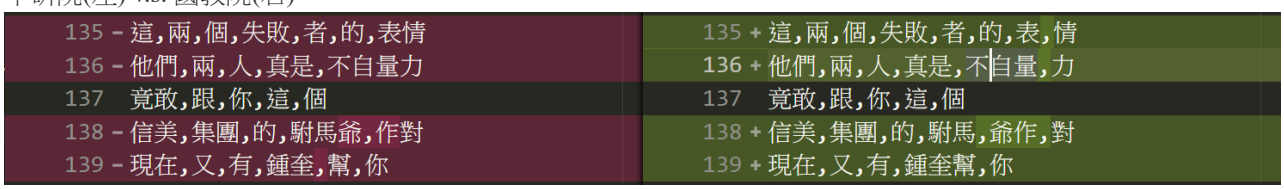
本次主題為電視劇對話中的台詞，給出上文(可能1句~4句不等)，在六個選項中找出正確的下文，其中上文可能是同一個人的台詞，也可能是對話。其實這個題目就是常見的"多輪對話"問題，不同於"QA問題"是單次問答，這種題目需要考慮到前後文的關係，所以更加困難。這種問題的model也是目前在generative language model 還表現不好的狀況之下，可以用大量語料庫搜索適合的response，來實現聊天機器人效果的好方法，是目前較為實際且能夠直接應用的方法。

Data Preprocessing/Feature Engineering : 2%

(1)分詞:

- **[jieba]**
  效果較差，很多詞都分錯，錯誤率高，約50%句子會跟中研院分的不一樣。
  1. 分詞長度偏大，有時候會分成很怪的詞組，如"天就亮,了"，"誰,要,妳愛,上無藥,可救"。 2. 會把人名亂切開，如"林,明德"，"鍾,世民"。 (在presentation時，聽到某一組使用jieba，實驗發現**跟不分詞直接一字一字斷開結果差不多**，可能就是jieba分詞不夠好的緣故，畢竟是針對大陸簡體語料所做的套件)
- **[國教院中文分詞系統 NAER]** https://github.com/naernlp/Segmentor
  (採純統計式模型，執行速度快)
  比jieba好很多，分詞較細，且人名不會被切開(可能有特別針對NER做處理)，但還是有少部分分錯的時候，有時是一個詞包到前後錯字，有些是分得太細，如："有,樣,學,樣"。
- **[中研院中文斷詞系統 CKIP]** http://ckipsvr.iis.sinica.edu.tw/
  (採經驗法則模型，執行速度較慢)
  分詞結果九成都跟國教院一樣(都正確的)，但少部分國教院分錯的，中研院也能分對，效果最好，見下面例子。

  比較 **example**
  中研院(左) v.s. 國教院(右)



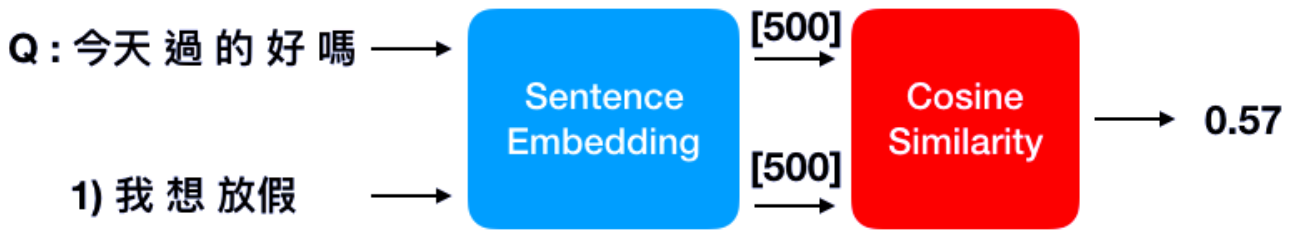
  可發現國教院把"不自量力"分成"不","自量力"；"駙馬爺"分成"駙馬","爺"；"鍾奎,幫,你"分成"鍾奎幫,你"。

(2)Word2vec:

- 使用Gensim，embedding size一開始使用1200(MLDS做seq2seq的經驗，維度盡量開大)，針對直接算sent2vec的相似度的架構來說(Model 1)，size比1200大和比1200小，效果都較差(數據見Experiment and Discussion)，1200算是最好的size。
- 但後來才發現，在針對RNN的架構時(Model 3、Model 4)，似乎不適合開那麼大，反而100~200就能train得不錯，越大的維度反而訓練不起來，最後我們表現最好的model是使用100維作為embedding size。
- Word2vec的方式，skip-gram跟CBOW都嘗試過，發現skip-gram效果較好，CBOW稍微差一些。
- 其餘參數：
  - window = 7
  - min_count = 10
  - iter = 100

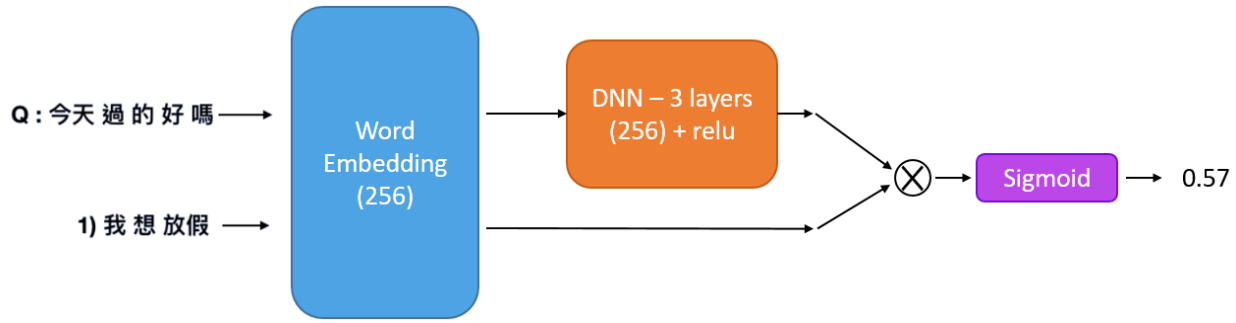
Model Description (At least two different models) : 4%

1. [Model 1] Sentence embedding -> cosine similarity --- 最簡單的model



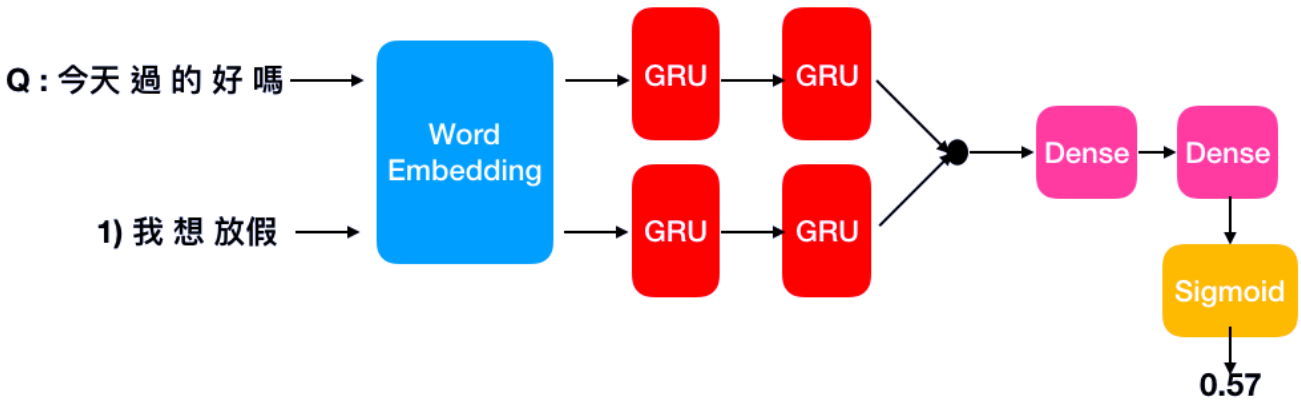
- **Sentence embedding**：將一句話斷詞之後，每個詞在訓練好的word2vec模型中會有一個向量，將這些詞向量做 $weight(w) = \frac{a}{a+p(w)}$ 的加權平均直接就當成我們的句子向量。(a是一個常數，$p(w)$是該詞在訓練資料中出現的機率) 見[參考paper][1]
- **Cosine similarity**：判斷是否為該問題的答案是看兩個句子向量的相似程度，這裡是用兩個高維向量的cosine similarity當分數，選擇分數最高的選項當作答案。
- **Model 參數細節**：$a = 6 \times 10^{-4}$，embedding size=1200
- **Issue**：
  1. 因為我們是用加權平均來生成句子的向量，所以這個模型中缺少句子中詞與詞的前後關係。
  2. 其次，由於這個方法的問句跟答句所用的sent2vec方式是一樣的，所以沒有考慮到需要從問句轉換為答句的問題，導致只是選出跟問句比較像的句子。

2. *[Model 2] DNN model --- 第二簡單的model*



- **Word Embedding** : 將每個詞以訓練好的word2vec模型轉成向量
- **Training data**：我們會從training data中挑選真的是有前後關係的兩句話、和隨機挑選的兩句話，分別標上1(True)和0(False)的label，網路在看到兩句話後會給定一個分數(0~1之間)，在做testing的時候，選擇六個選項中分數最高的那一個當答案。
- **DNN model**：為了減少訓練時間，我們在開始使用RNN來訓練前，先嘗試直接用DNN的方法，也就是先用上面方法一(詞向量平均)做sent2vec後，把問句接DNN，希望透過DNN，能把句子transform成答句，然後直接跟答句的sent2vec比較cosine similarity。
- **Model 參數細節** ：
  - embedding size = 256
  - DNN: 3層dense，output dim=256
  - activation function: relu
- **Issue** : 此作法可以避免 Model 1 中沒有考慮到從問句轉換為答句的問題，但缺點是sent2vec的方法仍然缺少句子中詞與詞的前後關係。

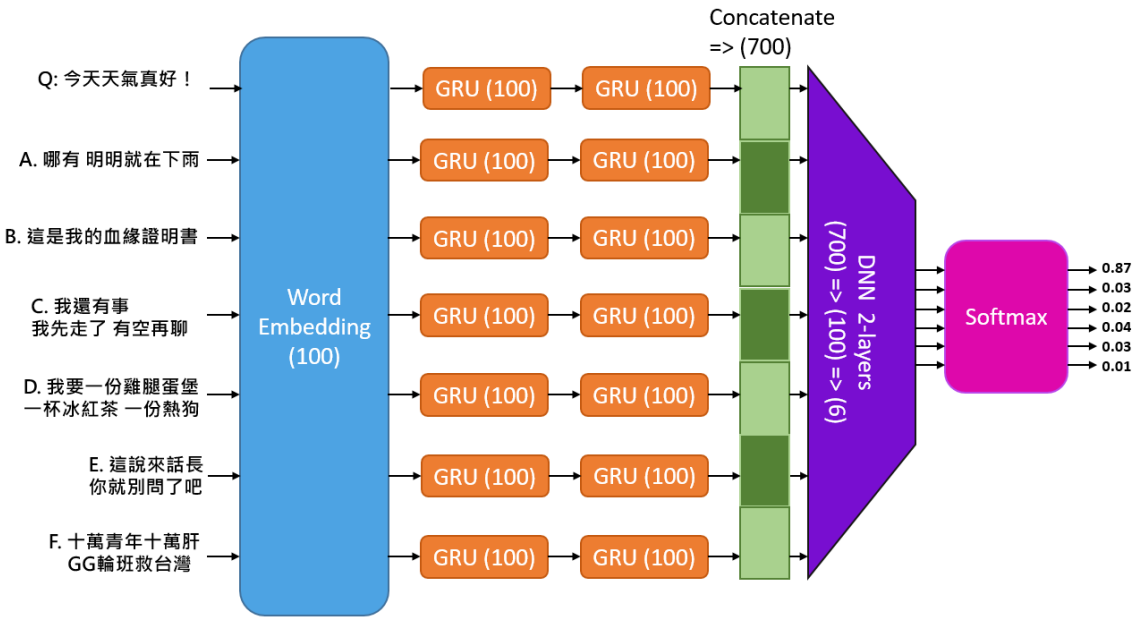
3. *[Model 3] RNN model --- 正式model*



- **Word Embedding** : 將每個詞以訓練好的word2vec模型轉成向量
- **Training data**：我們會從training data中挑選真的是有前後關係的兩句話、和隨機挑選的兩句話，分別標上1(True)和0(False)的label，網路再看到兩句話後會給定一個分數(0~1之間)，在做testing的時候，選擇六個選項中分數最高的那一個當答案。

- **RNN model**：input的句子把每個字embedding完後，接上RNN(GRU)，問句和答句所接的RNN是不同的RNN(沒有share weight)，問句RNN的output再接三層DNN讓他有某些transoform，輸出與答句的RNN的output作cosine similarity，得到答案(0~1之間的機率分布)。
- **model 參數細節**：
  - embedding size:100
  - 第一層GRU:128 cells
  - 第二層GRU:64 cells
  - 第一層Dense(64->32)
  - 第二層Dense(32->1)
- **Issue**：此model架構有考慮句子中詞與詞的前後關係，理論上已經避免了剛剛其他前兩個架構所擁有的問題。但追根究柢，這種給兩句input算單一output的方法還是可能導致一些問題：
  - **Training與Testing的差異**：
    首先，Training跟Testing的方式是稍微不一樣的，Training時是單純指定1 or 0的label，並且希望True case的output是1，False case的output是0；而在Testing時，其實只需要**選6句中機率最高的**，並不需要每一句True case的target都是1。**=>只需要optimize相對分數，而非絕對分數。**
  - **用絕對分數來Train髒data易混淆網絡**：
    如果訓練資料乾淨，針對絕對分數來訓練的問題不大。但是畢竟這次的訓練資料很髒，而且False case也是隨機產生，因此訓練資料中，可能還是有很多True case但實際上語意不相關(*可能恰好截到對話結束的地方*)，或是False case的答句是還可接受的(*可能問句是比較開放，可接受多種答句*)諸多狀況。但我們這樣一視同仁地希望True case的output是1，False case的output是0，於是就容易對要訓練的網絡產生混淆，可能害他在training的時候就已經無法對這些data做很好的判斷。其實我們所需要的，應該只是讓True case的成績大於False case的成績就好了
  - **解法：使用多output+softmax，當你每次訓練都是拿6句話的相對分數來比較，就能避免單一筆data品質不佳的狀況**
  - **作法：End-to-end model**。

4. [Model 4] End-to-end RNN model --- 最強model



- **model架構**：input為一個問句+六個答句，embedding後經過兩個GRU，輸出7個100維的vector，直接concatenate成為700維的vector，最後接上兩層Dense，轉為6維後經過Softmax，得到預測結果。

- **Improvement**：針對前述RNN架構的缺點，我們修改為End-to-end的RNN model，分成6個答句選項的output做Softmax，而非原本需要讓True case的output越大越好

  > **single-output Sigmoid v.s. multi-output Softmax** 原先RNN model用單一output+Sigmoid，而Sigmoid是將$[-\infty, +\infty]$都映射到$[0, 1]$之間，而且是output接近無限大的時候，Sigmoid出來才是1，結果會讓他變成output越大越好。 反之Softmax是看各個output之間的比例，如果其中一個output明顯大於其他，則他Softmax出來的value就會很接近1，不需要train到讓他output趨近無限大。

- **Training data量與Testing data量的平衡**： 在這種架構中，我們產生訓練資料的方式，也是挑選連續句子作為True case並label 1，然後隨機亂數挑選5句其他句子作為False case並label 0。這樣除了能夠改以相對分數訓練外。另外也能讓Training data中False case的data數量是True case的五倍，也就是與Testing data完全一致。在前面的單純RNN架構中，我們並沒有把True:False case的比例弄成1:5，也因此可能他所訓練到的False case不夠多，導致在testing時效果不如預期。
- **training data處理**：
  - 最長句子長度 = 30
  - 將句子補齊的padding加在最前面
- **model參數細節**：
  - embedding size: 100
  - GRU output dim: 100
  - Dense 1: 700 => 100
  - Dense 2: 100 => 6
  - activation function: Swish

Experiment and Discussion : 6%

1. [Model 1] Sentence embedding -> cosine similarity

- 不同的**embedding維度:** 可以看出維度大的在使用cosine similarity後表現比較好，比較能判斷句子的相近程度。

embedding維度	**private score**	**public score**
100	0.33122	0.30039
500	0.36758	0.36916
1200	0.40869	0.40513

- 不同的a值(embedding維度:1200)：常數在這裡可以看成一個詞的機率的調整，越小的話，詞出現的機率就越能影響結果。而在實驗的過程後，發現6e-4能達到最好的performane。

a	**private score**	**public score**
6e-2	0.35098	0.34624
6e-3	0.36719	0.36640
6e-4	0.40869	0.40513

- 不同的**training data**
  - 將training data改成以下的形式，除了增加training data的數量外，在Question和answer的task中，這樣的training data還可以加強前一句和後一句的關係。

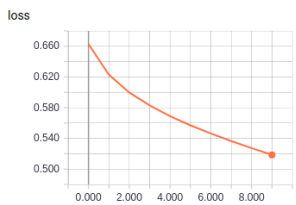
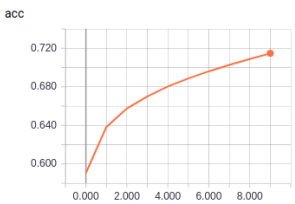
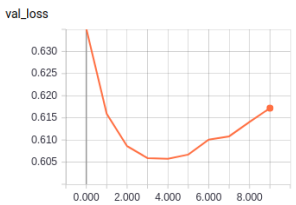
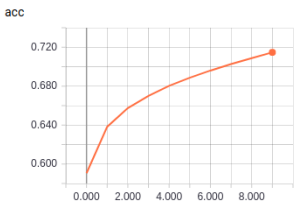
```
今天 過 得 好 嗎 -> 今天 過 得 好 嗎 我 很 好
我 很 好       -> 我 很 好 真的 嗎
真的 嗎        -> 真的 嗎
```

	private score	**public score**
before	0.39367	0.38814
after	0.40869	0.40513

- 結果討論：此model雖然準確度不比其他的高，但就其model大小及付出的運算量來說，已經是個CP值很高的做法。

2. [Model 2] DNN model

- 訓練參數 :
  - optimizer: Adam
  - learning rate=1e-4
  - epoch = 10
  - batch size: 256
  - loss function: cross entropy
  - checkpoint: on val_acc
- 訓練過程 :

Training loss	**Training accuracy**	**Validation loss**	**Validation accuracy**
			

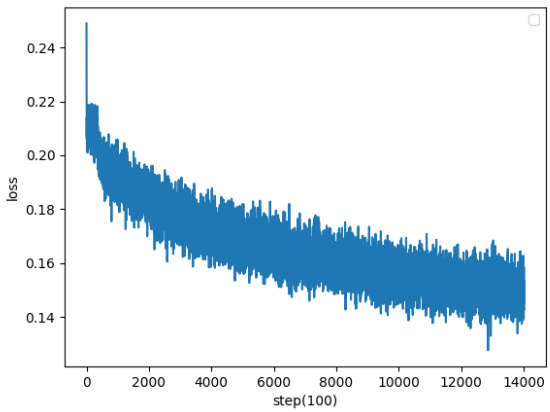
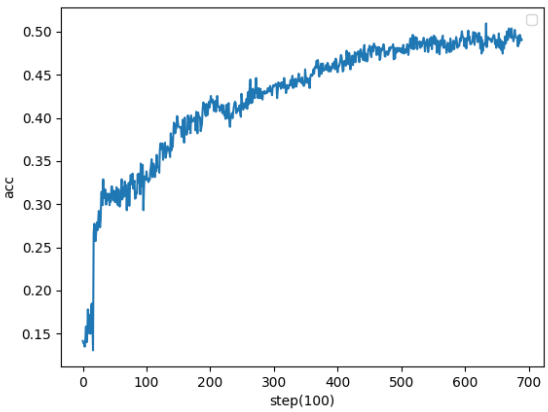
- **kaggle score**

public score	**private score**
0.40513	0.40869

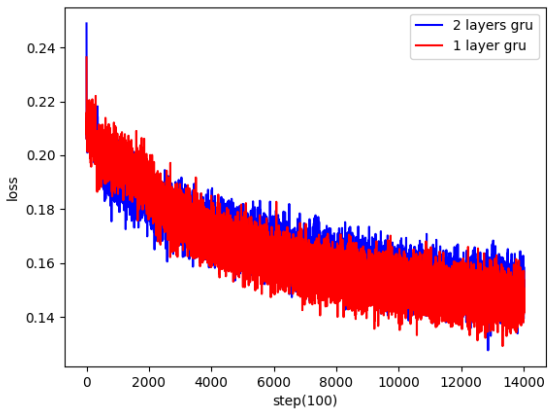
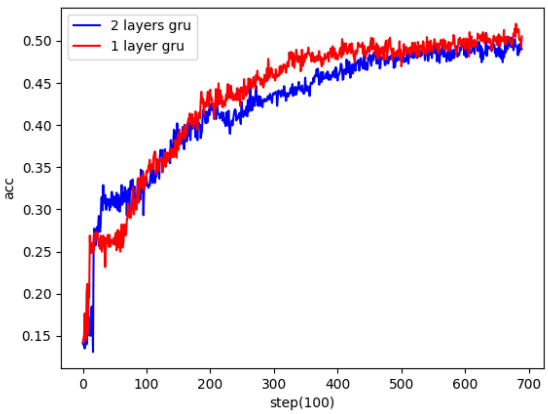
- 結果討論： train了半天，沒有比Model 1好多少，效果頗差，可見sent2vec直接接DNN並不是個好方法，畢竟沒有考慮字序的關係。

3. [Model 3] RNN model

- 訓練參數 :
  - optimizer: Adam
  - learning rate:1e-4 (Adam)
  - batch size:32
  - loss function: MSE
  - epoch:30
  - checkpoint: on val_acc
- 訓練過程 :

MSE loss	Validation accuracy
	

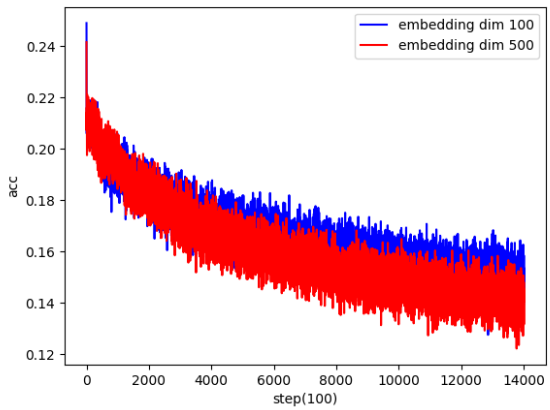
- 二層GRU與一層GRU的比較：
  - 由下圖可以發現，在計算loss的時候，兩種model的MSE loss差不多低，可是在validation的準確率上，只有一層GRU layer的表現是比較好的。
  - 從Kaggle上的成績來看，平均起來一層GRU的準確率稍微高一點。

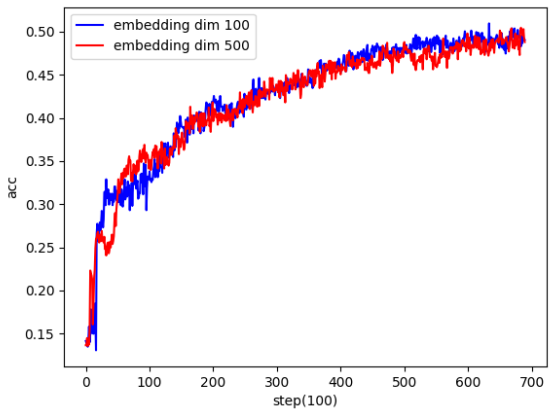
MSE loss	Validation accuracy
	

	private score	public score
1 GRU layer	0.48142	0.47470
2 GRU layer	0.46996	0.47588

- 不同**embedding size**之間的比較：
  - 由下圖可以發現embedding維度較大的在訓練過程中loss比較低，但是在準確率上是差不多的，為了訓練時間的考量，會選擇維度較低的方法來訓練。而在embedding維度1000以上的時候，訓練到一半準確率就會卡在一個很低的值，就再也升不上去了。
  - 從Kaggle上的成績來看，兩個model其實是差不多的。

MSE loss	**Validation accuracy**





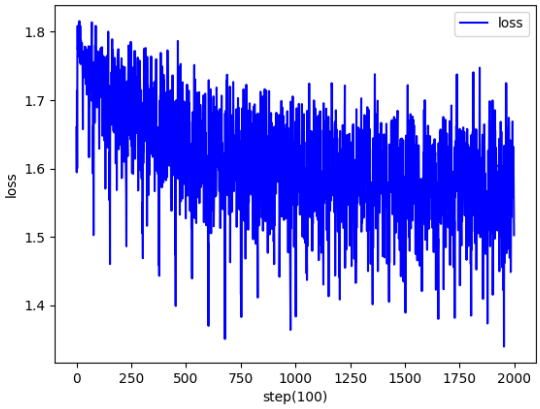
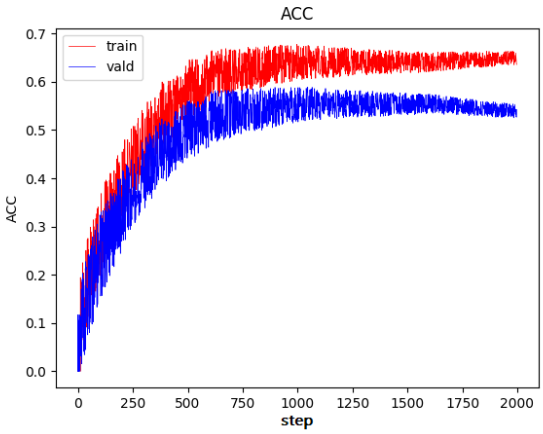
	private score	**public score**
embedding維度:500	0.45968	0.47312
embedding維度:100	0.46996	0.47588

- **結果討論**：RNN model的成績，明顯比前兩model提升了將近8%左右，可見有將字序考慮進去是非常重要的環節。

4. [Model 4] End-to-end RNN model

- 訓練參數：
  - optimizer: Adam
  - learning rate=1e-4
  - batch size: 100
  - epoch = 20
  - loss function: cross entropy
  - checkpoint: on val_acc

- 訓練過程 :

CrossEntropy loss	Validation accuracy
	

- 訓練結果 :

train acc	validation acc
0.62394	0.58169

- **kaggle score**

public score	private score
0.54426	0.55415

- **結果討論**：此model架構最為龐大，但效果也最佳，又比單純RNN model進步了7%~8%的正確率，推測他是藉由針對相對分數訓練的優勢，才能有如此顯著的進步，能有效避免單一RNN model訓練絕對分數的弊病。

Ensemble

這次final因為時間緣故，Model 4是在kaggle deadline後，聽完別組分享後才做出來的。因此在kaggle上最高分的成績是由Model 1 + Model 2 + Model 3 ensemble而來。最高分成績如下：

public score	private score
0.53913	0.54664

ensemble之比例為：(2個Model 1預測平均 + 4個Model 2預測平均 + 3個Model 3預測平均)/3

在ensemble時我們也發現，一直拿同一model ensemble的效果並不好，頂多提升個1%~2%，而使用不同做法的多個model做ensemble效果就比較好，即便其中有些model的表現並不好，像是我們原本最佳的Model 3也只有0.48附近的準確率，但在ensemble不同參數的model以及加入許多Model 1、Model 2的預測機率，一起投票後，就有效提升了近5%的正確率。

Conclusion : 1%

在這次final中，我們前後嘗試了4種model：從一開始不考慮詞序及問句答句差異的Model 1，進入考慮問句答句差異的Model 2，後來到考慮詞序的Model 3，最後是修正Model 3而採用相對分數訓練的Model 4。由40%左右的正確率，最後提升到約55%，我們了解到**1.考慮詞序**及**2.問句答句差異**以及**3.以相對分數訓練，避免部分data品質不佳**是後面model能贏過前面model的原因，也是這次題目的關鍵。

另外，還有一些因素也許是我們未來可以努力的目標，在ACL 2017 的 paper --- *Sequential Matching Network: A New Architecture for Multi-turn Response Selection in Retrieval-based Chatbots* 中，還引入了**多句問句分別與答句匹配**以及**利用conv層及pooling層做feature extraction**，有特別著眼在"**多輪**"對話的特質，因為多輪對話中，有很多句上文(這次final也是，以\t分隔多句上文)，但答句可能只取決於其中一句上文，因此需要好的feature extraction方法來挑出決定性的上文(詳見reference之paper大意)。這也是目前多輪對話題目較好的解法之一，這次final原本想使用該架構，但後來因model過大，一直會crush掉，最後只好半途而廢，但這個課題非常值得日後研究。

Reference : 1%

- [參考paper][1] *A Simple but Tough-to-Beat Baseline for Sentence Embeddings* [ICLR --- 2017 conference paper] (https://openreview.net/forum?id=SyK00v5xx)

  > paper大意：一個sentence to vector的簡單方法，跟word averaging有點像，但是針對每個字的出現機率做加權，其中$weight(w) = \frac{a}{a+p(w)}$，$a$在$[10^{-3}, 10^{-4}]$之間，最後再把所有sent vec做PCA，並且減去first conponent (類似把所有句子中相同的部分減去，也就是重複出現的詞，以凸顯句子之間的差異)。

- [參考paper][2] *Sequential Matching Network: A New Architecture for Multi-turn Response Selection in Retrieval-based Chatbots* [ACL --- 2017 paper] (https://arxiv.org/abs/1612.01627)

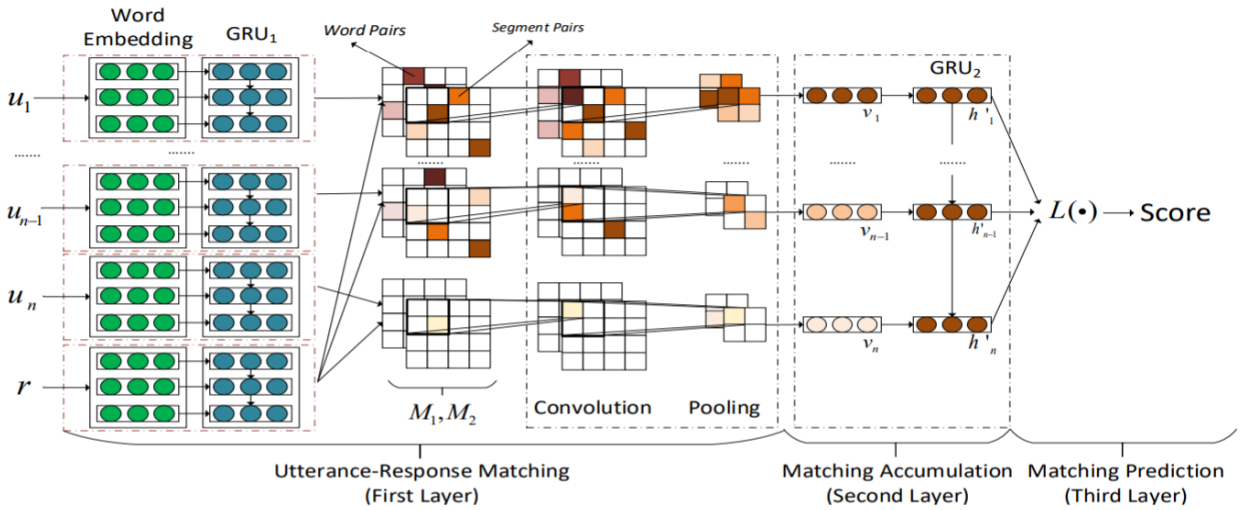


Figure 1: Architecture of SMN

> paper大意：將每一上文$(u_1, \ldots, u_n)$分別在**1.剛未進入GRU前**以及**2.已通過GRU後**，做sequence之間的matching，分別得到兩個matching的矩陣，將此二矩陣透過convolution以及pooling做有效的feature extraction，轉為向量後，再通過GRU做冗餘資訊的去除或有用資訊的保留，最後得到prediction。