Machine Learning final project

Chinese TV Dialogue Multiple-choice Probelm

隊伍：NTU_b05611033_耕耘戰車開起來

杜杰翰 B05611033	張育堂 B05611038	林宏陽 B04611003	張邵瑀 R06921081
主要貢獻： ensemble model training 協助 ranking 協助	**主要貢獻：** ensemble model training 主要 model tuning 協助 ranking 協助	**主要貢獻：** 報告內容整理、匯編	**主要貢獻：** Simple base line 主要 Strong base line 主要 ranking 主要

- # Introduction & Motivation :

在過去幾千年裡，語言在人類生活中扮演著不可或缺的角色。語言扮演著人與人之間的橋樑，而世界也在多變的地形下產生了各種不同的語言，進而發展出各種不同的文化。在持續發展人類未來的同時，借鏡於過去的經驗總是能使我們在進步中走得更穩。

在今日科技爆發的時代，機械及電腦技術的卓越使人們在工作及研究上更有效率，而使用電腦對語言做學習與分析是再好不過的方式了。在未來的科技發展上，語言分析將被更多的應用在日常生活中，而在面對大量的文字資料時也能進行更佳化的分類和預測。在這次的實驗中，我們讓電腦分析五個不同的劇本，從中學習基礎的對話能力，再將訓練好的結果用於判斷一段句子後的下一個回應可能為何。

問題描述：

給予一句話，並從接下來的6個選項內選擇正確的句子作為回應。

訓練資料為五份劇本，並將每句話以標點符號斷開，除這五份資料以外不得使用其他資料。

- # Data Preprocessing/Feature Engineering

1. Jieba

為目前最廣為被使用的中文分詞，例如將：“我来到北京清华大学”分成“我/ 来到/ 北京/ 清华/ 清华大学/ 华大/ 大学”
目的是因為在拼音文字中，單詞大多都由空白斷字，而在全音素文字文字中則沒有此種分隔標示，導致若干詞序沒有經過人類判斷很難辨別，抑或是就算經過人腦也抑難判讀，如經典的：下雨天留客天天留我不留，等等，所以在分詞方面我們採用此一套件。

2. Genism - Word2Vec

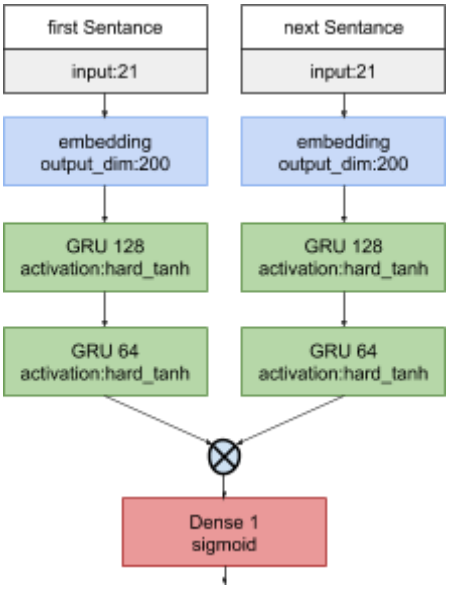
為python套件中，用於從文字中有效地提取語意，而其中的Word2Vec能透過機器學習的方式，將文字轉化為具有語意相關性的高維向量。我們將分詞處理完的train data經過Word2Vec做Embedding，透過調整各項參數以得到效果較佳的輸出向量。

3. train data

分別1-5個別的train data file中的N個句子先放入第1句話並去掉最後一句，而將其複製後第1句去除，這樣就有N-1筆train data，此為true label的部份，而false label則是在後面接續的那筆資料做np.roll製造出上下句不為連續的句子，接著做supervise learning。不過一開始的時候效果很不好，並且想到一般來說選擇題中錯誤的label本來就比較多，接著我們嘗試增加錯誤label的數量，所以我們就將原本1:1調整到1:5與這次題目6個選項的比例相當，得出單個model在kaggle score上可達到0.49369的分數。

- # Model Description:

model 1:



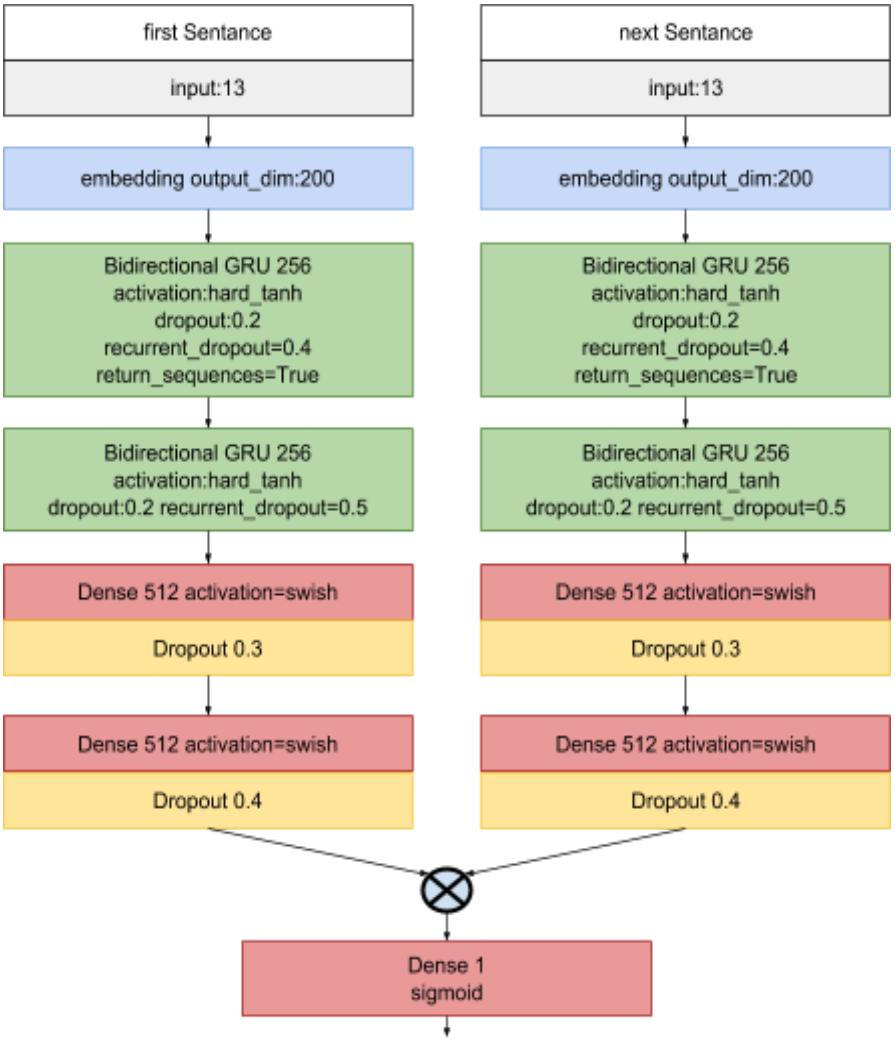
tokenize:

(1)斷詞處理：
以單一個中文字為單位直接截斷，沒有對應中文詞性做分段。

(2)Embedding：
word2vector中設定window=7, min_count=2, sg=1，取得約

3800個的token，將每個詞轉為200維向量

model 2:



tokenize:

(1)斷詞處理：
以Jieba做分詞處理，將中文字詞根據詞性作分段。

(2)Embedding：
word2vector中設定window=5, min_count=1, sg=1，取得約66400個左右的token，將每個詞轉為200維向量

以上model 1與model 2除了架構不同，在input與output層面的處裡都一樣。首先input面，由兩個tokenized的句子，一個為上句，另一個為下句，接著是各自的embedding layer，然後是output。在output之前用Dense層把最後兩個tensor做完內積的結果做sigmiod若是1則為此兩句為上下句關係，若為0則不是，以分數最高的為正確答案。

model 3 PCA:

tokenize:

(1)資料分類：
由1_train.txt讀進的資料給予tag 1，2_train.txt讀進的資料給予tag 2，以此類推。

(2)斷詞處理：

以Jieba做分詞處理，將中文字詞根據詞性作分段。

(3)Embedding：

將分好的詞轉為300維的向量，接著做padding把每個句子做好轉換。

(4)PCA:

在PCA中降到250維，再用kmeans分成5類，接著在predict的時候則是把每一句答案對問題做分類的比對，若是同一類則比相似度score，若非同一類直接給0，接著在對六個選項的score做篩選選出最相似的選項。

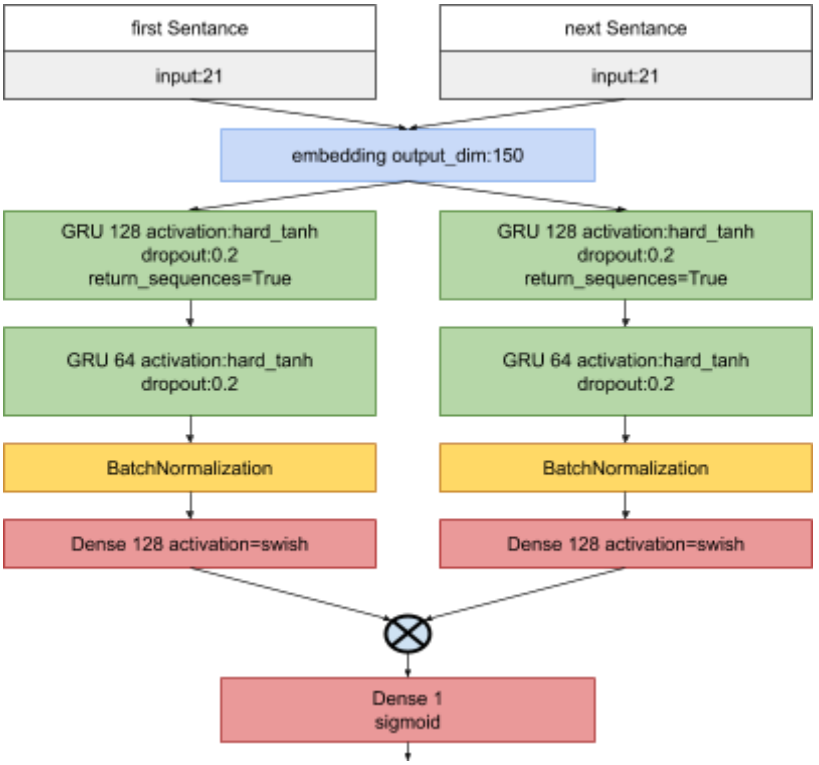
由前面幾個model來看這樣做似乎是一個合理的作法，因為主要是取出上下句在空間中的向量相似程度，但其結果並不盡理想，嘗試多種取法後效果仍然不夠理想，並且這種方法所需要的CPU資源非常巨大，所以並沒有以這個做法為主。

model 4:※為較後期發展之model固以補充方式補上

Without jieba,
Embedding 100維向量

model 5:

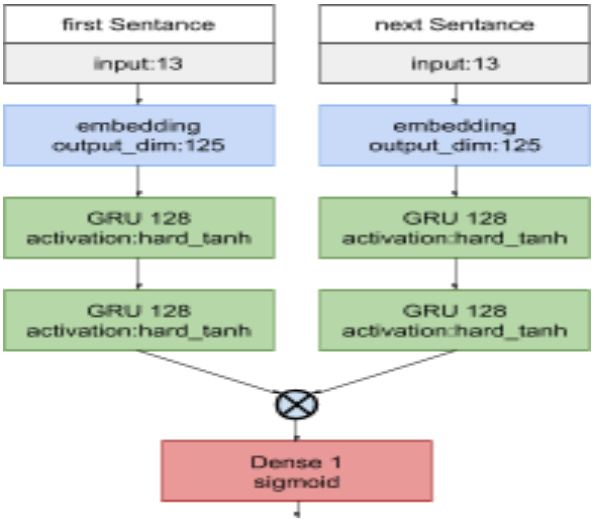
Without jieba,
Embedding 150維向量



model 6:

With jieba,

Embedding 150維向量



model 效能比較:

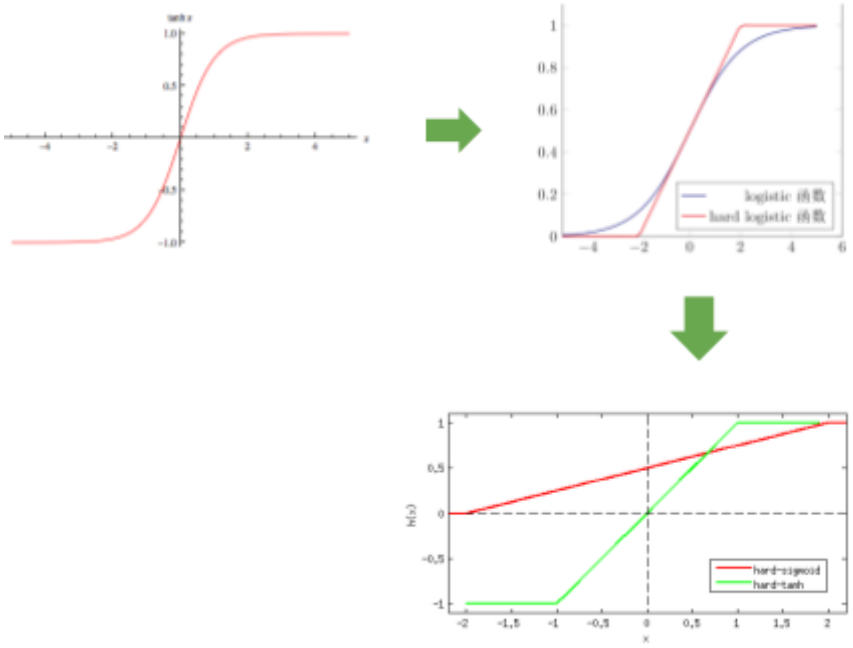
	model 1	model 2	model 3	model 4	model 5	model 6
kaggle score	0.465~0.479	0.490~0.501	0.227~0.320	0.490~0.508	0.513~0.519	0.306~0.311
one epoch	250 sec	1600 sec	780 sec	780 sec	230 sec	200 sec

備註:

customized activate function:

- hard_tanh:

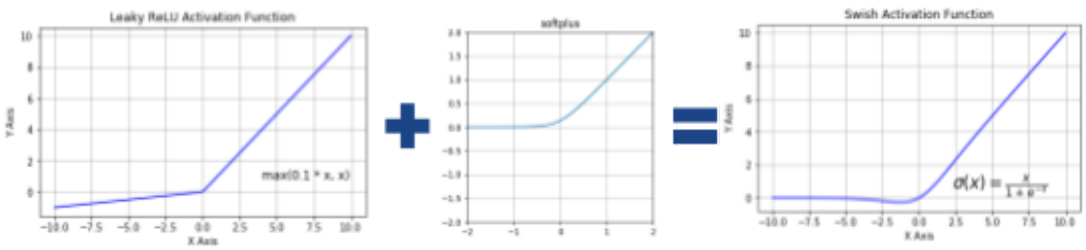
  此問題要分成兩類，那就用分割更明確的hard_sigmoid將RNN層的keras GRU預設的tanh activation取代，但在這個問題上面使用hard_sigmoid會做不起來，所以我把它改成hard_tanh



  - 在keras 中實作hard_tanh :
    hard_tanh= keras.backend.hard_sigmoid(x)*2 - 1

- swish:

  因為之前使用過leakyRelu跟softplus後想要找看看有沒有結合兩者優點的activation



- 在keras 中實作Swish :
  Swish = keras.backend.sigmoid(x)*x

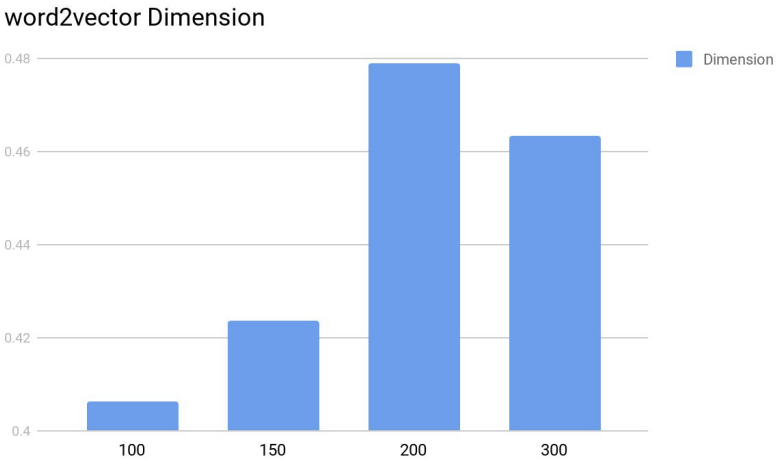
- # Experiment and Discussion :

dimension of word2vector:

Dimension:

首先在做任何需要使用到word2vector的work時，dimension是一個非常重要的參數需要調整，我們做了幾種嘗試(此處以model 1做實驗)。

dimension	100	150	200	300
score	0.40632	0.42371	0.47905	0.46324



然而在維度的挑選上，是引用之前HW5中大家普遍使用與我自己用過的維度，因為理論上300維就算在空間散佈上有較多的空白，但就feature的抽取上理應能得到更多的訊息。但就此層面來看，在training的過程中需要對各個feature做抽取所需要花費的時間應該較長。所以基於硬體的限制，我們還是採用200維的向量為主，也有試過降低維度去做。雖然訓練速度提升了，但效果不怎麼理想。

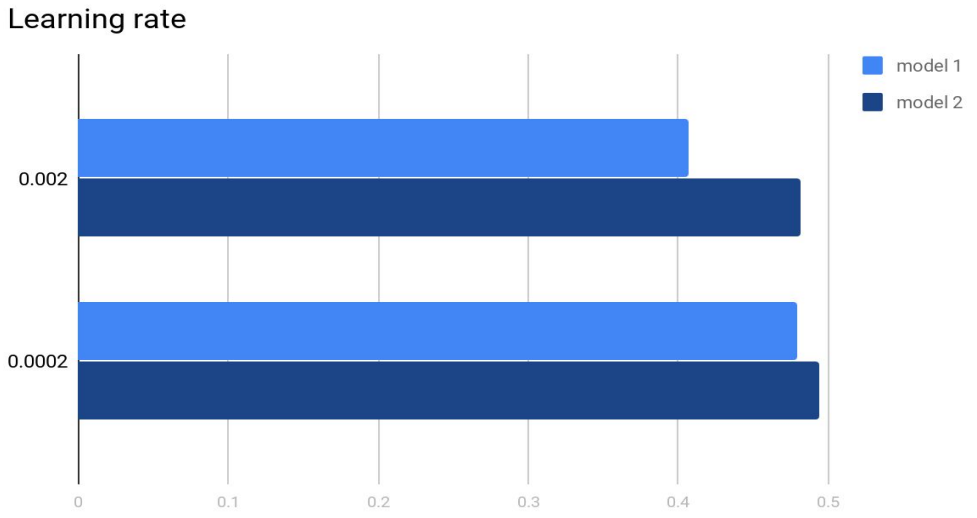
longest Input:

另外我們對於word padding的長度也做過各種嘗試。我們試過15、17、18、20、21，但其差異並不明顯，並且這些調整參雜在各個model中，但是對於分數的影響並沒有達到足以明顯影響結果的地步，其結果可能為在中文中，

整個語句中重要的字彙不常出現在頭尾，又因為我們是做end padding ，所以在做padding的時候就算被長度不夠被切掉也不影響整句話的關鍵詞句。

Learning rate:

learning rate的調整是這次project我們看得見差異最大的部份。由於硬體與時間的限制，在model 2的訓練上往往需要較長的時間才能完成一個epoch(大約1600秒左右)。在時間的壓力下我們試著提高learning rate。原本一個epoch 只會上升0.002左右的val_acc，但我們將lr從0.0002調到0.02之後，第一個epoch 就可以到0.013的val_acc，不過結果是train出來的model準確度比lr低的model少了大概1%，我們也在model 1嘗試了一樣的作法，結果更懸殊準確度大概差了有7%左右。所以我們還是決定使用較低的lr做training



ensemble:

在最後面的階段我們試著使用ensemble來提高準確度，主要以model 1與model 2為主，因為我們發現在單一model下做ensemble的效果很有限，若是混雜不同model可以提取各個model的優點，在最後其我們加入model 4之後，將原本卡在0.550左右的分數直接推到0.561，所以後來我們轉而訓練model 4他的訓練速度快，並且分數也不亞於model 2，而後我們為了增加model多樣性又加入了，model 5、model 6，但是我們把每個model取一樣比例加進去後發現效果不彰，而後改用weight voting以不同model個數去調整比例，得出model 1*5、model 2*5、model 4*5、model 5*2、model 6*2可以達到public 0.57549的成績，但private 只有到達0.55612 總體而言有蠻嚴重的overfitting，以下為各個model各取4個做ensemble結果。

	model 1 x4	model 2 x4	model 4 x4	model 5 x4	model 6 x4
score	0.52015	0.51027	0.54110	0.54980	0.34743

- # Conclusion :

這次project 的題目，主要是從題目給出的6個選項中，選出最有可能的下一句話與上一句呼應。在我們嘗試的幾個方法中，發現其中最有效的是在embedding完接RNN後再最後一層做Flatten之後再做內積。我們還發現有分詞與無分詞的方法相差的分數不多，所以第2種方法我們嘗試對於句子做PCA，嘗試對於上下句中的主成份做分解，再做K-means對各個選項做分類，但效果並不好。後來又K-means的score做評分，但結果仍然不佳。所以，後來只採用內積的方法來做此次的project，不過一開始的時候效果很不好，可是在加入上下句不連續的句子作為錯誤的label後效果才有明顯的進步，接著我們嘗試增加錯誤label的數量，並且想到一般來說選擇題中錯誤的label本來就比較多，所以我們就將原本1:1調整到1:5與這次題目6個選項的比例相當得出單個model在kaggle score上可達到0.49369的分數，但因為與前面的分數仍然有落差，所以我們試著將手邊的model做ensemble，達到了目前0.55059的成績，而後來繼續增加新的model試著衝高public score但最後的結果卻有很嚴重的overfitting，但其平均算起來的分數仍然很不錯，但對於這個問題來說，我們就沒有使用什麼特別的作法來完成此次的project，算是有點可惜的地方。

- # Reference :

1. https://blog.csdn.net/YhL_Leo/article/details/51736830
2. https://github.com/fxsjy/jieba