Machine Learning 2017 Fall Final Report

Listen & Translate

Team Name : NTU_r06922105_LeooooooWang
組員：R06922105 王捷生 ｜ R06922075 翁瑋 ｜ R06723025 林耘寬

1. **題目概述：**

進行台語翻譯，給定一句台語音訊檔(經過MFCC)，要在答案的選項中選取出最有可能的中文翻譯，答案為四選一的選擇題，每題都有一個相對應的正確答案。

2. **資料前處理**

在這個題目當中，我們可以從助教那邊得到台語音訊跟相對應的中文句子當作training data，期中台語音訊的部分，助教們已經幫我們整理成mfcc格式，每一句話被切成長度不等的向量，每一個維度又各自代表不同的39維向量，training data中總共有45036句。

- Mfcc：

對於不同長度的vector進行padding，檢查發現training data跟testing data最長的為246維，將其他不夠長的vector padding成246維的vector，padding的方法是在不夠長的vector後面補上39維的0。

經過padding後的mfcc trainging data 格式為：(45036 , 246 , 39)

- 中文句子：

助教給的中文data中，每一筆句子都是用空格給切開，在考慮要使用字還是詞做word embedding時，發現testing的選項中，同一題的不同選項，並不是詞的意義很相近，而是字數相同，因此決定以字為單位做word embedding，使用fastText提供的dictionary，將每個字轉成300維的向量，因為training data跟testing data中最長的句子為13個字，將不足13個字的句子後面padding上300維的0。

padding後的chinese trainging data 格式為：(45036 , 13, 300)

- Testing data ：

因為每題是四選一的選擇題，於是將mfcc向量各複製四遍，同一題對每一個選項去預測出是否為該相對應中文句子的機率，在四個機率中選擇最高的那項當作我們的答案。

3. 想法與解法

在仔細看過助教投影片上面建議的方法後，我們決定試試看seq2seq。畢竟只看投影片上面的介紹，似乎只是兩段RNN接在一起就可以完成。原以為能輕鬆通過simple baseline，但不論用斷詞還是斷句抑或是用Fastext和gensim做不同的word embedded所建構的的model。將model產生的sequence與選項做比對選擇效果都非常的不理想。

於是我們馬上改用投影片裡面給的另一個retrieval model，一樣是用fasText(也有試過gensim)和jieba去做word embedded的部分，在通過LSTM後與MFCC做內積，並將得出來的結果與選項計算cross entrophy選出正確答案。雖然模型的部分很快就完成了，但我們將validation loss比較低的model拿去做預測，出來的結果還是只有0.25左右，跟用猜的差不多。

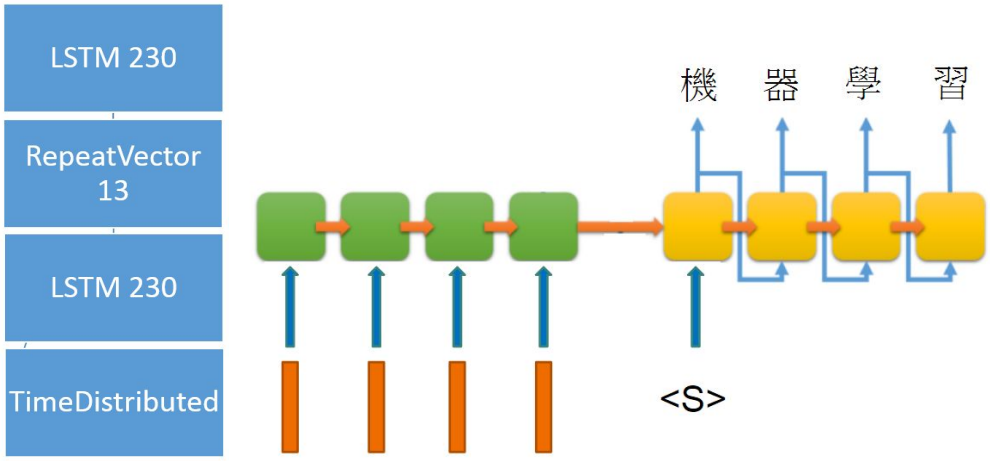
深受打擊的我們拿結果去做分析發現，由於所有的training data都是正向的，所以機器並沒有學習到所謂"錯誤"的選項，於是我們決定加入一些打亂過的錯誤資料，結果就得到了爆炸性的進步，我們的預測正確率來到的0.46，然而此時仍不足以讓我們通過strong baseline。

接著我們將想說在4選1的選擇題，加入3倍的錯誤training data似乎蠻合理的，結果讓我們又增加了0.02的正確率，距離strong baseline只有一步之遙。於是我們將錯誤的training data挑出來看發現model似乎也將有些正確答案的字數也學起來了，機器只靠答案的字數就能去判斷正確與否。

我們只好重寫我們生成錯誤data的方式，只選擇有相同長度答案去生成錯誤的training data，這項改動讓我們的成績順利來到0.56順利的衝破strong baseline。

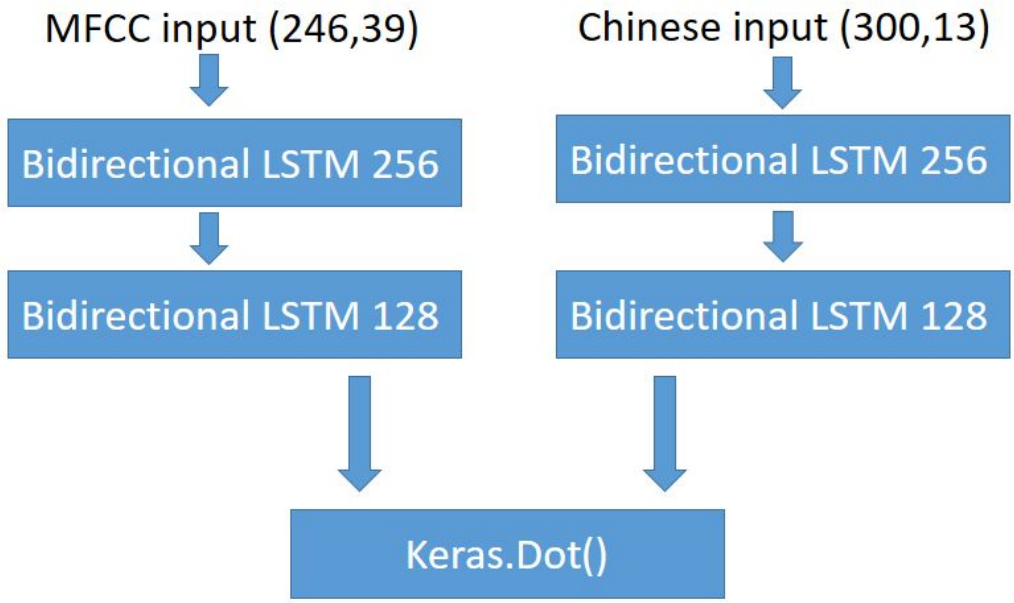
4. 模型

一開始對於模型的使用上，選擇了Seq2Seq的model。



但因為實作上的種種困難(例如：中文字的翻譯並不像拼音語系一樣有固定數量的字)，在歷經幾次失敗的Seq2Seq模型之後，決定改投向retrieval model的懷抱，相較於Seq2Seq model中input跟output的長度不固定，retrieval model接收

兩個不同的輸入，經過RNN後輸出兩個長度相同的向量，再去判斷相對應的training data所產生的向量相似度應該要較高，而非相對應的training data所產生的向量相似度應該要較低，以此為目標來訓練我們的模型，下圖是我們的retrieval model示意圖：



5. 訓練結果：

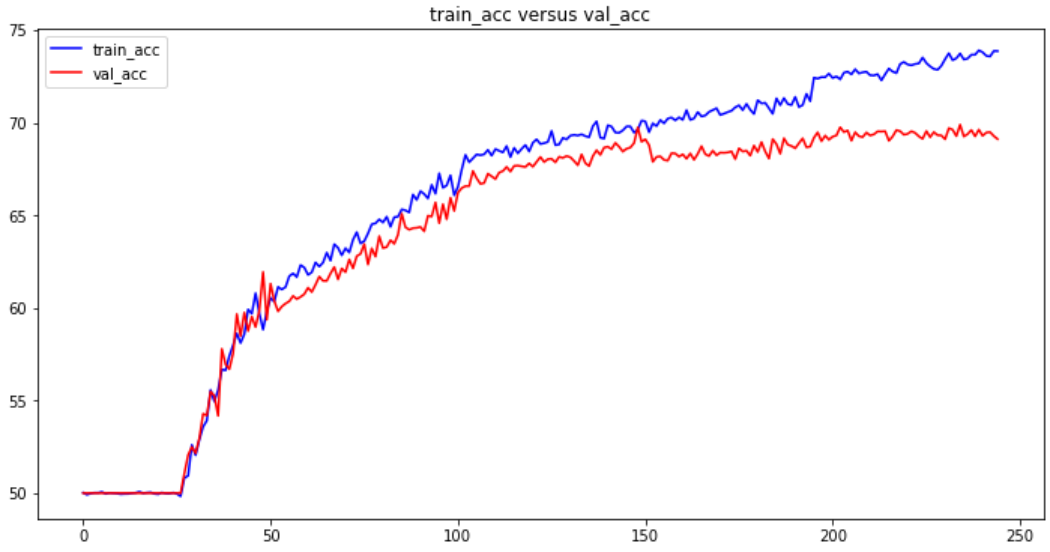


Fig. Training中的accuracy：後來一直卡在0.7左右

Training時各有一半是正確一半是錯誤的資料

這裡的Accuracy指的是判斷單一筆音訊是否是相對應句子的正確率

跟kaggle上的分數不一樣

models	Kaggle Public
LSTM*1	0.57099
GRU*1	0.59800
LSTM*2	0.58099
GRU*2	0.60300
LSTM*2 + GRU*2	0.62100
LSTM*4 + GRU*4	0.61000

(註:表格中的LSTM指的是Bidirectional LSTM)

6. 問題討論 (錯誤data的產生對模型的影響)

在使用retrieval model時，除了原本的training data之外，還必須提供一些錯誤的data給模型，讓模型要有分辨對錯的能力，避免模型學到不管如何的資料都只要回答true即可，我們提供給模型的正確與錯誤資料筆數為1:1，也就是在每一筆的traing data中，都有另外給定一筆錯誤的資料。

那麼該如何決定錯誤資料的生成方式呢?在這個產生錯誤資料的部份我們共經歷了三個階段：

First Stage：

第一個想法當然就是把training data中相對應的中文句子向下位移一位，之後因為出來的成果不佳，而在檢討錯誤資料的產生方式時，重新發現了testing選項中每個長度都相同，於是開始懷疑我們的模型是否靠著句子長度就可以分辨出正確與否，而當每個選項的長度都相同時，模型當然就無所適從。

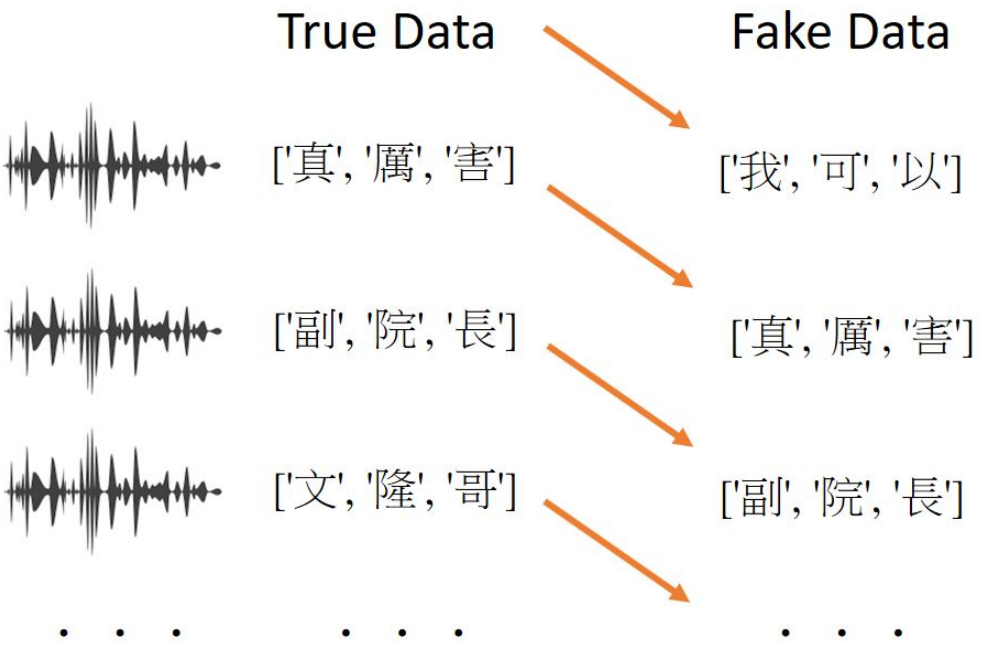
Second Stage：

於是開始更改錯誤資料的內容，目標是把相對應錯誤的中文句子長度也變的相同，在這一個階段中，我們將正確的中文句子進行shuffle，當作該筆data錯誤的資料，如此保證了錯誤資料的長度一定跟正確資料相同，在這一階段training過程中相當的順利，validation的accuracy甚至超過了0.8，0.8這個數值代表的意義，相當於每一題全隊的機率為(0.8)^4 = 0.4096，也就是說在kaggle上的期望分數會有0.4左右，足以讓我們通過simple baseline，但事情沒那麼簡單，不然也就不會有第三階段了，丟上kaggle的分數不斷的在0.25左右徘徊，也就是跟用猜的沒兩樣。

Third Stage：

這個階段我們又開始思考為何在validation上的表現跟丟上kaggle的表現差這麼多，最後對於模型做了各式各樣的試驗之後，發現我們的模型只要是餵給他正確的中文句子(並不一定要是相對應的句子)，通常預測出來的值都會是True，這一部分很明顯出了很大的問題，後來我們發現了是打亂中文句子時，只會得該句變成一個毫無意義的文字排列，於是我們的模型就因此只學會了分辨出有意義跟沒有意義的中文句子這件事。

最後我們將training data中的所有句子依照字數做分類，同樣長度的句子會向後位移個K位，當作該筆trainging data的假資料，另外在train了10~20個epochs左右時，會更改K的值，讓模型可以學習到更多分辨錯誤的資料，資料的產生如下圖：



7. 使用到的Library跟Toolkit

- Keras 2.0.8
- Tensorflow 1.3.0
- Numpy 1.13.3
- Panda 0.21.1
- fastText wiki.zh.vec

8. 執行環境

Ubuntu 16.04
Cuda 8.0
Python 3.5.2
Google Cloud Platform

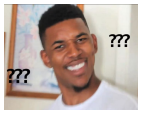
- CPU：6 core
- RAM：32GB
- GPU：NVIDIA K80 x1
- GPU memory：11GB

9. 心得

在看過其他組別的期末發表後，發現很多雖然同樣都是用retrieval model，但高分的組別幾乎都有另外對MFCC的training data做降維，藉此將MFCC的特徵值提取出來，或是利用MFCC隨機取樣產生額外的training data來提高model的準確率。

在這邊稍微介紹一下我們在使用Google Cloud Platform前的系統環境。

在只有一張完全不支援Cuda的AMD Radeon HD 7700 Series顯示卡和一張windows體驗只有5.9分 i5-3470 CPU的硬體環境下，這次Final Project的前段時期我們可說是過得生不如死(在寫hw3 CNN時也是生不如死)。直到聽到其他組建議我們可以使用Google Cloud Platform提供給偏鄉窮苦學生的免費300鎂體驗方案，使用一個小時只要大約美金1塊錢，比我去網咖寫ML final project 還要便宜。原本在桌機上要跑8到9個小時的training，丟到GCP上面跑只需要30分鐘左右。我們不禁開始懷疑GCP是不是架設在另一個平行時空，並產生之前到底幹嘛以及為啥我爸爸不是連戰的感慨。



ML有三好，頭腦好，運氣要好，錢越多越好

10. Reference

https://blog.keras.io/a-ten-minute-introduction-to-sequence-to-sequence-learning-in-keras.html
http://www.zmonster.me/2016/05/29/sequence_to_sequence_with_keras.html
https://github.com/farizrahman4u/seq2seq
https://www.tensorflow.org/tutorials/seq2seq