Machine Learning 2017 Fall － Final project

Listen & Translate

NTU_r05229016_南無阿彌陀佛

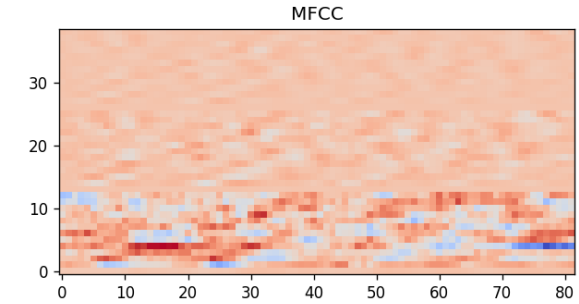
R05229016 羅章碩
B03701221 王逸庭
R06942128 許祐銘
R05229014 鄒適文

2018.01.22

題目： Listen Translate

三個題目，我們一開始先嘗試 Listen Translate 和 Question Answer ，但最後我們選擇做 Listen Translate。

此題目標是利用台語音訊檔的MFCC選出相對應的中文語句。



1) 真厲害
1) 你很棒
2) 是飛碟
3) 你打我

Preprocessing / feature engineering：

使用的模型不一樣，準備訓練資料的處理方式也不一樣。以下是兩個模型的前處理方式：

1. Seq2Seq Model

A. one-hot-encoding :

因為我們字典裡沒有英文，所以我們在每一題的最前面加上 'B' 當作是 Begin of sentence，並在最後面加上 'E' 當作是 End of sentence，之後將 training data 中的每一個"字"包括('B'、'E') 編號並做成字典，再利用 np.zeros 開兩個陣列 (sample數量, max_decoder_length(max_decoder_target), 字典大小)

將每個字代表的位置設成 1，在餵給 decoder 的 data 最前面是'B'，而最後的 label 的第一個字則沒有給'B'。

B. Gensim word2vec :

用 gensim 的 word2vec 將 training data 訓練出一組詞向量之後當作 model 中 embedding layer 的初始權重，word2vec 的參數設定為詞向量維度 64、每個字只要出現一次就會納入字典中、slide window 給5、使用 skip gram 的方式做 training、訓練次數給 10、最後 negative sample 給10代表有10個'noisy words'。我們在這邊有測試分"字"或是利用 jieba 分"詞"去做 training，討論以及結果會在後面的地方詳述。

2. Retrieval Model :

文字部分處理同前面有用 one-hot-encoding、word2vec，不同的部分在於在 Retrieval model 的架構下，我們必須自己創造 training data，而其實只要使用 np.ones 就可以很簡單的創造對的 training data，但我們還需要錯誤的資料，所以我們生成錯誤標記的方式如下：

A. Negative Sample Generation 1 :

原始 mfcc_data = [feature1, feature2, feature3, feature4, feature5]
正確答案 captions = ['caption1', 'caption2', 'caption3'] 標記為 [1, 1, 1]
np.roll(captions, 1) = ['caption3', 'caption1', 'caption2'] 標記為 [0, 0, 0]
np.roll(captions, 2) = ['caption2', 'caption3', 'caption1'] 標記為 [0, 0, 0]
而這邊在選答案時是將每個選項的 list 加起來並平均後選出最高分的那個選項當答案，例：
有兩個選項 A = [0.7, 0.6, 0.8] , B = [0.4, 0.5, 0.1, 0.2]
此兩個選項平均分別為 A=0.7, B=0.3，則將會選A當成預測結果。

B. Negative Sample Generation 2 :

原始 mfcc_data = [feature1, feature2, feature3, feature4, feature5]
正確答案 captions = ['caption1', 'caption2', 'caption3', 'caption4', 'caption5'] => 標記為1
np.roll(captions, 1) = ['caption5', 'caption1', 'caption2', 'caption3', 'caption4'] => 標記為0
np.roll(captions, 2) = ['caption4', 'caption5', 'caption1', 'caption2', 'caption3'] => 標記為0
np.roll(captions, 3) = ['caption3', 'caption4', 'caption5', 'caption1', 'caption2'] => 標記為0
np.roll(captions, 4) = ['caption2', 'caption3', 'caption4', 'caption5', 'caption1'] => 標記為0

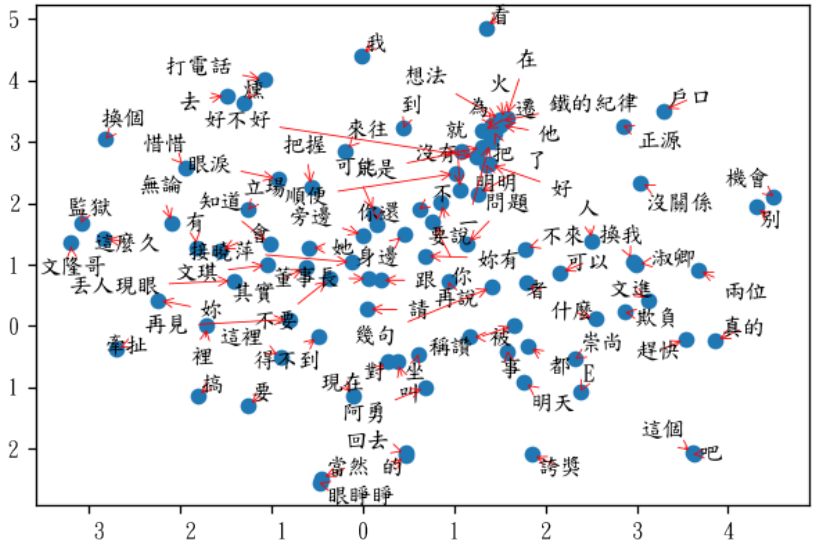
兩種方法不一樣的地方為，第一種是將 answer 開為一個矩陣，第二種方法是使 answer 直接是 1 或 0。之後我們經實驗發現，第二種方法會比第一種方法還要好，我們覺得可能是因為經 np.roll 旋轉之後，即使順序不對 (理論上應該要標記為 0) 在平均過後可能分數還是蠻高的，這樣會使的明明是錯誤的答案卻被當成正確答案，因此表現可能會不太好。

所以我們最後是選擇第二種方法來訓練，且在我們的正確 data 數量 : 錯誤 data 數量是 1：4 時分數最高。

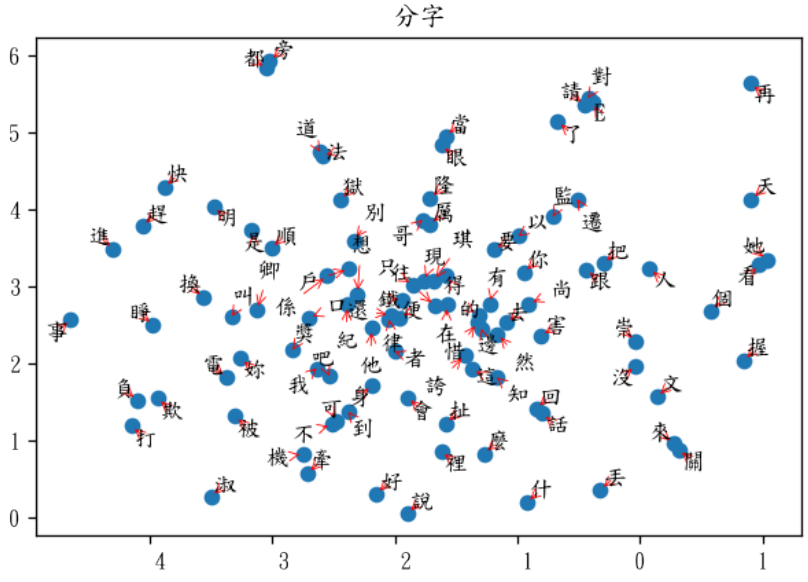
前面提到我們也有嘗試過使用 jieba 分詞來測試：之前我們在做前處理的部分都是拆成一個字一個字去看，但是真正有意義的是詞，而我們好奇整個句子用分詞或分字去拆一個句子對結果的影響如何，所以我們利用 jieba 來幫助分詞，測試對結果的影響如何。

下圖為字和分詞之後利用T-SNE做降維之後畫出來的結果。

分詞



分字



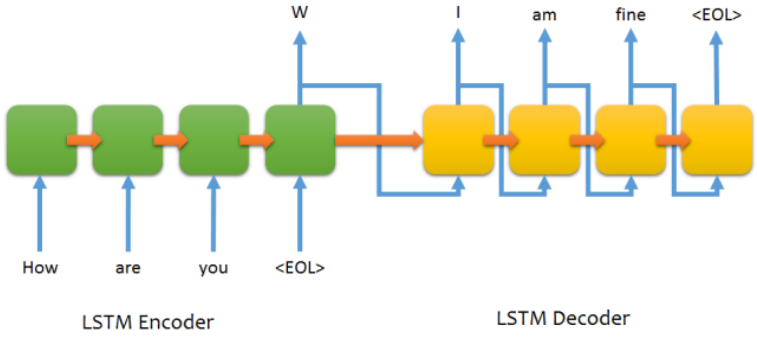
從上面的圖可以看到詞跟詞之間的關係似乎比較明顯，但是字跟字都分散的比較開。而在kaggle上的分數用分字的方法分數為 *0.769*、而用分詞的方法則為 *0.629*。乍看之下用分字的方法結果比較好，但是仔細去看 Predict 的時候發現用jieba去分 testing data 的結果時出現了OOV的問題，大約有2948個詞沒有出現在字典裡，而沒出現在字典裡的詞我們就給0去處理，而這可能導致在kaggle上的分數差這麼多，因為利用分詞的方法在訓練過程中loss和accuracy的數值其實跟分字不會差太多，甚至還比較好一些，因此如果將OOV的問題給處理掉的話或許結果會好一些。

除此之外，jieba的分詞似乎也會有些問題，有些句子在分詞的過程中會分的不好導致出現OOV的問題，像是：完手術、總比文隆...，所以如果有將分詞分得好的話應該也可以讓kaggle分數再往上爬。

模型與實驗：

我們主要使用兩種模型，分別是Seq2Seq(使用 teacher forcing)和 Retrieval model，一開始我們是使用seq2seq去做預測，但準確率並無法有效的提升。但因為這題是選答案，於是我們最後採用Retrieval Model 做處理。

A. Seq2Seq



此模型特點輸入序列和輸出序列的長度可以有變化。將輸入encoder，輸入decoder，輸出預測序列，。

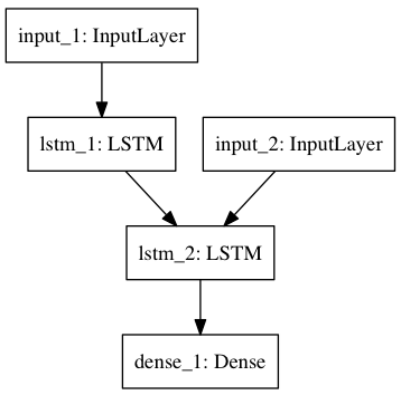
首先我們先採用基本Seq2Seq模型：

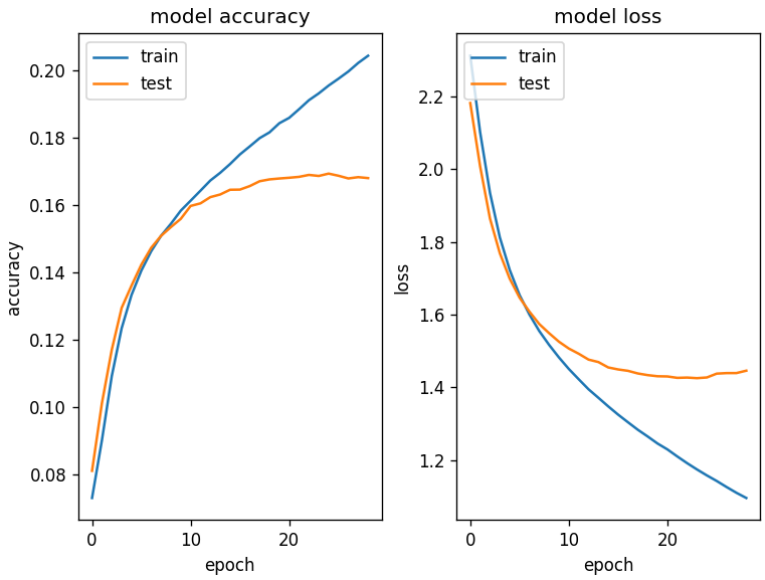
對MFCC，疊了一層LSTM，並輸入decoder，與輸入的文字檔，輸出預測結果。使用梯度下降法Adam 以及Categorical Cross entropy 作為損失函數。

- LSTM Hidden layer = 256
- Dense layer = 2391
- Batch = 128

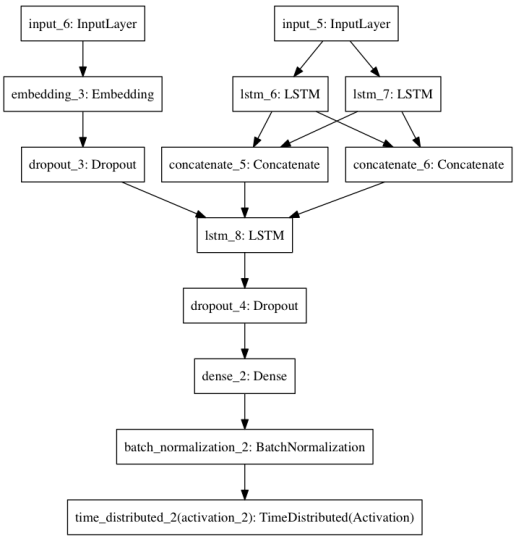
我們基於上面的基礎，將MFCC的LSTM改成Bidirectional，並加上了Dropout ：

- MFCC : LSTM (128) + LSTM (128) >> Bidirectional (256)
- LSTM hidden layer = 256
- Dropout = 0.2
- Dense = 2391
- Batch = 128





接著我們對MFCC疊了Bidirectional LSTM，對中文利用Fasttext 提供的中文字 model，對文字檔進行Embedding。並在過程加入Dropout，以及在最後加入Time Distributed。

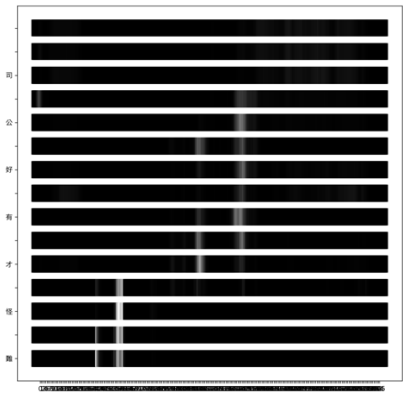


- LSTM_6 (128) + LSTM_7 (128) = Bidirectional (256)
- Embedding_3 dim = 100
- Dropout_3 = 0.1
- LSTM_8 layer = 256
- Dropout = 0.2
- Dense = 2391
- Batch = 128

最後我們加入Attention的機制，但因為機器和時間的限制，我們Attention只嘗試一次。而因為Keras並沒有提供Attention相關函數，必須自己疊（如右圖）

- LSTM layer = 64
- Embedding = 100
- Batch = 512

但跑出的結果卻比沒有Attention更糟糕，Val loss : 1.4376 / Public acc : 0.24900，



於是我們檢驗是否真的有達到Attention的作用。

下圖為attention顯示的樣子，看起來是有做到Attention，推測是參數的問題。

	Seq2Seq(1)	Seq2Seq (2)	Seq2Seq (3)
Val loss	1.4407	1.4456	1.7408
Public accuracy	0.37300	0.37500	0.30100

結果比較：

在 Test 的部分，我們採取將每個答案和音檔輸入model，選出loss(Crossentropy)的最少的答案，而因為 Attention 因為用的不是最好的參數所以在此不討論。下表為結果：

下表使用(1) (2) (3) 模型，行的文字分別為四個選項，其下面的數字分別是將答案和MFCC丟進的model算出的Loss，而Decoded sentence則是將MFCC丟進去模型產生的預測序列。

model	妳說	有啦	妳會	焢肉	Decoded sentence
1	0.553	0.549	0.811	0.439	我告訴妳
2	0.354	0.591	0.470	0.561	我們都不會擔心
3	0.678	0.748	0.851	1.213	我們不可以

model	開動完手術又怎麼樣	別想搞那些有的沒的	所以我不忍心看到她	我們真的不是故意的	Decoded sentence
1	2.975	2.694	2.104	1.222	我們都不希望
2	2.649	2.886	1.890	1.096	我們都不會
3	3.385	3.592	2.838	1.819	我告訴你

model	醫生交代不可以爬樓梯	已經有一些證據和線索	妳還記得我們的孩子嗎	因為今天文隆要回來了	Decoded sentence
1	3.234	2.974	1.940	1.711	你們不要緊張
2	3.509	2.963	2.022	1.645	你們不要這麼多年
3	3.974	3.673	2.655	1.969	不然你們

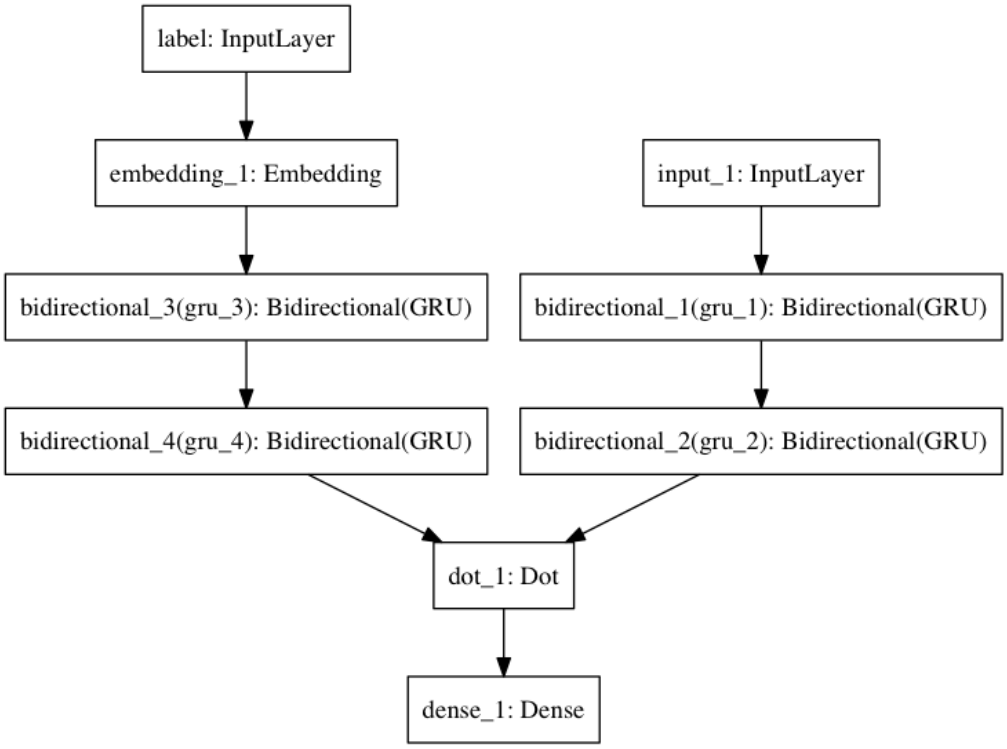
model	老師	記住	難產	還沒	Decoded sentence
1	0.737	0.643	1.369	0.356	我告訴妳
2	0.550	0.478	1.427	0.379	我們都不會
3	0.745	1.279	1.458	0.503	我們告訴你

model	不對	好是	爸媽	來走	Decoded sentence
1	0.581	0.658	0.286	0.609	爸你先走了
2	0.474	0.555	0.408	0.667	謝謝
3	0.516	0.708	0.556	0.714	你們是

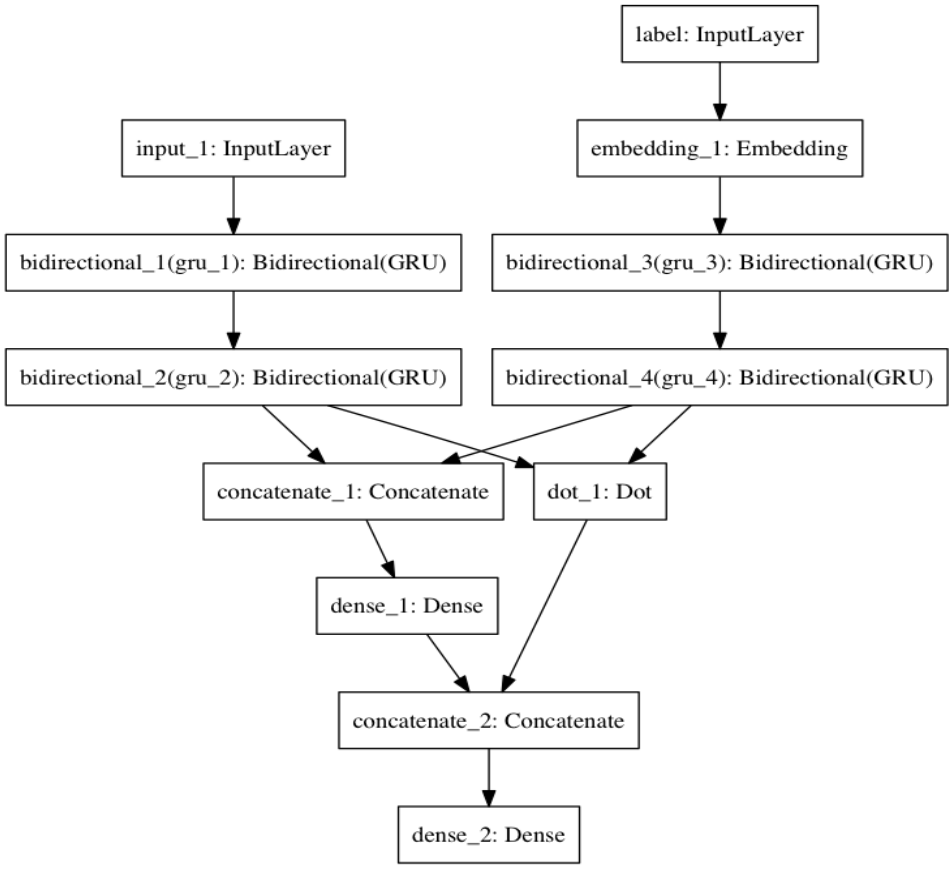
我們可以看到其實加入 Bidirectional 對結果並沒有太大的進步，而加入fasttext結果反而退步。於是我們拿Test前五題，觀察他們預測狀況差異：
Model 1, 2, 3 : 1, 2 的 loss 比較相近，加入fasttext(3)產生的 loss 都相對較大。
從上面五個表格可以發現直接進行預測和給答案找出的答案相似度很低，像是產生的字數不一致，以及decoded sentence大多數給的都是常見不特殊的語句，並無法有效辨別答案。這顯示利用Seq2Seq達到此題的目標是有一定的難度的，所以我們嘗試另一個Model – Retrieval 。

B. Retrieval Model

以下是我們的 model 架構圖，我們嘗試了兩種不同的疊法。
第一種：使用dot當作輸出，疊法是先將 MFCC 通過兩層(或三層) Bidirectional GRU，以及將我們的選項通過 embedding 之後再接兩層 Bidirectional GRU，最後兩者 dot 起來，而參數部分：
activation = sigmoid ，optimizer = adam，loss 則是 binary_crossentropy



第二種：由於不想只有使用dot當作最後的輸出，嘗試了一些不同的接法，而最後試出來比較好的結果如下圖疊法。



結論及心路歷程:

我們花了很長的一段時間再研究要怎麼使用keras寫出 seq2seq、bidirectional seq2seq 及 attention seq2seq model，雖然最後終於寫了出來，但發現不管怎樣調整參數(最後聽別組報告才發現不應該用 teacher forcing)，分數都沒有往上的趨勢，所以我們決定使用路邊阿伯都會搭建的 retrieval model，經過 ensemble 七個model之後最後得到了超越 0.8 的正確率!!!

而在retrieval model我們測試了：

1. 不同的Negative Sample Generation方式：
   發現不考慮每個字，只考慮一整句話，會有比較高的正確率。(詳見上面preprocessing)。
2. 對文字使用 tokenizer 與 wor2vec 做 embedding：
   使用 word2vec 在 Embedding layer 有 pretrain 的weights會有比較好的表現。
3. 對文字使用jieba分詞測試：
   使用jieba分詞不會有比較好的結果，推測可能是因為jieba是依據語意去做分

詞，而且jieba有些分詞的結果並不是很好。除此之外，只拿training data做字典時，在testing data也會出現OOV的問題，所以要有大量pretrain好的字典才可以避免testing data有太多OOV的問題，在前處理上要花更多時間。而如果只用字做成字典時，結果也不會太差，可能是每個台語字的發音可以對到每個中文的字，所以結果也還會不錯。

4. 對音訊檔做down sampling：
   由於不太懂downsampling怎麼做會比較好，我們只測試了把長度減半的方法(長度若200做完會變100，隔一格取一個feature的概念)，而這種方法會使得表現下降，可能是在 down sampling 時沒有抓到重要的feature。
5. 不同深度的retrieval model：
   由於計算資源的不足，我們只測試過兩層GRU、三層GRU，而三層的表現是比兩層的還要好。
6. 不同結構的retrieval model：
   因為最後想不到進步的方法了，所以開始嘗試不要只用dot當作retrieval model的output，我們嘗試在後面做一些變化，而最後表現有上升大約0.05，最後的模式結構詳見上面 retrieval model 架構部分。

最後以下是我們 Retrieval model 的各種實驗的數據：

Negative Generation	方法 1	方法 2	方法 2	方法 2	方法 2	方法 2	方法 2
Model	第 2 種	第 2 種	第 2 種	第 1 種	第 2 種	第 2 種	第 2 種
Method	兩層 GRU	兩層 GRU	三層 GRU	兩層 GRU	Down sample、三層 GRU	兩層 GRU、Jieba	三層 GRU、W2Vec
Public Score	0.597	0.756	0.768	0.694	0.693	0.629	0.769

分工

B03701221 王逸庭：模型搭建(seq2seq)，報告撰寫

R05229016 羅章碩：資料前處理、模型測試、報告撰寫

R06942128 許祐銘：模型搭建(retrieval model)、報告撰寫

R05229014 鄒適文：模型搭建(retrieval model, bidirectional seq2seq)、資料前處理、報告撰寫