機器學習

期末專題報告

題目:Listen and Translate

組員:電信碩一 黃釋平 R06942082 (Retrival model)

電信碩一 陳建榜 R06942020 (Sequence to Sequence)

電信博一 彭正安 D06942013 (data proccessing)

1. Preprocessing/Feature Engineering

- 音訊檔(train.data)

  共有 45036 筆資料，每筆資料的每個瞬間都是 39 維的向量，但由於每筆音訊長度不相同，我們必須先對其做 padding 到 246 筆，缺少的在後面補上 np.zeros(39)，最後資料 shape = (45036, 246, 39)。

- 翻譯檔(train.caption)

  我們分別使用 Seq2seq 和 Retrival model 來進行，詳細的 model 在下段會介紹，而這兩種模型對中文轉向量的方式不同，但同樣都有做 padding 使句子長度都為 13。

  在 Seq2seq，我們建立一個 one-hot 的中文字典，共有 2391 個字，每個中文字以長度 2391 的向量代表，其中只有一個值是 1，即代表那個字，最後資料 shape = (45036, 15, 2391)。

  在 Retrival model，我們使用 gensim.word2vec 加上以 wiki 資料做好的字典 model，將字轉成長度 300 的向量，並做 normalization，不在字典中的字則補 np.zeros(300)。在訓練時要製造三個錯誤選項，我們以隨機方式取其他行句子作為選項，並且對四個選項 shuffle，讓機器自己學該如何選，最後資料 shape = (45036, 4, 15, 300)

  這樣訓練出來的 model，雖然在 validation 有相當好的表現，但實際測試上傳準確率卻非常低，剛開始相當不明白，我們將訓練資料的隨機選項都故意挑相同長度的再進行測試，結果準確率下降非常多，才發現是句子長度問題。

  因此，我們又把生成三個錯誤選項的 function 加上「同長度」的規則，使每個選項像 test.csv 一樣，句子都一樣長，詳細 trainning 狀況會在第三段描述。

- 測試檔(test.data, test.csv)

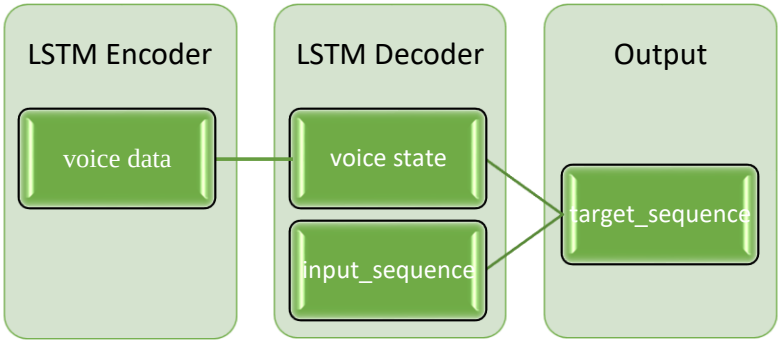
  test.data 是音訊檔，同樣必須做 padding 至 246 筆，缺少的補上 np.zeros(39)，最後 shape = (2000, 246, 39)。

  test.csv 每行有四個選項，在 Seq2seq 中以 one-hot 字典轉成向量，在 Retrival model 則用 gensim.word2vec 轉，處理方式同 train.caption。

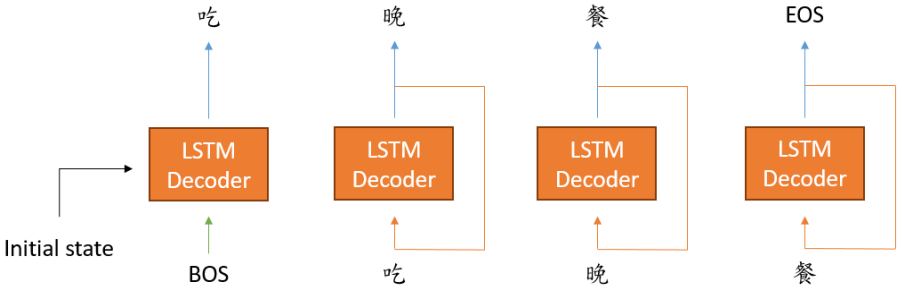
2. Model Description

- Sequence to Sequence (Seq2Seq)

I. 原理

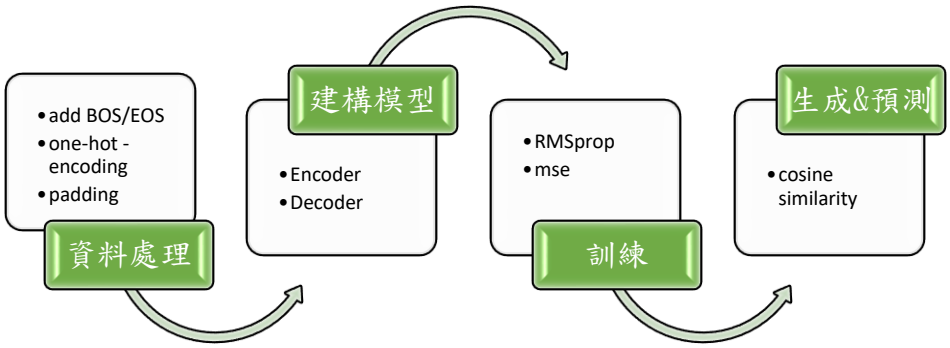


Voice data 是聲音檔轉成向量表示，藉此能拿來做運算，一串聲音向量進入 Encoder 後，全部跑完輸出 LSTM state，這個 state 就是代表這段聲音的意義，直接指定給 Decoder 當作 initial state。假設 sequence 是這段聲音的翻譯，input_sequence 則要在 sequence 前面補上開頭符號(BOS)輸入給 Decoder，target_sequence 要在結尾補上結束符號(EOS)作為 Decoder 的目標輸出。



也就是說，Decoder 要學到的是，在給定的 Initial state(代表情境、意義)之下，不斷預測下一個字，直到出現 EOS 代表句子完畢;同時，Encorder 必需要能生成含有正確情報的 LSTM state，提供給 Decoder 做預測，所以這兩者是相輔相成的。

II. 程式流程



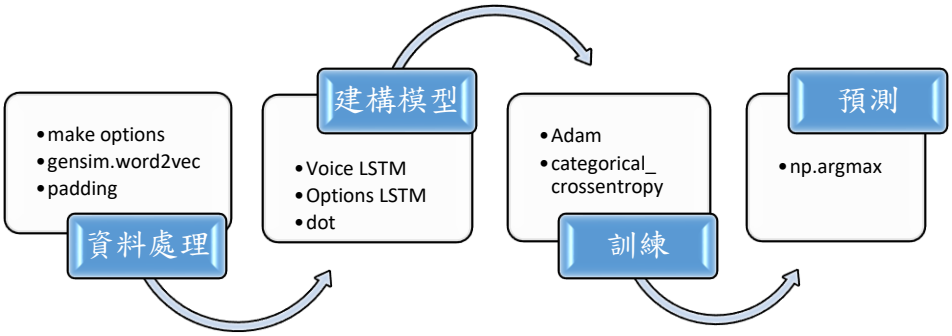
- Retrival model

I. 原理

Information Retrieval 原先應用在圖書館系統，使用各種方式將圖書資料的 feature 擷取出來，用來做檢索跟比對。其中一種叫做 vector model，將查詢條件與書目分別 encode 為同長度的向量，並比對相似度(dot,cos…)，便可知道兩者的相關性。

此法亦可應用在我們的期末專題中，因為是選擇最相關的選項，可以將聲音與選項 encode 後做比對，相似度最高的即為答案。但在訓練時，由於給定的 train.caption 只有正確的翻譯，所以我們要刻意製造錯誤選項，將正確的 label 標為 1、錯誤的標為 0。

II. 程式流程



3. Experiments and Discussion

- Sequence to Sequence (Seq2Seq)

先將音訊檔讀進來後，每一筆長度 pad 成一樣，也就是每一筆音訊檔的 shape 為 (246, 39)。另外在 train.caption 的部分，我們先利用 gensim.models.Word2Vec 套件將文檔中出現過的中文字記錄下來，得到共 2391 個不同的字，之後將每一個在 train.caption 檔案裏頭的句子以 one-hot encoding 的方式將每個文字都以一個(2391,)的向量表示。另外，若是句子長度小於最大句子長度時，將在其後補上全為 0 的(2391,)向量，直到長度與最大句子長度相同。

在實際 model 架構中，encdoer 及 decoder 各為一層的 LSTM cell，其中有兩個 training 參數:latent_dim、go_backward。而在 encoder 的輸出部分，我們取出其最後的 state 並將其直接指定給 decoder 當作 initial_state。而在 decoder 的輸出部分，我們用一層 Dense Layer 作為輸出，並且為了配合我們使用得 one-hot encoding 方式，我們採用'softmax'作為 activation function。架構可參照下圖。

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	(None, None, 39)	0	
input_2 (InputLayer)	(None, None, 2391)	0	
lstm_1 (LSTM)	[(None, 256), (None,	303104	input_1[0][0]
lstm_2 (LSTM)	[(None, None, 256),	2711552	input_2[0][0] lstm_1[0][1] lstm_1[0][2]
dense_1 (Dense)	(None, None, 2391)	614487	lstm_2[0][0]

在預測的部分，對於測試音訊一樣將其讀進來後 pad 成與原訓練用的音訊檔長度相同，將其作為 encoder model 的 input 做 predict 可得到此音訊檔的[state_h, state_c]，再將此 state 搭配起始向量["BOS"]餵入 decoder model 中，讓其不斷產出新的字元以及[state_h, state_c]，將新的輸出再次作為 input 餵入 decoder model，直到產生終止字元[“EOS”]或是句子長度超過最大句子長度。最後使用在訓練時訓練好的 Word2Vec model 來做相似度的判斷，如下：(可參照實際的程式碼)

I. 將 decode 出來的 sentence 與相對應的四個選項取出
II. 將每個屬於 decode sentence 的字元與選項句子中的字元算 similarity，並累加。
III. 將上一步驟中得到的數值除以每個選項的句子長度，因此可以得到 sentence 對於四個選項的平均相似度。
IV. 選擇相似度大的作為預測的答案。

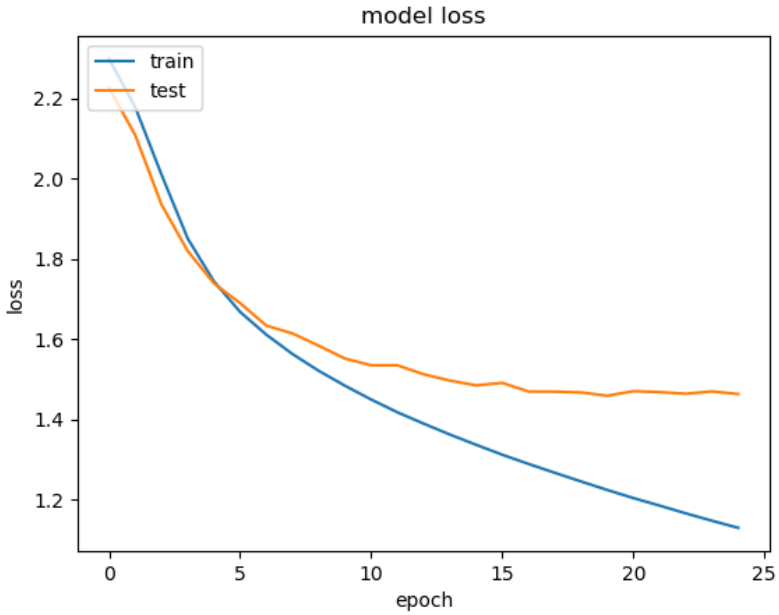
```
Q_A = np.zeros((2000,),dtype = 'int')
for i in range(2000):
    sentence = result[i]
    similarity = np.zeros((4))
    for j in range(4):
        option = _options[i][j]
        for word in sentence:
            for opt in option:
                if (word in bag.wv.vocab) and (opt in bag.wv.vocab):
                    similarity[j] = similarity[j] + bag.wv.similarity(word, opt)
            similarity[j] = similarity[j]/float(len(option))
    index = np.argmax(similarity)
    Q_A[i] = int(index)
```

訓練過程:

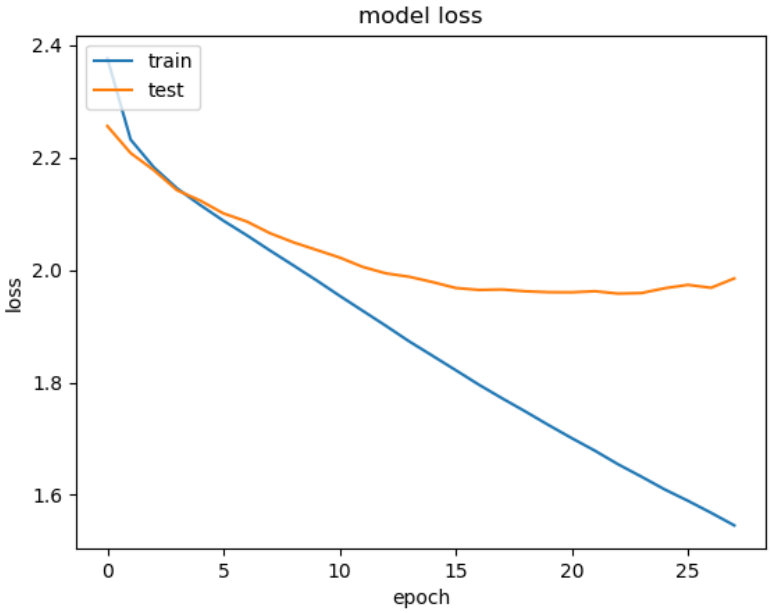
首先在 training 過程中，有調整的參數如下:

I. Latent_dim : 128、256
II. Go_backwards : True、False
III. Optimizers : rmsprop、sgd、adam
IV. Kernel_regularizer : l2

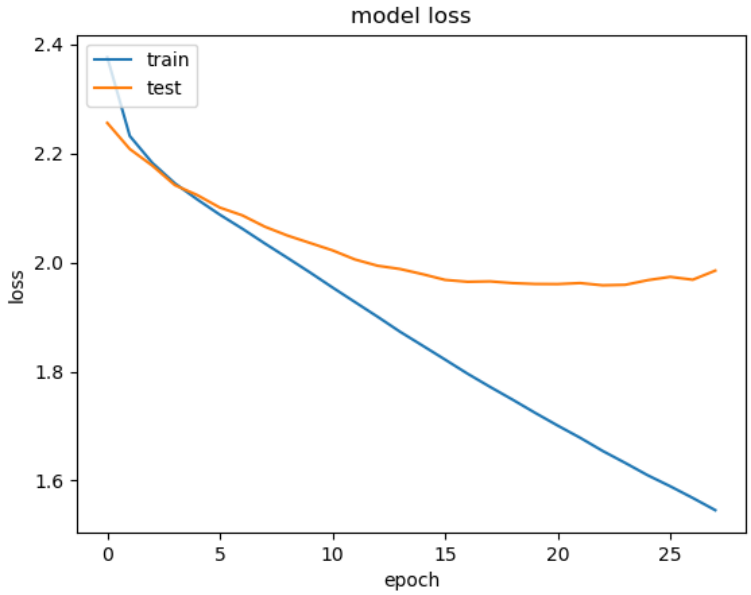
在最初我們先以 Latent_dim=256 的 LSTM 作為主架構，並沒有加上 regularizer 及 go_backwards 等參數。得到的結果如下。



可發現 train_loss 雖然持續下降，但 val_loss 在到達 1.6 左右時便收斂，有 overfitting 的情況產生。因此我想知道是那些原因造成此種情形。首先測試是否為參數過多導致 overfitting，所以藉由將 Latent_dim 降至 128 調整了參數多寡，並重新訓練得到下圖。



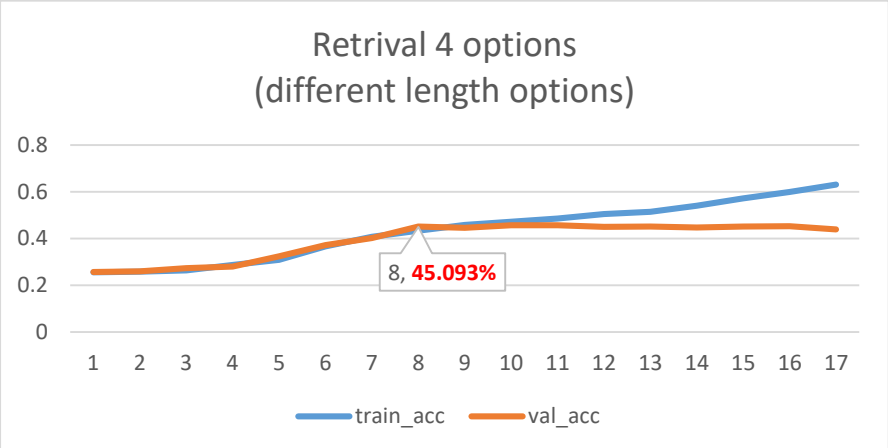
發現 val_loss 不降反升，因此轉而考慮使用 regularizer 並搭配較大的 learning rate，卻發現此時的 train_loss 與 val_loss 皆上升，上傳 kaggle 之後得到的分數也不如最初版本。所以我回歸到沒有加任何參數、latent_dim 為 128 的 model，嘗試使用 go_backwards 方法來訓練，原因在於 **voice data padding 的部分我們會在不足長度的資料補上 0 向量，而使用 go_backwards 方法可先把 0 向量的部分讀完後再進行訓練**。但 val_loss 與 loss 皆跟原本 model 相同。



而在 optimizer 選擇方面，在每一種模型中皆嘗試過 rmsprop、adam、sgd，發現 rmsprop 可使得 loss 降得較低，而 kaggle 的分數也較其餘兩者高出一些，因此最後的結果都是使用 rmsprop 作為 optimizer。

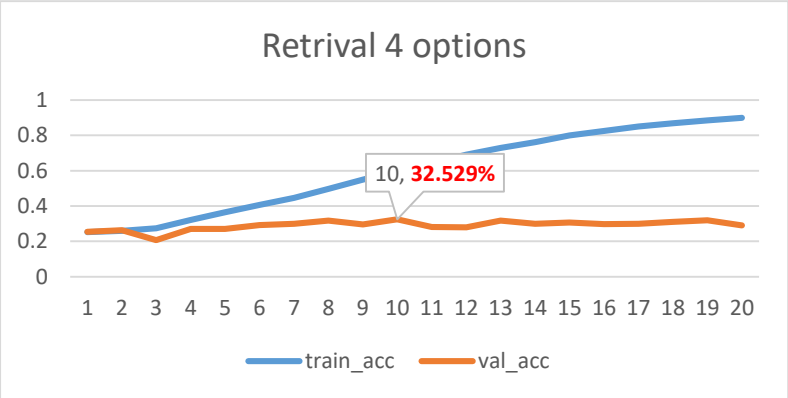
- Retrival model

剛開始嘗試生成三個錯誤選項，含正確的共有四個。音訊和選項都先經過 LSTM(256)-Dense(128)-Batchnormalization()-Activation('linear') 轉成 128 維的向量，音訊向量再分別和選項向量做 dot，輸出四個值，最後接 Activation('softmax')，代表**要輸出四個選項的為答案的機率，在預測時就選擇機率最高的做為答案**，以下是訓練的情形：

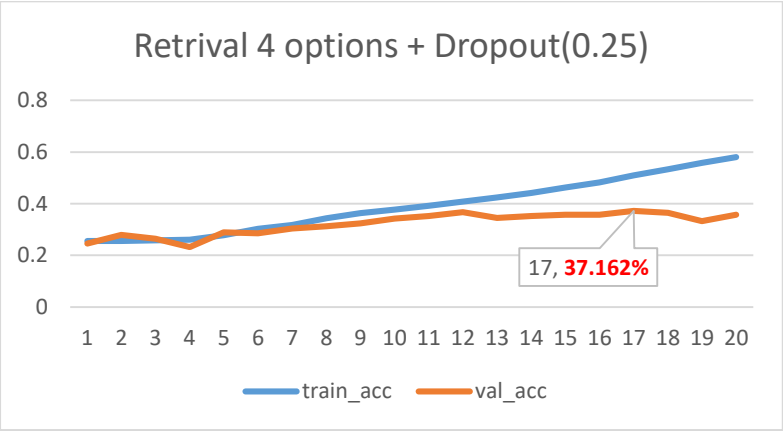


看到 45.093%實在欣喜若狂，但實際測試上傳竟只有 **22.8%**，與預期差異非常大。經過不斷測試，我們發現是句長問題，test.csv 中的選項都是相同句長，我們在生成錯誤選項時並無這規定，將 trainning set 生成為相同句長選項後代回 model 測試，果然 Accurancy 下降非常多。

我們猜想可能機器學到直接以語音長度和選項長度來做判斷，並沒學到語意，因此在之後都以相同句長的選項來做訓練，並將 validation set 與 trainning set 完全隔離(在隨機生成選項時，不會選到彼此的)，以下是以相同架構重新訓練的結果：

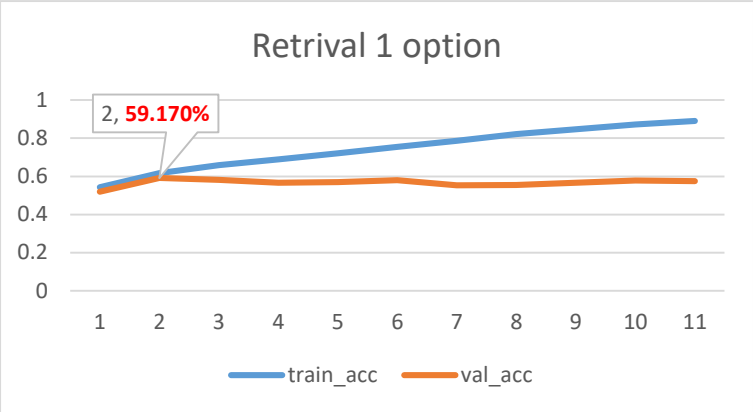


Validation accurancy (val_acc)卡在 32.529%無法上去，Training accurancy (train_acc)卻可以到將近 1，很明顯 overfitting，因此我們在音訊、選項的 LSTM 後都加上 dropout(0.25)，再試一次:

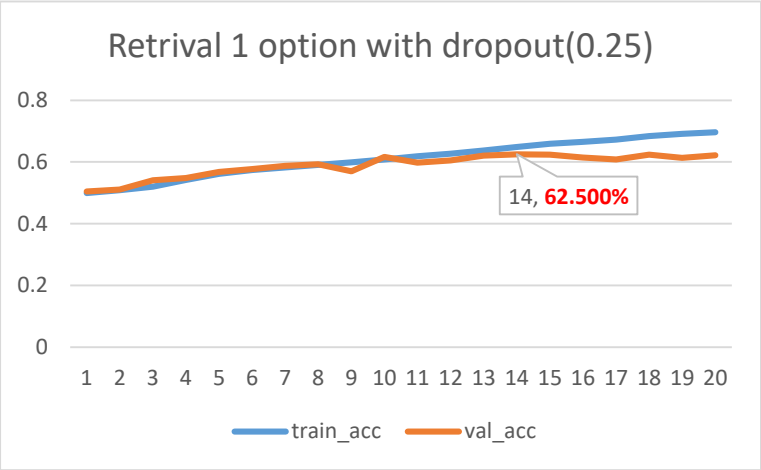


val_acc 上升至 37.162%，上傳結果 **37.3%**，相當符合預期。但是，之後無論如何調整 LSTM、Dense、Dropout、Activation，或將 dot 換成數層的 DNN，都沒能改進。

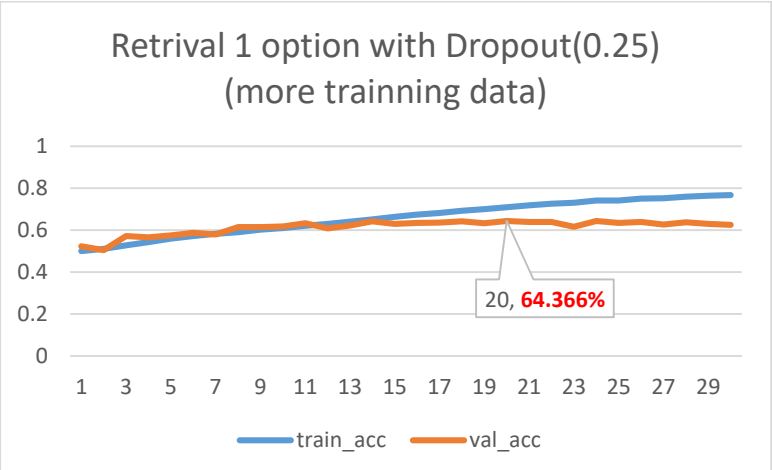
於是我們決定改另一種方式，每次只輸入一個句子，輸出就是「對」的機率(0~1)，**在訓練時每個聲音檔只給一次正確、一次錯誤選項，預測時一樣用 argmax 來判斷哪個機率最高做為答案**，其餘架構都不變。**注意這種方式 acc 會從 0.5 左右開始，因為一筆聲音只對應兩筆資料**，實際訓練結果如下:



在第二圈時達到 59.17%，而後不會上升了，我們大膽的以這個結果 predict 上傳，得到 41.8%，相當意外。而一樣觀察到後面有 overfitting 的現象，試著加上 Dropout(0.6)，但無法上升，懷疑抽取過多，而後又調成 Dropout(0.25)，結果稍稍上升：



val_acc 上升至 62.5%，但上傳卻只有 41.5%，甚至不如上個 model。**我們懷疑是 initial state 不太好，所以重新訓練一次，並將 trainning data 量增加 500 筆**，訓練的結果：



val_acc 能達到 64.366%，上傳為 **44.3%**，是目前的最佳紀錄。

Retrival model 總整理

model	Best epoch	train_acc	val_acc	kaggle
4 options (different length)	8	43.39%	45.09%	22.80%
4 options (same length) + ~~dot~~→DNN	8	34.94%	27.99%	不值得浪費
4 options (same length)	10	59.92%	32.53%	32.90%
4 options (same length) + dropout(0.25)	17	50.93%	37.16%	37.30%
1 option (same length)	2	61.73%	59.17%	41.80%
1 option (same length) + dropout(0.25)	14	64.84%	62.50%	41.50%
1 option (same length) + dropout(0.25) with more 500 training data	20	70.89%	64.36%	44.30%

以上 LSTM 皆為 256、Dence 皆為 128，有試著調整過這兩個參數，但沒有好成果，在此不附上。

- Discussion

  Q :為何讓模型直接輸出四個選項各為答案的機率，表現會不如讓模型只預測單一輸入是答案的機率，手動預測四次後再比較?

  ➢ ANS

  我們剛開始認為，也許這樣更能在模型中學到好的結果，但是在四個選項的模型中，選項的神經網路都是共用的，並不是一個選項對應一組 LSTM，照理說兩種模型訓練效果應該要完全一樣。

  其中唯一的差別，就在於我們餵的資料，**後者只餵一正確一錯誤，前者因為 LSTM 共用，相當於會拿到 25%要輸出「1」和 75%要輸出「0」的資料**，或許因此「1」的選項較難上升。舉例來說，如果讓四筆資料都輸出 0，loss 也會下降很多，機器就以為學好了，實際上會學錯方向。

  我們也有觀察到，四個選項的模型最高的機率的確沒有很顯著，例如[25% 26% 19% **30%**]，而一次只預測一個的則就比較明顯，例如[30% 45% 25% **61%**]。

  因此，我們認為**關鍵點就在於訓練資料的平不平均。越是不平均的資料，越難 fitting 出好 validation 的結果，較容易開始 overfitting。**