Machine Learning Final Project：
Listen and Translate

NTU_r06725015_成為 ML 大師

李尚恩，江孟軒，張葳凡，陳怡文

1. Introduction

近年來語音翻譯的方法有許多種，包含先讀入音訊後，再翻譯成該國語言的文字，接著再做詞義的翻譯，然而各國也會有一些方言但是所使用的文字是一樣的，像是中國的方言就有很多種：廣東話、台語、客家語等等，因此語音的翻譯問題仍有翻譯不同方言至文字的需求。

此次報告我們選擇台語語音翻譯做為我們的題目，做報告的過程中我們嘗試使用 Seq2Seq Model、Seq2Seq + Attention Model、及 Retrieval Model 來翻譯出最符合語音內容的翻譯選項，Seq2Seq Model 的最為簡單，雖然有過 Kaggle 的 simple baseline 但翻譯出來效果並沒有很理想。再來，我們嘗試了使用 Pytorch 實作 Seq2Seq + Attention Model ，但我們發現此模型在訓練資料中的部分音訊可以翻譯的很準確，不過也有預測出來語句邏輯不清的句子，而在預測測試資料時，其翻譯結果不容易看出哪個是正確答案，考量到此份報告的時間，所以我們並沒有再繼續調整 Attention Mode。最後我們嘗試 Retrieval Model，由於此報告的評分方式是從選擇題項中選出最適合的答案，我們發現使用 Retrieval Model 的架構能夠讓這項任務做得最好。此外我們也針對 Retrieval Model 與訓練資料做了一些調整，包含 down-sampling、不使用 pretrained word-embedding 與 negative sampling 等，最後再透過 ensemble 的方式提升模型準確度。此份報告將著重在 Retrieval Model 的部分。

2. Data Preprocessing/Feature Engineering

此 task 的 training 資料集中提供了音訊檔與其對應的翻譯 (caption)，在資料預處理的部分，我們對不同的模型架構做了不同的處理，以下為每個模型的 input：

(1) Seq2seq Model：

1. Training/Testing 音訊 input：作為 encoder 的 input
2. Embedding matrix：作為 decoder embedding layer 的 weight
3. Decoder data (input/target)：作為 decoder 的 input 及 output

(2) Retrieval Model：

1. Training pairs (positive/negative)
    a. Training/Testing 音訊 input
    b. Sentence input : 給定的音訊翻譯
    c. Label : 代表給定的音訊翻譯是否為真，1 (正確的翻譯) / 0 (錯誤的翻譯)

接下來將分別說明每個模型在音訊前處理與文字前處理的方法。

2.1 Audio Preprocessing

在音訊前處理的部分，Seq2Seq Model 與 Retrieval Model 所使用的資料是相同的，因此我們在這裡統一作說明，處理方法敘述如下：

(1) 標準化：將讀入的音訊檔整合成一個 array ，計算音訊檔的 39 個維度中每一個維度的平均與標準差，將原本的值減去平均後除以標準差，得到標準化後的結果。 我們測試後發現有做標準化的預測能力確實比沒有做標準化的更好一些。

(2) Padding：每一筆音訊檔的維度是 (sequence_length, 39) ，sequence_length 表示音訊有幾個 frame，我們希望使 sequence_length 長度相同，因此把每個音訊檔的最後補上數個維度為 39 ，且資料值都是 0 的 frame，使其長度等於最長的音訊長度 246 。

(3) Down Sampling：我們取出每一筆音訊資料的偶數 index 來增加音訊的 training data，最後一樣將音訊的長度 pad 成最長的音訊 246。

2.2 Text Preprocessing

在文字前處理的部分，兩種模型的差異較大，因此以下我們分別進行說明：

(1) Seq2Seq Model：

1. Embedding Matrix：
    a. 讀進 Training caption 後，在每一句訓練句子的句首加上 <sos> tag 代表該句子的開頭，在句尾加上 <eos> tag 代表該句子結束。用這些句子來建立 Word2Vec 模型，embedding 的 dimension 設為 300。
    b. 我們使用 Facebook pre-trained 的 word vector[1] 替換掉 Word2Vec 模型中的 word embedding，來建立我們的 embedding matrix。另外在 embedding matrix 中也額外加入 <pad> 與 <unk> 的 tag 用來代表 padding 和未知的 token 。
2. Decoder Data (input/target)：
    a. input 的部分利用 Word2Vec 模型將句子轉換為 Vocab ID sequences，並在句尾（ <eos> ）後補上 <pad> 在 Word2Vec 字典中的 index，最後 padding 成一樣的長度 14。
    b. target 的部分使用 np_utils.to_categorical 將 ID sequences 轉換成 one-hot vector (大小為 vocab_size)，以作為 Seq2Seq 的 output label。

(2) Retrieval Model：

1. Sentence input 的部分，我們先讀進 Training caption 建立 Word2Vec 模型，單純使用此模型的 Vocab list 作為我們的字典，並無使用其 word embedding。然後用此字典將 caption 轉換成 Vocab ID sequences。

[1] facebook fastText https://github.com/facebookresearch/fastText/blob/master/pretrained-vectors.md

2. Training pairs 的部分，我們隨機抽取與正確句子字數相同的錯誤句子，並讓正確與錯誤句子的比例為 1：5，以下為示意圖。

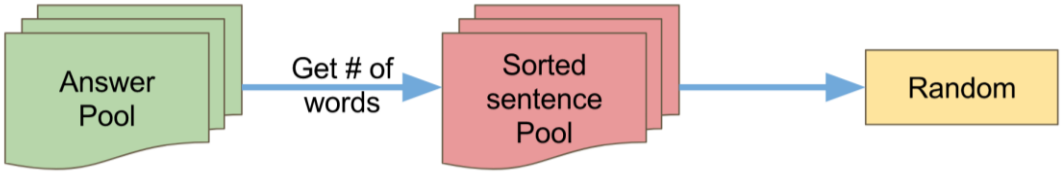

圖 1：Training Data 中錯誤翻譯的抽選示意圖

3. Model Description

在模型的部分，因為考慮到篇幅與實作的完整程度，我們以下僅針對 Seq2Seq 與 Retrieval Model 分別作說明：

3.1 Seq2Seq Model

首先先說明 Encoder 的部分，Encoder 的 input 為前處理完的音訊檔，因為其長度較長 (246 遠大於 decoder input 的 14)，因此先對其做了 Convolution 與 MaxPooling 的處理，這樣的處理也有助於增加我們訓練的效率。即透過不同大小的 filter 去學習出不同的 feature maps ，每一個 filter 為一個 $h * d$ 的矩陣，其中 h 為 filter size (一次看的字數)，d 為 300。Convolution 定義為下列式子：

$$c_i = f\left(\sum_{j,k} w_{j,k}\left(X_{[i:i+h-1]}\right)_{j,k} + b\right)$$

其中 b 為 bias, 然後 f 為一個 non-linear 的 function，Activation 函數則使用 SeLU。在通過 Convolution 後，再將其通過一層 LSTM 以取得 Encoder 的 states。以上即為 Encoder 的架構說明。

在 Decoder 的部分，先將前處理完的句子切成 <sos> + 句子 (decoder input) 與 句子 + <eos> (decoder target)，作為 Decoder 的輸入以及最終的輸出結果。Decoder input 會先經過一層 Embedding Layer，其 weights 即為前處理所產生的 embedding matrix。此時整個句子可以看成一個 $s * d$ 的矩陣，s 為 14, d 為 300。接著再通過一層 LSTM，其初始的 state 設為 Encoder 所產生的 states ，而其 input 則為通過 Embedding 後的句子。最後通過一層 Dense，Activation 函數設為 softmax，output 即為 one-hot 的向量，代表預測下一個字的機率分佈，下圖為模型架構圖：

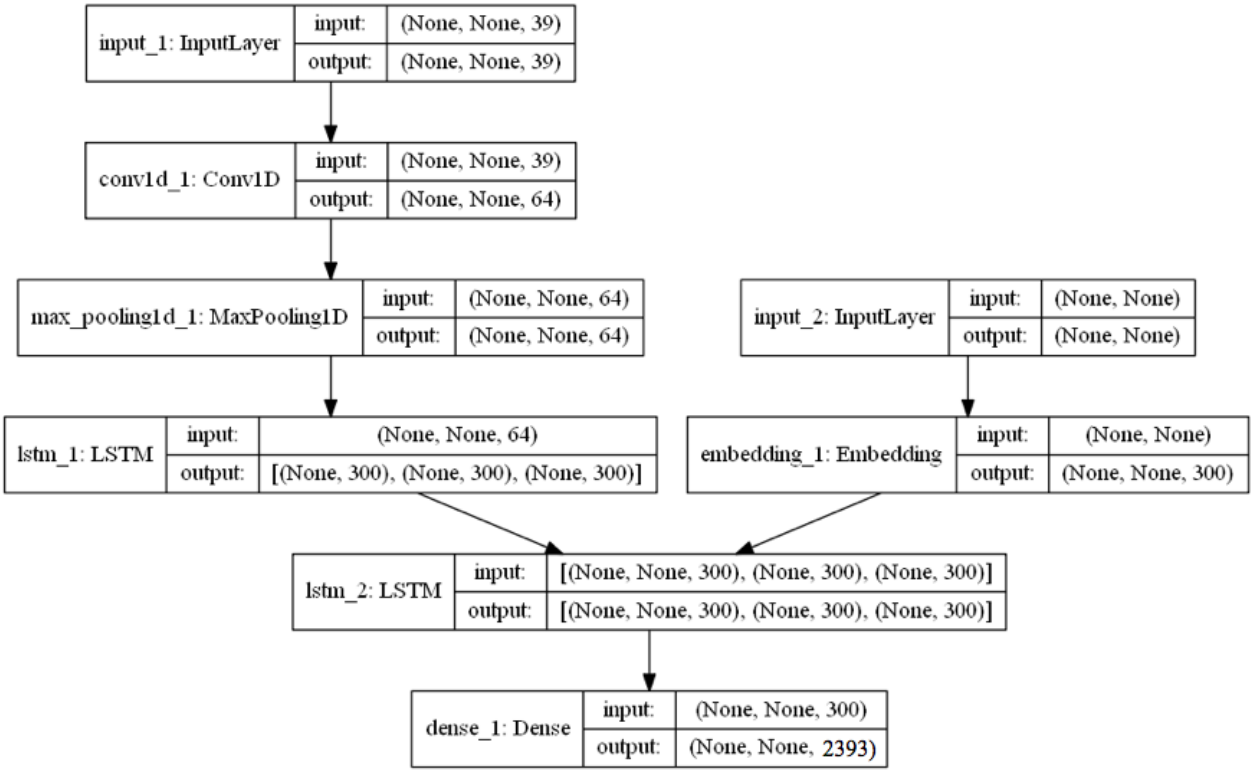

圖 2：Seq2Seq 模型架構圖

在預測的部分，我們是將四個選項分別進行預測計算分數。我們將音訊檔與 <sos> 分別丟入模型中的 Encoder 與 Decoder 中做預測，產生出第一個字的機率分布，將這個機率分布乘上該選項的第一個字，得到該選項第一個字的機率，將此機率加到該選項的分數，接著將該選項的第一個字再丟入模型做預測，以此類推，直到跑完選項所有的字。最後看哪個選項的加總分數 (機率) 最高，則認定此選項為答案。

3.3 Retrieval Model

我們參考了 Ryan Lowe el at.(2016) 的模型，input 分別是音訊與對應的翻譯。翻譯的部分會先通過 Embedding Layer 其 weights 會跟著模型一起訓練，音訊檔的部分一樣會先過 Convolution 與 MaxPooling 以加速訓練效率。接著將音訊及翻譯都通過 Bidirectional GRU，再分別把音訊與翻譯的 Bidirectional GRU states 合併，最後透過下列式子得到是否為正確翻譯的機率：

$$p(\text{flag} = 1|c, r, M) = \sigma(c^T Mr + b)$$

其中 b 為 bias，M 為一個 $d * d$ 的矩陣(d 為 GRU 的維度)，c 為音訊通過 GRU 合併後的 states，r 為翻譯通過 GRU 合併後的 states。將翻譯的 states 通過一層 Dense 後和音訊的 states 做內積，最後通過 Sigmoid 函數得到機率。

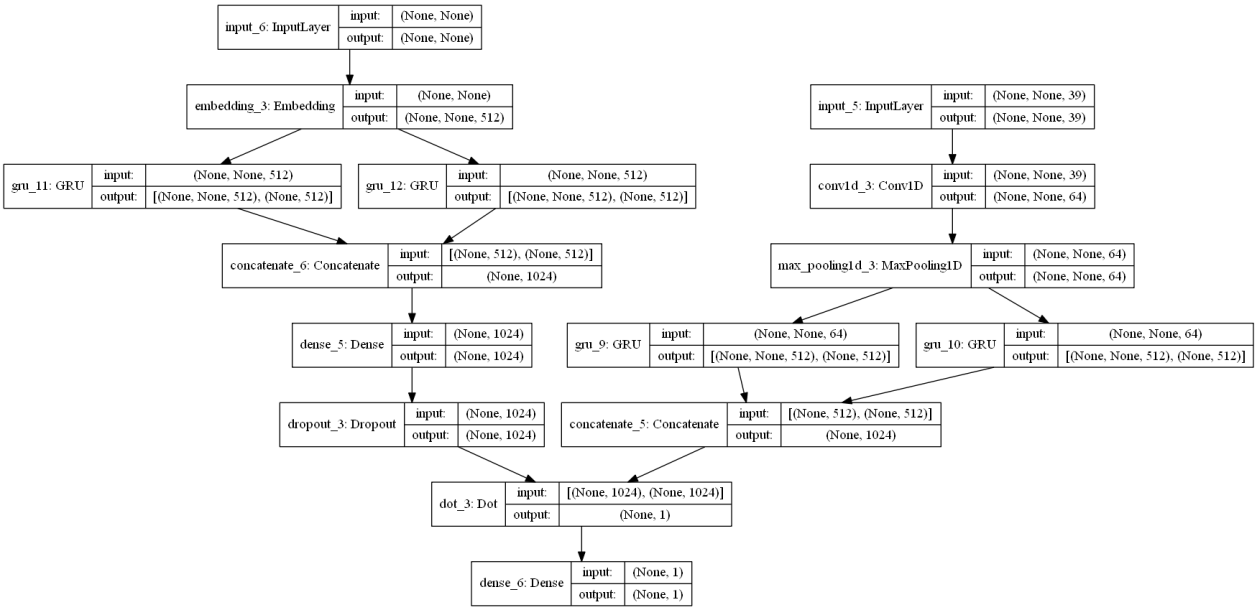

圖 3：Retrieval 模型架構圖

預測的部分是分別把 4 個選項與對應音訊配對後丟入模型中，將模型預測出來最高分的選項作為我們的答案。

4. Experiment Results and Discussion

在這次報告中我們嘗試了許多不同的方法來改善現有模型的表現，以下將會針對我們嘗試與改善的部分作討論。

4.1 使用 Pretrained Embedding 之影響

我們想要探討使用 pretrained 的 embedding matrix 對模型預測結果的影響。而在此實驗我們是使用 Facebook 所提供的 word embedding 來製作 embedding matrix，比較有無使用 pretrained embedding matrix 的準確率。GRU 的維度統一為 512，而正確/錯誤配對比例則是統一為 1:1，結果如下：

	有使用	無使用
Kaggle Accuracy	0.518	0.71

由上表結果可以發現，沒有使用 pretrained embedding matrix 的表現會比較好。我們推論可能是因為 Facebook 所提供的 word embedding 是考慮該 term 在句子中的文意產生出的特徵向量，而在此模型中我們要的是單一個字的特徵向量，因此會變成如：可以，在訓練模型時會被切成可、以，而分別取出可跟以的特徵向量，但是這樣產生出來的結果，就失去原本句子的意思了。再來我們的目的是要翻譯音訊，或許我們需要的特徵向量並不是意思上的特徵，而是音訊上的特徵，也就是根據每個字發音不同所產生的特徵。

4.2 正確/錯誤配對比例不同之預測結果

Retrieval Model 在訓練時採用不同比例的訓練資料，且考慮字數長度，其訓練資料的答案比例與預測的結果正確率如下：

資料比例	1:1	1:3	1:5
Kaggle Accuracy	0.71	0.782	0.786

一開始嘗試挑選與正確翻譯數量相同的錯誤翻譯 (1:1)，但後來發現提高錯誤翻譯的比例至 1:3（和測試選項的比例相同 ）可以提高約 0.1 的準確率。由於比例的上升很明顯，於是我們嘗試了比測試四選一的錯誤選項比例更高的 1:5 ，發現正確率又提高。我們推測是因為提高錯誤選項的比例除了增加了訓練的資料量，也可以讓模型判斷錯誤翻譯的能力提高。

4.3 不同 GRU Dimension 之影響

下表為我們嘗試不同 GRU dimension 的結果，在其他參數與處理都不變的情況下，我們發現當維度設為 512 的時候 Kaggle 的準確率最高。在訓練模型時，我們也曾嘗試維度為 1024 的 GRU，但在訓練時的 training loss 就變糟了，因此就沒有再放到 Kaggle 上做預測。由此我們可以發現，雖然 dimension 設得越高讓模型越複雜，某種程度可以有效提升我們的準確率，但過度複雜的模型反而也得不到太好的效果，有可能是因為隨著模型變複雜，其變異程度變得太大了，導致預測結果不佳。

Dimension	128	256	512
Kaggle Accuracy	0.543	0.557	0.678

4.4 Ensemble 之影響

Ensemble 是一個非常簡單又有效的方式提升模型的準確率。在此次報告中我們是透過改動每一次隨機產生 negative samples 的 seed，來產生不同的訓練資料，進而訓練出不同的模型。在預測時將每一個模型預測的 output 加總後取最大的選項作為我們的答案。透過這樣的方式我們等於變向生出很多不同的測試資料，讓每個模型訓練出來的結果都有些微的不同，進而提升整體預測的準確率，而下表是我們比較完全沒有 ensemble 的模型與 ensemble 5 個模型的結果：

	無 ensemble	有 ensemble
Kaggle Accuracy	0.825	0.868

4.5 Seq2seq v.s. Retrieval

最後我們想要探討為什麼 Retrieval 模型普遍在這個題目上的表現會比 Seq2Seq 模型來得好。我們認為可能是因為此題目是一個選擇題，因此對於 Retrieval 模型這種分類的模型較簡單，其只需要判斷音訊與對應的翻譯句子的相似程度，但是對於 Seq2Seq 模型來說，它是聽完音訊之後產生出一個翻譯出來的句子，這個難度比起判斷相似程度難上許多。再者，Seq2Seq 模型並無法直接用來產生此題目的答案，需要再透過 Cosine Similarity、Euclidean Distance 或一些文本比較的技巧，來找出哪個選項的句子與 Seq2Seq 翻譯出的句子最相像，而這一個步驟又可能會再產生誤差，導致預測的結果不理想。

Reference

[1] The Ubuntu Dialogue Corpus: A Large Dataset for Research in Unstructured Multi-Turn Dialogue Systems Ryan Lowe el at.(2016) https://arxiv.org/pdf/1506.08909.pdf

[2] Attention-Based Models for Speech Recognition Jan Chorowski el at.(2015) https://arxiv.org/abs/1506.07503

[3] A ten-minute introduction to sequence-to-sequence learning in Keras Francois Chollet (2017) https://blog.keras.io/a-ten-minute-introduction-to-sequence-to-sequence-learning-in-keras.html

Work Division

R06725015 李尚恩　書面製作與統整、各模型實作、上台報告

R06725019 江孟軒　書面製作、各模型實作、GitHub 檔案整理與上傳

R05725033 陳怡文　書面製作、各模型測試、投影片製作、文獻探討

B02507032 張葳凡　書面製作、各模型測試、投影片製作、文獻探討