Machine Learning Final Project

Topic : TV conversation (chat-bot)

1. Team name, members and your work division

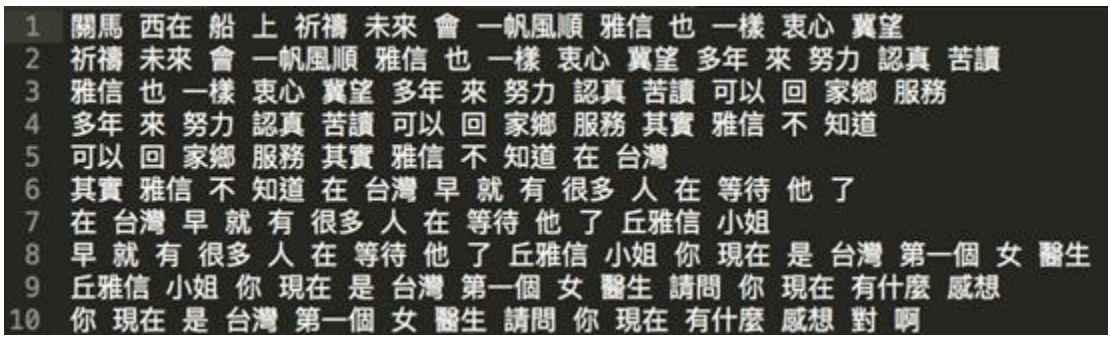
Team Name : NTU_r06946009_Bang		
Name	**Student ID**	**Assignment**
林庭宇	R06946009	設計與實作 model 1
呂承翰	R06946008	設計與實作 model 2
黃永翰	R06946015	資料前處理

2. Preprocessing/Feature Engineering

Training Data

首先將所有的 training data 合成一個檔案，再使用 jieba 來進行斷詞，並且用繁體中文版的字典，藉此達到更好的效果，斷詞後的結果如下圖所示：

斷詞完成之後，我們將每三句 (有 overlap)拼成一個句子，組合成新的 training data，如下圖所示：



最後我們使用 gensim 來訓練一個 word2vec 的模型，參數採用 skip-gram、設定 vector 的 size 為 64 維、min_count 為 0，並將 window 設為 10。

Testing data

與前述的 training data 相同，我們將 testing data 使用 jieba 斷詞，並且將每個選項拆開存進 list 中，再進行後續的 predict 動作。

3. Model Description

First Model

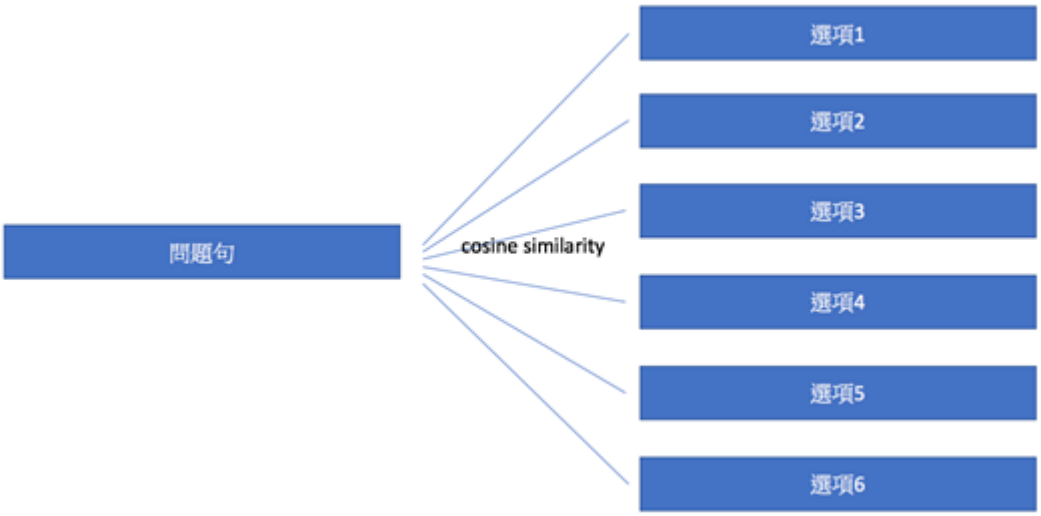
使用訓練完成的 word embedding matrix 將 testing data 中的問題與選項都轉換成 64 維的 vector 後，計算問題句裡面各單詞與每個選項中各單詞的 cosine similarity 並且取絕對值

在此舉一個例子，如果問句當中包含 6 個 vector，選項 1 有 5 個 vector，就要計算 30 個 cosine similarity 值，如下圖所示)，最後加總再進行比較，選擇最大的作為答案。

Second Model

使用訓練完成的 word embedding matrix 將 testing data 中的問題與選項都轉換成 64 維的 vector 後，將每個句子裡面的單詞向量加總後平均，以作為代表該句子的向量，

之後我們同樣會比較問題句向量與各選項句向量的 cosine similarity 數值，取出最大的作為答案，如下圖所示：



4. Experiments and Discussion

Training word2vec with/without overlap

我們分別使用兩種 training data 來訓練 word2vec 的模型，第一種就如同 Preprocessing/Feature Engineering 一節所述，將三個句子拼湊成一句，並且有 overlap，來組成新的 data，而第二種則是使用無 overlap 的資料進行，如下圖所示：

```
2  祈禱 未來 會 一帆風順 雅信 也 一樣 衷心 冀望 多 年來 努力 認真 苦讀
3  可以 回家 鄉 服務 其實 雅信 不 知道 在 台灣
4  早就 有 很多 人 在 等待 他 了 丘雅信 小姐 你 現在 是 台灣 第一個 女醫生
5  請問 你 現在 有 什麼 感想 對 啊 你 讀 的 是 名校
6  是 目前 日本 最 有名 的 醫科大學 請問 你 對 未來 的 規劃 是 什麼 你 一個 人 在 日本
7  會 不會 覺得 很 辛苦 對 啊 你 在 日本 會 不會 很 辛苦
8  你 穿 的 這件 衣服 是 日本 最 流行 的 嗎 請 你們 注意
9  女醫生 也 是 醫生 一個 醫生 最 重要 的 就是 他 的 技術
10 專業 的 技術 就是 這 幾年 我 在 日本 學習 的 重點 是 誰 跟 你們 說
```

以下為兩種方式在 kaggle 上的表現：

	Public score
With overlap	0.50671
Without overlap	0.46916

由此可以發現，使用有 overlap 的資料訓練 word2vec model 可以相當有效的提升判斷正確率。

Seq2Seq

除了做資料前處理和調整 word2vec 的參數外，我們還有嘗試使用 Seq2Seq 的方法，在做過一些調查後發現 github 上有一個別人寫好的套件 farizrahman4u/seq2seq，並且此套件是基於 keras 來實作的，但是這個套件除了不太好安裝外，使用上也不是很方便，當要訓練 model 的時候，需要把所有訓練資料中的 X 與 Y，分別變成一個（samples , input_length , input_dim）與(samples , output_length , output_dim）的三維結構，也因此非常吃記憶體，在 Word2Vec 的 Size 設 200 的前提下，將訓練資料轉成三維結構就耗盡了我實驗室 Server 96.6% 的記憶體（ 實驗室 Server 共 32G 記憶體 ）。

```
  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
 9971 mirlab    20   0 56.561g 0.030t 107348 S   4.3 96.6   7:03.70 python
 5682 mirlab    20   0 1383336  57032  20708 S   0.7  0.2   3:35.27 compiz
    7 root      20   0       0      0      0 S   0.3  0.0  29:14.89 rcu_sched
   97 root      39  19       0      0      0 S   0.3  0.0   3:21.42 khugepaged
```

在訓練完 Seq2Seq 的 model 後，卻發現不管丟進去的測試資料是什麼，最後 predict 出來的結果都一樣，之後做了各種不同的嘗試，包含將<GO>Token 拿掉，調整 padding 的方向(pre 與 post)，使用不同的 Seq2Seq 結構(SimpleSeq2Seq、Seq2Seq、AttentionSeq2Seq)，調整結構的參數（depth、peek），最後還是不管丟進去的測試資料是什麼，predict 出來的結果都一樣。

```
In [27]: # seq2seq
         generated_seq = model.predict(xx_test)
         for i in range(10):
             seq = ""
             for i in range(16):
                 seq += word2vec_model.most_similar(positive=[generated_seq[0][i]],topn=1)
             print(seq)

我<EOL><EOL><EOL><EOL><EOL><EOL><EOL><EOL><EOL><EOL><EOL><EOL><EOL><EOL><EOL>
我<EOL><EOL><EOL><EOL><EOL><EOL><EOL><EOL><EOL><EOL><EOL><EOL><EOL><EOL><EOL>
我<EOL><EOL><EOL><EOL><EOL><EOL><EOL><EOL><EOL><EOL><EOL><EOL><EOL><EOL><EOL>
我<EOL><EOL><EOL><EOL><EOL><EOL><EOL><EOL><EOL><EOL><EOL><EOL><EOL><EOL><EOL>
我<EOL><EOL><EOL><EOL><EOL><EOL><EOL><EOL><EOL><EOL><EOL><EOL><EOL><EOL><EOL>
我<EOL><EOL><EOL><EOL><EOL><EOL><EOL><EOL><EOL><EOL><EOL><EOL><EOL><EOL><EOL>
我<EOL><EOL><EOL><EOL><EOL><EOL><EOL><EOL><EOL><EOL><EOL><EOL><EOL><EOL><EOL>
我<EOL><EOL><EOL><EOL><EOL><EOL><EOL><EOL><EOL><EOL><EOL><EOL><EOL><EOL><EOL>
我<EOL><EOL><EOL><EOL><EOL><EOL><EOL><EOL><EOL><EOL><EOL><EOL><EOL><EOL><EOL>
我<EOL><EOL><EOL><EOL><EOL><EOL><EOL><EOL><EOL><EOL><EOL><EOL><EOL><EOL><EOL>
```

經過調查後我們發現，在此套件 Github 網頁提供的範例連結中，他最後得到的結果也是不論丟進去的測試資料為何，predict 出來的結果都會是一樣的。

Results

No good results were achieved so far:

```
[why ?] -> [i ' . . $$$ . $$$ $$$ $$$ $$$ as as as as i i]
[who ?] -> [i ' . . $$$ . $$$ $$$ $$$ $$$ as as as as i i]
[yeah ?] -> [i ' . . $$$ . $$$ $$$ $$$ $$$ as as as as i i]
[what is it ?] -> [i ' . . $$$ . $$$ $$$ $$$ $$$ as as as as as i]
[why not ?] -> [i ' . . $$$ . $$$ $$$ $$$ $$$ as as as as i i]
[really ?] -> [i ' . . $$$ . $$$ $$$ $$$ $$$ as as as as i i]
```

在 2018/1/19 的 ML 發表會上，組別 NTU_r05525066_kaggle 分身 2 也得出了跟我們一樣的結論，因此我們能夠確定 Seq2Seq 套件對於這一次的題目而言並不合適。

Ensemble

透過 ensemble，我們能夠將數個 model 預測出來的結果進行綜合考慮，來達到提升準確率的效果。在這個題目中，我們一開始採用了投票的方式來進行 ensemble，然而這個方式下會遇到許多答案是相同票數的問題。因此採取了另一個方法，將每一個選項的 cosine similarity 進行相加。

最終，我們以相同參數訓練 4 個 word2vec model，以前述 model2 的方法計算各個選項與問句的 cosine similarity，將四個 model 的結果相加，取最大值作為答案。

以下為 kaggle 上的分數：

	Public score
Without ensemble	0.50671
With ensemble	0.51024

由此可見，如果將 4 個 model 做 ensemble，可以些微提升預測的準確率，提升的幅度約在 0.03 ~ 0.04%左右。

後續我們又進行了只用 2 個及 10 個 word2vec model 的 ensemble，kaggle 上的表現比較如下圖所示：

	Public score
2 model ensemble	0.50830
10 model ensemble	0.50790

從上述的圖表中，我們發現這兩種的表現都有比單一 model 進步。然而 10 個 model 進行 ensemble 的結果略差，經過一番討論，我們得到了可能的原因是 Public 和 Private 中的 data 可能有偏差，有可能在 private 的 data 中，10 model ensemble 會有更好的表現。