Machine Learning Final Project

- Conversations in TV shows -

Team Name: DeepQueueing

Team Members and Work Division:

	學籍學號	姓名	分工
隊長	資工三 B04902089	林政豪	
隊員	資工三 B04902099	黃嵩仁	
隊員	資工三 B04902021	陳弘梵	
隊員	資工三 B04902090	施長元	

Preprocess / Feature Engineering

(1) 我們將五份 training data 讀入，經過 jieba 套件對每句話進行分詞切割。分詞時，我們嘗試過採用或不用 stopwords；雖然普遍認為 stopwords 可以幫忙去除一些多餘的無意義單詞；然而實作後，發現採用後反而會讓訓練出來的 word2vec 效果較差；但如果只去掉標點符號可以使效果稍稍變好，但差異不大。有可能是 gemsim 的隨機性導致這樣的結果。

(2) 接下來，我們要準備即將放入 word2vec 當中進行訓練的 list，shape 為：(N, M)，其中 N 為 data 總共有 N 句話，M 代表一句被 jieba 切為 M 個詞。

(3) 放入 gensim.models 的 Word2Vec 函式當中，並花了許多時間調整其中的參數：

a) vector_size = 32~128，透過 word2vec 訓練之後，產生的每個單詞所擁有的向量維度大小，越多的訓練資料可以讓更高維的向量表現越佳，我們總共有 757000 句，因此考慮調設較大的訓練維度。

b) window_size = 3，代表每一句話訓練時，每個單詞會考慮前面與後面各多少的單詞，即是 N-gram 的概念。而考慮到 jieba 的分詞(甚至有 stopwords)，我們雖然將三句話接在一起，還是使用了比 default 要少的 window size。

c) negative_size = 3，代表訓練時設置的 noise word 個數，我們調設的比 default 要小一些，我們認為這樣雜訊較少，並且希望能夠留下更多的詞句內容。

d) min_count = 0~4，可以過濾掉出現次數太少的單詞。

e) iteration = 15~50，對句子訓練的 epoch，我們將它調高希望他能夠更 fit data。

f) alpha=0.025，即為訓練的初始 learning rate，經過測試，調整這項參數不太容易讓我們訓練的結果更進步。

g) sg=1，我們採用的則是 skip-gram，這個模型是以中間的字為 input，周圍為 label 進行訓練，而 cbow(sg=0)，則是以中間的字為 label，周圍的字為 input。我們實作後，發現 skip-gram 效果好上許多。

Model Description

Gemsim Part:

(1) 如上所述，將每句話用 jieba 斷好詞，丟進 gemsim train word2vec，此時使用 cbow。

(2) 除了使用 skip gram 以外，其餘皆與(1)相同。

(3) Word2vec 時將五份每份有 N 句的 data，依序依照以下的方式合併，存入陣列中：

第一句
第一句+第二句
第一句+第二句+第三句
...
第 N-2 句+第 N-1 句+第 N 句
由於題目常常是以兩句的形式出現，回答的選項再有一句，因此我們使用這樣的預處理，讓他訓練 word2vec 的時候，能夠盡量貼近 testing 時問答的情形。

(4) 將 training data 全數使用套件 SnowNLP 轉成簡體中文之後，不另使用字典或 stopword，使用 jieba 斷好詞，丟進 gemsim train word2vec。

(5) 使用上面(1)、(2)所 train 出的 word2vec 重新組合句子，將句中的每個詞的向量加起來平均，如果前一句和後一句的 cosine similarity>0.95 就併為同一句，否則就把後一句當作新句子的開頭，training 時因為句子有時會很長(長度>100)，所以 window 使用 100。

(6) 發現 training data 中有一些句子和上一句完全相同，所以讀輸入時會檢查是不是和上一句完全相同；如果相同就丟掉。

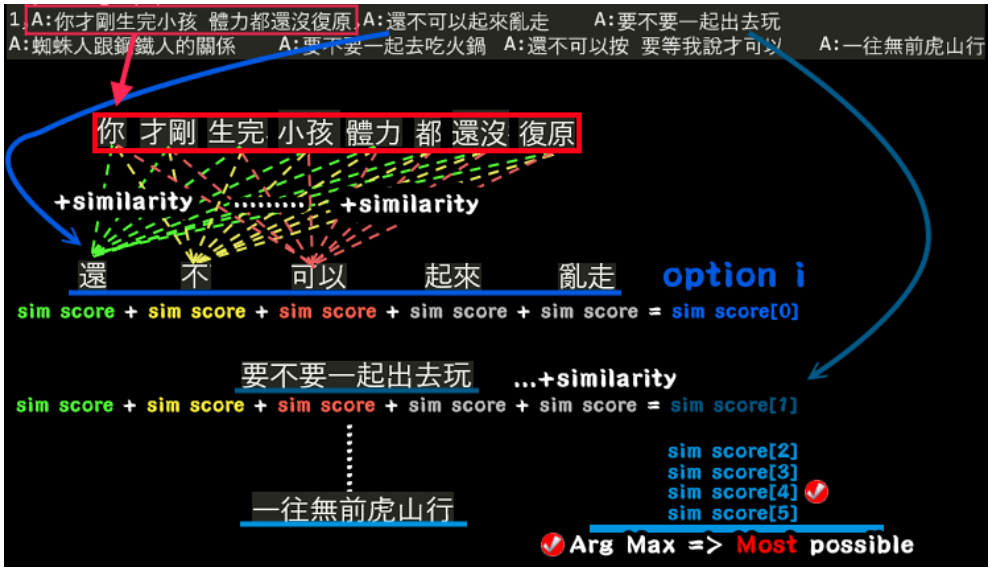
(7) 在禮拜五的分享會中，得知去掉標點符號會比較好，所以使用只有全、半形標點符號的 stopword。

Test Part:

(1) 助教手把手

a. 將題目與選項各自讀入，並用 jieba 進行切割。
b. 對於每一項答案選項，我們將每一個單詞，與題目的單詞，個別計算出 gensim word vector 的 similarity。
c. 這個選項中出現的單詞與題幹中單詞，若 similarity 高於我們所設定的一個 threshold，則將它累加入這個選項的 sum_sim 當中。

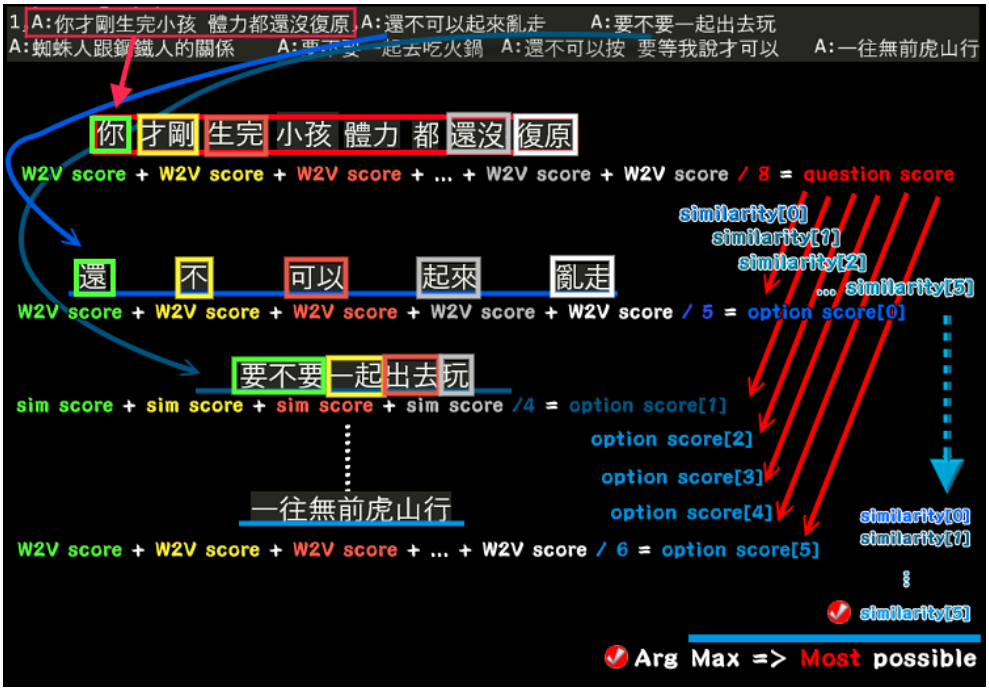
d. 如此反覆，直到六個選項的 sum_sim 都計算完成，會得到六個浮點數，分別即代表這個選項在我們 model 計算後，可能為答案的機率。



▲ 每筆 testing data 的 model 流成示意圖

(2) Cosine Similarity

將輸入的句子用 jieba 斷好詞之後，將問句和選項中的每個詞的向量加起來平均，作出六個向量，並使用 1- scipy.spatial.distance.cosine 計算相似度。



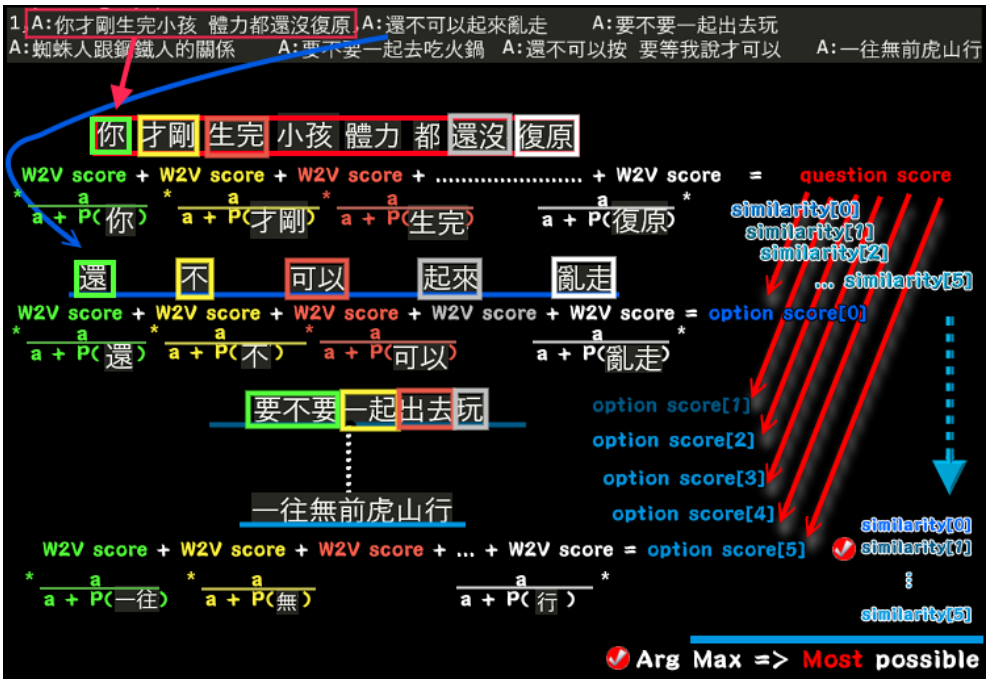
▲ 每筆 testing data 的 model 流成示意圖

註:在其餘條件相同的情況下，用 np.dot 並有除以向量長度 normalize 時結果很

差，但換成 scipy.spatial.distance.cosine 計算效果就好很多，目前還沒發現原因，可能是原本的程式有 bug。

(3) Weighted sum

這個方法是在禮拜五的分享會中看到，在將詞向量相加平均時以$\frac{\alpha}{\alpha+p(w)}$為比重相加，p(w)為$\frac{\text{該詞出現次數}}{\text{所有詞的出現次數和}}$，在這裡發現 alpha 的大小有很大的關鍵，在下一部分會說明。



▲ 每筆 testing data 的 model 流成示意圖

Experiments and Discussion:

- **Experiments：**

一開始我們實驗性的上傳了 label 都一樣的答案，發現 public 資料的分佈如下：

label	Kaggle Public Score
0	0.25770
1	0.16205
2	0.15177
3	0.15770
4	0.14505
5	0.12569

從上面的分佈，我們得知 0 選項出現的比率非常高(在 public 中已經是 5 的兩倍)，因此在我們曾經測試的 Model 當中，算出 similarity 的機率分佈之後，有著重提升 0 的出現機率，發現能讓 model 預測的效果一定程度的提高，所以我們決定以後 test 時會把 0 的分數都稍稍提升一些。

- **Testing Procedures：**
  1. 一開始決定採用 Gemsim part (1) + Test part (1)
     a. 在 Test 的部分，最先決定是，採用助教手把手提供的方法：累加每個選項對於題目的 similarity，選出最高分數的作為答案。另外我們也嘗試過，將每個選項的 word vector 先平均之後，再與題目計算 similarity，但是發現效果比較差，所以之後只採用手把手一陣子。
     b. Word2vec 的測試部分，我們發現不採用 stopwords 效果較佳，最後經過一番參數調整之後，發現 word2vec 的訓練維度較低時，比較能 fit。
     c. 單個 model 的表現一開始大概約 39%，在 tune 參數之後，可以達到 41~43%附近，在做出 21 個 ensemble 之後上船後可以達到 45%，很勉強的過了 strong baseline。
  2. 後來看了 training data 時發現，很多句子只看單句的話會語意不清
     a. 所以決定把使用 Gensim part(3) + Test part(1)，改為採用三句話列在一起訓練，單個 model 的表現一開始大約 43%。
     b. 但是這時 code 有寫錯，使得 training data 1 出現較多次。Code 斷詞錯誤的部分修正完之後，發現原本的模型表現也不差，於是將他們一起 ensemble。在 ensemble 的部分，我們寫了一個程式，在我們生成的許多預測的 csv 當中，每次如果票數相同，會隨機選出一個，最後可以達到 48.8%，但由於當時使用的 model 過多，加上有些 word2vec 檔案丟失，所以幾乎無法重現。
  3. 輾轉得知 gensim 使用 skip gram 會比 cbow 好，之後在禮拜五的分享時有同學說 cbow 是使用在 training data 超過一億個字或是檔案>500MB 時，這次的 training data 非常少，skip gram 會比較好，採用了 Gemsim Part(2+3) + Test part(1)，單個 model 上傳就可以有 48.5%，解決了我們無法重現的問題，在做出幾個 ensemble 之後可以達到 48.9%，但還不夠好。
  4. 再來聽說把 training data 轉換成簡體中文效果會比較好，畢竟 jieba 原本是使用在簡體中文上，想想覺得很有道理，所以就使用了 Gensim Part(2+3+4) + Test Part(1+使用 SnowNLP 轉換成簡體字)，單個 model 的表現與上個部分差不多，ensemble 的結果也比較差，加上會讓程式執行時間加長許多，所以之後就沒有採用。

5. 接下來突發奇想，覺得有些句子可能一句話就可以表示，有些可能需要五句話接在一起才完整，全部都用三句雖然很好，但一定也會造成一些犧牲，所以決定使用之前 train 好的 word2vec model 重新分句子，也就是 Gensim part(2+5) + Test part(1)，上傳後約 43%，遠不如 3 中的結果，所以放棄。
6. 已經不知道怎麼辦了，決定把 cosine similarity 加入 Testing 中，也就是 Gensim part(2+3) + Test part(1+2)，且 cosine similarity 的結果先除以他們的 maximum，將其變成 0~1 之間，再乘上神奇數字 8.88*1.126*1.337*5.01 當比重，加上原本的手把手的分數，上傳後發現單個就有 51.8%，雖然不明白箇中道理，但或許這也是一種 ensemble 吧，之後做了五個後 ensemble 可以達到 52%，發現了新世界。
7. 在週五分享時有聽到可以在 cosine similarity 中加上 weighted sum，且聽說 alpha 很重要且他們提示了 alpha 大約在(1e-3,1e-7)，如果選對了正確率會起飛，反之會很差，一開始 alpha 使用 1e-4，單個上傳只有 45%，在經過測試之後發現 alpha = 1e-3 * 1.337 時效果最好，之後再加上 Gensim part(6+7) 將標點以及重複句子去掉，單個可以達到 52~54%，將九個 ensemble 之後可以達到 55.889%！

Methods	Kaggle Public Score
No stopwords + 手把手 + cbow	單個 41~43%
No stopwords + 手把手 + 三句話 + cbow	單個 43~44%，之前寫錯所得到的和之後正確的 emsemble 後可以到 48.972%
Stopwords + 手把手 + cbow	單個 38~40%
No stopwords + 手把手 + cosine sim. + cbow	單個 43~44%
No stopwords + 手把手 + skipgram	單個 48.5%
No stopwords + 手把手 + 簡體中文 + skipgram	單個 48%
No stopwords + 手把手 + cosine sim. + skipgram + 用之前 train 好的 word2vec 分句子	單個 43%
No stopwords + 手把手 + cosine sim. + skipgram	單個 51.8%，ensemble 之後可以到 52%
No stopwords + 手把手 +cosine sim.+ weighted sum+ skipgram	單個 54%，ensemble 後可以到 55.889%

- **Seq2Seq and RNN**
  一開始我們使用 keras 實作了 seq2seq 或是直接使用 RNN 做 training，原先的構思是把 Training data 中下一句的 Vector 當作這一句 Vector 的答案，而後在 Predicting 時放入 Testing data 後，用生出的語句來與選項比對相似度。但這個方法效果奇差；Training 之後有兩種可能，一種是輸出完全牛頭不對馬嘴，生出的語句根本不知道在說什麼，可能原先的 Word2Vec 當中並不夠完善，無法用那些 Vector 完整表達語句意思。另一種是 Model 會找到一組與所有答案都相差不大的答案，不論輸入為何都會輸出那組答案；有懷疑過是不是因為句子數量太多所導致，所以改成只單獨訓練前 20 句，發現非常成功，慢慢擴大到 500 句時就開始 train 不起來，因為 training data 總共有 75 萬多句話，數量遠大於 20，所以果斷放棄此作法，開始研究 word2vec，和上述的方法。
- **Surprise(!?**
  在經過討論後發現我們一開始和”b04902011_大學長一家人”的做法幾乎相同，只有參數有些微的差距，但他們的正確率卻少我們 2%，後來經過多方比對之後發現，我們使用的 gensim 套件版本是 39.0，而他們是 32.0，在他們 pip install genism –upgrade 之後，結果就差不多了，由此經驗我們發現套件版本很重要，要記得更新。