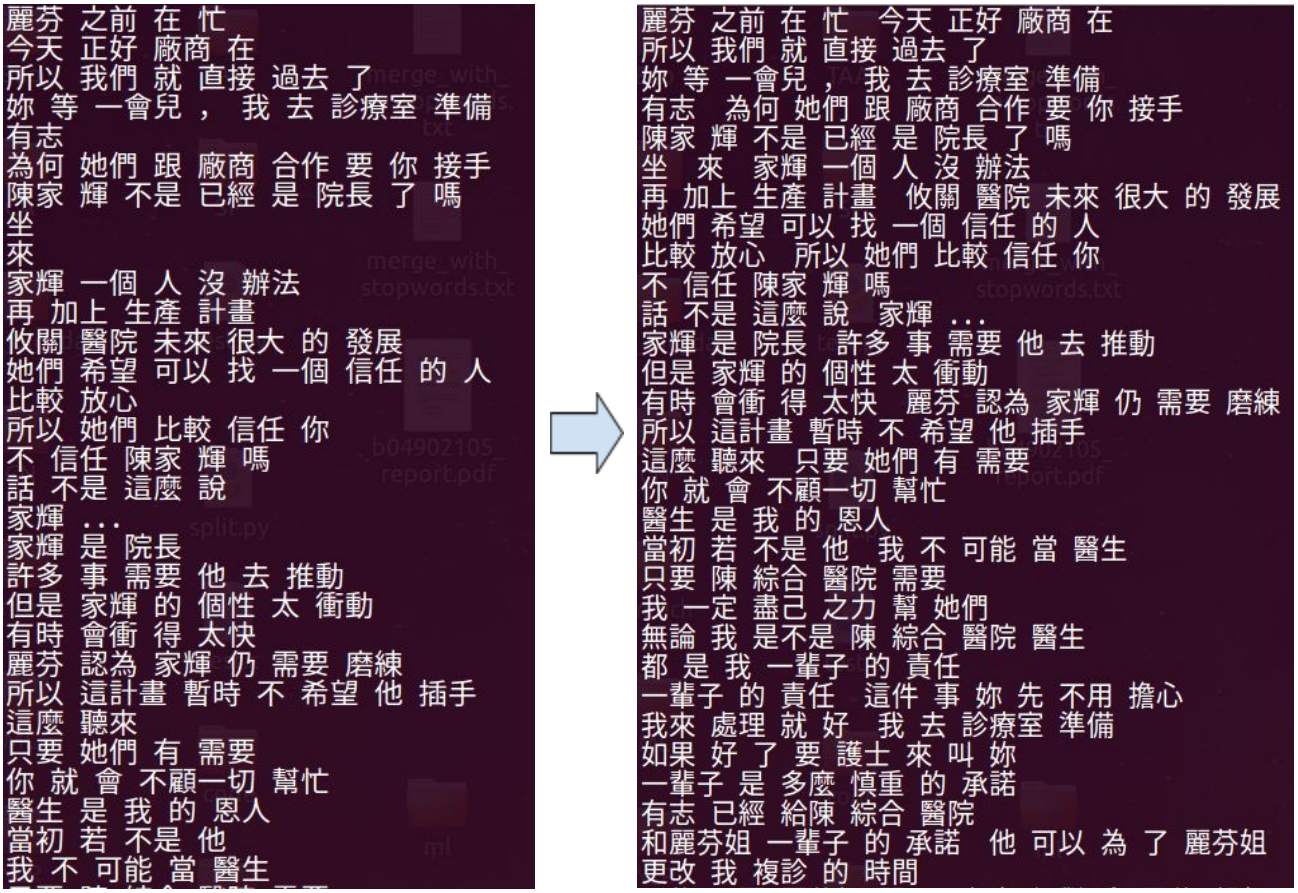
Conversations in TV shows

Team name : NTU_b04902105_TarngLaoIaNo2
Members : b04902025 施博瀚、b04902043 謝宏祺、b04902105 戴培倫
Work division :

RNN model:	謝宏祺、戴培倫
Word2Vec model:	施博瀚、謝宏祺、戴培倫
Report:	施博瀚、謝宏祺、戴培倫

Preprocessing / Feature Engineering:

1. 我們先將所有training data用jieba斷詞。
2. 使用jieba/extra_dict中的stop_words.txt做了兩種實驗。
    a. 去掉stopwords。
    b. 保留stopwords。
3. 考慮到斷詞後很多行只有少數幾個單詞，單一行沒辦法表達什麼意思，且training data為連續劇台詞，上下句多有關聯，而因為gensim Word2Vec的預設的window size為5，因此我們將全部句子上下句相連直到每一行長度皆大於等於5個單詞。



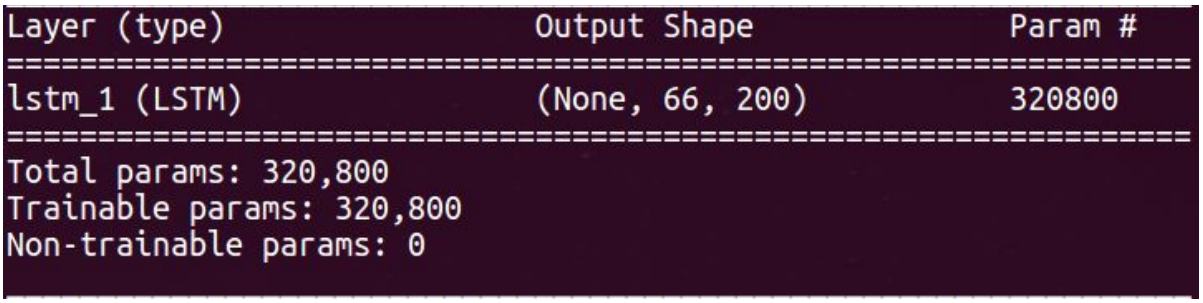
Model Description (At least two different models)

1. RNN model

   由於資料量太大，每次只將10%的data轉成word vector丟進RNN裡train。

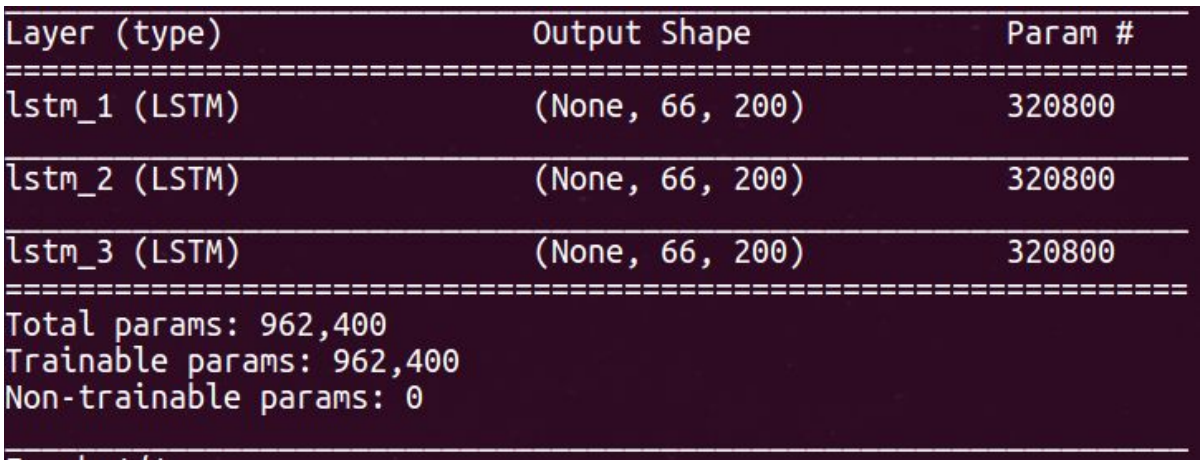
   1.1. 一層LSTM

   activation='relu'，loss='cosine proximity'，optimizer='adam'，約train 5個epoch便收斂。



   1.2. 三層LSTM

   activation='relu'，loss='cosine proximity'，optimizer='adam'，約train 5個epoch便收斂。



2. Gensim Word2Vec model

   2.1. 使用gensim的Word2Vec訓練詞向量，training data為5個劇本(1_train.txt ~ 5_train.txt) 的merge版本(將每句連接至5個詞以上)。

   2.2. 將testing data中的每個問題、選項經過 jieba 斷詞之後再使用 Word2Vec model 將其轉變成為 vector後進行比對，若是遇到不在字典裡的單字，則直接忽略它，將他轉為零向量。

   2.3. Train Word2Vec時設定iter=20。

   2.4. 比較相似度方法為將每個句子取平均，比如說一句長度為5個單詞，Word2Vec 輸出為200維，則將這5個單詞取平均，這一句即變成1個200維的向量，最後再將這200維做cosine similarity轉換，相似度最高的即為我們預測的答案。

Experiments and Discussion (8)

1. 不同的比較相似度的方法（以下實驗皆使用這兩種方法選答案）
   1.1. 使用助教手把手中的方法，將每個單詞作 Word2Vec.similarity比較之後大於threshold的加總起來，最後再來比較相似度，相似度最高的選項為答案。
   1.2. 將一個句子的每個單詞先轉為向量之後，取平均，最後再比較相似度，相似度最高的選項為答案。

2. RNN
   2.1. 用gensim的Word2Vec將training data轉成100維
   2.2. 以training data的前一句為source，下一句為target，希望能學出前後句的對應關係。
   2.3. 用1、3層的LSTM，activation='relu'，loss='cosine proximity'，optimizer='adam'，約train 5個epoch便收斂。
   2.4. 使用1.2的方法選出最相似的選項
   2.5.

Model	Kaggle Public Score
一層lstm、包含stopwords	0.26284
一層lstm、去掉stopwords	0.26640
三層lstm、包含stopwords	0.25494
三層lstm、去掉stopwords	0.26324

   2.6. 都沒辦法train出好的結果，但發現去掉stopwords成績會稍微好一點點，進步不顯著可能是因為jieba的stopwords量滿少的，只有18個。還有很多不太能代表意義的詞沒有被收錄。
   2.7. 參考了很多網路上的方法，也試過加上attention，但做出來的結果都很差，應該算是完全失敗的。因此最後轉而直接使用gensim的Word2Vec比較相似度。

3. Word2Vec CBOW
   3.1. 在最一開始時我們使用的是 CBOW 的model，Word2Vec預設的模式即為CBOW，因此我們只有將iter設為20，並調整size做實驗。

3.2. 下表為我們測試幾種不同輸出維度的 Word2Vec的結果

3.2.1. (1.1的相似度比較方法)

Dimension	Kaggle Public Score	
	With Stopwords	Without Stopwords
100	0.40988	0.40513
200	0.41422	0.40553
250		0.41264
300	0.42845	0.42648
500	0.40909	

3.2.2. (1.2的相似度比較方法)

Dimension	Kaggle Public Score	
	With Stopwords	Without Stopwords
100	0.43913	
200	0.44207	0.44129
250	0.43504	
300	0.42371	0.42648

3.3. 結果發現輸出維度大約在200的時候有最好的準確率，原因應該是訓練資料量不足的關係，由於訓練資料量不足夠，導致維度增加時overfit。

3.4. 由於kaggle有次數上傳限制，沒辦法將全部實驗做完。

4. Word2Vec skip-gram

4.1. 後來我們又測試了skip-gram model，在gensim Word2Vec 中，只要設定 sg=1 就可以指定在訓練詞向量的時候使用的是 skip-gram。另外，我們將iter設為20，並調整size做實驗。

4.2. 下表為我們測試幾種不同輸出維度的 Word2Vec的結果

4.2.1. (1.1的相似度比較方法)

Dimension	Kaggle Public Score	
	With Stopwords	Without Stopwords
100	0.44624	0.41673
200	0.43083	0.40553
300	0.42608	0.39837

4.2.2. (1.2的相似度比較方法)

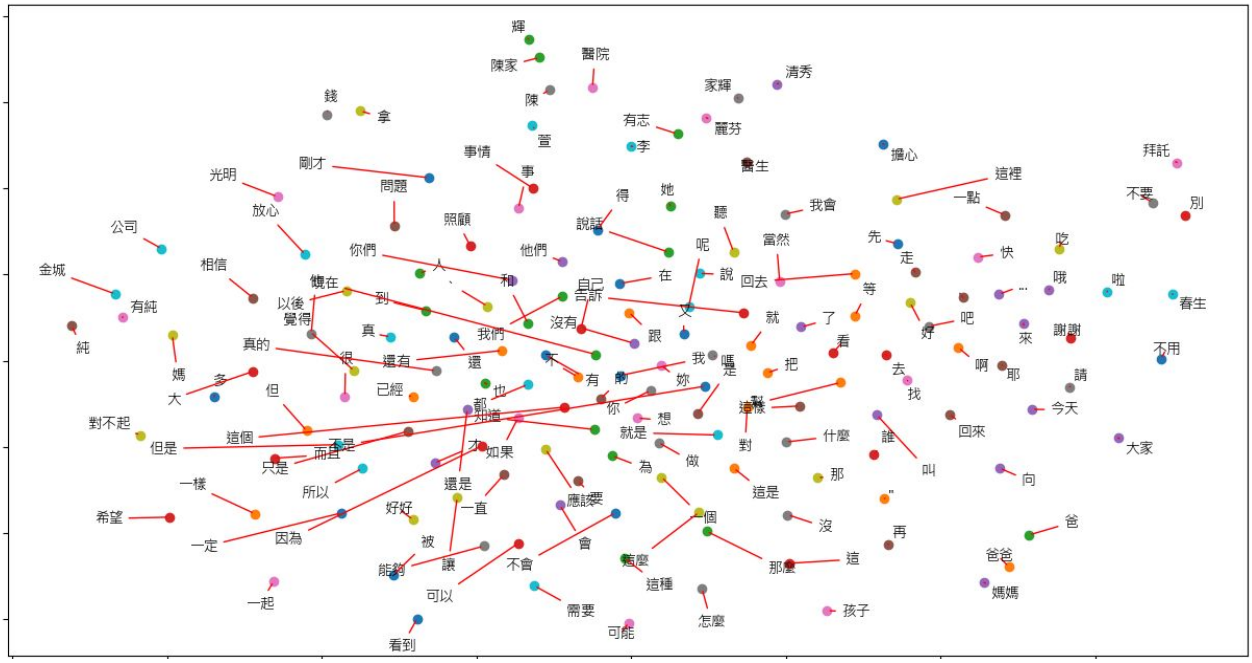
Dimension	Kaggle Public Score	
	With Stopwords	Without Stopwords
100	0.50079	
200	0.50474	0.50000
250	0.49209	
300	0.49328	0.49130

4.3. 同樣發現大約在維度200的時候有著最好的準確率。
4.4. 由於kaggle有上傳次數限制，沒辦法將全部實驗做完。

5. 針對不同長度句子處理方法
   5.1. 整句填成一樣長度 flatten 之後作cosine similarity，比如說全部問題及選項全部填成長度50，flatten之後就是50*dim下去做cosine similarity。
   5.2. 整句先平均之後作cosine similarity，因此每個問題及選項都是只有長度為1，dim維度的向量。
   5.3. 第一種處理方法中我發現因為特別長的句子屬於少數，所以在填充之後，每個選項之間的已經幾乎沒有差別了(因為填充都是填同一個字)，所以原本句子想表達的東西已經被我們的填充給弄不見了；而第二種處理方法比較能表達這個選項大約落在甚麼地方，原本的內容不至於被弄不見，而我們的實驗也表示出第二種處理方法比較好
   5.4.

Method	5.1	5.2
Kaggle Public Score	0.31	0.44

6. Word2Vec 可視化
   6.1. 我們想藉由將 Word2Vec 可視化來觀察出training data出現頻率較高的是甚麼樣的詞類。
   6.2. 下圖為 Word2Vec min_count=3000的 model 所做出來的圖 (min_count=3000代表training data中出現次數小於3000的單詞將不會被訓練到)
   6.3. 觀察出出現頻率高的單詞都是一些語助詞、人名、代稱等等，這是可以預期到的，因為我們的training data是類似劇本的東西，劇本是人講的話，而我們平常講話本來就會帶有許多的助詞。
   6.4. 我們上面所作的實驗 min_count 皆設為1，原因是有很多單詞可能就只有在其中一句裡面出現過，而那個單詞又是那一句的精髓，因此如果設高一點的話，就會省略掉許多重點，因此我們設為1。



7. 結論

7.1. 直接用Word2Vec比較相似度做出的成果比RNN好非常多。

7.2. Word2Vec設定iter=20比預設的iter=5要好。

7.2.1. 同樣用最好的參數做測試，(sg=1, size=200, 不去掉stopword),

iter = 5: kaggle public score = 0.49802

Iter = 20: kaggle public score = 0.50474

7.3. Skip-gram的Word2Vec比CBOW的好很多，因為要解決的題目是：給一個前句，推論出下一句。

CBOW是由附近的word vector選出此處最有可能的word vector，而Skip-gram是input一個word vector，選出附近最可能的其他word vector，與題意較為相近。故Skip-gram結果比較好我們覺得相當合理。

7.4. 沒有去掉stopwords結果稍微好一點點。可能是因為jieba的stopwords量滿少的，只有18個。且可能有些字在這次的training data中是能代表某些意義的，而也有很多不太能代表意義的詞沒有被收錄。因此去掉jieba提供的stopwords反而造成一點反效果。