

題目：TV Conversation

隊名：NTU_b04901136_onedayatime

隊員：b04901136 張家銘、b04901155 鄭閔、b04901056 張承洋

分工	張家銘	33.33%	RNN Autoencoder
	鄭閔	33.33%	Word2Vec preprocess
	張承洋	33.33%	Weighted sum

一、 Preprocessing / Feature Engineering

這部份包含如何利用有限又品質不好訓練資料訓練出一個好的 word embedding。包含以下步驟：

1. 切詞：比較使用繁體的 jieba 直接切和將文字轉成簡體字後用簡體的 jieba 切。
2. line window and stride：觀察到訓練資料中有些句子太短而和前後句合併後仍有 gensim 可能可以辨識的前後詞關係，因此用 line window 控制訓練資料中每幾行當成一句，stride 控制每次 window 要往後移動幾行。
3. size (word vector dimension)：設定輸出的 word vector 有幾維。
4. word window：設定在同一個句子當中 gensim 要看前後幾個詞做訓練。

testing data 同樣轉成簡體字後用簡體的 jieba 切詞，再進訓練好這個 gensim 的 Word2Vec model 得到每個詞的向量。我們評估參數效果好壞的依據其實就是我們下面的第一個模型 Average and Cosine Similarity 的結果。

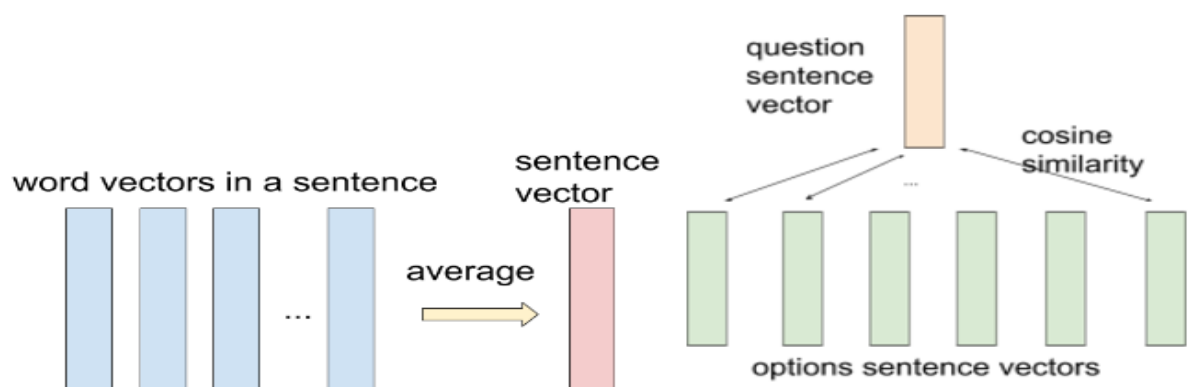
二、 Model Description(At least two different models)

1. Average and Cosine Similarity

我們將 training data 利用 gensim 的 Word2Vec 套件訓練出一個 model，可以將 jieba 斷詞後詞轉成向量，將一句話的各個向量經過平均之後便得到該句的 sentence vector，利用 scipy 的 cosine similarity 功能算出問題與選項的 sentence vector 之間的相似程度，取最大者為輸出答案。(對於 OOV 的處理我們直接忽略)

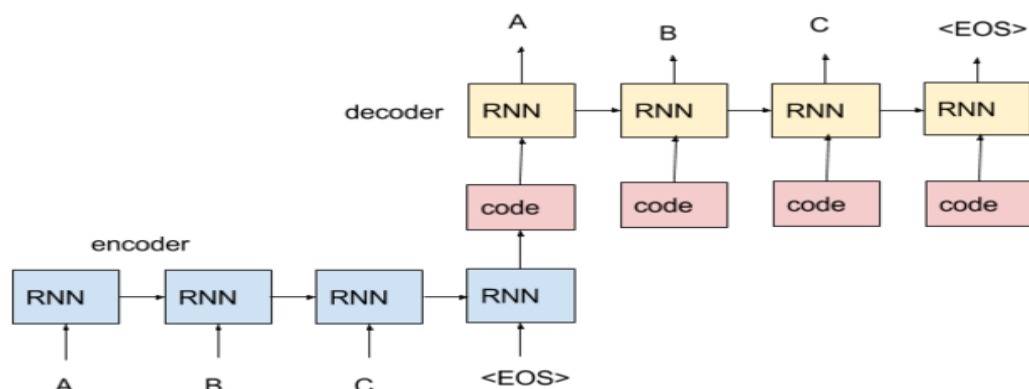
2. Recurrent Neural Network Autoencoder

a. 模型架構



因為第一種模型直接平均取得句向量的方式並沒有考慮到詞之間的先後順序關係，我們設計了一個 RNN 的 autoencoder，希望能在有考慮到先後順序的條件下將原本的詞向量序列 encode 成比較好的句向量。

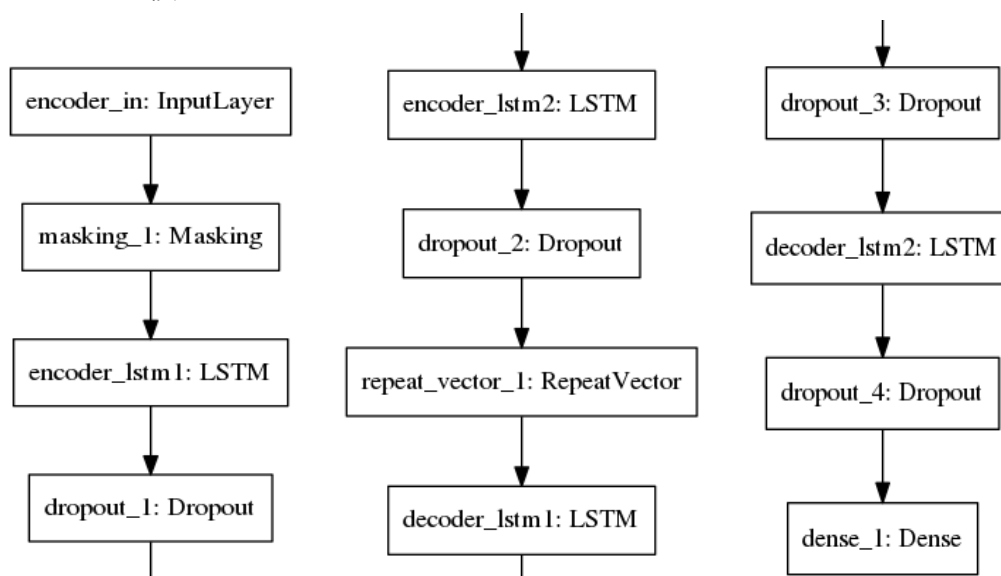
evaluate 時把問題和選項的詞向量序列輸入 encoder，再把得到的結果彼此做 cosine similarity，取最大者為輸出答案。



b. 訓練細節

- (1) autoencoder 是非監督式學習，只要在輸入和輸出都放同樣的序列就好。
- (2) encoder 的 RNN 代表多層 LSTM Cell，每一層的 unit 從 256、128、64 遞減。
- (3) encoder 的 RNN 代表多層的 LSTM Cell，除了最後一層的 LSTM layer 以外，每一層的 LSTM 都 return sequence，而最後一層 LSTM layer 僅有 return state。
- (4) decoder 的 RNN 代表多層 LSTM Cell，每一層的 unit 從 64、128、256 遞增。
- (5) decoder 的 RNN 代表多層的 LSTM Cell，第一層的 LSTM layer 每一個 time step 的輸入都是 encoder output。因為希望這個 code 含有資訊包括出現什麼詞以及他們的先後關係，我們認為 autoencoder 必須在沒有其他 input 的情況下，把 input 降維，再 reconstruct。所以我們認為 decoder 不能像 sequence to sequence 的 generator，將 output sample 完再餵回給 RNN。
- (6) Loss Function 用的是 MSE，希望輸入和輸出越像越好。

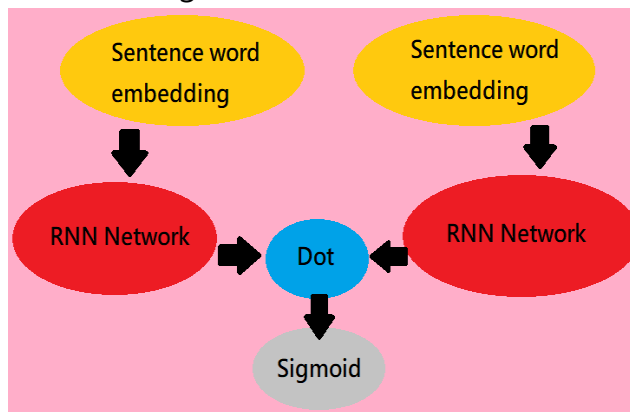
c. keras 模型



3. Recurrent Neural Network using Dot and Sigmoid

a. 模型架構

模型有兩個輸入，一個是問題的詞向量序列，另一個是回應的詞向量序列。這兩個輸入分別經過一個 RNN 之後的結果互相內積，通過一個 sigmoid 之後得到 0 到 1 的分數。



b. 資料選取

因為大部份資料庫中的句子都很短，不足以拿來預測連接的句子，所以我們把 N 個資料庫中的句子合併在一起成為一個句子當題目。同樣的也把 N 個資料庫中的句子合併在一起成為一個句子當答案。

c. 資料標記

需要 0 和 1 的標記資料，我們假設訓練資料中鄰近的資料是正確的回應，標記為 1，再從遠處 random 挑選一個不相干的回應，標記為 0。

d. 訓練細節

(1) RNN 代表多層的 LSTM Cell，每一層的 unit 從 256、128、64 遞減。

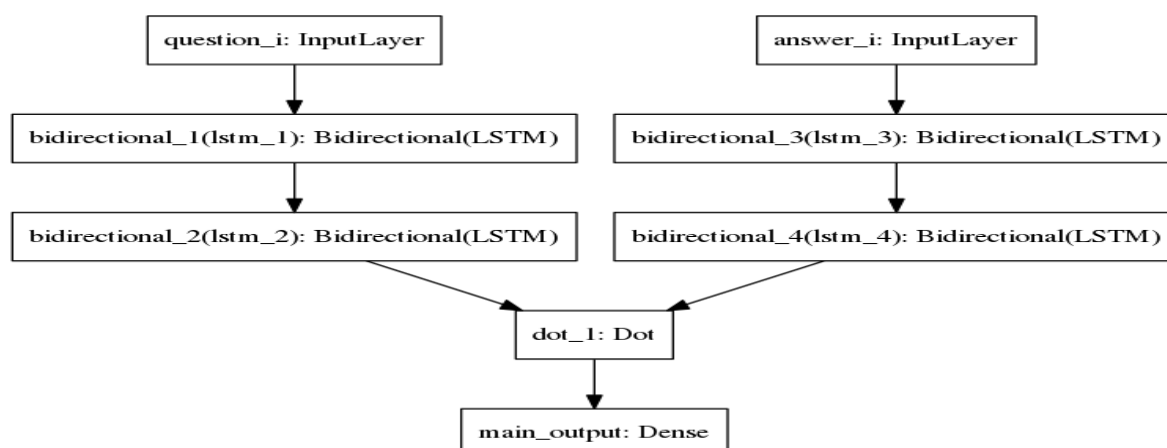
(2) RNN 代表多層的 LSTM Cell，除了最後一層的 LSTM layer 以外，每一層的 LSTM 都 return sequence，而最後一層 LSTM layer 僅有 return state。

(3) Loss Function 用的是 hinge，因為我們希望輸出不只可以將可行的答案的機率調到很高，還要把不可行的答案的輸出降到愈低愈好。

e. 模型精神

此模型設計的概念是希望 RNN 出來得到的結果能代表該句子，dot 的步驟是讓問題和回應之間的關聯性越高越好。

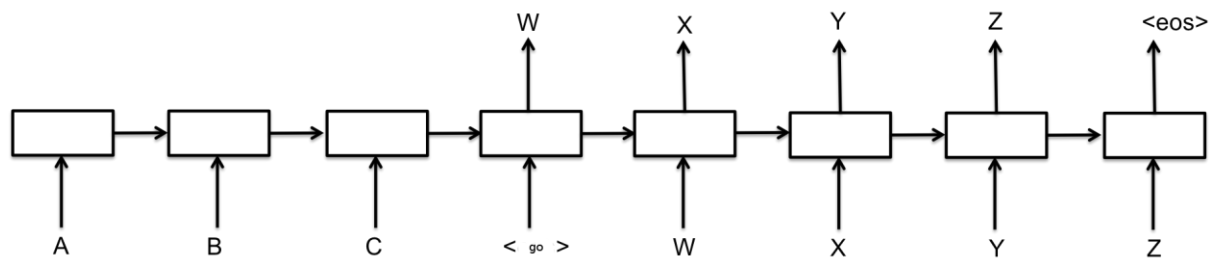
f. keras 模型



4. seq2seq

a. 模型架構

我們也嘗試了基本的 seq2seq 的架構。模型輸入問題後直接輸出回應。



圖片來源：<https://www.tensorflow.org/versions/r1.1/tutorials/seq2seq>

b. 資料標記

將鄰近的語句當坐下一句的回答，但是因為 training data 中有些資料辭彙量太少，所以我們將 N 個 training data 中的句子合併當成問句，再把接續的 N 個句子合併當成回答。

c. 訓練細節

(1) encoder RNN 代表多層的 LSTM Cell，每一層的 unit 都是 1024，但是只有最後一層不 return sequence，但是最後一層要 return state，因為要當成 decoder RNN 的 initial state。

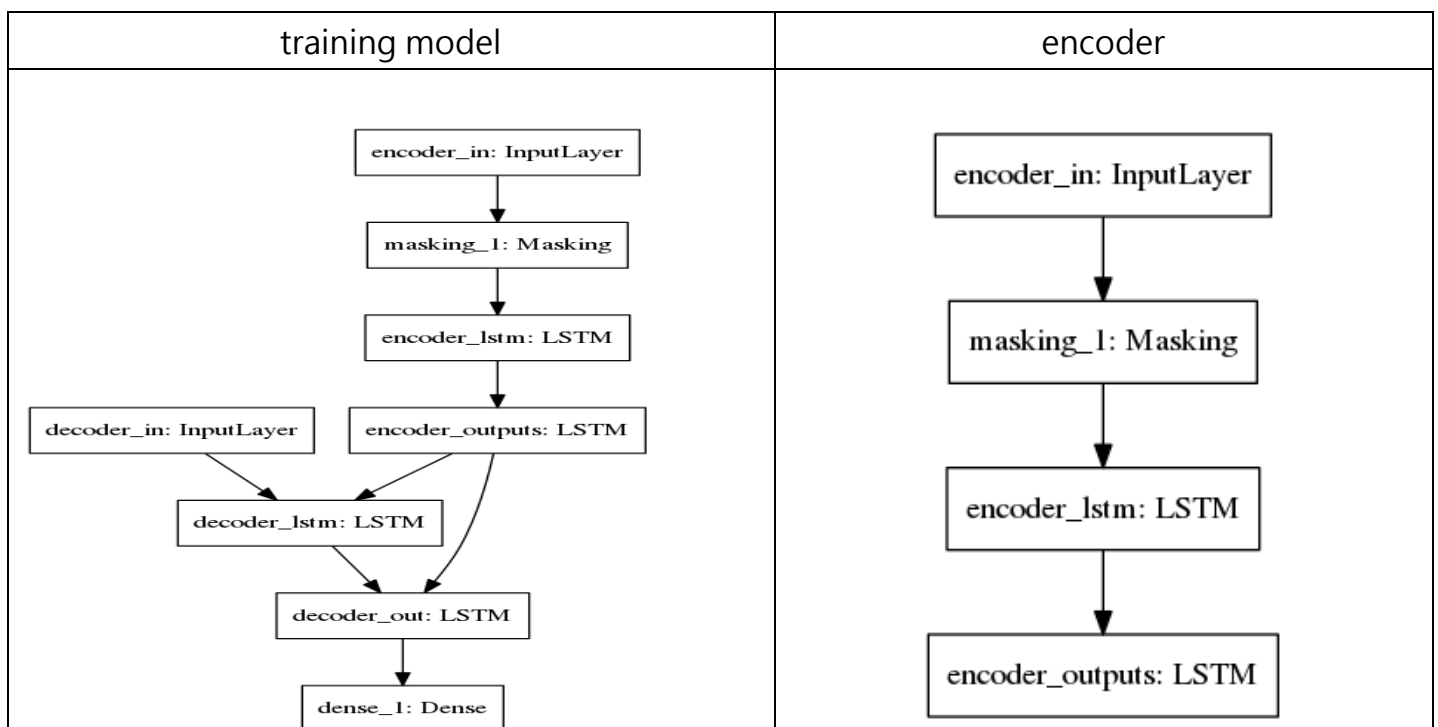
(2) decoder RNN 代表多層的 LSTM Cell，每一層的 LSTM 都 return sequence，因為最後一層 layer 需要被 sample 成文字，並且 return state，因為在 decode 的時候需要 state 當作輸出。

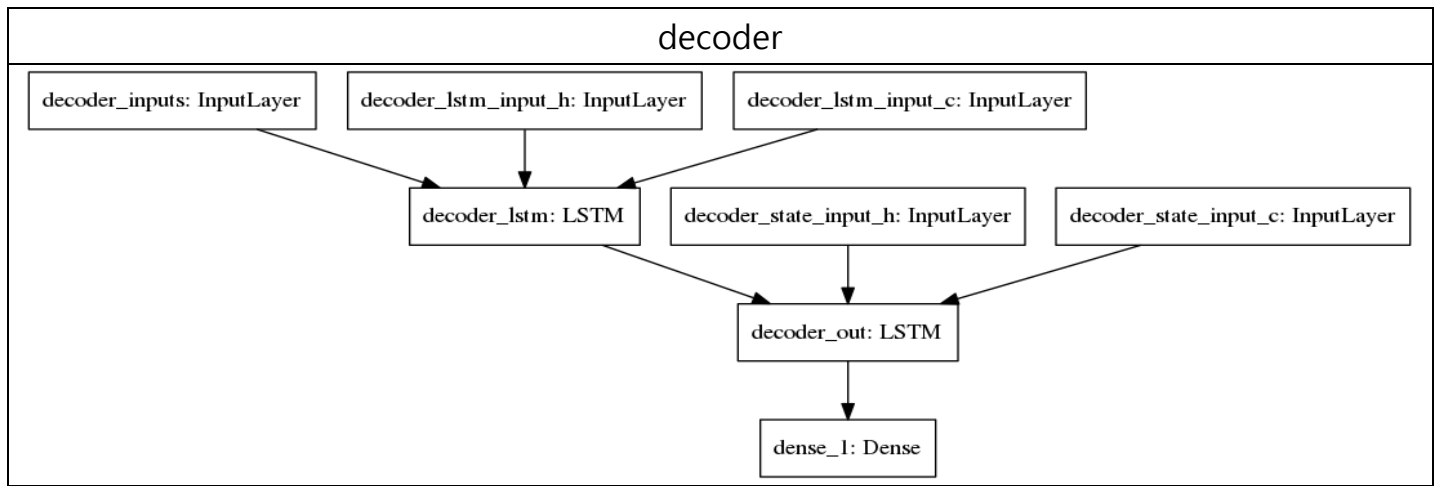
(3) Loss Function 用的是 mse，因為我們希望輸出的答案可以跟回答一樣像。

d. 模型精神

希望輸入問題給模型後，模型能輸出回應。我們選擇選項中和模型輸出的回應最接近的選項作為答案。

e. keras 模型





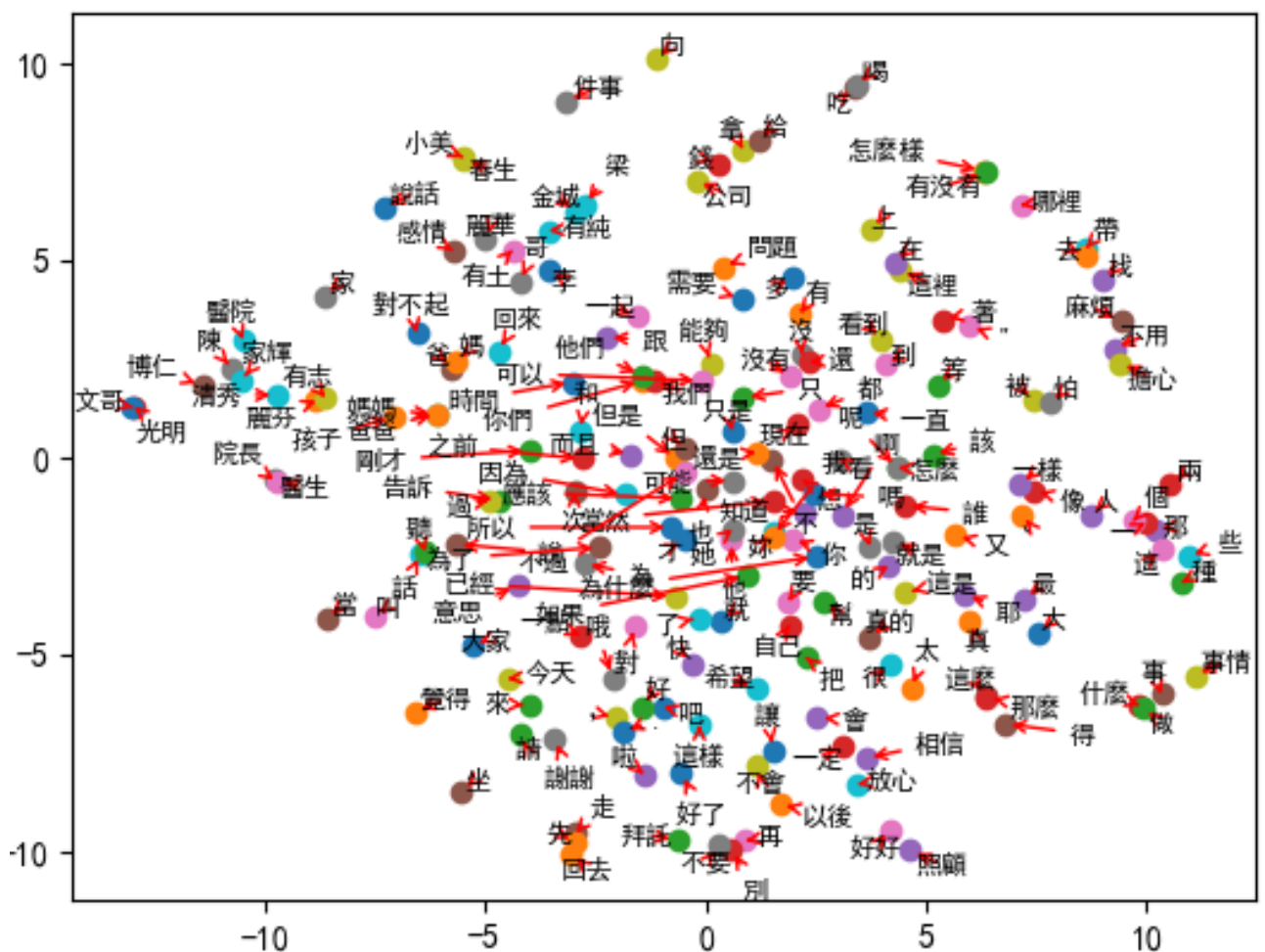
三、Experiments and Discussion

● Word embedding + simple average + cosine similarity

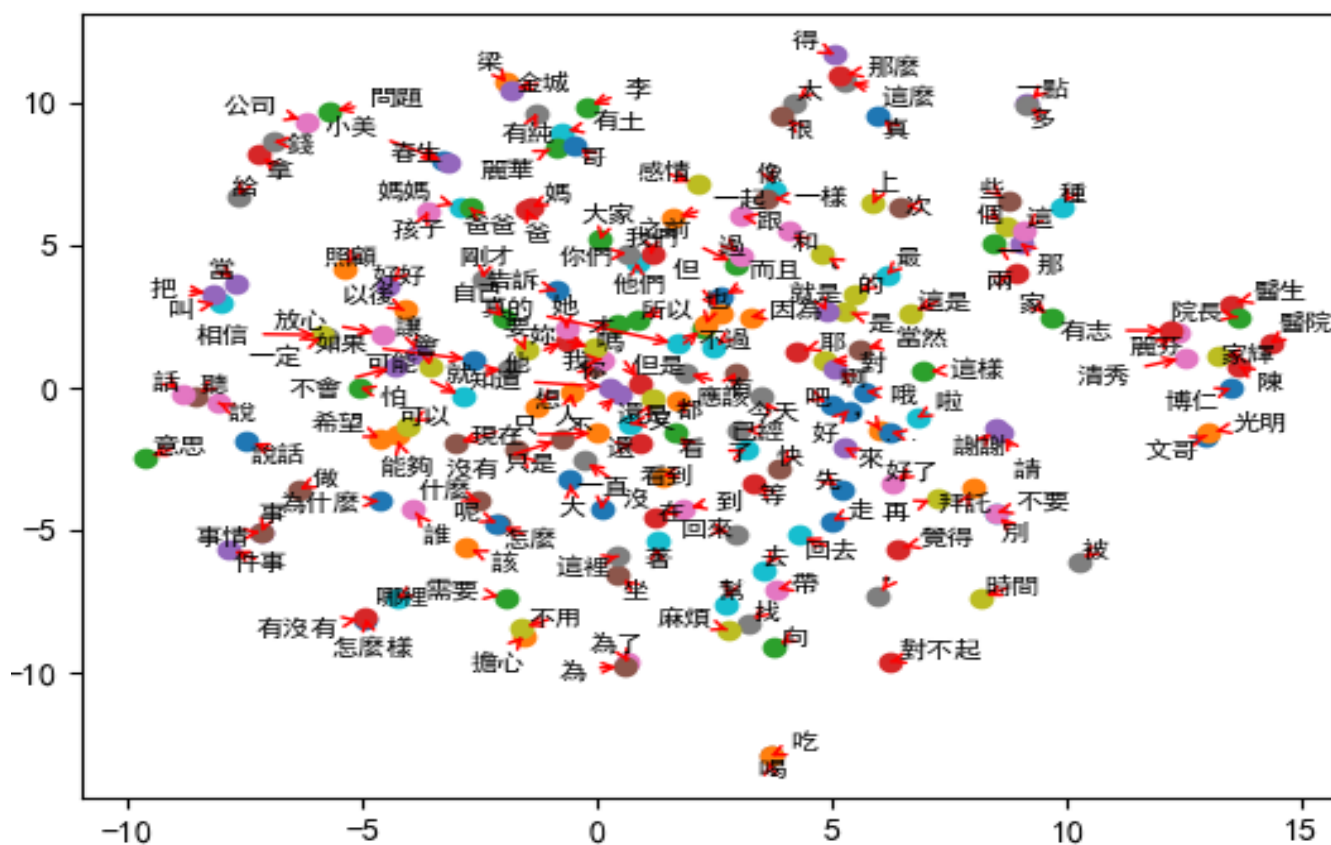
此部份主要的目標是訓練出好的詞向量，並用 simple average 和 cosine similarity 後的結果來評估詞向量的好壞。可以調整的參數如第一部份 Preprocessing/Feature Engineering 所述。由於 Kaggle 上傳次數有限，

1. Word embedding visualization

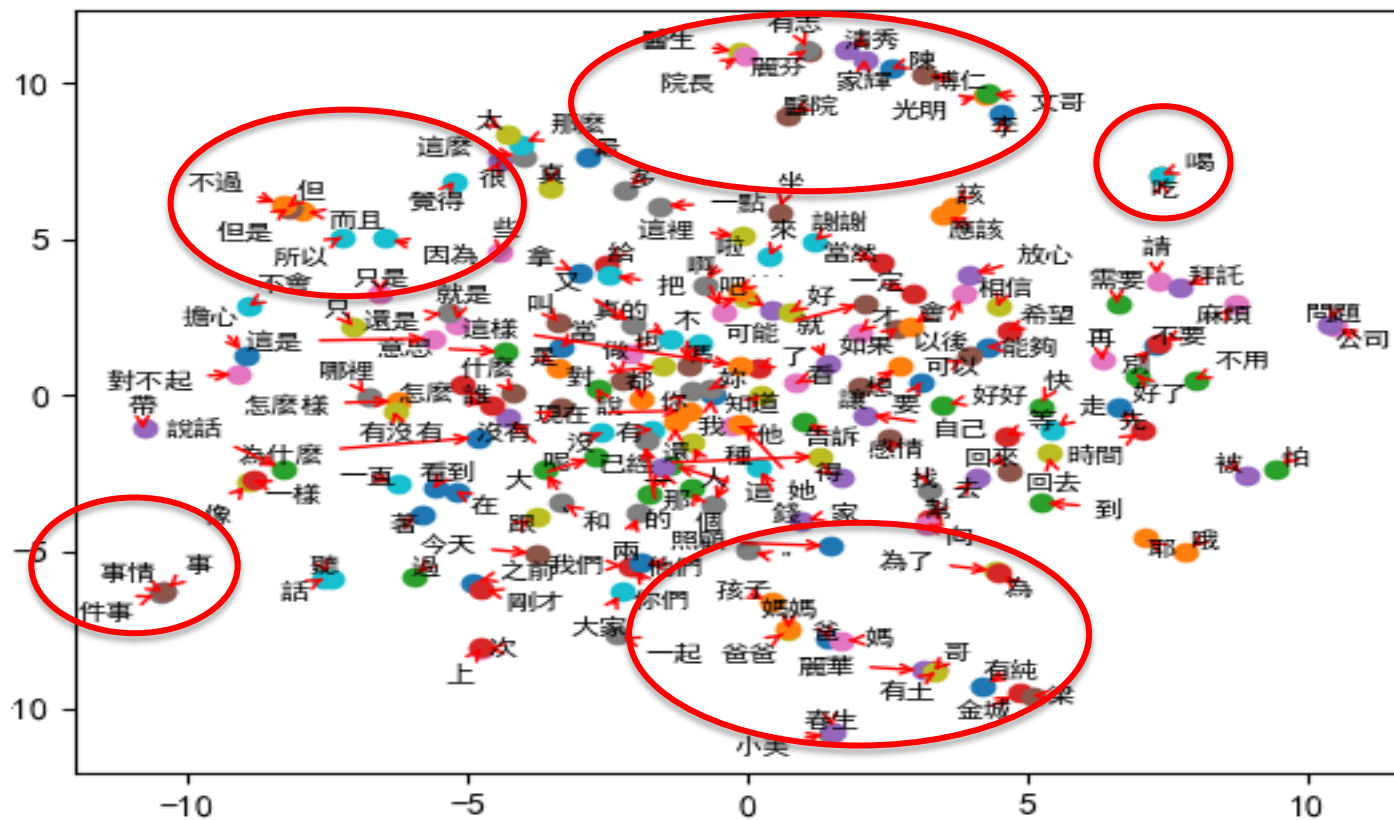
(dim : 32 , window : 3 , stride : 1)



(dim : 64 , window : 3 , stride : 1)



(dim : 256 , window : 3 , stride : 1)



從上面三張圖可以看出，關係相近的詞會被 cluster 在一起，例如：上圖中紅框中的詞。

2. 繁體 jieba、簡體 jieba (其他參數未調整)

使用 jieba	繁體 jieba	簡體 jieba
正確率	0.44	0.42

訓練條件:

line window	stride	size	Word2Vec window
3	2	64	5

我們以 cosine similarity 的架構比較繁體與簡體中文 jieba 的訓練結果，由上表可以發現，兩者訓練出來的結果相近，過程中我們嘗試印出斷句後的結果，發現兩者結果幾乎相同，另外我們嘗試使用繁體 jieba 去切簡體字的 corpus，準確率完全爆炸，跌到只剩 30% 左右，反之亦然，因此我們認為只要使用正確的字典斷句，便會得到不錯的結果。而在準確率由於方面繁體中文表現稍微優異，在以下的實驗當中我們都以繁體中文作為 jieba 切詞的字典。

3. line window、stride

(表 2-1)

正確率	line window 1	line window 2	line window 3	line window 4
stride 1	0.42	0.46	0.44	0.40
stride 2	-	0.47	0.44	-

(表 2-2)

正確率	line window 2	line window 3
stride 1	0.46	0.44
stride 2	0.47	0.44
stride 3	-	-
stride 4	0.44	0.48
stride 5	-	-

訓練條件:

jieba 字典	size	Word2Vec window
繁體字典	64	5

由表 2-1 可以看出，在 stride=1 or 2 的情況下，準確率都在 line window = 2~3 之間達到高峰，而往兩側遞減；而 stride=2 的表現比 stride=1 好一些，但兩者相差接近。因此我們認為這驗證了我們的假設，training data 中確實有許多句子長度太短，甚至有些句子只有一個動詞或是一個專有名詞，所以需要適當大小的 line window 把完整的資訊納入一個句子當中，然而當 window 過大 (>4)時，可能會將不同對話情境中的句子合在一起，使得 Word2Vec Model 在訓練時將類型迥異的詞屬於的 cluster 拉近，造成誤差。

另外，由於 stride 對於準確率的影響不明顯，我們針對 line window = 2~3 的條件，觀察準確率與 stride 的關係，結果如表 2-2 所示，可以看出當 stride=3~4 時有最好的表現，推測是因為 training data 中的對話很多都是以 3~4 句作為一個單位，不同的單位即代表不同的對話情境，不當的 stride 會造成上述 model 在分類時混淆造成表現較為低落的情況。

4. size(word vector dimension)

size	32	64	128	512	1024
正確率	0.41	0.45	0.43	0.41	-

訓練條件:

jieba 字典	line window	stride	Word2Vec window
繁體字典	3	2	5

由上表可以看出，此條件下的準確率大約落在 42%上下，在 size = 64 時達到最大值，而在 size>64 的情況下準確率隨著 size 增加而降低並出現震盪的現象，因此我們認為 64 維的 word vector 便足以儲存文本的資訊，事實上我們觀察 training data 是出自某個連續劇的劇本，與大部分的 testing data 出題方向不盡相同，因此如果使用更高維度的 word vector 可能會考慮到不必要的資訊，造成誤差。

5. Word2Vec window

Word2Vec window	2	3	4	5	6	7	8	9	10	11
正確率	0.33	0.38	-	0.42	-	0.47	-	0.43	-	0.41

訓練條件:

jieba 字典	line window	stride	size
繁體字典	3	2	64

從上表可以發現，Word2Vec window 對於我們 cosine similarity 模型的影響並不大，大約都落在 43% 上下並在 window=7 時達到高峰，而後則隨著 window 增加而逐漸降低，推測是因為我們有考慮到 line window 的關係，所以出現過短句子的機率偏低，大部分丟進 Word2Vec model 的訓練資料長度都超過 10，有些長度甚至會達到 20，因此過短的 window 可能無法準確地抓到某個字在對話情境中的定位，造成分類上的誤差。

5. weighted sum

vector size \ evaluation method	average	weighted sum
64	0.45	0.47
128	0.45	0.47
512	0.46	0.46

我們的 weighted sum 方法來自此篇 paper : <https://openreview.net/pdf?id=SyK00v5xx>

Algorithm 1 Sentence Embedding

Input: Word embeddings $\{v_w : w \in \mathcal{V}\}$, a set of sentences \mathcal{S} , parameter a and estimated probabilities $\{p(w) : w \in \mathcal{V}\}$ of the words.

Output: Sentence embeddings $\{v_s : s \in \mathcal{S}\}$

1: **for all** sentence s in \mathcal{S} **do**

2: $v_s \leftarrow \frac{1}{|s|} \sum_{w \in s} \frac{a}{a+p(w)} v_w$

3: **end for**

4: Form a matrix X whose columns are $\{v_s : s \in \mathcal{S}\}$, and let u be its first singular vector

5: **for all** sentence s in \mathcal{S} **do**

6: $v_s \leftarrow v_s - uu^\top v_s$

7: **end for**

訓練條件:

jieba 字典	line window	stride	size	Word2Vec window
繁體字典	3	3	64	5

我們的作法是先將訓練好的 Word2Vec model 列出所有字的出現次數(假設為 max)，則 $p(w) = \text{model.wv.count} / \text{max}$ ， α 大約落在 $10^{-3} \sim 10^{-4}$ 之間，將所有詞經過紅框內的處理之後再求一般平均，便可以得到 weighted sum。

由上表可以得知，整體來說由於 weighted sum 多考慮了詞頻，所以在表現略優於單純的 average。

● RNN Autoencoder

1. RNN layer 數量：括弧內每個數字代表一層 RNN 的 unit

RNN layer 數量	(16,32)	(32,64)	(32,64,128)	(32,64,128,256)
正確率	0.33	0.38	-	0.26

訓練條件：

Size	Negative	Word2Vec window	corpus 相鄰三句合併在一起
64	10	7	

只要調高 Model 複雜度，訓練出來的模型就會有較差的表現，我們猜測是 因為 training data 跟 testing data 在本質上有許多差異，所以只要 model 的複雜度到一定程度以上，那訓練出來的結果會造成 testing 時很多的誤判。

2 earllystop：以 model reconstruct 結果與原本語句的 mse 誤差，選擇 earllystop

earllystop	(16,32) with earllystop	(16,32) without earllystop	(32,64,128) with earllystop	(32,64,128) without earllystop
正確率	0.25	0.33	0.28	0.30

原本我們都有加 earllystop，可是 training 時，1~2 epochs 過後，reconstruct 的表現就幾乎沒有再增加，但是我們發現就算 reconstruct 的結果沒有在增加了，autoencoder 還是有在增加 embedding 的表現。

我們猜測 reconstruct 過後與原本語句的 mse 誤差，不能當作 reconstruct 的表現指標。

3. Decoder RNN 的 initial state：我們嘗試了隨機與以 0 為 initial state

Decoder RNN layer initial state	initial state=0	initial state=random
正確率	0.36	0.35

基本上沒有太多差別。

● RNN Using Dot and Sigmoid

1. training data 形成問句的方式：

為了將 training data 中相鄰的句子視為上下句，但是有些 training data 中的語句辭彙量太少了，所以我們把 training data 中相鄰的 N 個句子合併作為一句，這裡的 N 就是 line window。(每一個問句的第一個語句相鄰的間隔就是 stride)

正確率	line window 2	line window 3
stride 1	0.36	0.32
stride 2	0.33	0.28
stride 3	-	-
stride 4	-	-
stride 5	0.29	0.26

- a. 我們認為合併兩句會有最佳的結果，合併過多時，會使語句過長無法判讀，合併過少時，因為有時候一句只有一個詞，會使得語句極難判讀。
- b. 因為 stride 數增加，只是純粹的減少 data 數量，只會使表現下降。

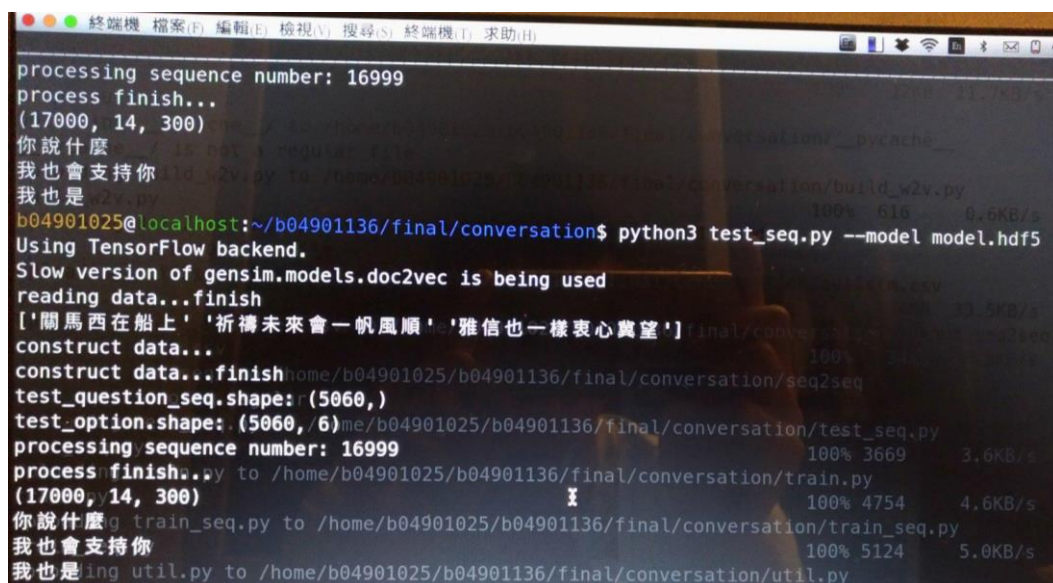
2. 將問句、答案丟到 RNN 之後，使用不同的方法融合兩向量

Merge layer	dot	concatenate
正確率	0.35	0.39

我們原本使用 dot 但是準確率在經過一個 epoch 之後就不再改變，所以我們改用 concatenate，準確率有明顯的改變。

● seq2seq

我們訓練出的模型能產出合理且有意義的回應，例如：

A terminal window showing the execution of a Python script for training and testing a seq2seq model. The output includes progress bars for training and testing, and the final results for the test set. The test set results show a sequence number of 16999, a process finish time of (17000, 14, 300), and a test question sequence shape of (5060, 6). The test option sequence shape is (5060, 6). The test results show a sequence number of 16999, a process finish time of (17000, 14, 300), and a test question sequence shape of (5060, 6). The test option sequence shape is (5060, 6). The test results show a sequence number of 16999, a process finish time of (17000, 14, 300), and a test question sequence shape of (5060, 6). The test option sequence shape is (5060, 6).

```
processing sequence number: 16999
process finish...
(17000, 14, 300)
你說什麼
我也會支持你
我也是
b04901025@localhost:~/b04901136/final/conversation$ python3 test_seq.py --model model.hdf5
Using TensorFlow backend.
Slow version of gensim.models.doc2vec is being used
reading data...finish
['關馬西在船上' '祈禱未來會一帆風順' '雅信也一樣衷心冀望']
construct data...
construct data...finish
test_question_seq.shape: (5060,)
test_option_seq.shape: (5060, 6)
processing sequence number: 16999
process finish...
(17000, 14, 300)
你說什麼
我也會支持你
我也是
```

A: 關馬西在船上 B: 你說什麼

A: 祈禱未來會一帆風順 B: 我也會支持你

A: 雅信也一樣衷心冀望 B: 我也是

雖然看似有些無厘頭，但其實都是合理的回應。尤其是第二個例子的回應很具體，不是套用在任何問題都適用的回應，表示我們真的有訓練到。不過我們用這個模型產生的句子和選項比較發現，模型輸出的句子和題目的正確答案都是合理的回應但是，不過並不一定是相似的句子。因為面對同一個問題同時可以有很多的合理的回應，而這些回應的方向不一定相似。後來準確率都最好大約只有 0.20 左右，比起隨機選一個答案的 0.167 只好一點。

結論是選擇題並不適合使用自由度高的 seq2seq 模型。

結論：

總體來說，直接比較 word vector 的 cosine similarity 的表現比 neural based 的表現還要優異，我們推測主要有兩個。首先，由於這次的 project 主題比較單純，只是要從 6 個選項中挑選 1 個當作回答，這種偏向選擇題的方式通常比較會讓簡單的模型有好的表現；其次，這次的 training data 性質偏向劇本，而 testing data 大部分是同學出題，內容迥異，再加上訓練資料中各個情境的段落之間沒有空白，導致 neural based 的模型會發現很多問題與回答毫無關聯的情況，導致我們的 RNN 準確率一直拉不起來，如果這次的 training data 內容更有系統，我們認為 autoencoder 跟 seq2seq 的表現能夠優於單純比較 similarity 的方法。