Final Project - Conversation in TV Shows

(一) Team name, members and work division

（依學號排序）

b04902008 曾千育：cosine similarity model (simple baseline)、report

b04902011 張立暐（隊長）：cosine similarity model (strong baseline) 、report

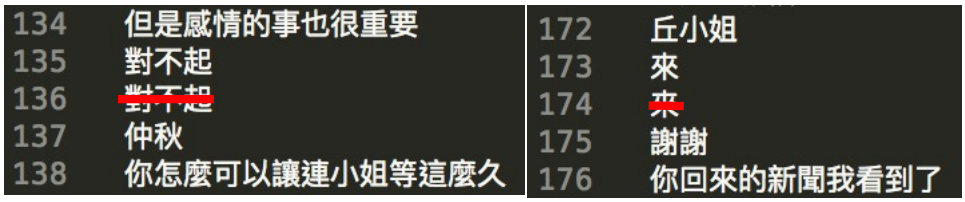
b04902012 劉瀚聲：retrieval model、report

(二) Preprocessing/Feature Engineering

在我們實作的兩種模型中，都有對訓練資料作各種預處理，其中**刪除重複的句子**、**刪除有雙引號的句子**、**使用 jieba 套件進行斷詞**這三點是兩種模型皆有使用，**連接句子**則有不同的處理方法，以下依照預處理的順序對各點分別進行說明：

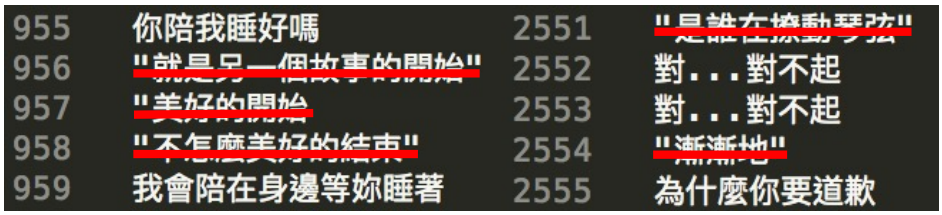
1. **刪除重複的句子**

   因為我們在訓練資料中發現有少數重複的句子（如下圖所示），而這樣的句子通常都是較短、且對整體意思並不是很重要的句子，因此我們把這些重複的句子刪除。



2. **刪除有雙引號的句子**

   因為我們在訓練資料中發現有少數被雙引號括起來的句子（如下圖所示），而這些句子通常是歌詞、且會中斷原先劇本對話（也就是其他沒有被括起來的句子）的連貫性，我們認為這樣會在後續訓練詞向量時，影響結果的好壞；若是將被雙引號括起來的句子刪除後，也可以明顯發現句意會通順許多，因此決定把有雙引號的句子刪除。



3. **使用 jieba 套件進行斷詞**

   由於 jieba 原先的預設字典是簡體字典，但是我們的訓練資料卻是繁體劇本，因此我們決定使用助教在 hw6 時提供的繁體字典"dict.txt.big"，使其斷詞效果較好。

（使用預設字典）

```
10    你現 在 是 台灣 第一 個 女 醫生
```

（使用繁體字典）

```
10    你 現在 是 台灣 第一個 女醫生
```

4. 連接句子

i. Cosine Similarity Model：

因為我們在訓練資料中發現有不少過短的句子，若將前後句連接後，會使其句意較完整且連貫（如下圖所示），因此我們將每 3 句相鄰的句子合併為一句新的訓練資料，預期後續訓練詞向量時會因為句子的連貫性而有較好的結果。

```
273    各位
274    為 了 表示
275    我們 對 女醫生 的 歡迎
276    我們 要 他 做 什麼 呢
277    做 什麼
```

（←連接前）　　（↓連接後）

```
273    各位 為 了 表示 我們 對 女醫生 的 歡迎
274    為 了 表示 我們 對 女醫生 的 歡迎 我們 要 他 做 什麼 呢
275    我們 對 女醫生 的 歡迎 我們 要 他 做 什麼 呢 做 什麼
```

ii. Retrieval Model：

由於要將訓練資料整理成可以放入 retrieval model 的樣子，我們將訓練資料中每 4 句為單位，前兩句合併為 Request Sentence，後兩句合併為 Response Sentence，轉換為一筆新的訓練資料，接著向下位移一句，以此類推。

經由上述的處理將訓練資料轉變成一一對應的 Request 和 Response sentence 後，便能生成「正確匹配」的資料；但因為訓練時還需要告訴機器哪些是錯誤的配對，故將不對應的 Request 和 Response sentence 配對來產生「不正確匹配」的資料（藉由 Request sentence 不動，Response sentence 向後平移 10~50 個句子來達成此目的），並將其分別標上 Label 1（正確匹配）、Label 0（不正確匹配）來完成正反資料的生產。

(三) Model Description

1. 使用 gensim 套件訓練詞向量

因為我們實作的兩種模型中皆是奠基在詞向量的基礎之上，第一種 cosine similarity model 是將訓練好的詞向量以各種方法來比較問答句間的關係，第二種 retrieval model 則是以訓練好的詞向量來初始化 embedding layer（兩種模型的實作方法會在下兩點會做詳細介紹），因此詞向量的優劣十分重要。

而我們是使用 gensim.models.word2vec 中的 Word2Vec 函式來訓練詞向量，且在經過多次的測試結果後（在(四)中會詳細說明調整參數的過程），調整參數 size=32、window=3、min_count=1、sg=1、negative=3、iter=35，其他則維持預設。

2. Cosine Similarity Model

一開始在訓練模型時，因為沒有特別的想法，因此依據助教公告的提示「根據訓練資料訓練一個 word embedding，比較問題與各答案選項間 word embedding 的關係」來訓練模型，準確率便達到 39%，成功超過 simple baseline。接著改為使用繁體字典後，因為斷詞的優劣進而影響詞向量的優劣，再加上 ensemble 多次訓練的結果，準確率便提升至 45%，超過 strong baseline。再經過多番測試後一路提升至 54%，其中我們嘗試過各種計算句子相似度的方式，詳細作法如下：

i. 詞向量平均作為句向量

將每一句的詞向量加總平均作為句向量，得到問句與各選項的句向量後，將問句對各選項分別計算餘弦相似度，餘弦相似度最高的選項即作為預測答案。

$$V_s = \frac{1}{|s|}\sum_{w \in s} V_w$$

$$similarity_1(s_1, s_2) = \frac{V_{s_1} \cdot V_{s_2}}{\|V_{s_1}\|\|V_{s_2}\|}$$

ii. 兩句中的每個字兩兩計算餘弦相似度

參考助教的方法，改為將問句的每個字對各選項的每個字兩兩計算餘弦相似度，再將其加總，選擇最後結果最高的選項作為預測答案。

$$similarity_2(s_1, s_2) = \sum_{w_i \in s_1}\sum_{w_j \in s_2} \frac{V_{w_i} \cdot V_{w_j}}{\|V_{w_i}\|\left\|V_{w_j}\right\|}$$

iii. 前兩種方法混用

與組別 b04902089_DeepQueueing 討論，將前兩種方法作加權總和。

$$similarity_{\text{mix}}(s_1, s_2) = \beta \cdot similari\text{□}y_1(s_1, s_2) + similarity_2(s_1, s_2)$$

iv. 詞向量加權作為句向量

參考組別 b03902096_我用 GPU 你用 deepQ 在 Top 10% 報告中提及的方法，保留第一種以詞向量來表達句向量的想法，但將計算方式由平均改為加權，讓較獨特（出現頻率越少）的詞向量對句向量的影響較大；接著一樣與第二種方法混用。

$$V_s = \frac{1}{|s|}\sum_{w\in s}\frac{\alpha}{\alpha + p(w)}V_w\,,\ \ \alpha = 10^{-3} \text{ or } 10^{-4}$$

$$similarity_1(s_1, s_2) = \frac{V_{s_1}\cdot V_{s_2}}{\|V_{s_1}\|\|V_{s_2}\|}$$

$$similarity_{\text{mix}}(s_1, s_2) = \beta \cdot similarity_1(s_1, s_2) + similarity_2(s_1, s_2)$$

3. **Retrieval Model**

我們 Final Project 一開始是三個主題並行，儘管最後我們因 TV 的表現較好，而選擇 TV 這個主題，但一開始我們在主題 QA 所嘗試的模型很大程度的影響了我們在 TV 實作模型的選擇。以下介紹我們在 QA 嘗試並實作的模型：對於每筆訓練資料，計算兩個 one-hot 向量，分別代表答案區間開始和結束的位置，作為訓練資料的答案。接著訓練兩個模型，其輸出的維度與文章的詞數相同，而每個詞對應到的輸出表示那個詞作為答案的開頭、結尾的機率。而訓練的細節如下：

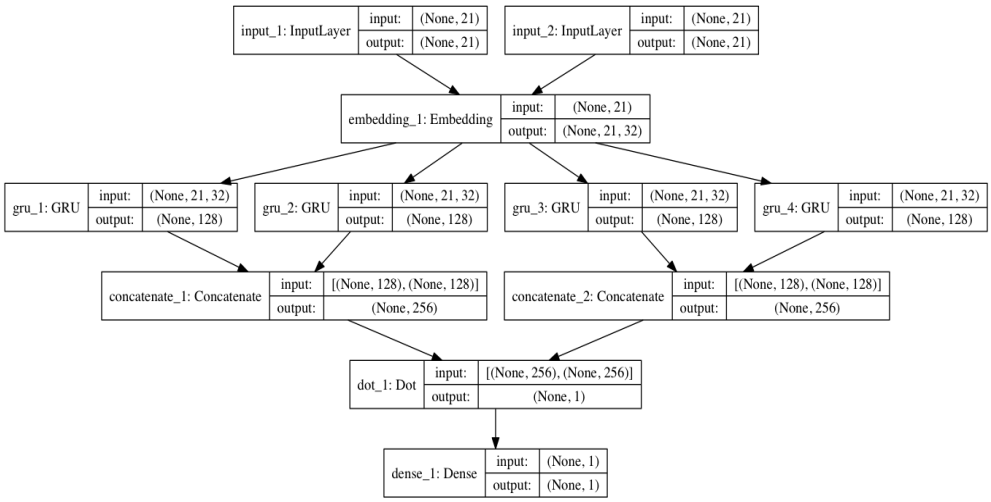
先用 jieba 進行斷詞，然後將文章 C 和問題 Q 用 gensim 的 Word2Vec 做 word embedding，之後將 embedded 過後的文章 C_E 和問題 Q_E 通過不同的 Bidirectional RNN R_C, R_Q 作為 encoder，產生的向量分別是 C' 和 Q'，其中 C_i' 和 Q_i' 各是一個 32 維的向量代表第 i 個詞。然後將 Q' 的平均 $\bar{q}'$ 作為代表問題 Q 的向量。之後，計算 $C_i' \cdot \bar{q}'$ 作為詞 C_i 所對應的輸出。

因此我們將一樣的概念運用在 TV 這個主題。以下開始描述實際用在 TV 的模型細節：

首先使用 keras.preprocessing.text.Tokenizer 將輸入資料中的每個詞轉換成一一對應的數字編號（Embedding Layer 再進一步將數字編號對應成詞向量），並透過 keras.preprocessing.sequence.pad_sequences 來將長短不一的輸入 padding 成相同長度，這樣即完成訓練前的資料處理。

下圖為我們以 Keras 實作的 retrieval model 模型架構，其中 request/response sentence 分別通過兩個 Input Layer 輸入，再通過同一個由 gensim pretrain 好的詞向量初始化的 Embedding Layer 後，分別輸入兩邊的 GRU Layer，其中兩邊皆有兩個 GRU Layer 是為了實作 Bidirectional GRU（其中一個透過參數 go_backwards=True 進行反向讀取輸入），並由

Concatenate Layer 進行合併，最後再將兩邊合併後的輸出先做內積（Dot）再過 sigmoid 函數使其輸出落在 0 到 1 之間以代表輸入是否為匹配資料的機率。



(四) Experiments and Discussion

1. gensim 版本問題

在與別組同學討論過作法後，發現在差不多的作法下，我們的準確率卻比他組同學的準確率低 2% 左右，在仔細檢查之下，驚覺我們使用的 gensim 版本為較舊的 3.0.8，而他組同學使用的版本為較新的 3.2.0，因此在更新套件版本後，準確率就上升至與他組同水準，看來 gensim 在版本更新後有優化其算法內容，才導致這樣的表現差距。

2. 在不同 iteration 數量下的 CBOW v.s. skip-gram

public 準確率	CBOW	skip-gram
iter=5	45.652%	44.940%
iter=35	46.916%	47.233%

前面提過因為我們實作的兩種模型中皆是奠基在詞向量的基礎之上，因此我們花較多的時間在改善詞向量的訓練。我們發現在 iteration 較低的時候（預設為 5），CBOW 和 skip-gram 對於結果並無明顯的差異，甚至後者略低；但在我們將 iteration 改為 35 後，不僅兩者的結果都上升約 1~3%，skip-gram 的效果也較 CBOW 好。因此接下來的討論我們皆使用 iter=35 和 sg=1 的設定。

且經過實驗發現，在 sg=1 的情形下，維度設定若過高，會使結果下降約 1%，因此維持一開始 size=32 的設定。

3. cosine similarity 的算法比較

算法敘述	單筆準確率
詞向量平均	38~40%
每個字兩兩計算	44~47%
詞向量平均＋每個字兩兩計算	50~52%
詞向量加權＋每個字兩兩計算	52~54%

一開始最直覺的想法便是一句話是由多個詞所組成，因此若將一句話涵蓋的詞向量做加總平均，便能簡易快速的表達一句話的句向量，這樣訓練出來的結果大約落在 38~40%左右。然而我們發現，這種方法存在某種問題無法解決，也就是當兩句話所涵蓋的詞向量都不接近，但是加總平均的句向量卻恰好較接近時，便會導致答案誤判。

因此我們改以助教的方法，將兩句中的每個詞兩兩做餘弦相似度計算，這樣的做法的確有使準確率上升至 44~47% ，推論其成效是因這種計算方式可以充分考慮到兩句之中所有涵蓋的詞是否有相關性，能避免上述平均造成的問題。但我們同時也發現，在測試資料中，有不少題目中的錯誤選項會包含跟題目相似度很高的詞，這種情形會造成這個算法的誤判，我們的準確率也因此暫時停滯。

在與別組討論過後，發想將前兩種方法透過以下方式並用：

$$similarity_{\text{mix}}(s_1, s_2) = \beta \cdot similarity_1(s_1, s_2) + similarity_2(s_1, s_2)$$

我們猜測是因為第一種方法可以考量到整句話的意思、第二種方法則可以避免兩句話涵蓋的詞向量相去太遠但句向量相近的巧合，故將兩者同時考量可能可以得到更多完整的資訊，進而提高準確率。實際測試下的確使準確率提升至 50~52%，而其中發現 β 設置約為 50~65 之間的值可使其表現最好。

在聽完 Top 10% 報告後，我們參考他組所提及的「Weighted Sum」方法取代原先使用的「詞向量平均」來產生句向量，也就是上表所述的「詞向量加權」，並同樣混合第二種方法來計算相似度，最後使得準確率上升至 52~54% 。此算法的初衷是認為出現頻率太高的詞所能提供的資訊相對較少，故我們推測使用此算法產生出的句向量能更聚焦在比較重要的詞向量上。

4. **訓練資料預處理的成效**

public 準確率	詞向量平均＋ 每個字兩兩計算	詞向量加權＋ 每個字兩兩計算
保留所有句子	51.501%	52.608%
刪除重複的句子	51.699%	51.976%
刪除重複或有 雙引號的句子	51.422%	52.513%

雖然我們推測刪除重複或有雙引號的句子，能使句意較完整且連貫，進而使詞向量的訓練結果進步，不過經過實驗我們發現其實並沒有顯著的效果，差異都在 1%內。我們推測是因為相對於全部約 75 萬筆的訓練資料來說，重複和有雙引號的句子分別只有 3 千多與 4 千多筆資料左右，所佔的比例不到 1%，因此並不會有多顯著的影響；而準確率的變動可能僅來自於 Word2Vec 函式中其他參數的隨機性。

另外由上表也可看出使用「詞向量加權」的確比使用「詞向量平均」的表現來得好。

5. **retrieval model 中單向、雙向、比例比較**

單向 GRU	雙向 GRU
45.612%	45.652%

一開始實作的 retrieval model 是只有單向的 GRU，表現普通。為了提升準確率，我們嘗試使用雙向 GRU，但發現準確率並沒有顯著的進步。

正反資料數 1:3	正反資料數 1:5
45.652%	48.300%

在聽完 Top 10% 報告後，我們發覺對 retrieval model 來說，訓練資料的正反比例很重要，故我們將一開始的 1:3 改成與測試資料一致的 1:5，發現準確率顯著提升，但還是沒有突破 cosine similarity model 的表現。

6. **cosine similarity 與 retrieval model 選擇**

由結果來說，retrieval model 的表現沒有 cosine similarity 來得好，雖然在 Top 10% 報告中，第一名組別使用 retrieval model 達到很好的表現，但我們礙於訓練時間及設備效能的限制，更改模型架構及調整參數並重複測試所帶來的時間成本遠高於其準確率進步的效益，故我們沒有繼續深入研究此 model；另一方面，使用 cosine similarity model 在這個主題上不需大量的運算時間即可獲得還不錯的準確率，是相對經濟實惠的選擇。

7. Future Works

雖然我們在這次的 project 中將 cosine similarity model 做到還不錯的表現，但就進步的空間而言，可能並不如使用 RNN cell 實作的 model，因此我們未來如果可以克服設備上的限制，我們會朝向發展以 RNN cell 甚至 CNN based 這些可能有前景的 model，透過研究 NLP 相關的論文，並使用更好更廣泛的 dataset 來做訓練，以期達到更好的成效。