題目:Final-TV-conversation
Team name: NTU_r06922093_網媒所黃鼻地大師
Members:
學號: R06922093 系級: 資工碩一 姓名: 陳禹齊
學號: R06942048 系級: 電信碩一 姓名: 林彥伯
學號: R06922096 系級: 資工碩一 姓名: 洪子翔
Work division:
word vector model: 洪子翔
Sequence to sequence model: 林彥伯
Data preprocessing + report: 陳禹齊

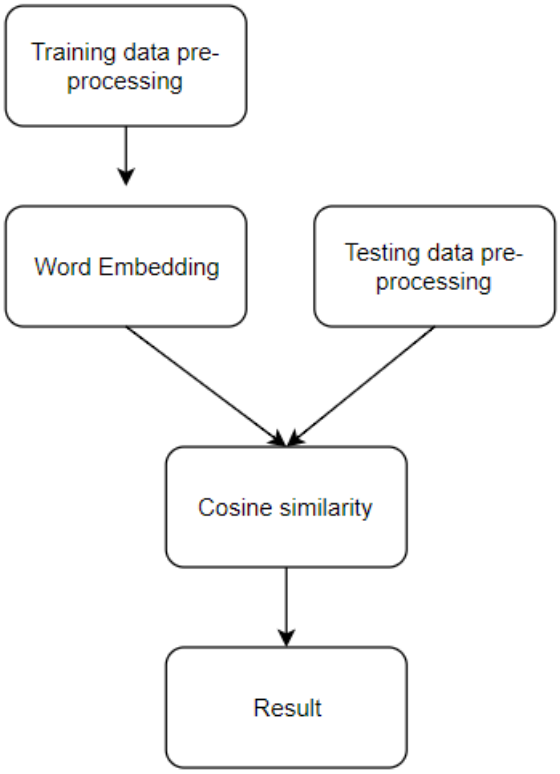
Preprocessing/Feature Engineering

在資料前處理這方面我們花了很多的時間在找出要如何切割和組合我們的 training data，才能讓我們訓練的 word vector 能夠很好的表現出詞與詞之間的相對關係，根據 HW4 的經驗，我們先將所有的標點符號以 python 內建的 Re 將所有的標點符號去除後，再以 jieba 分詞處理每一個句子，加入適當的 stopwords，將沒有意義的詞從 training data 中去除以提高 word vector 的品質，而 word vector 包含幸福來了這部電視劇中各個主角的名字的姓名、名字以及名字中的單詞，也加入了一些常見的 stopwords 像是"也"等等，去除後的結果能得到一些適當的提升。

至於在 testing data 上我們先將每個選項的句子切出後一樣以 jieba 分詞處理，若在訓練好的 word vector 中找不到切好的詞，就以 0 向量表示，也因為這樣的做法因此並沒有必要在 testing data 的前處理加上 stopwords，最後加總每一個選項中的向量再計算其平均值，最後再和問題的 word vector 計算 cosine similarity，並選出最高的選項做為答案。

Model Description

System overview



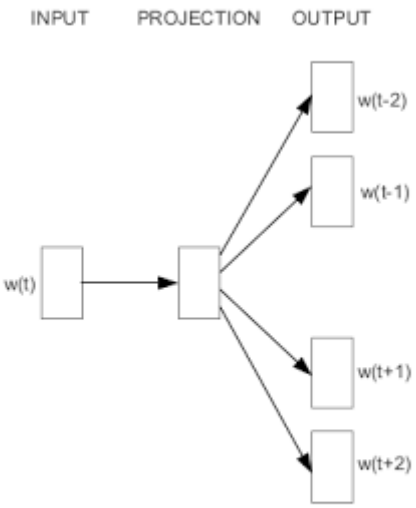
先利用 gensim 的 word2vector 將經過 preprocessing 的 data 做為 input，並經過實驗後挑選最好的 word2vector 中的參數設定。

而之所以選用 word embedding 來做為 vector representation 的原因是在於不管是在效率抑或是保留語意兩方面 word embedding 的效果都較 one hot 等方法好，而透過適當的 size 選擇，可以最佳化此 model 的向量維度大小，且利用 windows 可以考慮到前後語意之間的關係。

其中 gensim 的 wordvector 的實作又可以分為 CBOW 和 skip-Gram 兩種模型，兩個模型的不同之處在於 skip-Gram 是給定一個詞來預測該詞的上下文，而 CBOW 則是相反，給定前後文來預測應該要被填入的詞。

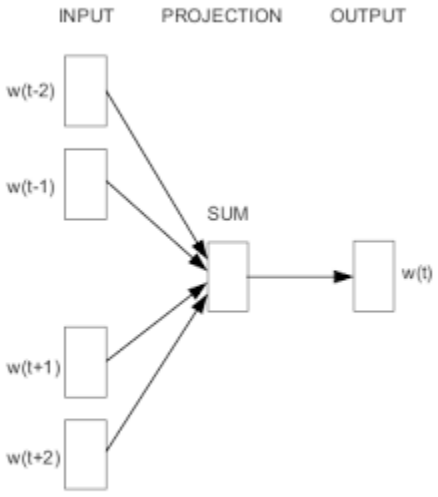
a. Skip-Gram

   先對每個經過 pre-processing 的詞以 one-hot 方式索引編碼，得到的每個詞都會是一個維度為辭典大小的向量，其中每個值都會是 0 或 1，而每個詞跟目標的距離也會是參數之一。示意圖如下：



Skip-gram

b. CBOW 和 skip-gram 恰好相反，CBOW 是將一段句子中的詞當做 ground-truth，而其上下文為 input，且句子的長度可以由自己控制



CBOW

c. Sequence to sequence
和前兩個 model 一樣，都需要利用到訓練好的字典，先到字典中比對前後兩個句子的 cosine similarity，若是兩者的 cosine similarity 大於 0.9 才將此句子加入 training data 中訓練，而 predict 則是將問題餵進 seq-to-seq model 中產生一段 output 後再以此 output 為基準和每個選項比較 cosine similarity，並選擇最高的做為答案。不幸的是在我們的訓練過程中此

seq-to-seq 一直訓練不好，推測是因為此 model 無法處理自問自答的問題，再加上若單純只以 cosine similarity 做為基準，有可能在訓練時上下兩句話並無關連但他們的相似度極高也會被加入訓練集當中，e.g. ['我們', '可以', '去', '看', '布袋戲', '了'] ['我們', '可以', '去', '看', '關公']，這兩句話雖然相似度很高，但是卻無上下文的關係。

比較此兩種不同 model 訓練的結果如下

	CBOW	Skip-gram	Seq-to-seq
accuracy	0.4425	0.4835	0.2322

由結果可以看出 skip-gram 在這樣的文本訓練中有較好的表現，也印證了 CBOW 需要在更大量的 training data 中才能更有優勢的說法。

Experiments and Discussion

主要針對訓練 word vector 的各種方式進行實驗與討論，而我們最後 ensemble 的方式也是以不同參數訓練出來的 word vector 所產生的結果來投票，以下將分別比較不同的訓練方式所導致不同的結果，而每次比較時只更動一個變數，其他未提到的變數固定如下

Data pre-processing	size	window	iter	negative	min_count
句子的處理方式	latent size	考慮前後幾個詞之間的關係	訓練的次數	Negative sampling 的參數	出現次數大於多少的詞才參與訓練
將所有句子前後串接. e.g. (1,2),(2,3),…	64	7	5	3	0

針對不同的 Data pre-processing，比較結果

Type	全部接成一個句子	兩句串接(連續)	三句串接(連續)	六句串接(不連續)	七句串接(不連續)	八句串接(不連續)
accuracy	0.2837	0.4835	0.4714	0.4756	0.4714	0.4688

由實驗可以發現，將所有文章接成一個句子在結果上完全壞掉，直觀上這樣的結果也相當合理，因為整個文本的大小遠遠超過一個句子的長度。

而將兩句連續串接的結果在實驗中可以得到最好的結果，之所以會有這樣做的動機是在於許多句子都是上下連貫的，而且有不少句子的長度是小於我們預設的 window，試過兩句之後理所當然接著試試看三句串接，但是結果並不如兩句串接好，因此就沒有再往下繼續試誤。

最後一種串接方式是基於觀察 training data 後發現整個文本是由一段一段的劇本所組成，可能有六到十句是基於同個主題的對話，因此一個簡單的想法是如果能將這些相關的對話分別切出來當做一個句子的話應該可以得到比較好的結果，但在實作上要做到相當困難，因此最後決定實驗每六到八句接成一個句子，可以發現六句接成一串的結果是最好的，因此推論 training data 中的劇本可能以六句為一個故事的機率最高。

針對不同的 latent size 比較結果

Latent size	32	64	96	128
accuracy	0.4646	0.4788	0.4835	0.4728

由實驗可以發現在 96 維的時候結果為最佳，推論在 32、64 維的情況下壓縮過度使得一些重要的 feature 已流失，而到了 128 維整個向量又太過稀疏。

針對不同的 window 比較結果

Latent size	7	10	20	30
accuracy	0.4835	0.4698	0.4712	0.4704

在 window 為 7 得時候有最佳結果，但隨著 window 上升，其實就算在 6 句合併的情況下有超過 30 個詞在同個句子內的情況也是不多見，因此在實驗中我們只測試到 window 為 30 的情況。

針對不同 iter 比較結果

iter	5	20	100	2000
accuracy	0.4835	0.4783	0.4812	0.4679

由實驗結果可以發現當 iter 極大時，整體的 accuracy 並沒有顯著提升，反而略有下降，推估是由於 testing data 是由同學隨機亂出，和 training data 的相關程度有限，因此就算訓練的很貼近 training data，在 testing data 上反而造成 overfitting

針對不同 negative 比較結果

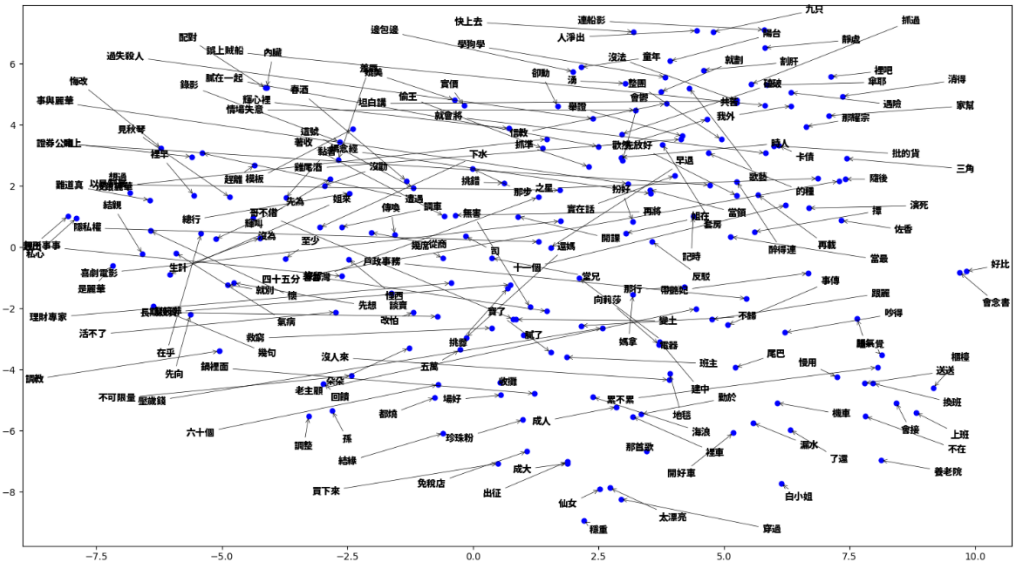
negative	3	4	5	20
accuracy	0.4781	0.4835	0.4689	0.4521

針對不同 min_count 比較結果

min_count	0	2	14	20
accuracy	0.4835	0.4614	0.4822	0.4763

由實驗可以發現把所有出現過的詞都加進訓練可以得到最好的結果，而透過實驗可以發現其實這個參數和結果的好壞並沒有一定的規律可循。

將 word vector 降維後可畫出下圖



為了觀察此 word vector 的正確性，以關鍵詞”安全”到 word vector 中搜尋相似度前 7 高的詞，列表如下

Rank	word	similarity
1	會派	0.7174065113067627
2	危險	0.6815862655639648
3	救出	0.6719944477081299

4	危險性	0.6422066688537598
5	勤務	0.6406013369560242
6	行動不便	0.6390127539634705
7	人身安全	0.6348111629486084

以關鍵詞”死亡”到 wordvector 中搜尋相似度 7 高的詞，列表如下

Rank	word	similarity
1	休克	0.8676049709320068
2	脈搏	0.8592756986618042
3	手術過程	0.8557049036026001
4	斷層	0.8530095815658569
5	輸血	0.8530054092407227
6	判定	0.8528964519500732
7	引流	0.8524845838546753

另外也透過詞語之間的關係確認此 word vector 的品質，例如高興之於搖尾巴，就像是難過之於什麼呢？在詞向量裡的計算就可以被表示成高興 – 搖尾巴 = 難過 -?，列出最高的五個候選者如下：

Rank	word	similarity
1	狗屎	0.7470333576202393
2	單挑	0.7354824542999268
3	不擦	0.7304470539093018
4	答話	0.724448025226593
5	舌頭	0.7213054299354553

由此結果可以觀察出似乎在語意的分析上在這個題目上面此 word vector 表現並不是很好。