

Conversations in TV Shows

▪ Team name

NTU_r06943153_球哥與他的夥伴們

▪ Members and work division (1)

學號	姓名	分工
R06943147	李明庭	word2vec model, tune parameters
R06943153	蘇旻彥	tune parameters, report, presentation
R06943084	游彥勝	tune parameters, produce answer
R06943118	賴俊丞	tune parameters

▪ Preprocessing / Feature Engineering (3)

在資料的 Preprocessing 當中，我們將讀入的五個 train.txt 檔案串在一起做為 training data，並使用中文斷詞套件 - jieba(結巴)將每個句子做切割、分詞。由於 jieba 是大陸的套件，預設分詞的字典是簡體中文，所以我們改變預設字典為斷繁體中文較為友善的 dict.txt.big。分詞套件 jieba 總共有三種不同的分詞模式，分別為：(1)全模式 (2)精準模式 (3)搜尋引擎模式，我們所使用的是預設的精準模式。

在分詞之後，我們使用了 stop words 的功能，引入 stopwords.txt，並將 training data 的句子中所有列在其中的單詞給去除。在 stopwords.txt 中所存放的詞為一些對文章主題無關緊要，與其他詞相比也顯得較不重要，去除掉也不會影響文義的詞，像是“你、我、他”這些代名詞，以及介係詞、語末助詞等。使用 stop words

可以讓之後的 word2vector model 訓練時可以將每個字與字之間的關聯訓練得更為精準。

▪ Model Description (7)

在這個題目當中，我們主要使用的是套件 `gensim` 中的 `word2vec` 功能來訓練詞彙，並使用兩種不同的方式在題目的選項中找到解答。以下會先敘述 `word2vec` 的實作，接著講解兩種不同的找解答方法。

1. word2vec

這是一個訓練詞向量(word vector)的套件，在訓練詞向量的過程，既是最簡單也是最困難的。簡單的是程式碼本身並不複雜，而困難的地方是各種參數的微調，使訓練出來的詞向量效能達到最高。在 `word2vec` 當中有許多不同的參數，這裡我們簡單介紹一下各參數的意義，而實驗結果會放在後面的段落。

(1) `size`：控制訓練出的 word vector 的維度。

(2) `window`：控制在訓練的時候一次能往前後看幾個字。

(3) `min_count`：每個詞出現的數字要高於這個參數才會視為訓練對象。

(4) `iteration`：訓練時更新參數的週數。

(5) `sg`：控制要使用 `cbow` 或是 `skip-gram model`，我們使用的是 `skip-gram`。

(6) `alpha`：也就是 learning rate，我們使用 default (0.025)。

2. Euclidean distance

在訓練完詞向量之後，我們接著要找出每個題目的答案。首先先把題目與選項使用 jieba 分詞後(處理方法和 training data 一樣)，將題目以及每個選項中的每個詞向量加總，做為這個句子的向量。接著要評估哪一個選項和題目的向量最接近，這裡我們使用的第一種方法是計算每個選項與題目的 Euclidean distance。

Euclidean distance 的原理是將每個向量視為在多維空間中的一個點，並計算兩點之間的距離，計算的公式如下：

$$d(x, y) := \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_n - y_n)^2} = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}$$

3. Cosine similarity

我們使用的第二種方法是估算題目與選項的向量之間的 cosine similarity，也就是餘弦相似度，這兩種方式的差異會在實驗結果的部分做說明。cosine similarity 的算法是直接計算兩個多維向量之間的 cosine 值，計算公式如下：

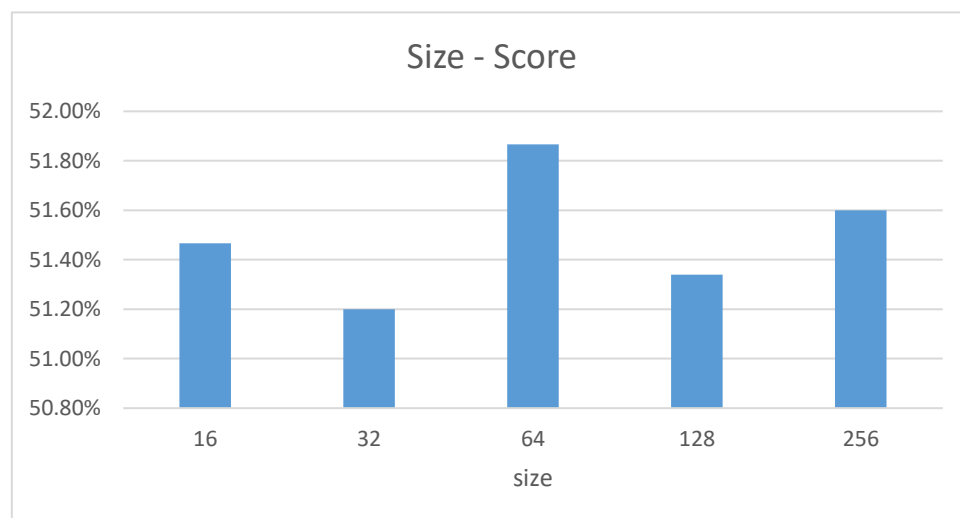
$$\text{similarity} = \cos(\theta) = \frac{A \cdot B}{\|A\| \|B\|} = \frac{\sum_{i=1}^n A_i \times B_i}{\sqrt{\sum_{i=1}^n (A_i)^2} \times \sqrt{\sum_{i=1}^n (B_i)^2}}$$

▪ Experiments and Discussion (8)

接下來將呈現我們所做的不同實驗結果，皆是以 cosine similarity 所預測的答案。礙於 Kaggle 一天只能上傳五次，所以我們取出 15% 的 testing data 並以肉眼辨識賦予 label，並在過程中順便觀察題目與答案的關聯性最為參考(很多題目根本都亂出阿...)。以下實驗作為評估的分數皆是以這 15% 的樣本所做的評估。另外，這些 label 並沒有參與 training 的過程，僅為了以下實驗所用。

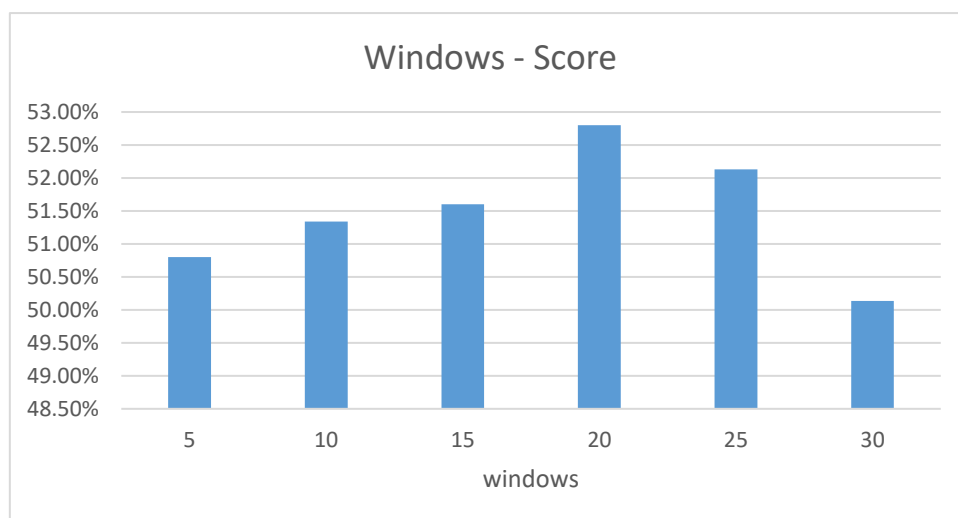
1. Size

使用 windows=10, min_count=1, iteration=50，並改變不同的 size 所做的實驗結果如下圖。可以看出在 size=64 的時候分數最高。



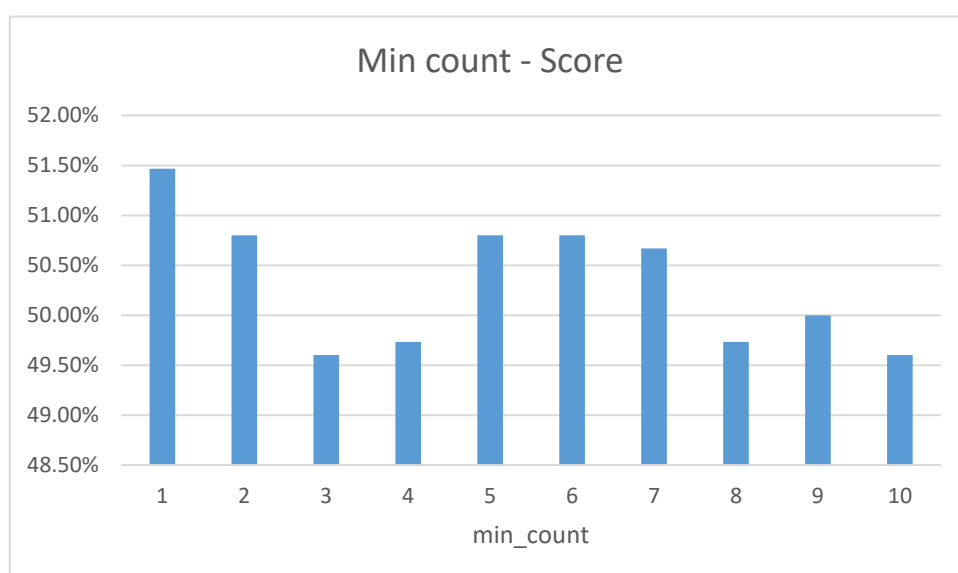
2. Windows

使用 size=128, min_count=1, iteration=50，並改變不同的 windows 所做的實驗結果如下圖。可以看出在 windows=20 的時候分數最高。Windows 所代表的是一次往前和往後看多少個字作為資訊來訓練，若太少則資訊不足，但是太多卻又可能找到相隔一兩句、比較不相關的字，使訓練成果變差。



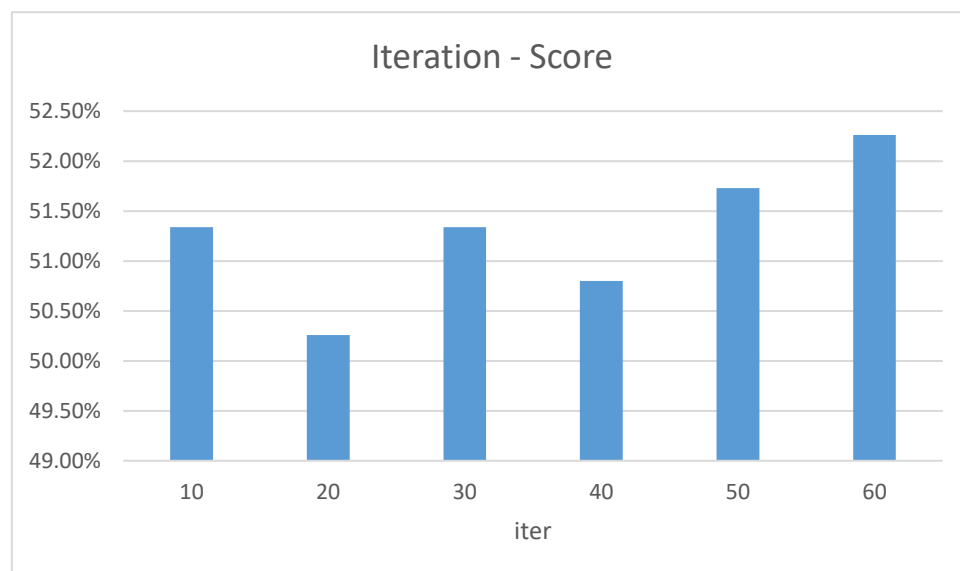
3. Min count

使用 size=64, windows=10, iteration=50，並改變不同的 min_count 所做的實驗結果如下圖。可以看出 min_count=1 的時候分數最高。Min_count 的原意是可以過濾一些出現次數極少的字詞，但由於已經使用 stopword 過濾了一些字，因此剩下的字應該都是相對重要的名詞、動詞等等，若調高 min_count 可能會過濾掉一些重要資訊，使訓練成績下降。



4. Iteration

使用 `size=64, windows=10, min_count=1`，並改變不同的 `iteration` 所做的實驗結果如下。雖然感覺 `iteration` 越高分數越高，但因為 `training time` 太高，加上分數並沒有明顯的變化，因此沒有繼續嘗試下去。



5. Others

接下來我們列出我們上傳到 **Kaggle** 所做的一些嘗試，包含一些完全失敗的做法以及探討可能失敗的原因。

(1) Euclidean distance

使用 `Euclidean distance` 的結果非常失敗，在探討數學公式的計算後，我們認為由於 `Euclidean distance` 直接計算兩點之間的距離，並沒有經過 `normalize`，而 `cosine similitaty` 是計算兩個向量的夾角，在計算的過程中有類似 `normalize` 的過程，因此結果可能較為準確。

	euclidean	cosine
Kaggle Public Score	0.10632	0.48063
size=128, windows=10, iteration=50		

(2) 不同參數的嘗試

以下是在決定要使用一開始的方法，取出 15% testing data 自己賦予

label 做實驗之前，分數較高的幾次結果。

size	windows	min_count	iteration	Kaggle Public Score
100	15	1	100	0.48695
100	15	1	50	0.46521
200	10	5	50	0.46521

(3) 過濾選項中和題目相同的字(關鍵!)

在最開始的時候，我們的組員提出了一個假設，若選項中出現和題目完全相同的字詞，會使這兩個句子的向量太過接近，因此若選項中出現了題目中出現過的字，則應該要將之去除。但是我們的分數始終沒辦法突破 0.5，於是我們嘗試不要去重疊的字，沒想到分數有了突破性的成長！

	有去除	沒去除
Kaggle Public Score	0.48063	0.52727
size=128, windows=10, iteration=50		

▪ Conclusion

最後我們用兩組參數做出的結果進行 ensemble 得到我們的最後結果，分別是由以上實驗推測出最佳的參數，以及我們在過程中胡亂嘗試出的最好成績(如下表(1)及(2)。Ensemble 之後的最高分數為 **0.53675**。

	size	windows	min_count	iteration	Kaggle Public Score
(1)	64	20	1	100	0.52292
(2)	128	10	1	50	0.52727