HW2 Report

109550167 陳唯文

1. Preprocessing

在這次的作業，我總共使用了下列 4 個 preprocess 的方式

i. 移除 stopword： 使用 nltks 內的 stopword 列表將句子中的部分單字去除
ii. 移除標點符號和 html 標籤 ： 將句子中的標點符號和 html 標籤如
 以空格取代
iii. 去除多餘的空白，並將每個字元都轉為小寫
iv. Lemmatization： 詞性還原，首先使用 pos_tag 判斷每個詞的詞性，再配合 nltk.stem 內的 WordNetLemmatizer 使用。還原後的範例結果如下
watching -> watch
ate -> eat
prettier -> pretty
cars ->car

Example sentence

Original :

One of the other reviewers has mentioned that after watching just 1 Oz episode you'll be hooked.

After Preprocessing :

one reviewer mention watch 1 oz episode hook

2. Number of Feature

Number of feature	F1-score	Precision	Recall
350	0.6885	0.6963	0.6907
500	0.7033	0.7108	0.7053
800	0.7194	0.728	0.7214
1000	0.7312	0.7412	0.7334
1250	0.7381	0.7487	0.7403

1500	**0.7468**	**0.7575**	**0.7489**
2000	0.6647	0.6791	0.6693

上面的表格為經過 preprocessing 後的資料更改不同數量的 feature 結果。如同上方表格所示，隨著 feature number 不斷升高，F1-score 的表現也越來越好，然後在 feature number = 1500 時，達到最佳的 F1-score。但當 feature number 增加到 2000 之後，F1-score 便大幅度下降。這可能是因為 feature 越多，越能夠幫助 bigram 模型判斷 sentiment，但當 feature 達到一定數量時，繼續增加的 feature 反而會變成多餘的資訊，並使 bigram model 無法準確的判斷。

3.Comparison

Model	preprocess	F1-score	Precision	Recall
Bert	0	0.9317	0.9324	0.9311
Bert	1	0.9026	0.9047	0.9027
Bigram	0	0.7057	0.7088	0.7065
Bigram	1	0.7033	0.7108	0.7053

從實驗出的結果可以發現 Bert 在每個方面的表現皆是遠勝於 Bigram 模型的。可能是因為 Bert 是一個深度雙向的模型，可以根據不同的上下文產生不同的詞向量，使它能夠更加精確的判斷一段話的情緒。而相較之下，我們寫的 Bigram 模型只是利用相鄰詞彙的機率，統計出機率較大的字詞組合，對於整段話的邏輯和語意辨識並沒有太大的幫助。從兩個模型的執行時間也可以知道 Bert 的計算量應該是遠遠超過 N-gram 模型的。

4.Discussion

a. Can bi-gram model outperform DistilBert ?

不行。由於 Bert 在進行訓練時還會考慮到上下文的關係，並能分辨具有不同意義的相同的詞彙，產生不同分類。而 bi-gram 模型只會考慮到前一個詞出現時的條件機率，對語意的辨識自然不如 Bert。

b. Can bi-gram consider long-term dependencies ?

不行。如同上一題提到的，bi-gram 模型只會考慮相鄰兩個詞出現的機率，因此當句子越長，越前面的詞彙對於當前詞彙的影響越會降低，而且 N-gram 模型也沒有透過前後文分辨語意的能力。

c. Would the preprocessing method improve the performance of the bi-gram model?

	Perplexity	F1-score
全部 preprocess	31.726	0.7033
不做任何 preprocessing	45.901	0.7057
保留 stopword	60.933	**0.7166**
不做詞性還原	21.942	0.7004
保留標點符號	38.271	0.6973

由上面的表格可以得知，不同的 preprocessing method 確實可以改變 bi-gram model 的表現，甚至可以透過採用不同組合的 preprocessing 方法得出最佳的 F1-score。而從實做出來的結果，發現當保留 stopword 時會有最佳的 F1-socre，可能是因為常用詞彙的增加，使判斷的結果較之前精確。保留標點符號時的 F1-score 則是最差的，從這裡可以推論去除多餘的資訊，也就是 preprocessing，對於模型的精確度增加確實有它的必要性。
但從結果也可以發現，perplexity 和 F1-score 的好與壞沒有相當的關連性，而這或許是因為 perplexity 的大小是用機率來判斷一個句子出現的可能性，也就是句子的通順程度，因此對判斷 sentiment 並沒有太大的關聯。

d. If you convert all words that appeared less than 10 times as [UNK], would it in general increase or decrease the perplexity on the previously unseen data compared

to an approach that converts only a fraction of the words that appeared just once as [UNK]? Why or why not?

Perplexity 會降低，因為將出現頻率較低的詞轉換成 [UNK]，使得相乘的機率提高，entropy 降低，最後的 perplexity 也會降低

此方法相較於將只出現一次的詞轉為[UNK]會更有效，因為將無效詞的範圍提高，會將更多低頻率的詞減少，相乘的機率也會因此提高。

5.Problem

在寫這次的 hw 一開始遇到的困難就是要先去搞懂 Bert 跟 N-gram model 的功能，使用的目的和演算法，才能慢慢了解我們這次 hw 的目標和應該要實行哪些 function，如果沒有先去查好資料，很容易不懂自己現在寫的東西應該具備什麼樣的功能。還有一開始在寫 part1 時不太了解 model 跟 feature 應該要回傳什麼東西，因此卡了一段時間，幸好網路上有蠻多資料在講解 n-gram，而且在寫的過程也會對整個模型更加熟悉，所以經過慢慢摸索之後，終於完成這次的 HW。