Data Mining Research and Practice Final Project: Movie Review Sentiment Analysis

組別: 第三組
組員: 莊順麟、廖家緯、張祐瑋、謝天霖、洪翊誠

March 9, 2022

Contents

Chapter 1

Introduction

1.1 Topic

電影評論語意分析 (Movie Review Sentiment Analysis)

1.2 Motivation

Machine Learning (ML) 為人工智慧的其中一個分支，透過現有數據自動學習，以發現其中規律，並推論預測結果。其廣泛應用於資料探勘、電腦視覺、自然語言處理、生物特徵辨識、搜尋引擎、醫學診斷、檢測信用卡欺詐、證券市場分析、DNA 序列定序、語音和手寫辨識、戰略遊戲和機器人等領域。其中 Natural Language Processing (NLP) 為人工智慧的子領域，專注於分析大量的自然語言數據，進而預測客戶情緒之判斷。早期 NLP 的作法是建立一套詞彙資料庫，用程式語言寫好人工訂定的規則，讓電腦依指令做出反應。但這種人工方式不可能包含所有語言的歧異性。1980 年 NLP 領域引入機器學習方法，不再使用程式語言指定電腦所有規則，而是建立演算法模型，讓電腦學會從訓練的資料中，尋找資料所含的特定模式和趨勢。NLP 主要應用於語音辨識、自然語言理解、機器翻譯、自然語言的生成、語意分析... 等。

語意分析出現在現實許多場景，比如做一個電商網站，賣家需要時刻關心使用者對於商品的評論是否是正面的。再比如做一個電影的宣傳和策劃，電影在鍵盤俠們中的口碑也至關重要。網際網路上關於任何一個事件或物品都有可能產生成千上萬的文字評論，如何定義每一個文字的情緒是正面或是負面的，是一個很有挑戰的事情。挑戰體現在以下幾個方面，區別結構化資料、評論資料的長短不一、很難限定到固定的維度且很難通過某個詞判斷使用者的情緒。

1.3 Dataset

我們使用的資料集來自 Kaggle 的 Bag of Words Meets Bags of Popcorn [6]。總共有 50,000 條 IMDB 電影評論以及對應的情緒評分。情緒評分為二元分類，1 表示正面情緒，0 表示

負面情緒，兩種情緒各有 25,000 筆資料。因為計算資源有限，我們從中 10,000 筆資料出來分析，並維持 1 與 0 類別的比例。

1.4 Flow Chart

我們將資料前處理及實作方法繪製成 Fig. 1 流程圖:

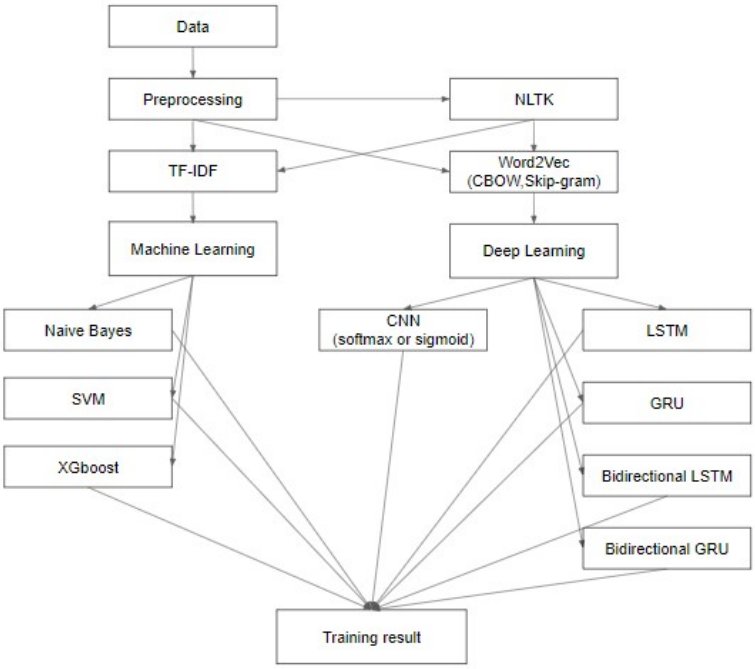

Figure 1. Flow chart of our method

1.5 Using Package

以下為我們這次報告所主要使用的套件:

- 資料處理與視覺化: pandas, matplotlib, seaborn
- NLP: nltk, gensim
- machine learning: scikit-learn, xgboost
- deep learning: tensorflow, pytorch
- 其他: multidict, joblib, pyspark

nltk 與 gensim 為 NLP 常用套件，我們使用 nltk 去除 stop word，並用 gensim 訓練 word2vec model。而 scikit-learn 是常見的 machine learning 的套件，而 tensorflow 與 pytorch 分別為 google 及 facebook 所開發的 deep learning 套件。通常學界喜好 pytorch，而業界喜好 tensorflow，在這次報告中，我們使用 tensorflow 訓練 CNN、LSTM、GRU... 等 model。此外，我們實作 NLP 領域 state-of-the-art 的 model，並採用 pytorch 框架。我們使用了 BERT pretrain model 做 fine-tune 與傳統 ML、DL 方法進行比較。

Chapter 2

Data Exploration & Preprocessing

2.1 Splitting Dataset

在分割資料集時，我們使用 holdout method，確保 training set, validation set 以及 test set 中的 0, 1 類別的比例一致。我們將 dataset 分成 training 64%, validation 16%, test 20% 的比例。Tab. 1 呈現 training, validation test set 中 data 的數量，Fig. 2 呈現三者所佔的比例。

label	training	validation	test	total
0	3200	800	1000	5000
1	3200	800	1000	5000
total	6400	1600	2000	10000

Table 1. 資料集數量

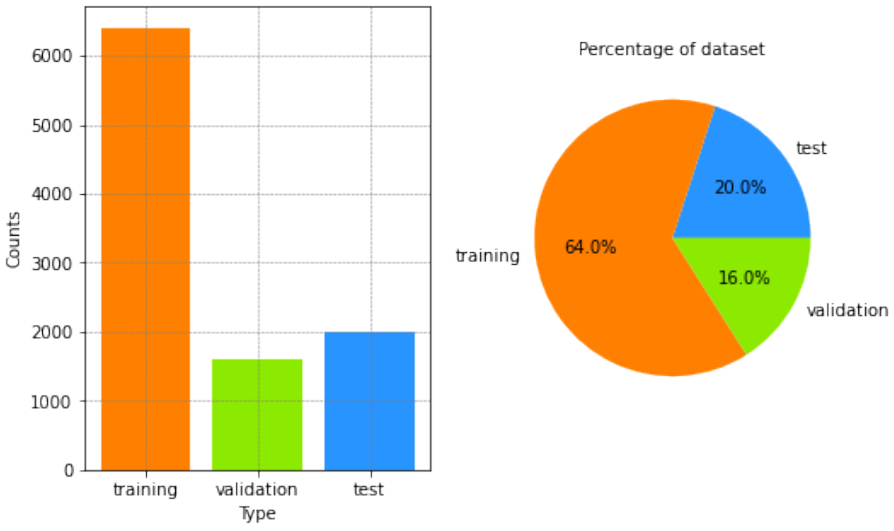

Figure 2. 資料集分布

2.2 Data Preprocessing

以下我們將文本進行前處理，原始評論內文如下:

> The film starts with a manager (Nicholas Bell) giving welcome investors (Robert Carradine) to Primal Park. A secret project mutating a primal animal using fossilized DNA, like Jurassik Park, and some scientists resurrect one of nature's most fearsome predators, the Sabretooth tiger or Smilodon.

我們使用以下兩種方式處理:

1. Ourselves
    - 句子分割
    - 去除標點符號與特殊符號
    - 大寫轉小寫
2. NLTK: 自然語言處理套件，去除 stop word

Ourselves	**NLTK**
the film starts with a manager nicholas bell giving welcome investors robert carradine to primal park a secret project mutating a primal animal using fossilized dna like jurassik park and some scientists resurrect one of nature's most fearsome predators the sabretooth tiger or smilodon	film starts manager nicholas bell giving welcome investors robert carradine primal park secret project mutating primal animal using fossilized dna like jurassik park scientists resurrect one nature fearsome predators sabretooth tiger smilodon

2.3 Exploratory Data Analysis (EDA)

經過兩種前處理方法後，我們利用柱狀圖呈現所有評論的單字量。透過圖表，我們可以清楚了解單字量的分布。去除離群值的後，我們標示 5 條鉛直線，黑色代表最小值與最大值，紅色代表第一四分位數與第三四分位數，藍色則是中位數。從 Fig. [3, 4] 我們可以發現，整理完之後的資料分布有些許不同，我們可以觀察到經過 NLTK 套件處理的單字量比用我們自己的方法還要短，前者範圍約為 0 ∼ 1000，後者則為 0 ∼ 600。

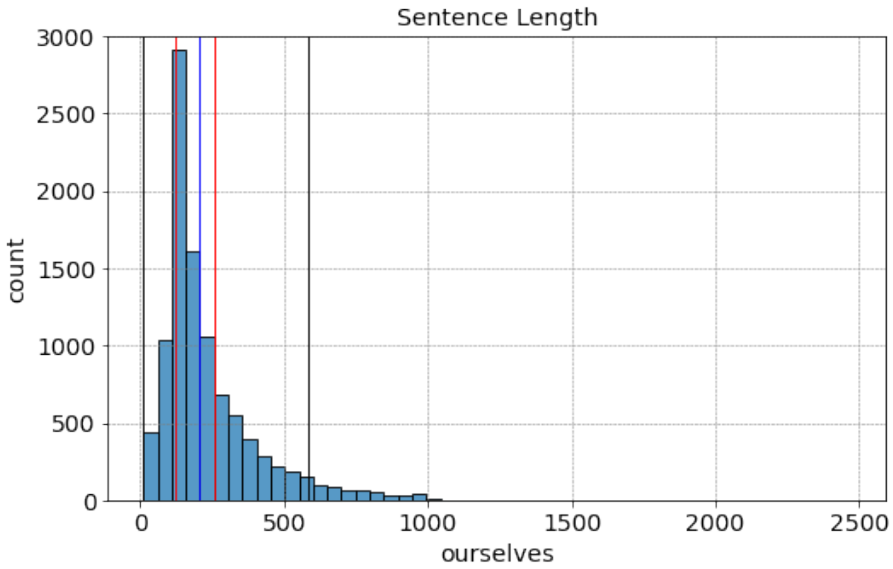

Figure 3. Length of sentence by ourselves

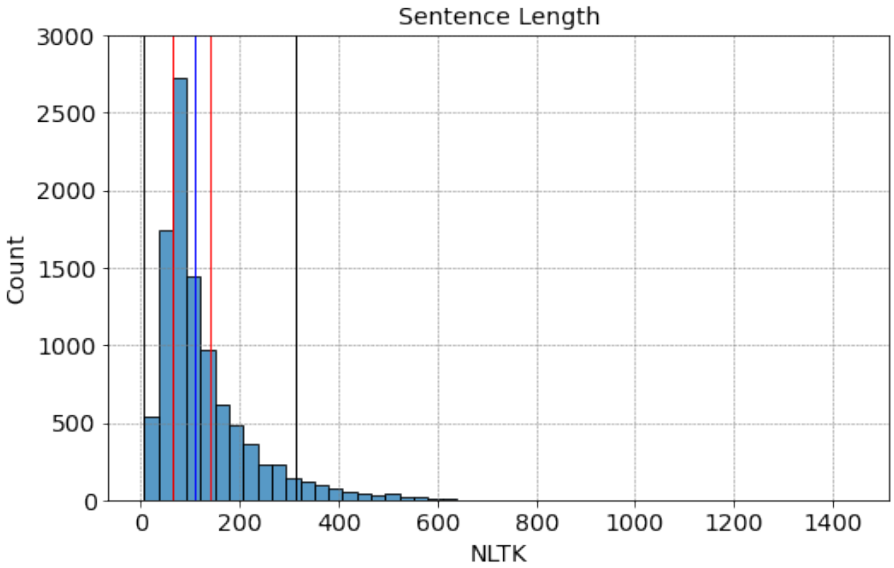

Figure 4. Length of sentence by NLTK

僅從評論長度看不出兩者資料處裡的明顯差異，我們分別統計兩種方法出現在所有評論中頻率最高的前 10 名單字。透過文字雲，我們可以快速捕捉文章中出現頻率高的單字，長條圖則呈現前十名數量間的差異。Fig. 5 中長條圖所呈現的都是沒有意義的一些單字，一般文章皆會不斷出現。而 Fig. 6 則是經過 NLTK 套件處理，幾乎都是具有意義的單字，並且是與情緒相關的形容詞。

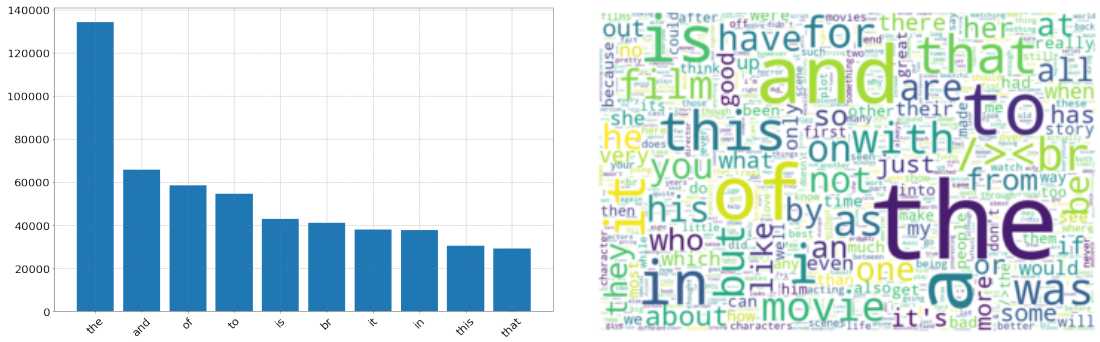

Figure 5. Top 10 highest frequency words by ourselves

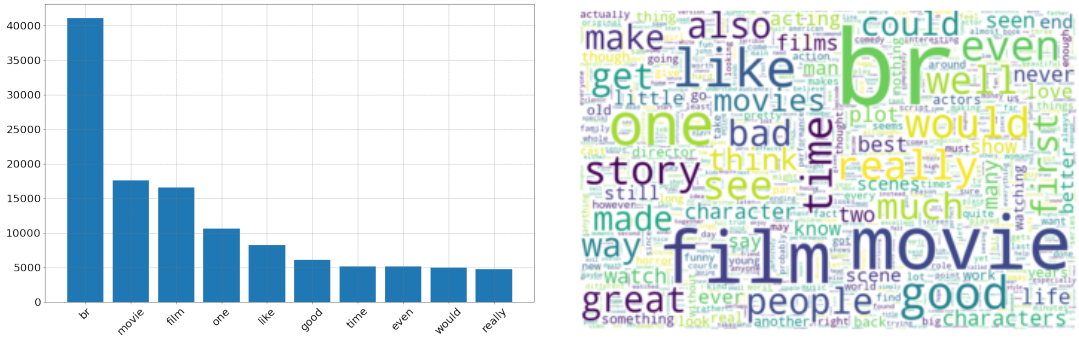

Figure 6. Top 10 highest frequency words by NLTK

Chapter 3

Approach

3.1 Word Embedding

在進入 training 之前，我們需先把 data 轉換成 AI 能讀取的型態，我們將這種文字轉換的方式稱為 word embedding。傳統 NLP 使用如 TF-IDF count-based 的統計方法，而近年來隨者 deep learning 爆炸性的發展，Google 在 2013 發表 prediction-based Word2vec 的 model [3]，並結合 RNN, LSTM... 等 deep learning 方法進行訓練。在這次報告中，我們使用 TF-IDF 搭配 machine learning 傳統方法與 Word2vec 搭配 deep learning 方法進行比較。以下簡單介紹 word embedding 的三個方法。

3.1.1 TF-IDF

設詞彙 t 出現在文件 d 中的次數為 $f_{t,d}$。

1. **Term Frequence (TF):**

$$\mathrm{TF}(t,d) = \frac{f_{t,d}}{\sum_{t=1}^{n} f_{t,d}}$$

在 Fig. 7 左圖中，計算 TF 值時，鎖定第 d 文件中的詞彙 t 的出現次數，除以 d 文件的所有詞彙數量。TF 值是 normalize 後的結果，透過 normalize 可以更清楚知道這個詞在某一個文件中所佔的篇幅比例，其目的能避免只有考量到次數決定重要性而忽略了該評論的長短。

2. **Inverse Document Frequency (IDF):**

$$\mathrm{IDF}(t) = \log \frac{m}{1 + |\{d \mid f_{t,d} > 0\}|}$$

在 Fig. 7 右圖中，計算 IDF 值時，鎖定詞彙 t 在所有文件 D 中出現的次數。取對數是為了不讓權重變化太劇烈，而分母加 1 可防止引入新資料時出現分母 0。

3. **TF-IDF:**

$$\text{TF-IDF}(t, d) = \text{TF}(t, d) \times \text{IDF}(t)$$

許多過於頻繁出現的詞彙，IDF 值非常小，因此 TF-IDF 可透過 IDF 來平衡 TF，解決 TF 過度關注的問題。

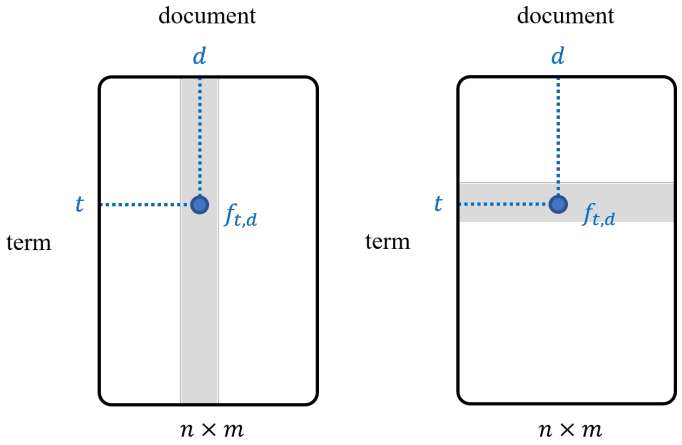

Figure 7. TF-IDF

Tab. [2, 3] 為 TF-IDF 矩陣的權重，行代表的每則評論，列代表對應文字的權重，我們可以觀察到 NLTK 前幾筆資料的權重皆比 ourselves 來得高，可由 Fig. [5, 6] 所解釋。

Word	Doc1	Doc2	Doc3	Doc4	Doc5
br	0.0358	0.0701	0	0.1052	0
movie	0.0176	0.0172	0.2562	0	0
film	0	0	0	0	0
one	0	0	0	0	0
like	0	0	0	0	0

Table 2. TF-IDF matrix by ourselves

Word	Doc1	Doc2	Doc3	Doc4	Doc5
br	0.04012	0.0835	0	0.1114	0
movie	0.0196	0.0204	0.2938	0	0
film	0.0208	0.0865	0	0.0865	0
one	0.0415	0.0431	0	0	0
like	0.0231	0	0.0576	0	0

Table 3. TF-IDF matrix by NLTK

3.1.2 Continuous bag of word (CBOW)

CBOW 的想法是利用 $w_{i-3}, w_{i-2}, w_{i-1}, w_{i+1}, w_{i+2}, w_{i+3}$ 來預測 w_i。input 是 one-hot encoding 的型式，而 output 是 w_i 的機率分布，可以把他視為 $p(w_i|w_{i-3}, w_{i-2}, w_{i-1}, w_{i+1}, w_{i+2}, w_{i+3})$ 的機率模型。

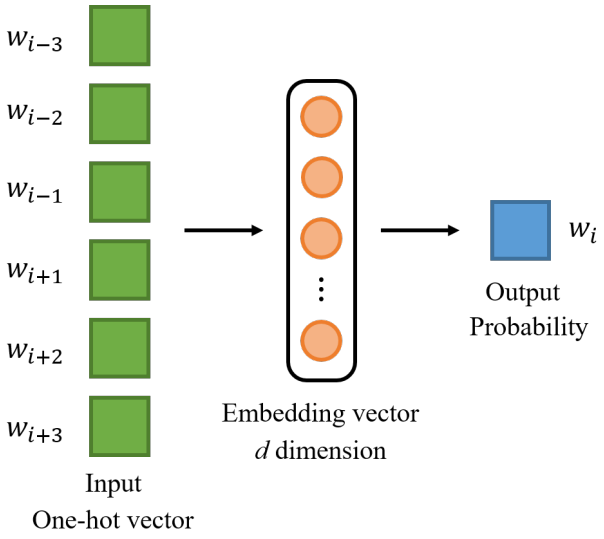

Figure 8. CBOW

3.1.3 Skip-Gram

skip-gram 則是利用 w_i 來預測 $w_{i-3}, w_{i-2}, w_{i-1}, w_{i+1}, w_{i+2}, w_{i+3}$。input 一樣是 one-hot encoding 的型式，而 output 是 $w_{i-3}, w_{i-2}, w_{i-1}, w_{i+1}, w_{i+2}, w_{i+3}$ 的機率分布，可以視為 $p(w_{i-3}, w_{i-2}, w_{i-1}, w_{i+1}, w_{i+2}, w_{i+3}|w_i)$ 的機率模型。

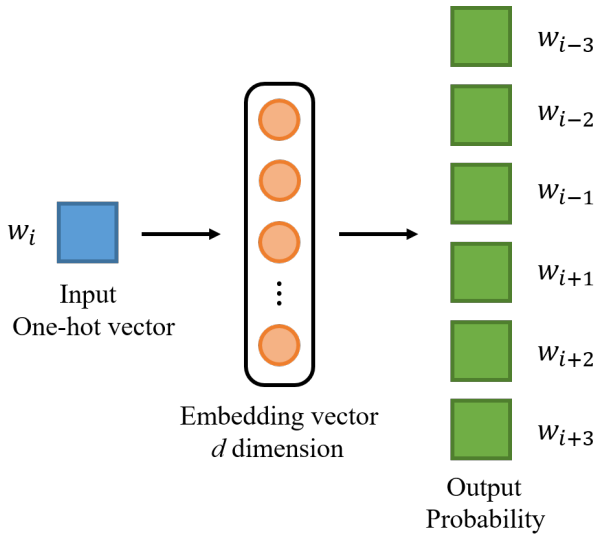

Figure 9. Skip-Gram

3.2 Machine Learning Model

3.2.1 Naive Bayes

Naive Bayes 是一個歷史悠久的分類方法，就是假設各特徵之間相互獨立，這使得樸素貝葉斯算法變得簡單，但有時會犧牲一定的分類準確率，很多高級自然語言處理模型也是從貝葉斯分類法演化而來。因此學習貝葉斯方法，是研究自然語言處理問題的一個非常好的切入點。

$$p(C|F_1,\ldots,F_n) = \frac{p(C)\ p(F_1,\ldots,F_n|C)}{p(F_1,\ldots,F_n)}.$$

3.2.2 Extreme Gradient Boosting (XGBoost)

XGboost 是基於 Random Forest 的演算法，並以梯度提升（Gradient Boost）為框架，在近幾年 ML 領域中大受歡迎。Random Forest 在多項實驗中被證明對於噪音比較大的分類問題上處理較差，噪音太多而導致劃分的節點下的分類劃分不好，而 XGboost 有內建的交叉驗證，可以自動迭代優化找出最優的參數，在每一個節點產生時去自動計算特徵增益，再去往上計算權重，並且因為並行計算，可以比 Random Forest 更快的方式找到最優參數。

3.2.3 Support Vector Machine (SVM)

SVM 是在特徵空間中間隔最大的線性分類器，而 SVM 的學習目標便是讓間隔最大化。目前支援向量機主要應用於模式識別得領域中，如文字識別、人臉識別；同時也應用在許多的工程技術和資訊過濾等方面。以下圖為

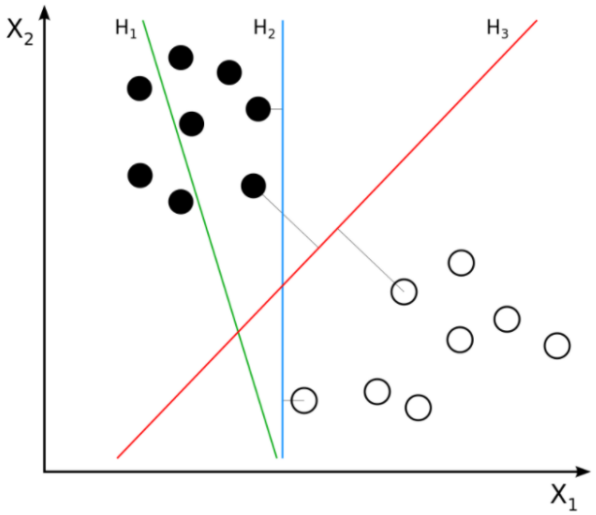

Figure 10. SVM

我們現在有 H1, H2, H3 三條直線，如果三條直線分別代表 3 個分類器的話，那麼很明顯地：H3 表現的最好。因為 H1 不能把類別分開；H2 可以，但只有很小的間隔；而 H3 能以最大間隔將它們分開。

3.2.4 Grid Search

在的訓練過程中，調參一直是一件非常困難的過程，這些參數代表著模型的配置變量，而隨著不同的超參數，訓練出來模型的結果也會不同。因此調參的方法除了手動的「佛系調參」，以及模仿大神在網路上已有的參數設置，其實還有兩種尋找最優參數的方法。其中之一的 Grid Search，就是透過回圈搜尋遍歷的每一個參數，直到找到最優的參數，而這個做法也相對耗時。若超參數較多時，則適合 Randomized Search，就是透過設定搜索次數，搜索超參數的特定數量隨機組合。

3.3 Deep Learning Model

3.3.1 Convolutional Neural Network (CNN)

CNN 是由 convolution layer、maximum pooling layer、fully connected layer、activation function 所組成的神經網路，以下我們簡單介紹每個 layer 的功能。

1. **Convolution layer:** 為一個平行的特徵圖，它通過在輸入圖像上滑動不同的卷積核並執行一定的運算而組成。此外，在每一個滑動的位置上，卷積核與輸入圖像之間會執行一個元素對應乘積並求和的運算以將捲積到的資訊投影到特徵圖中的一個元素。一張特徵圖中的所有元素都是通過一個卷積核計算得出的，表示一個特徵圖使用的權重與偏置式相同的。

2. **Pooling layer:** 為一種非線性形式的降採樣。有多種不同形式的非線性池化函式，而其中 maximum pooling 是最為常見的。它是將輸入的圖像劃分為若干個矩形區域，對每個子區域輸出最大值。直覺上，這種機制能夠有效地原因在於，一個特徵的精確位置遠不及它相對於其他特徵的粗略位置重要。池化層會不斷地減小資料的空間大小，因此參數的數量和計算量也會下降，這在一定程度上也控制了過擬合。

3. **Fully connected layer:** 在經過幾個卷積和最大池化層之後，神經網路中的進階推理通過完全連接層來完成。就和常規的非卷積人工神經網路中一樣，完全連接層中的神經元與前一層中的所有啟用都有聯絡。因此，它們的啟用可以作為仿射變換來計算，也就是先乘以一個矩陣然後加上一 bias。

4. **Activation function:**

   - Softmax：為歸一化指數函式，能將含任一實數的 k 維向量壓縮到另一個 k 維實向量中，使得每一個元素的範圍都在之間，並且所有元素的和為 1。
   - Sigmoid：也稱為邏輯函數，用在隱層神經元輸出，取值範圍在 $0 \sim 1$ 之間，可用於分類問題。

3.3.2 Long Short-Term Memory (LSTM)

LSTM 主要處理時間序列性質資料的神經網路模型，能夠同時考慮上一個時間點的數值計算進行計算，其特點為含有 Input gate、Memery cell、Forget gate 與 Output gate 四個單位，能夠控制數值的遺忘與記憶 (如下圖)，避免權重過大或是梯度消失的問題。

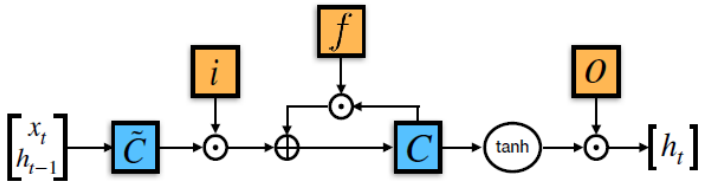

Figure 11. LSTM

3.3.3 Gate Recurrent Unit (GRU)

GRU 與 LSTM 的概念類似，但 GRU 主要含有 update gate 與 reset gate，相較於 LSTM 少一個單位，計算方法也有所不同，因此其優點在於執行速度較 LSTM 快，但分析的精確度並沒有明顯的差異。

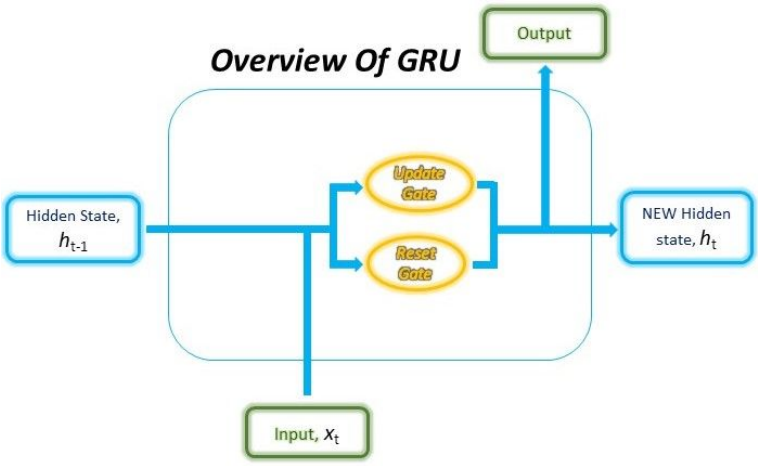

Figure 12. GRU

3.3.4 Bi-directional LSTM (BiLSTM) and Bi-directional GRU (Bi-GRU)

一般的 LSTM 與 GRU 屬於單向的分析模型，而 Bidirectional LSTM 與 GRU 提供雙向的分析，能將輸入資訊在過去的時間與未來的時間進行訓練，以文本分析為例，模型在一般文句從第一個字往後面分析，也從最後一個字分析到前面 (如下圖)，其優點在於更能夠藉此推斷語句之間的關係。

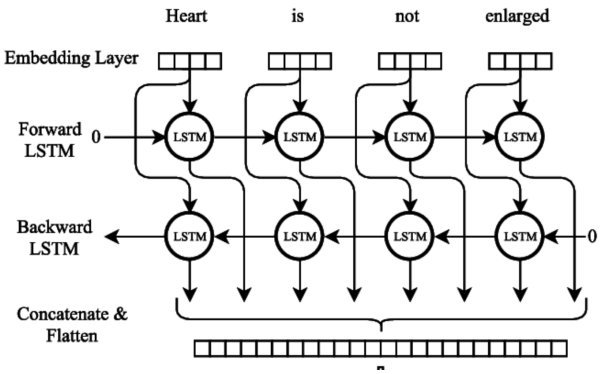

Figure 13. BiLSTM

3.4 State-Of-The-Art (SOTA)

接下來我們會簡單介紹 SOTA 的方法，由於方法過於複雜，我們僅點出 model 最核心的部分。

3.4.1 Transformer

Transformer model 為 Google 在 2017 年提出 [4]，其模型自帶 embedding layer，因此不須透過 word to vector 做處理。而在這個報告中，我們關注 encoder 的部分，其中包含 position encoding、multi-head attention、feed forward 等 module。

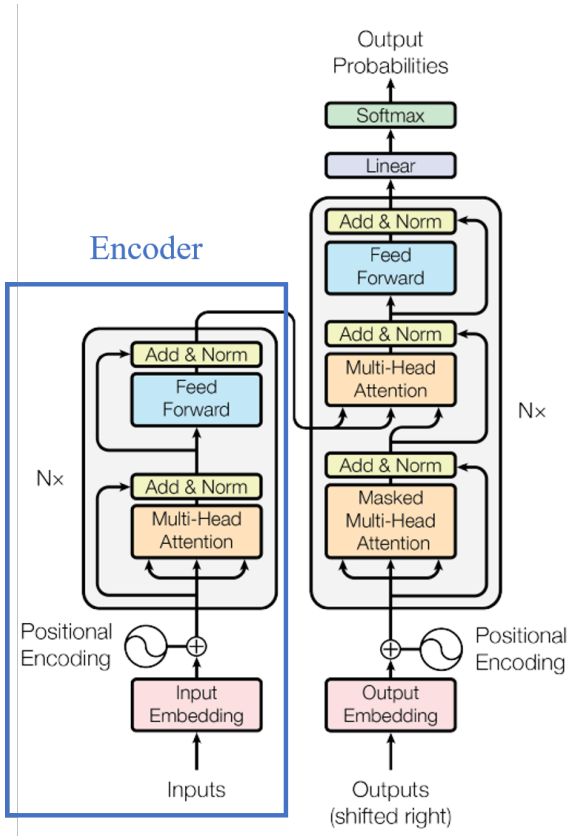

Figure 14. Transformer

Transformer 的核心在於 attention module，他透過內積運算比較 input 單字兩兩間的相似度，相似度高的相當於有較大的權重，反之權重較小，並重 softmax 將所有權種 normalize 到 0,1 之間。與 CNN 相比，CNN 受到 filter 局部性的限制，相近的單字才有機會做運算。而 Transformer 會將 input 兩兩做運算，透過內積計算相似度，為一種 non-local 的計算型式。

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V$$

其中 Q: queries, K: keys, V: values，d_k 為 keys 的 dimension

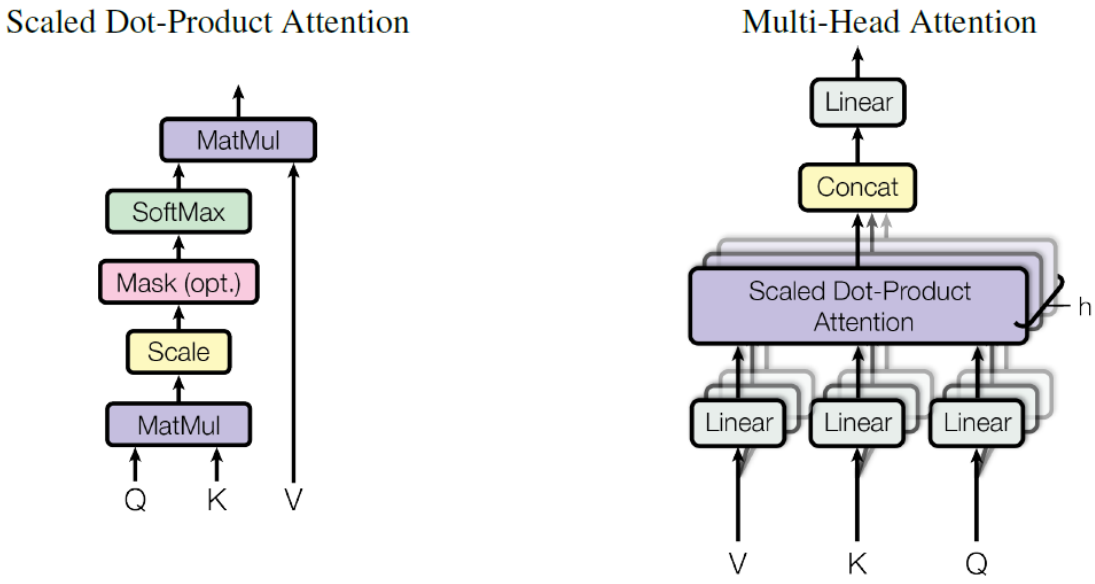

Figure 15. Attention module

3.4.2 Bidirectional Encoder Representations from Transformers (BERT)

BERT 為 Google 在 2018 年所提出，他前半部使用 transformer encoder，後半部接上一個線性分類器，通常我們會使用 pretrained 好的 model，並做 fine-tune 訓練。

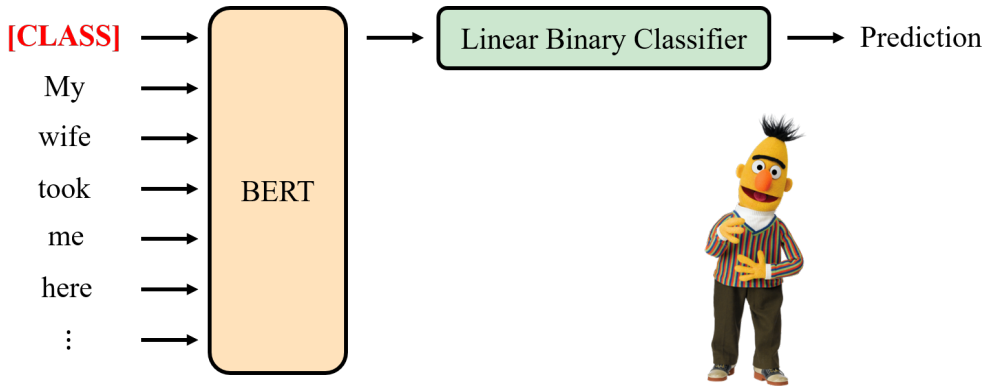

Figure 16. BERT

Chapter 4

Experiments and Result

4.1 Machine Learning Method

使用 TF-IDF 進行矩陣建構前，還有一項重要的參數需要調整: 最大特徵數個數。透過數值結果決定最大特徵數個數選取，在下方圖表可以觀察到 Naive Bayes 約莫 3,000~5,000 即可達到驗證集良好的效果，而 XGBoost 發現在 3,000 後就沒有在上升的趨勢，對 SVM 來說除了考量特徵數以外，還需要考量訓練時間，每當我提昇特徵數數量實，所需要計算的時間相較於其他兩者會大幅增加，所以保守觀察還是先以 8,000 下為主。

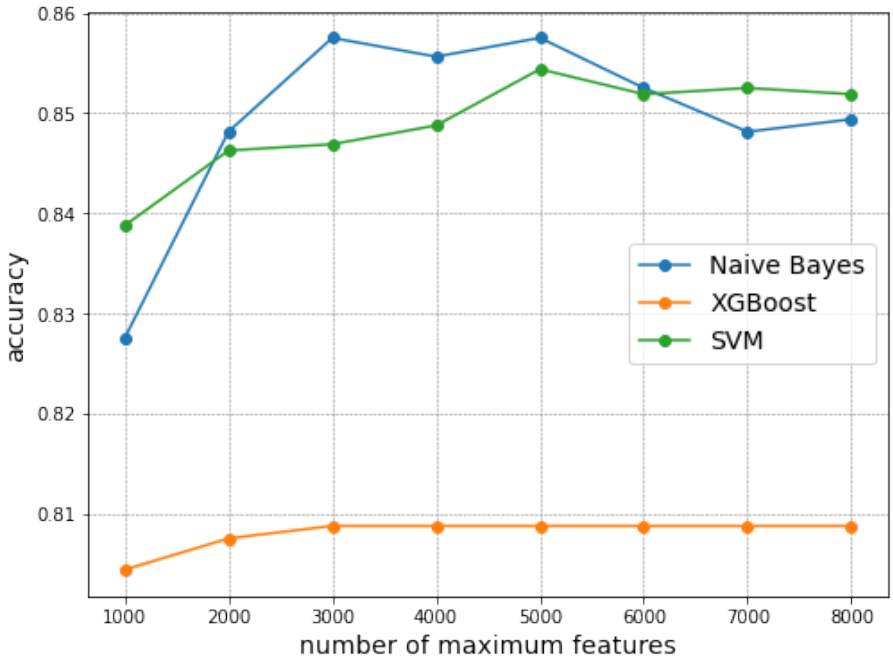

Figure 17. number of maximum feature

model	sentence	precision	recall	F1-score	accuracy
Naive Bayes	ourselves	0.8409	0.8510	0.8459	0.8450
Naive Bayes	NLTK	0.8669	0.8530	0.8599	0.8610
XGBoost	ourselves	0.7753	0.8350	0.8040	0.7965
XGBoost	NLTK	0.7755	0.8600	0.8756	0.8055
SVM	ourselves	0.8133	0.8540	0.8332	0.8290
SVM	NLTK	0.8341	0.8750	0.8541	0.8505

Table 4. Accuracy of ML method

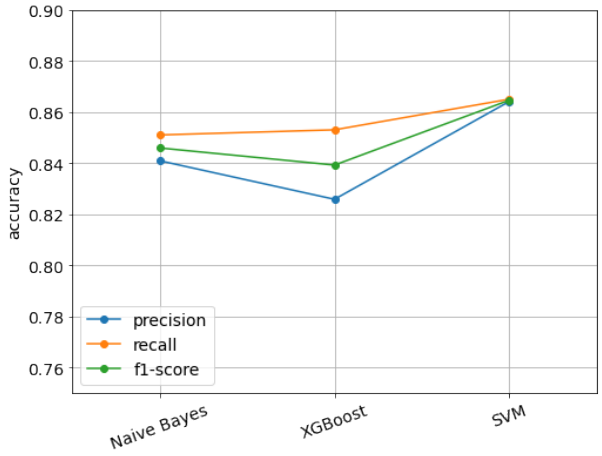

Figure 18. Ourselves

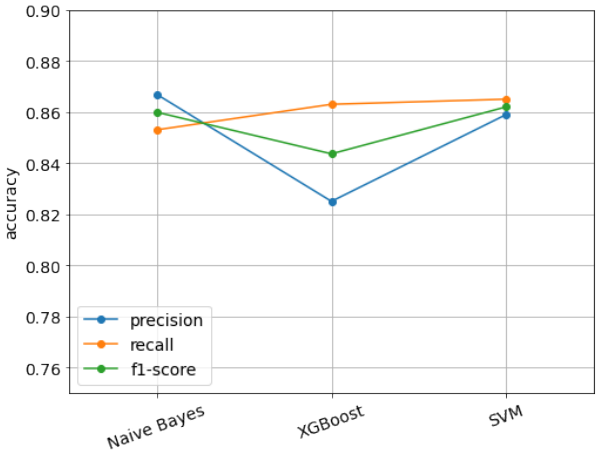

Figure 19. NLTK

從此結果可以得知，Naive Bayes, XGboost 在 NLTK 的表現比 ourselves 的方法好，而在 SVM 結果卻相反。比較三個 model 的 accuracy 以 SVM 的表現最佳。再看 precision, recall, F1 的數據，可知數據相當平衡，accuracy 的結果具有參考價值。

4.1.1 Grid search

model	sentence	precision	recall	F1-score	accuracy
XGBoost	ourselves	0.8258	0.8530	0.8392	0.8365
XGBoost	NLTK	0.8250	0.8630	0.8436	0.8400
SVM	ourselves	0.8641	0.8650	0.8646	0.8645
SVM	NLTK	0.8590	0.8650	0.8620	0.8615

Table 5. Accuracy of ML method with grid search

model	sentence	origin	grid search
XGBoost	ourselves	0.7965	0.8365 (+0.0400)
XGBoost	NLTK	0.8055	0.8400 (+0.0345)
SVM	ourselves	0.8290	0.8645 (+0.0355)
SVM	NLTK	0.8505	0.8615 (+0.0110)

Table 6. Comparison of origin and grid search

因 Naive Bayes 無超參數可調整，因此僅比較 XGBoost 與 SVM。從 Tab. 6 可知，使用 grid search 可顯著提升 accuracy。

4.1.2 Ensemble

就由先前最好的 Naive Bayes 模型搭配使用 grid search 找到最合適超參數的 XGBoost 和 SVM，進行投票，最終的結果

precision	recall	f1-score	accuracy
0.8676	0.8850	0.8762	0.8750

Table 7. Accuracy of ML method with ensemble

4.2 Deep Learning Method

4.2.1 CNN result and evaluation

embedding	activation	precision	recall	F1-score	accuracy
CBOW	softmax	0.7500	0.8250	0.7857	0.7750
CBOW	sigmoid	0.7716	0.7870	0.7792	0.7770
skip-gram	softmax	0.8201	0.8570	0.8381	0.8345
skip-gram	sigmoid	0.8538	0.8000	0.8260	0.8315

Table 8. Ourselves

embedding	activation	precision	recall	F1-score	accuracy
CBOW	softmax	0.7584	0.8320	0.7935	0.7835
CBOW	sigmoid	0.7661	0.8350	0.7990	0.7900
skip-gram	softmax	0.8228	0.8500	0.8362	0.8335
skip-gram	sigmoid	0.8543	0.8210	0.8373	0.8405

Table 9. NLTK

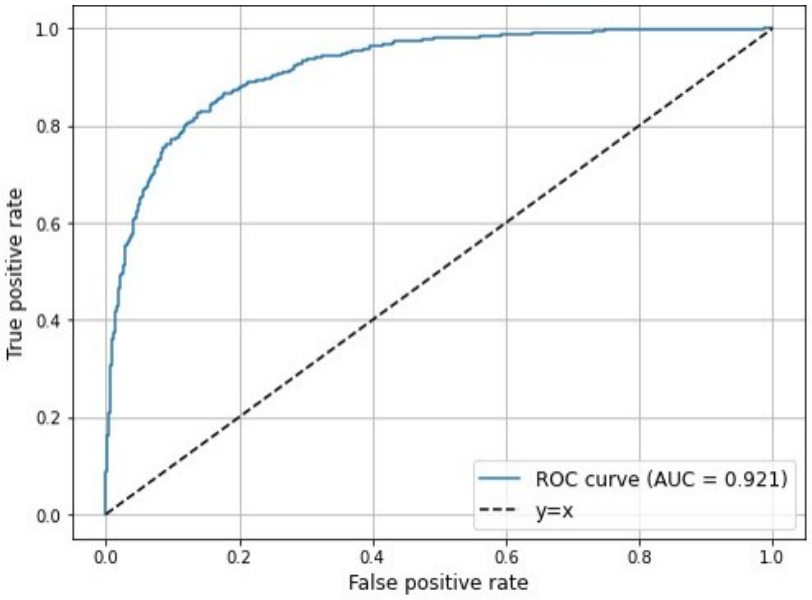

Figure 20. ROC curve of CNN

以下我們呈現 validation accuracy 的曲線，可以看出 model 收斂的過程。

在 Fig. 21 固定用 ourselves，比較使用激活函數 sigmoid, softmax 搭配預測模式 skip-gram, CBOW。可觀察到使用 skip-gram 的成效是比 CBOW 還要好的。而 Fig. 22 則是固定前處理方法使用 NLTK，那結果如同前者。

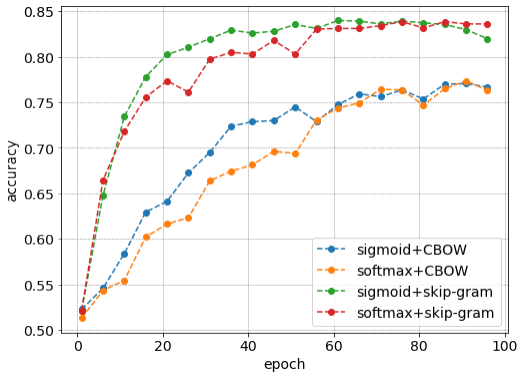

Figure 21. ourselves

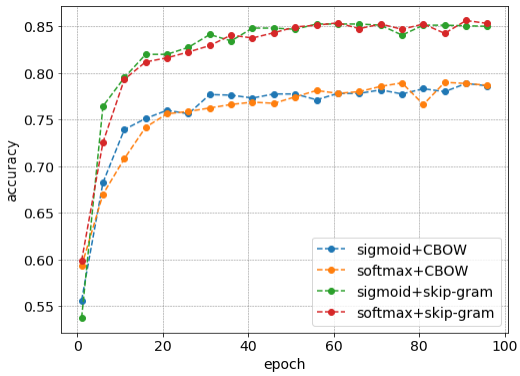

Figure 22. NLTK

Fig. 23 中，在使用 CBOW 則僅有 0.75∼0.8 區間，同時在 Fig. 24 固定 skip-gram 情況下，表現都是能到 0.8 ∼0.85 區間。

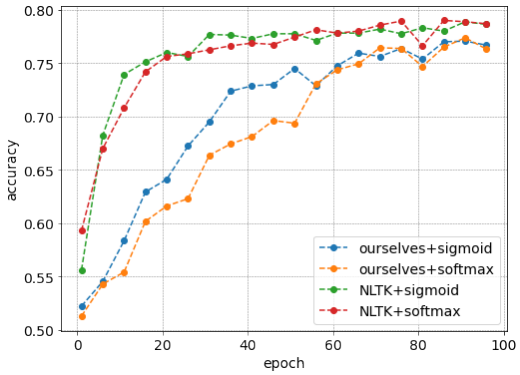

Figure 23. CBOW

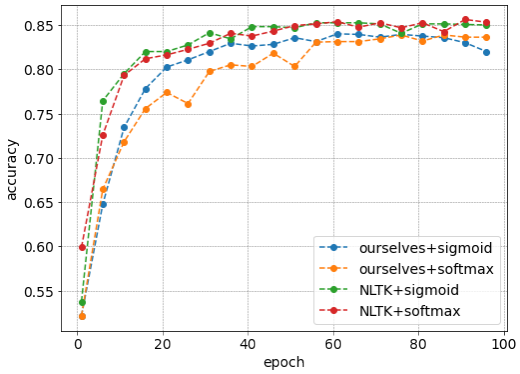

Figure 24. Skip-gram

最後在 Fig. [25, 26] 比較激活函數對整體表現的影響，發現在 NLTK+skip-gram 組合下，不管用哪一種激活函數效果差不多，比較明顯出現差異的則是在使用 ourself+CBOW 的組合上，對這組來說 sigmoid 會比較合適

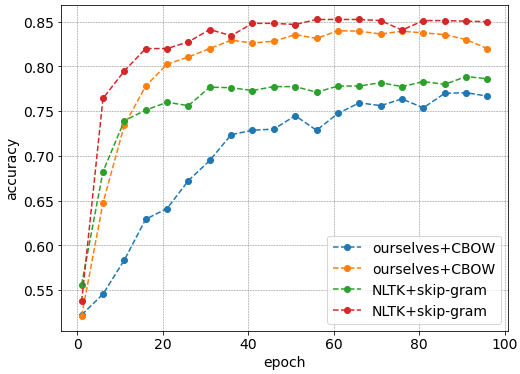

Figure 25. Sigmoid

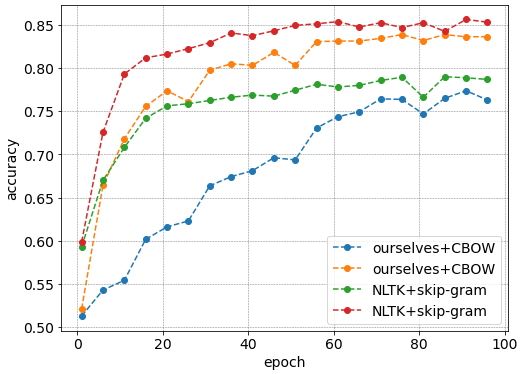

Figure 26. Softmax

4.2.2 LSTM and GRU result and evaluation

model	embedding	precision	recall	F1-score	accuracy
LSTM	CBOW	0.8854	0.6260	0.7335	0.7725
LSTM	skip-gram	0.8941	0.7260	0.8013	0.8200
GRU	CBOW	0.8519	0.6900	0.7624	0.7850
GRU	skip-gram	0.9127	0.6060	0.7284	0.7740
BiLSTM	CBOW	0.8454	0.7380	0.7880	0.8015
BiLSTM	skip-gram	0.8973	0.6990	0.7858	0.8095
BiGRU	CBOW	0.8377	0.7020	0.7639	0.7830
BiGRU	skip-gram	0.9120	0.6220	0.7396	0.7810

Table 10. Accuracy of LSTM、GRU、BiLSTM、BiGRU

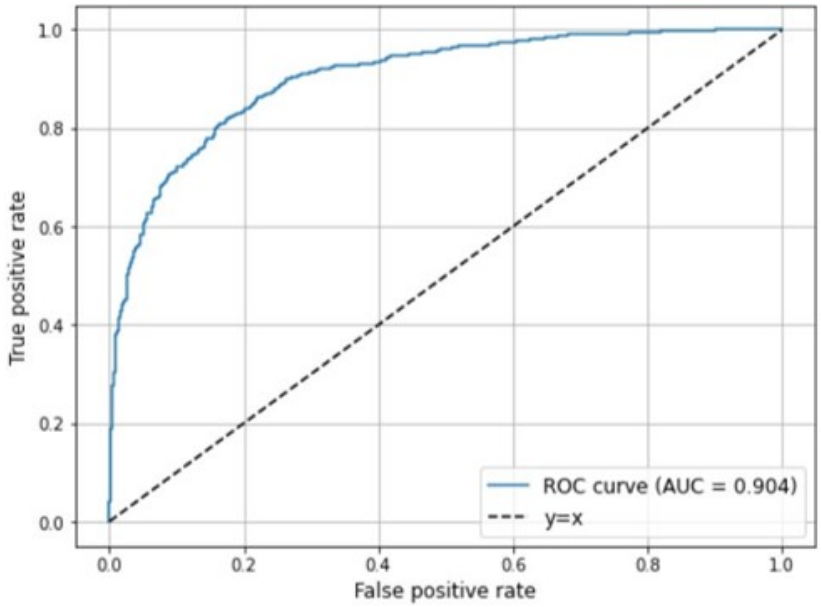

Figure 27. ROC curve of LSTM

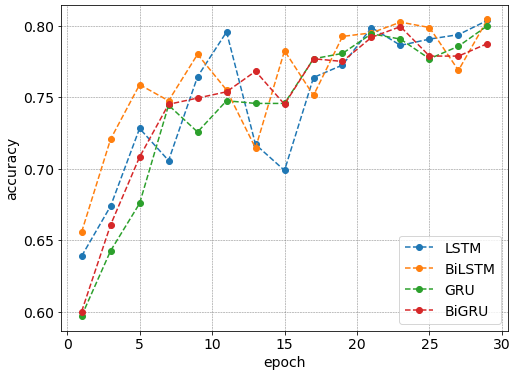

Figure 28. CBOW

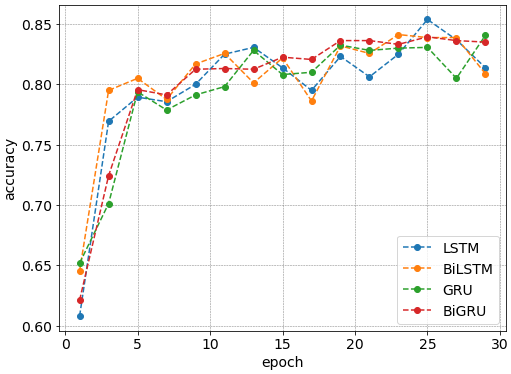

Figure 29. Skip-gram

根據上表結果，並沒有看出哪些特定條件的模型有明顯良好的表現。此外，利用 LSTM 等時間序列性的神經網路模型分析結果相較於其他模型的表現較差，推測可能原因為其他參數的需要調整，例如調整 dropout layer 的遺忘數值，讓重要的文句被記憶，不必要的文句遺忘，以免過多的無關的問句對干擾模型的分類判斷。

4.2.3 BERT Result

從 Tab. 11 可觀察到，BERT 為 SOTA 方法，其表現為所有 deep learning 方法中表現最優。因為 BERT 有自身的 embedgging layer，因此不需使用 word to vector，直接 input 句子即可。我們比較了沒有進行任何處理的句子、ourselves 以及 NLTK，可以發現到處理越乾淨的句子表現會越差，反而沒經過任何處理的原句，表現最好，這個其實和 BERT 本身訓練的方式有關 (這個比較複雜，因此沒特別提及)。另外，因為我們使用 pretrained model 做 fine tune，因此 Fig. 30 可以看到 validation accuracy 在前幾個 epoch 就來到 0.8 以上了，最高均可達到 0.87。

pre-processing	accuracy
None	0.8545
ourselves	0.8535
NLTK	0.8495

Table 11. BERT result

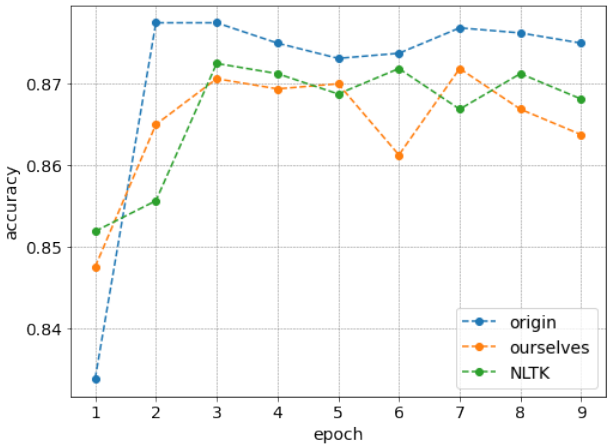

Figure 30. BERT

model	pre-processing	embedding	activation	accuracy
Naive Bayes	ourselves	tf-idf	×	0.8450
Naive Bayes	NLTK	tf-idf	×	0.8610
XGBoost	ourselves	tf-idf	×	0.8365
XGBoost	NLTK	tf-idf	×	0.8400
SVM	ourselves	tf-idf	×	**0.8645**
SVM	NLTK	tf-idf	×	0.8615
CNN	ourselves	CBOW	sigmoid	0.7770
CNN	ourselves	skip-gram	sigmoid	0.8315
CNN	NLTK	CBOW	sigmoid	0.7900
CNN	NLTK	skip-gram	sigmoid	**0.8405**
CNN	ourselves	CBOW	softmax	0.7750
CNN	ourselves	skip-gram	softmax	0.8345
CNN	NLTK	CBOW	softmax	0.7835
CNN	NLTK	skip-gram	softmax	0.8335
LSTM	NLTK	CBOW	sigmoid	0.7725
LSTM	NLTK	skip-gram	sigmoid	**0.8200**
GRU	NLTK	CBOW	sigmoid	0.7850
GRU	NLTK	skip-gram	sigmoid	0.7740
BiLSTM	NLTK	CBOW	sigmoid	0.8015
BiLSTM	NLTK	skip-gram	sigmoid	0.8095
BiGRU	NLTK	CBOW	sigmoid	0.7830
BiGRU	NLTK	skip-gram	sigmoid	0.7810
BERT	None	Transformer	softmax	**0.8545**
BERT	ourselves	Transformer	softmax	0.8535
BERT	NLTK	Transformer	softmax	0.8495

Figure 31. All of result

Chapter 5

Conclusion

在這份報告中，我們分析了電影評論的資料集，並使用多種方法進行預測。我們使用 TF-IDF 搭配 ML 方法及 Word to vector 搭配 DL 方法進行比較。我們挑選 ML 中具代表性的三個模型: Naive Bayes、XGBoost、SVM。Naive Bayes 算是較古老方法，速度上為最快，且不需要調參，在這次實驗中表現不錯；XGBoost 是 Kaggle 競賽中的常勝軍，但這次的表現較不亮眼；SVM 在這次實驗中的表現最佳，且背後具有深厚的理論支撐，可惜在訓練及預測上都需要花不少時間。另外，我們透過 grid search 方式得到最好參數，改善我們的 accuracy，並在最後做 ensemble，將 accuracy 提高至 0.8750，為這次實驗的最佳紀錄。

而在 DL 方法，我們使用了 CNN 與 RNN 家族系列的模型，並搭配不同的 Word to vector model，從結果上來看，skip-gram 優於 CBOW，和大部分人的經驗一致。而在 CNN 實驗中，我們在比較末端兩種不同的 activation function，第一種一般分類常用的 softmax，output 的是 0, 1 類別的機率分布，而第二種是 sigmoid，output 的是 0, 1 之間的機率值，其想法等同於做 logistic regression，結果可以發現接上 sigmoid 的表現略高於 softmax。然而 RNN 家族系列的模型在這次實驗表現中不如預期，我們推測以下幾種原因: 訓練不夠久、超參數設置非最優、優化遇到困難，希望未來能在這幾個關卡上突破。

除了 CNN 與 RNN 家族的模型，我們還實作 state-of-the-art 的方法，其效果也是 deep learning 中最好的，不過和 SVM 相比仍略遜一籌，因此我們認為在資料科學領域中，沒有最好的方法，只有適合的方法。

最後，我們對每個方法皆做了模型的評估，計算 precision、recall、F1-score 來衡量模型的好壞，以 ML 方法而言，precision 和 recall 的表現都還不錯，因此也得到不錯的 F1-score。若是 CNN model，大部分的 recall 高於 precision 不少，而 RNN 系列的則是 precision 遠高於 recall，可見模型在預測上已經偏掉了，不過從 ROC curve 來看，CNN 與 LSTM 表現最好的模型，AUC 都有上 90，表現還不錯。

關於 feaure work，我們希望可以做到的是 Explainable AI，許多 deep learning 方法雖然表現不錯，但他就像是個黑盒子，無法解釋現象，而 machine learning 方法較多具有解釋性，例如 tree-based 的方法，我們可以很明白知道他的分類依據，而又例如 SVM

具有深厚的數學理論支持，可以找出明確的 hyper-plane 做分類。因此我們也希望 deep learning 的方法也能有足夠的解釋性，不再是個黑盒子。而近年來許多學者發展出許多 Explainable AI 的方法，像是最知名的是對 input 做微擾動，觀察 loss 的變化，進而找出 model 預測的重要依據，諸如此類的方法都是我們未來想嘗試的。

Bibliography

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova, BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, Computation and Language, 2018.

[2] Laurens van der Maaten and Geoffrey Hinton, Visualizing Data using t-SNE, Journal of Machine Learning Research, 2008.

[3] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean, Efficient Estimation of Word Representations in Vector Space, Computation and Language, 2013.

[4] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin, Attention Is All You Need, Computation and Language, 2017.

[5] NLP: https://aiacademy.tw/what-is-nlp-natural-language-processing/

[6] Kaggle: https://www.kaggle.com/c/word2vec-nlp-tutorial

[7] SVM: https://www.itread01.com/content/1546204351.html

[8] Kernel SVM: https://medium.com/chung-yi/ml%E5%85%A5%E9%96%80-%E5%8D%81%E4%B8%89-svm-kernel-bcb4cfc64d19

[9] Word to vector: https://zh.wikipedia.org/wiki/Word2vec

[10] CNN: https://zh.wikipedia.org/wiki/%E5%8D%B7%E7%A7%AF%E7%A5%9E%E7%BB%8F%E7%BD%91%E7%BB%9C

[11] LSTM: https://ithelp.ithome.com.tw/articles/10223055

[12] GRU: https://ithelp.ithome.com.tw/articles/10242543

[13] BiLSTM: https://www.gushiciku.cn/pl/2EOL/zh-tw