Machine Learning Final Project

隊名：NTU_b03202047_百變怪

組員：物理三 王昱翔 李漪莛 盧靖 羅鈺凱

題目：Transfer Learning on Stack Exchange Tags

工作分配

- Model 1：李漪莛、王昱翔
- Model 2：盧靖
- Model 3：羅鈺凱
- 整合報告：王昱翔

Preprocessing、Feature Engineering

- Model 1
  1. 把所有科目的資料切割成四部分：id、title、content 和 tags。
  2. 預處理 title 和 content，包含：
     去除標點符號、 、 、網址（http 開頭等）和 stopwords。
     大寫轉成小寫。
- Model 2

  用 DictReader 將 test.csv 依照標頭存起來，分成 id、title 和 content。
  將整個 dictionary 丟進 for 迴圈裡 remove common stopwords 和 html 的開頭描述字元。
- Model 3

  主要跟 Model 1 的 preprocessing 一樣，另外還有使用 nltk 的 POS tagger，只留下名詞以及動名詞而已。

Model Description

- Model 1
  1. Count Common Words
     (1) unigram：計算去掉 stopwords 後的 title 最常出現的詞，取前 th1 名，存進“common words”中。
     (2) bigram：利用 nltk 套件，將 title 所有的詞兩兩利用 “-“ 連接起來，計算最常出現的雙字詞的前 th2 名，存進“common words”中。
     (3) 其中 th1、th2 先初始為 1000 和 300。
     (4) 事後遍歷各數量級的 th1、th2，再利用 training data 中六個科目取出的 tags，依照 Kaggle 的計分方式，計算出在 th1 = 1200、th2 = 600 時會有較高的正確率，以此值取代原先的 th1 和 th2。

2. Capture Tags

   把所有 title 中含有在 “common words” 中的單、雙字詞取出，最為最初版本的 tags。

3. Remove General Words

   (1) 此部分是在 test 這個科目找到 tags 後近一步的處理。

   (2) 另外六科目的 content 中，最常出現的 common words，取特定數目，從 test 中的 tags 去掉，可去除掉如：good、color 等字。
   若目標科目為 biology，則去除掉的字為除了 biology 外，另外六個科目的 common words；若目標科目為 cooking，則去除掉除了 cooking 外，另外六個科目的 common words。

   (3) 若從 biology 或 cooking 等科目去掉 general words，成效會變佳，然而將此法做在 test 上，Kaggle 的 score 反而稍差。(見下一部分 discussion。)

4. Reprocessing the output

   (1) 以下是把 Capture tags 所 output 的檔案再做處理。

   (2) 把所有的句子用 nltk 的套件去做 POS tagging。

   (3) 查看 output 的標籤在原句子中所對應的 POS 為何，如果標籤的 POS 是 MD, JJ, IN, VB, VBD, VBG, VBN, VBP, VBZ，則我們就把此標籤從 output 移除。

   (4) 把單數的標籤都換成複數的標籤。在每一個字的字尾都加上’s’，如果加上 s 後的字串有在原本的 output 檔案裡面，則新的標籤就是舊的標籤加上’s’的字串。

   (5) 在做處理的時候發現字如果有 Ö 會產生問題，後來發現原來是 schrÖdinger，所以把含有 Ö 的字都換成美式拼音，就變成 schroedinger（參考 https://en.wikipedia.org/wiki/Erwin_Schr%C3%B6dinger 所改）

   (6) 把舊的標籤用以上的步驟處理過後，輸出。

- Model 2

  1. 使用 nltk 裡的 FreqDist 將 title 和 content 分別計算做第一階段清除之後的最常出現詞語，將兩者的結果做 intersection 後放入 common 列表裡。

  2. 產生 Tags。

- Model 3

  1. 簡介

     此 Model 考慮在一篇文章內 Keyword 同時出現與其對應標籤的關聯性。將各 Keyword 之 Coherence 對於同主題內的文章做加總後，得知該主題

內重要的 Keyword 組合。特色如下：

(1) 不考慮詞語頻度而是考慮詞語間的關聯性，預測之標籤更具意義

(2) 可能抓出文章內不包含的字做為標籤

定量給出的關聯性統計是連續性的，而推算出來的標籤為離散性質（非有即無），其關係並不明確。在此便可使用 Off Topic 做 Training，得到「關聯性」->「標籤」的轉換關係。

2. 流程

(1) 透過 IDF，從眾標題與內容中取得該主題之 Keyword List，選定 N 作為 List Size.

(2) 對各文章之標題與內容製作 TF-IDF Word Bag 並 Normalize，作為該文章之 Feature

(3) 將 Feature 對自身進行直積，得到單篇文章之 Coherence List（Size 為 N*N）。將其對各文章進行加總並 Normalize，得到主題之 Coherence List。

(4) 以 Feature 與 Coherence List 作為 Input，透過一個 Unknown Function 得到標籤表（Boolean，Size 為 N）

(5) Output 各主題的預測標籤

3. Unknown Function 形式

其實上述的 Coherence List 即為一對稱之 Markov Matrix，而 Raw Feature 為 Markov Chain 的初始狀態。對於一個作為標籤的 Keyword，可預期其有多個相鄰的重要 Keyword，在 Markov Graph 上則對應到節點形成之 Cluster 與其中心。

對此，我們使用 Raw Feature 與其往後兩個 Markov 演化的 Sub-Feature，共三個向量作為文章之 Feature，考慮他們的線性疊加與平方項所得到之 Tag List。至此已成功將 Off Topic Tagging 抽象化並可進行 Training。

<u>實驗結果與討論</u>

- Model 1

1. 在各個階段的 Performance（顯示的為 Kaggle public score）

(1) 若只考慮 unigram（th1 = 1000、th2 = 0），score = 0.07760

(2) 若同時考慮 unigram 和 bigram（th1 = 1000、th2 = 300），score = 0.09679。

(3) 調整 th1 = 1200、th2 = 600 後，score = 0.10818。

(4) 移除有 A-B 句子中的 A 和 B 時，score = 0.11122。

(5) 移除 POS 是 MD, JJ, IN, VB, VBD, VBG, VBN, VBP, VBZ，score =

0.11365.

(6) 若利用 tf-idf 方法，成效較差，因為 tags 常常是那些在每個 title 中共同出現的詞。

(7) Remove general words：設定去除掉其他科目前 800 名的 general common words，score = 0.10272，成效稍微變差。（下部分有更詳細的數據比較）

(8) 如果做了 Reprocessing，可以大幅進步到 0.12299。(此為最終 Kaggle 上最高的成績。)

以上方法可以整理為下方的表格

方法	th1	th2	Kaggle
unigram	1000	0	0.07760
unigram、bigram	1000	300	0.09679
unigram、bigram	1200	600	0.10818
unigram、bigram、移除 A-B 句子中的 A 和 B	1200	600	0.11122
unigram、bigram、移除 A-B 句子中的 A 和 B、移除不符合的 POS	1200	600	0.11365
unigram、bigram、移除 A-B 句子中的 A 和 B、remove general words	1200	600	0.10272
unigram, bigram, 移除 A-B 句子中的 A 和 B, 移除不符合的 POS, 單數改複數，調整含Ö的字串	**1200**	**600**	**0.12299**

2. 不同 th1, th2 對於 training data 的影響

由於 th1、th2 的影響彼此獨立，因此一次調整一個變數去觀察正確率。此處的正確率指的是「全部找到的 tag 的集合」和「全部正確的 tag 的集合」的重複率，而並非單一筆資料找到的 tag 的正確率。其中：

p：precision rate

r：recall rate

s：f1 score = (2*p*r)/(p + r)

th1	th2		biology	cooking	crypto	diy	robotics	travel	ave
600	0	p	0.141	0.353	0.235	0.386	0.231	0.256	
		r	0.164	0.288	0.120	0.311	0.159	0.093	
		s	0.151	0.317	0.159	0.344	0.188	0.136	0.216
1000	0	p	0.112	0.306	0.194	0.311	0.168	0.233	

		r	0.231	0.416	0.165	0.417	0.192	0.140	
		s	0.151	0.352	0.178	0.356	0.179	0.175	0.232
1500	0	p	0.088	0.256	0.159	0.249	0.124	0.219	
		r	0.283	0.523	0.203	0.501	0.213	0.198	
		s	0.134	0.344	0.178	0.333	0.157	0.208	0.225
2000	0	p	0.075	0.222	0.131	0.208	0.099	0.212	
		r	0.337	0.603	0.223	0.558	0.228	0.256	
		s	0.122	0.324	0.165	0.303	0.138	0.232	0.214
0	300	p	0.129	0.196	0.156	0.179	0.090	0.153	
		r	0.067	0.080	0.040	0.072	0.031	0.028	
		s	0.088	0.114	0.064	0.103	0.046	0.047	0.0770
0	600	p	0.078	0.118	0.108	0.111	0.048	0.123	
		r	0.090	0.096	0.055	0.090	0.033	0.045	
		s	0.083	0.106	0.073	0.099	0.039	0.065	0.0779
0	1000	p	0.050	0.077	0.080	0.083	0.030	0.096	
		r	0.103	0.105	0.068	0.111	0.034	0.058	
		s	0.067	0.089	0.074	0.095	0.032	0.072	0.0715

3. 加上 Remove General Words 的影響

定義 thG 為去除掉的 general common words 數目（單字詞和雙字詞共同排名），則 thG = 0，表示沒有加上此做法時的 score，可作為比較基準；並且，在此部分，th1 = 1200、th2 = 600。

下表只顯示幾組較具代表性的數據：

目標科目	thG	score
biology	0	0.082
	500	0.086
	800	0.087
	1000	0.089
	1500	0.084
crypto	0	0.098
	800	0.150
	1000	0.152
cooking	0	0.182
	800	0.271
diy	0	0.173

	800	0.203
test (physics) score on Kaggle	0	0.11122
	800	0.10272
	1000	0.10197

從上表可看出，除了在 test 資料外，其他科目加入此做法處理後，準確度都會提高，推測 test 中的 tags 可能出現這些普遍存在於別的科目中的字。

4. 使用 POS tagging 以及單複數的影響

根據我們初步所預測的 tag，常常會預測到 will, red, blue 這種字，通常 tag 不會是助動詞或是形容詞，所以經過多次丟到 kaggle 上的結果後，決定把 POS 是 MD, JJ, IN, VB, VBD, VBG, VBN, VBP, VBZ 移除會得到最好的效果。 另外我發現 tag 裡面有單數跟複數的問題，曾經試過把複數改為單數(即字尾有 s 且去掉 s 後的字是有出現在 tag 裡面)，發現結果是變差的，所以後來一律都把單數改為複數。

- Model 2

1. 調整 stopwords 的來源，分別試了 nltk 內建 stopwords、standford NLP codebase 和 stopwords generated by Chris Buckley and Gerard Salton from Cornell University. 經過嘗試後以 Cornell 的效果最佳。
2. 檢視第一次產生的 tags, 發現有許多無意義的字眼無法在第一次 stopwords 時被去除。經過調整不同的 stopwords 後手動在最後增加了一個第二階段待去除的 wordlist. 最後這個方法在 Kaggle 上的 public score 是 0.0890。

我們人工所新增的 stopwords 如下：via, two, make, e, c, using, r, three, mu, eta, must, r, m, v。

- Model 3

1. 對於各 Topic 做初步測試

DF_THRESHOLD = 3

KEYWORD_SIZE = 1000

TAG_SIZE = 3

Topic	Articles	Mean F1 Score
Biology	13196	0.0937
Cooking	**15404**	**0.2754**
Crypto	10432	0.1347
DIY	25918	0.1864

Robotics	2771	0.1279
Travel	19279	0.1085

其中可見 Cooking 之成績最好，以下便以 Cooking 做參數測試。

2. DF_THRESHOLD

Test Topic = Cooking

KEYWORD_SIZE = 1000

Articles = 5000

TAG_SIZE = 3

DF_THRESHOLD	Mean F1 Score
3	**0.2928**
2	0.2928
5	0.2927
10	0.2888
20	0.2653
50	0.1968

此參數為 Topic TF-IDF 排名之 DF 閾值，2 與 3 差別不大，但往上調整時成績下降。可見有些重要 Keyword 只在 3~5 篇文章中出現。

3. KEYWORD_SIZE

Test Topic = Cooking

DF_THRESHOLD = 3

Articles = 5000

TAG_SIZE = 3

KEYWORD_SIZE	Mean F1 Score
1000	0.2928
500	0.2641
200	0.2133
2000	**0.3064**
4000	0.3041

此參數為 Topic TF-IDF 排名表之大小，決定了實際被採用的 Keyword 數量。其中 2000 時成績最佳，4000 卻反而下滑。可見若包含太多 Keyword，可能有些不重要字也能上排行榜，模糊了 Coherence List。

4. TAG_SIZE

Test Topic = Cooking

KEYWORD_SIZE = 1000

DF_THRESHOLD = 3

Articles = 5000

TAG_SIZE	Mean F1 Score
3	0.2928
2	**0.3273**
1	0.3244
4	0.2584
5	0.2290

此參數為輸出 Tag 之平均數量。當輸出數量太多或太少時，可能會擺入無關字眼或者放不進有信心的字眼，導致分數下降。

5. Some more tests

   Test Topic = Cooking

   DF_THRESHOLD = 3

   Articles = 5000

KEYWORD_SIZE	TAG_SIZE	Mean F1 Score
2000	2	0.3328

6. Kaggle

   Test Topic = Test (Unknown)

   KEYWORD_SIZE = 2000

   DF_THRESHOLD = 3

   Articles = 5000

   TAG_SIZE = 2

   Kaggle Mean F1 Score = 0.07007