

# Machine Learning

## HW4

r05922018  
黃柏智

### 1. Analyze the most common words in the clusters

我處理 raw data 的方法是把所有 terms 當中，不是英文字母的符號去掉，再把每個 term 都轉成小寫。例如 “MATLAB for kids?” 這個 title 會先被切成 “MATLAB”, “for”, “kids?” 三個 terms，然後去掉非英文字母並轉成小寫，最後這個 title 會變成 “matlab for kids”。經過這個轉換後，再來分析每個 term 出現的頻率，會得到以下的結果：

the	to	i	a	in	is	and	of	this	that	it	...
91517	72535	63360	57626	36435	34560	33243	27632	22009	21958	21663	

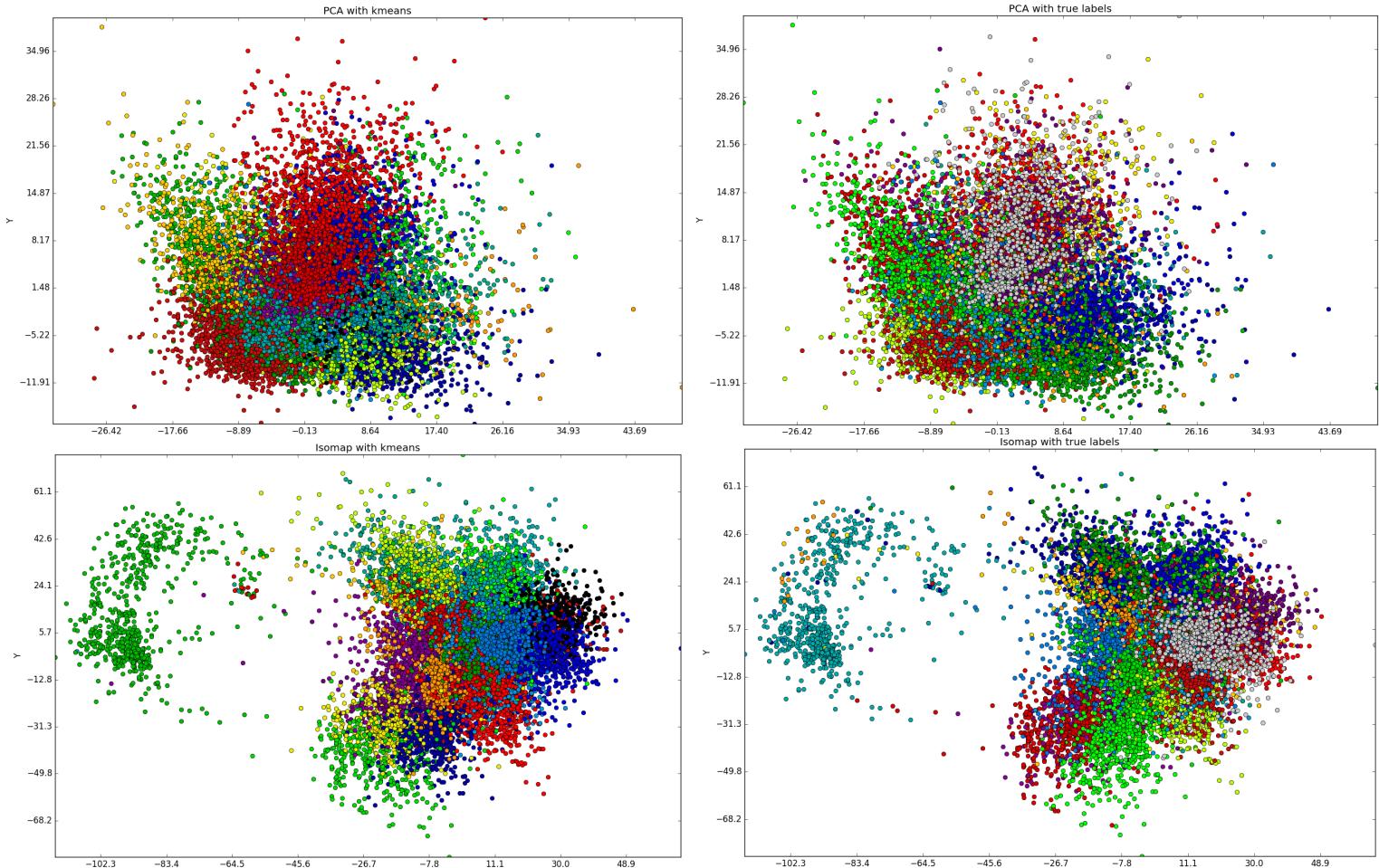
在整個 corpus 出現太高的字必須要過濾掉，不然會影響準確度。除了根據上面的結果之外，我也參考了 RANK NL 與 XPO6 的 stopwords，最後整理出一份 stopwords 列表，存成 `stopword.txt` 這個檔案。

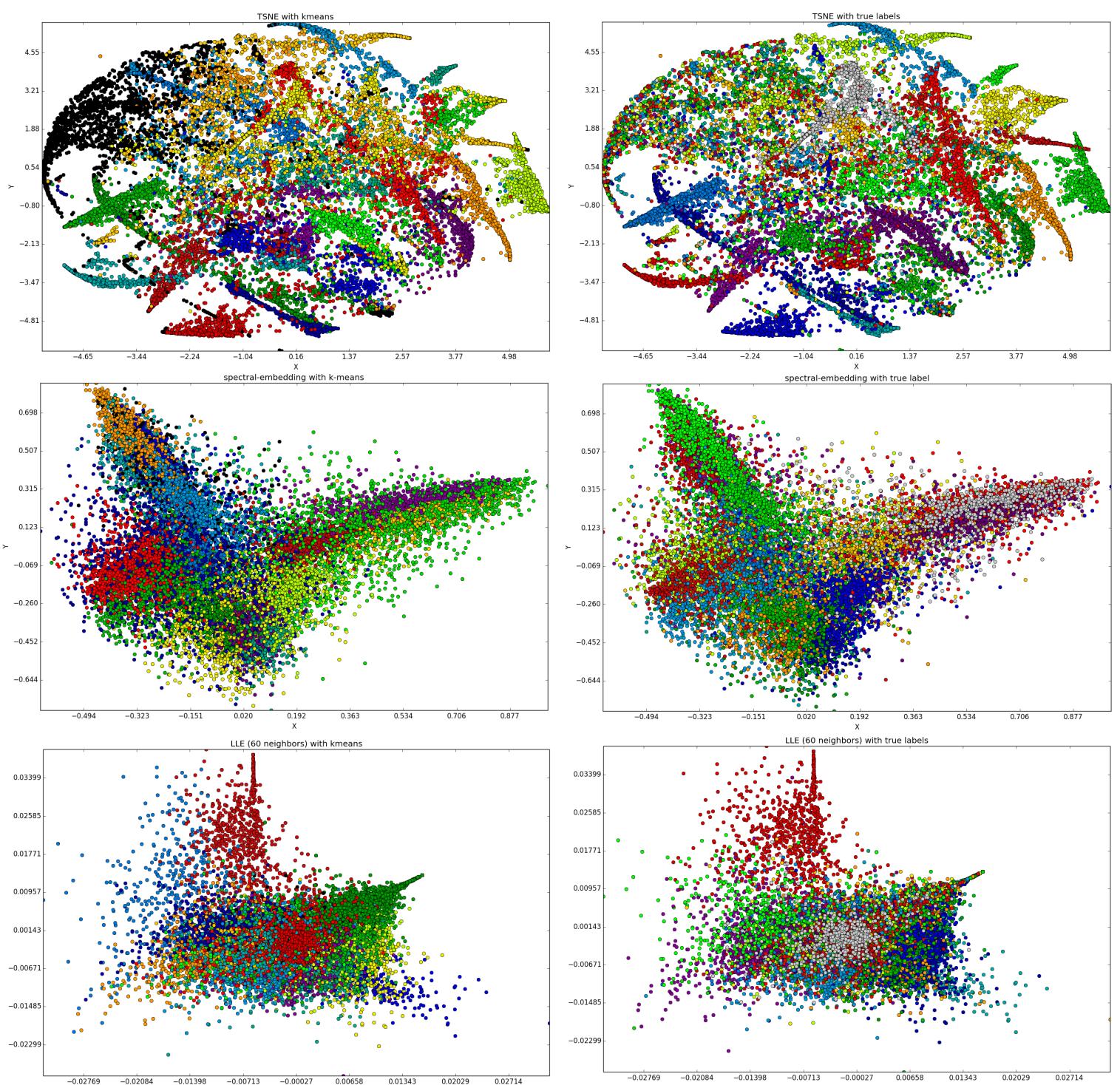
### 2. Visualize the data by projecting onto 2-D space

想要 visualizing data，我們就必須用一些數值來描述每一筆 data。在 report 第三題中我會說明我如何使用 word-embedding 的概念，以一個 200 維的 vector 來表示某一個 title。在能夠以數值來表示 data 後，就能開始做視覺化。

在視覺化的階段，可分成兩個步驟：聚類、降維。首先對用來表示 title 的 200 綴 vector 做 clustering，接著把這些 200 綴 vector 降維成二維的 vector，如此一來便能將所有 title 的 vector 表示在一個座標平面上，並根據先前 cluster 的結果來上色。

我嘗試過 k-means 與 agglomerative clustering 兩種聚類方法，以及 PCA, Isomap, tSNE (t-distributed Stochastic Neighbor Embedding), spectral embedding (Laplacian Eigenmaps) 以及 LLE (Locally linear embedding) 這五種降維方法。以下是結果：





可以看到這五種降維方式與 k-means 配合時，都能大概取得與 true label 相似的 classes 分佈模式，也就是說在 true label 中屬於同類的 data，在預測時也大概會被標記成同一類別，其中又以 t-SNE 取得的效果最佳。而在 agglomerative clustering 中，如果選擇 ward linkage 的模式，得出來的類別分布也會跟 k-means 相似。

### 3. Compare different feature extraction methods

我試過幾種方法，都有比較 remove stopwords 前後的差別。以下會依照 accuracy 由低至高做介紹：

#### (1) Naive Bayes

這個方法通常會用在文章分類，每一個類別都會建立一個 language model，這個 model 是機率模型。在這個作業中，我把每一個 title 都建立一個 language model，當要比較兩個 title 是否同類時，把其中一個 title 當作 query，用這個 query 跟另一個 title 的 language model 做相似度計算。這個相似度計算是根據 'Text Classification from Labeled and Unlabeled Documents using EM' 這篇 paper 裡的公式。公式的實作是 counter.py 裡的

countProbability() 這個 function。他會計算這個 query 屬於這個 model 的機率。如果機率超過 threshold，就認定這兩個 title 是同一類。在不 remove stopwords 時，F-measure 最高只能到 0.115，remove stopwords 後，F-measure 可以到 0.16。

### (2) Bags-of-Words with TF-IDF

這個方法是用一個長度是所有 unique term 總數的 vector 來表示一個 document。然後 vector 的 dimension 值是該 term 的  $tf \times idf$  值。得到每個 document 的 vector 後，再算兩個 vectors 間的 cosine similarity。算 cosine similarity 的 function 是 counter.py 中的 docCosineSimilarity() 這個函式，算法就是直觀的向量內積相乘除以向量長度。

此外，我也發現 sklearn 這個套件下面有 sklearn.feature\_extraction.text.TfidfVectorizer 這個物件，他可以直接得到所有 title 之間的 cosine similarity matrix，而且此套件算出來的值可以獲得比較好的 accuracy。不 remove stopwords 的 F-measure 可以到 0.364，remove 後可到 0.515。

### (3) Word Embedding

Word embedding 是 Tomas Mikolov 在 2013 年成功實作出 word2vec 這個好用的套件後開始在 NLP 領域走紅。他把每個 term 投影到高維空間中，特定詞彙之間的向量甚至具有幾何上的性質。我使用了 gensim 底下的 word2vec module 來做 word embedding。我選定的向量維度是 200。在算完每個 term 的向量後，我把每個 title 中扣除 stopwords 的 term 的向量加起來，得到一個累加的向量，用這個向量來代表一個 title。

有了代表 title 的 vectors，就能算 title 間的相似度了。我用了 cosine similarity 以及 word mover's distance (WMD) 兩種方式實作。前者使用了 scipy.spatial.distance.cosine 來計算，後者使用 gensim.models.word2vec.Word2Vec 這個 object 下的 wmdistance 來計算。實驗後發現還是 cosine similarity 的結果比較好，在 remove stopwords 前只能獲得不到 0.3 的 F-measure score，remove 後可以提升到將近 0.7，差異十分巨大。

### (4) Tag deciding

這次作業禁止使用 tag 的資訊，因此這個方法的上傳分數只能當作參考。我在瀏覽 title 的資料時發現，這 20 個 tag 在 title 中出現的頻率幾乎都很相似，幾乎都有 800 多個以上，因此這個方法的做法很直觀：如果兩個 title 中，都出現了某個 tag 的 term，就把他們歸成同一類。這個簡單的判斷方法拿到的 F-measure 超過 0.9，十分驚人。或許是 stackOverflow 的提問者在問問題時，都習慣把關鍵字打在 title，以求吸引該類別的精通者前來解題。這個結果或許能 imply 某些 domain knowledge 在解特定領域問題時會有奇效

## 4. Try different cluster numbers and compare them

當使用不同 cluster numbers 做比較的結果如下圖。結果十分直覺：原本被分類成不同類的點，在 cluster 數減少時，可能被誤判成同類的點，例如分成 5 classes 時左上角那一大片都被歸成同一類。而當 cluster 數增加時，原本同一類的 data 有機會被區分成不同類別。當 cluster numbers 跟預期的類別數目 (20 個) 相差不太多時，分數不會有明顯的下跌，因為原本的方法並非完全準確，cluster number 的小幅變動後的結果仍在合理誤差範圍。但當 cluster number 跟 20 差很多時，準確率就會大幅下降。

