HW4 report

R04922108 林俊佑

1.

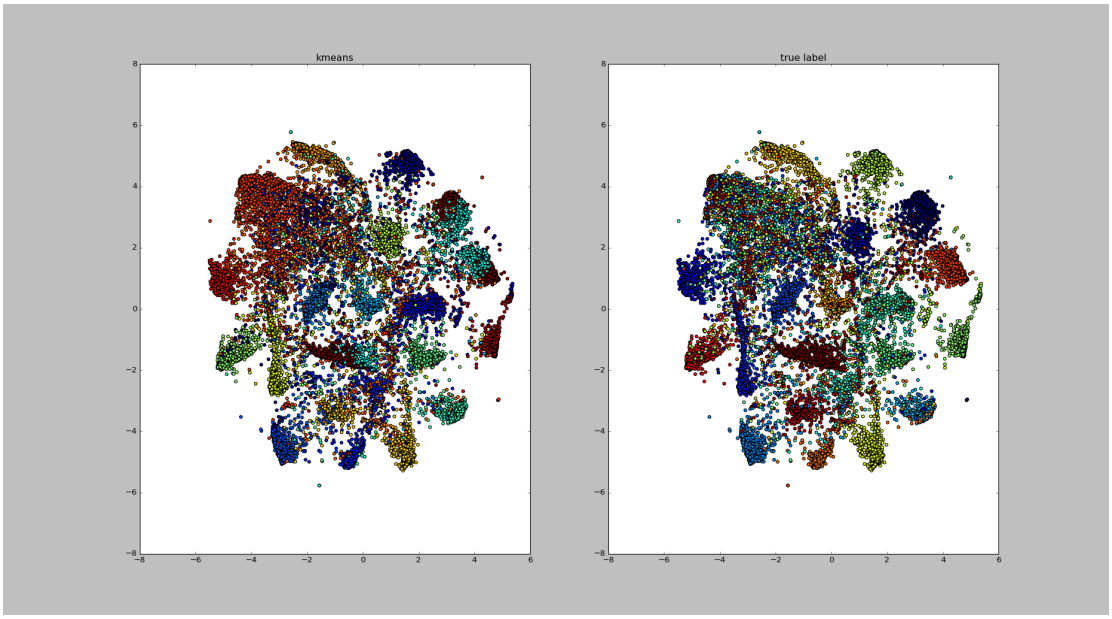
```
['', 'a', 'and', 'for', 'that', 'i', 'of', 'is', 'this', 'it', 'but', '1', 'to', 'can', 'have', 'in', 'the', 'on', 'with', 'how']
```

```
set(['', 'and', 'code', '0', 'be', 'is', 'it', 'an', 'as', 'at', 'have', 'in', 'any', 'if', 'what', 'from', 'for', 'how', 'would', 'there', 'when', '1', 'to', '2', 'way', 'get', 't', 'do', 'file', 'that',
 'use', 'but', 'not', 'using', 'with', 'name', 'a', 'on', 'like', 'i', 'of', 'm', 'this', 's', 'so', 'can', 'the', 'my', 'or', 're'])
```

```
set(['', 'all', 'code', 'just', 'thanks', 'file', 'web', '0', 'only', 'has', 'then', 'string', 'get', 'know', 'not', 'using', 'like', 'name', 'list', 'server', 't', 'x', 'where', 'page', 'set', 'some', 'a
re', 'out', 'what', 'for', '3', 'does', 'new', 'be', 'hibernate', 'here', 'org', 'by', 'on', 'c', 'of', 's', 'com', 'or', 'into', 'one', 've', 'use', 'from', 'would', 'to', 'there', '2', 'way', 'type', 'f
unction', 'that', 'but', 'with', 'me', 'this', 'work', 'value', 'will', 'can', 'error', 'problem', 'my', 'and', 'want', 'do', 'is', 'an', 'it', 'an', 'as', 'at', 'have', 'in', 'need', 'id', 'if', 'no', 'w
hen', 'any', '1', 'how', 'which', 'you', 'http', 'user', 'php', 'data', 'class', 'a', 'java', 'i', 'm', 'so', 'the'])
```

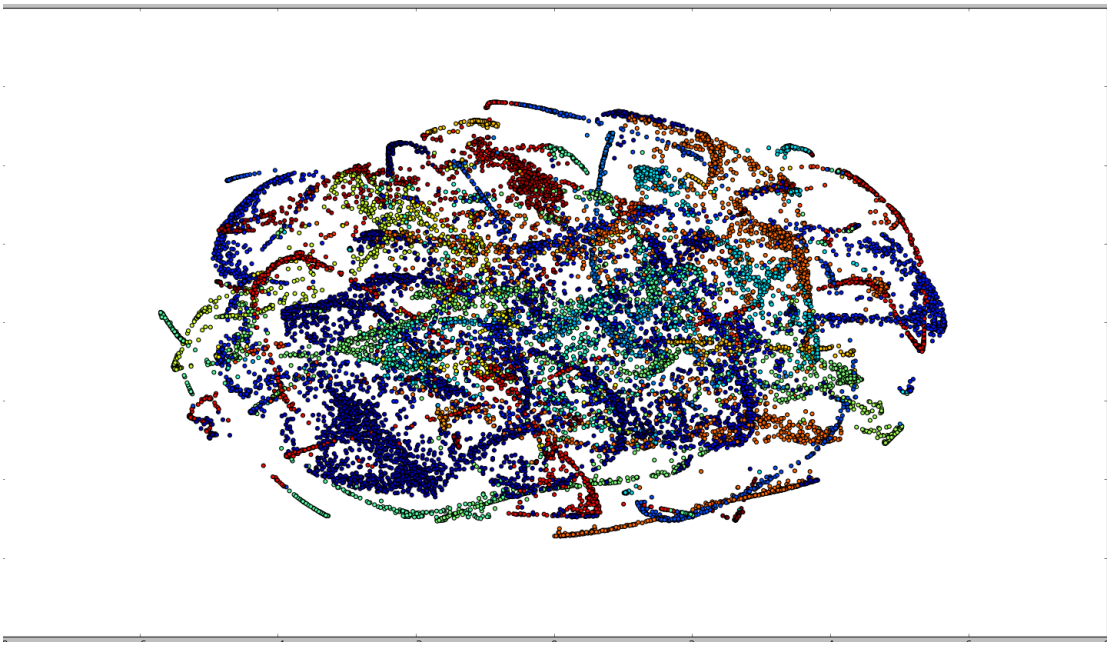
這次作業我的 stopword 都是使用 doc.txt 來獲得，如圖所示，第一張圖為使用 document frequency(df) 前 20 名的結果，第二張為前 50 名的結果，第三張為前 100 名的結果，可以看到使用前 50 名的結果就會出現一些比較重要的關鍵字，例如'code', 'file' 等等，最後我選擇使用前 20 名的 set 加上 nltk 的 stopword 再加上 df 小於 1 次以及 word 長度小於 2 的字作為我最後的 ignore word set.

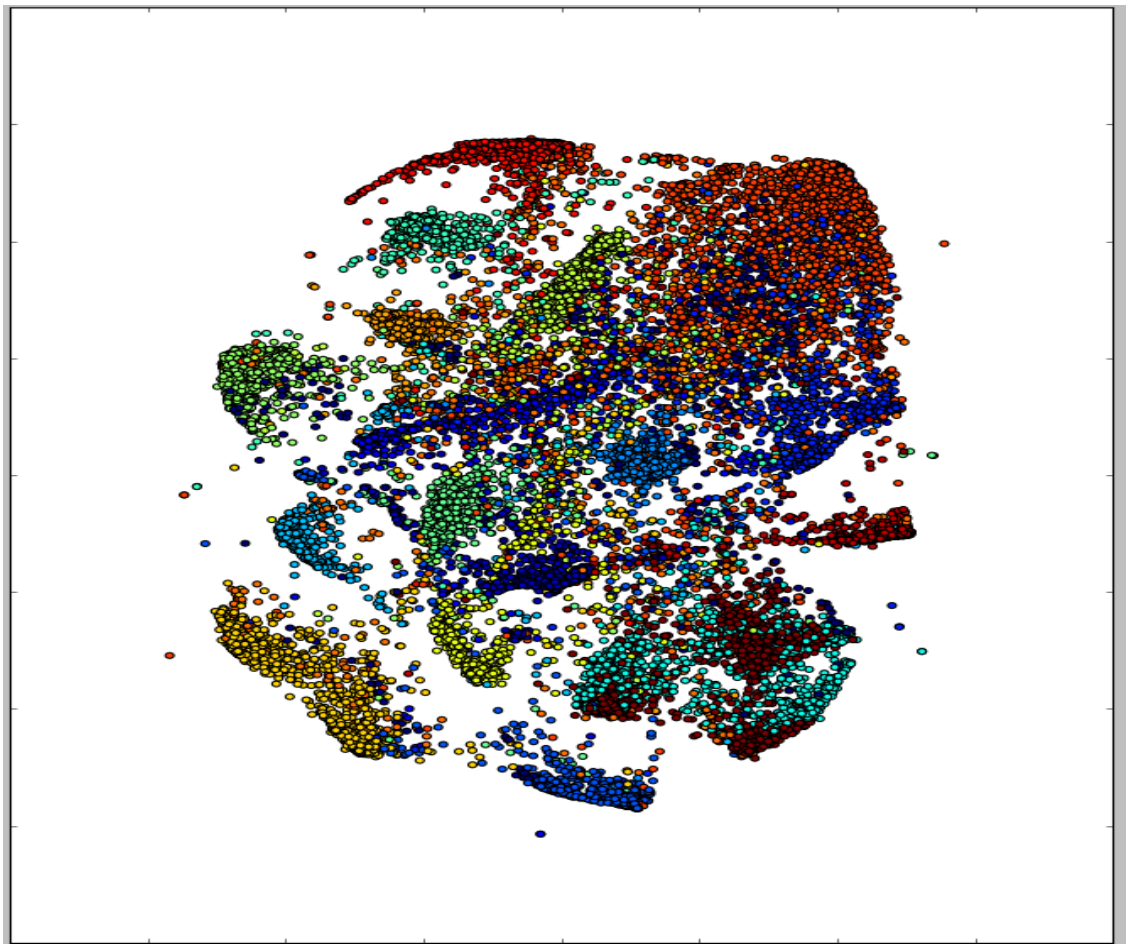
2. 下圖為 visualize 的結果，使用 tsne 降到兩維



左邊為我使用 100 維 latent 的 word2vec model 並使用 kmeans 分群畫出來的結果，右邊則為同樣的 features，並使用 true label 當顏色畫出來的圖，可以看到我的 model 最多群的那一類其實有相當多的分類錯誤，再 true label 上面重疊許多，在其他類別上的分類就較為正確.

3. 這次作業我時做了兩種方法, autoencoder 和 word2vec，以下是我使用兩種 model 出來的 feature vector 所畫出來的圖，第一張為 autoencoder 最後出來的 code 為 5 維，第二張為 word2vec，最後出來的 latent 為 100 維，兩個都使用 tsne 降到兩維





從圖中發現 word2vec 的效果比 autoencoder 好很多，我發現在使用 tf-idf 來 train deep autoencoder 的時候非常難 train，因為使用 tf-idf 會造成 input vector 非常 sparse，使得 network loss 非常低，造成難以 recover 的情況，所以結果會非常差也是很正常的

4.

```
[ 685  839  826 1029  911  773  820 1090  817  838  856  817  860 1495  754
  778 2557  845  863 1547]
```

```
[ 836 2730  788  847  766  794  881  988 1485  987  772  866  656  797  807
  819  982  799  844  846  710]
```

```
[ 740  861  336  835  632  745  795 2382  483  787  850  854  794  808  831
  844  624  797  848  713 1023  732  577  530  579]
```

上面三張圖為使用不同 cluster 數量後，每一群的總數量，第一個為 20 類，第二個為 21 類，第三個為 25 類，可以發現最多群的那個根據總類別數而下降，再根據第二題的結果，最多群那類有很多錯誤的，所以在 kaggle 上也是最多群的分數最高，可能原因為越多群造成每一群的精準度越高，所以會分得更好.