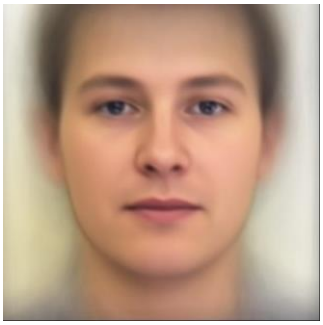
學號：B03901161　系級：　電機四　姓名：楊耀程

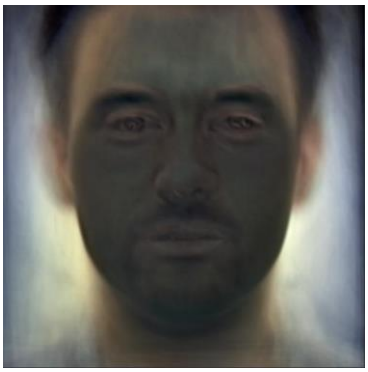
A. PCA of colored faces (collaborators: b03901165 謝世暐)

A.1. (.5%) 請畫出所有臉的平均。(collaborators: b03901165 謝世暐)

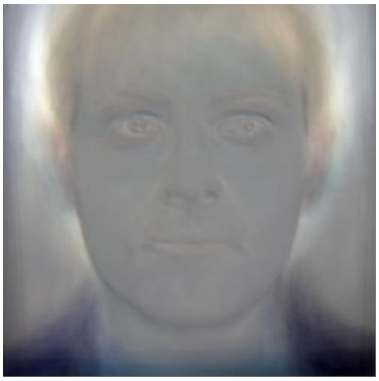


A.2. (.5%) 請畫出前四個 Eigenfaces，也就是對應到前四大 Eigenvalues 的 Eigenvectors。(collaborators: b03901165 謝世暐)

第一個

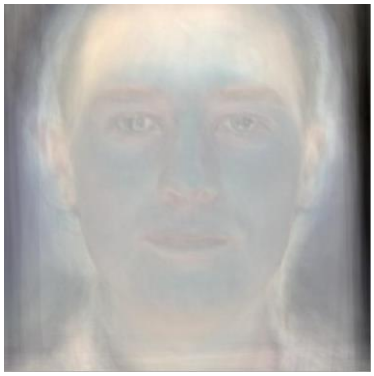


第二個



第三個

第四個



A.3. (.5%) 請從數據集中挑出任意四個圖片，並用前四大 Eigenfaces 進行 reconstruction，並畫出結果。(collaborators: b03901165 謝世暐)

第一組

Reconstruct 結果　　原圖(0.jpg)

第二組

Reconstruct 結果　　原圖(23.jpg)

第三組

Reconstruct 結果　　原圖(45.jpg)

第四組

Reconstruct 結果　　原圖(67.jpg)

A.4.　(.5%) 請寫出前四大 Eigenfaces 各自所佔的比重，請用百分比表示並四捨五入到小數點後一位。　(collaborators: b03901165 謝世暐)

編號 1-4 分別對應第一大到第四大的 Eigenfaces，比重以百分比表示，四捨

五入到小數點第一位

編號	ratio (%)
1	4.1
2	2.9
3	2.4
4	2.2

原始數據:

```
ratio:  0.0414462483826
ratio:  0.0294873222511
ratio:  0.0238771129321
ratio:  0.022078415569
```

B. Visualization of Chinese word embedding(collaborators: b03901165 謝世暐)

B.1. (.5%) 請說明你用哪一個 word2vec 套件，並針對你有調整的參數說明那個參數的意義。(collaborators: b03901165 謝世暐)

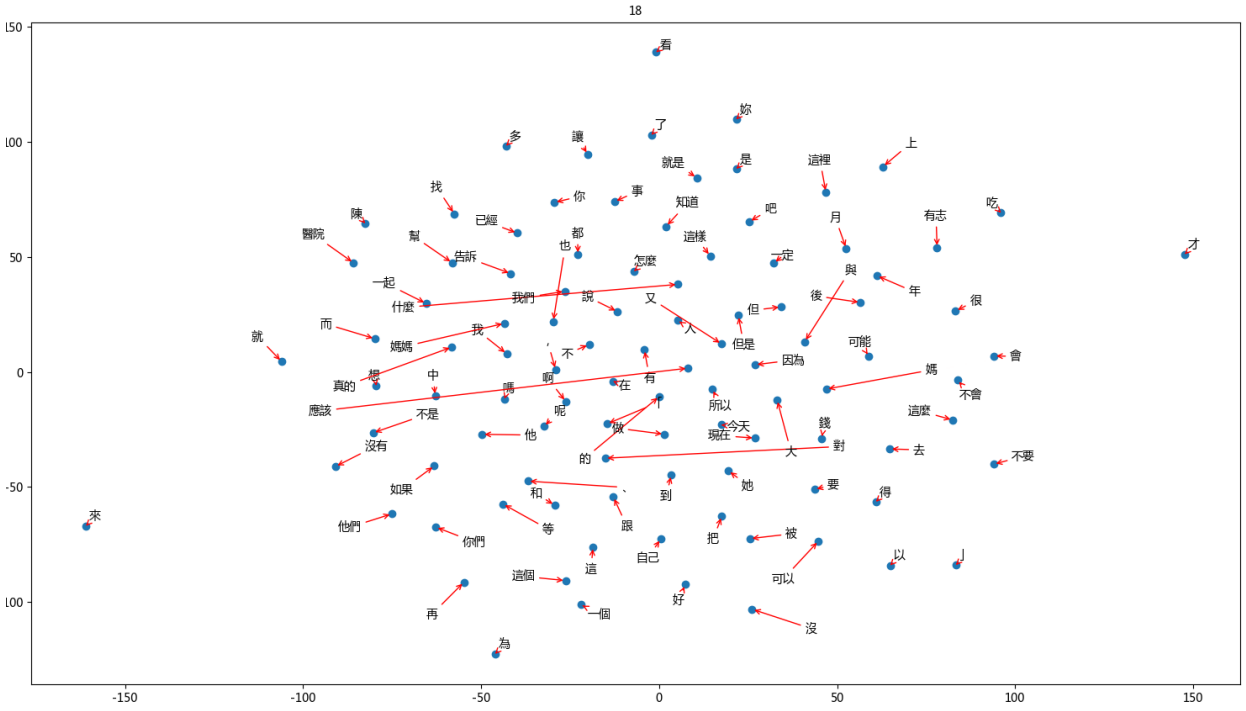
word2vec 套件: Genism 的 word2vec

調整的參數:

- 用 jieba 斷詞時使用 'dict.txt.big'
- 句子長度小於六的都拿掉
- word2vec 的 min_count =5，這邊的意思是次數多於 min_count 才會進去 word2vec training。
- 做畫圖時，只取出現次數大於 3000 的字詞
- Embedding dimension = 64 (embedding 的維度，一個字詞 embed 成 64 維的向量)
- TSNE 降維，降至 2 維(for visualization)

B.2. (.5%) 請在 Report 上放上你 visualization 的結果。(collaborators: b03901165 謝世暐)

Visualization 的結果如圖，觀察在 B.3.回答



B.3. (.5%) 請討論你從 visualization 的結果觀察到什麼。(collaborators: b03901165 謝世暐)

整體的 visualization 圖在 B.2，結果分群的效果蠻好的，很多相近類別(或是意思相近)的字 都有被分在附近，以下舉數個例子，以及放大圖:

相近意思與詞性 (意思相同或完全相反) “會 不會 不要“

相近意思與詞性 “但是 但 因為”

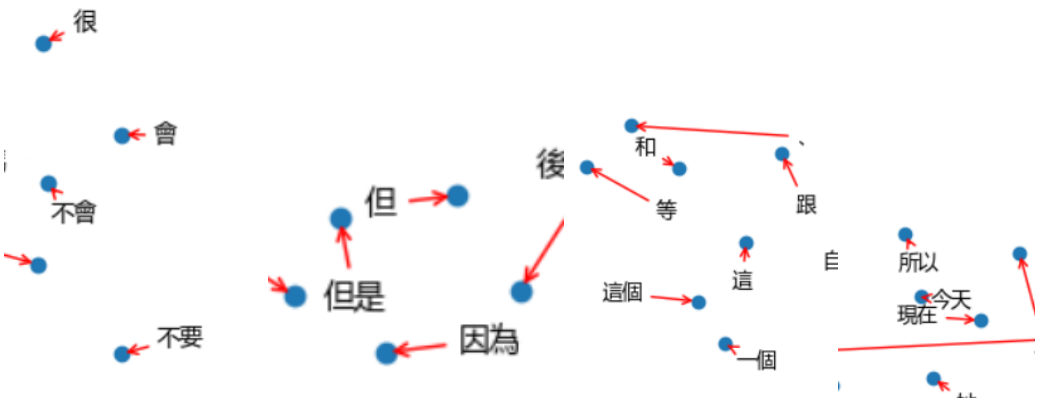
相近意思與詞性 “這個 這 一個”

相近意思與詞性 “跟 和”

相近意思與詞性 “現在 今天”

可見意思相近的字很容易被分在附近，在句子結構中，會在同一位置出現的字，也會被分的近，由上面例子可以看出，例如: A 跟 B，A 和 B，兩句話意思相近。現在要做什麼，今天要做什麼，這兩句話結構跟意思上也很相近。

放大圖



C. Image clustering(collaborators: b03901165 謝世暐)

C.1. (.5%) 請比較至少兩種不同的 feature extraction 及其結果。(不同的降維方法或不同的 cluster 方法都可以算是不同的方法) (collaborators: b03901165 謝世暐)

方法一:

先用 DNN auto-encoder 降維到 32 維，再用 kmeans 分成兩群，被 kmeans 分成同一群的即認定為同一類 dataset。

DNN 架構如下:

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	(None, 784)	0
dense_1 (Dense)	(None, 128)	100480
dense_2 (Dense)	(None, 64)	8256
encode_out (Dense)	(None, 64)	4160
dense_3 (Dense)	(None, 64)	4160
dense_4 (Dense)	(None, 64)	4160
dense_5 (Dense)	(None, 784)	50960

Total params: 172,176
Trainable params: 172,176
Non-trainable params: 0

kaggle 的分數為 0.985，kmeans 分群的數量比為: 70021 : 69979 ，兩個群的數量接近 1:1，正解是 70000:70000，因此非常接近正解。可以看到 DNN 的降維效果是很好的。

方法二:

先用 PCA 降維到 32 維，再用 kmeans 分成兩群，被 kmeans 分成同一群的即認定為同一類 dataset

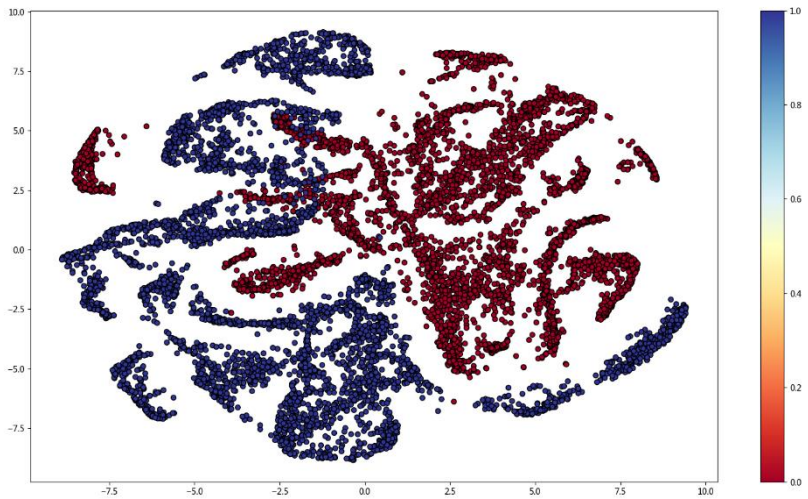
PCA 是用 sklearn.decomposition 的 PCA，kaggle 的分數為 0.03，kmeans 分群的數量比為: 103661 : 36339，兩個群的數量差很多。正解是 70000:70000，代表他跟正解還差蠻多的，分的很不平均，也造成的 kaggle 分數非常低。

結論: DNN-autoencoder 取 feature(降維)的結果好很多，在 kaggle 上可以看到顯著的差距，另外 kmeans 在兩者的分群結果，從數量比上(70021 : 69979 and 103661 : 36339)，也看到了很大的差距，DNN-autoencoder 分的好很多。

	kaggle public	kaggle private
DNN auto-encoder	0.98533	0.98538
PCA	0.03049	0.03024

C.2. (.5%) 預測 visualization.npy 中的 label，在二維平面上視覺化 label 的分佈。(collaborators: b03901165 謝世暐)

下圖為 DNN-autoencoder 降維至 32 維，再用 TSNE 降維至 2 維的結果。藍色與紅色分別代表兩個 class，可以看到基本上兩個 class 分的蠻開的，很少接近重疊的區塊。而用此 DNN-autoencoder + kmeans 在 kaggle 上可以得到 0.98 的結果，也證實他分的蠻開的，有把正確的群分開。

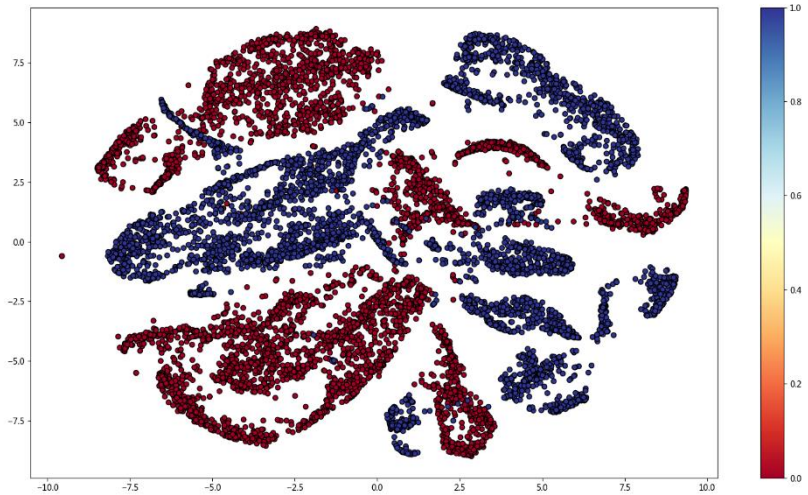


C.3. (.5%) visualization.npy 中前 5000 個 images 跟後 5000 個 images 來

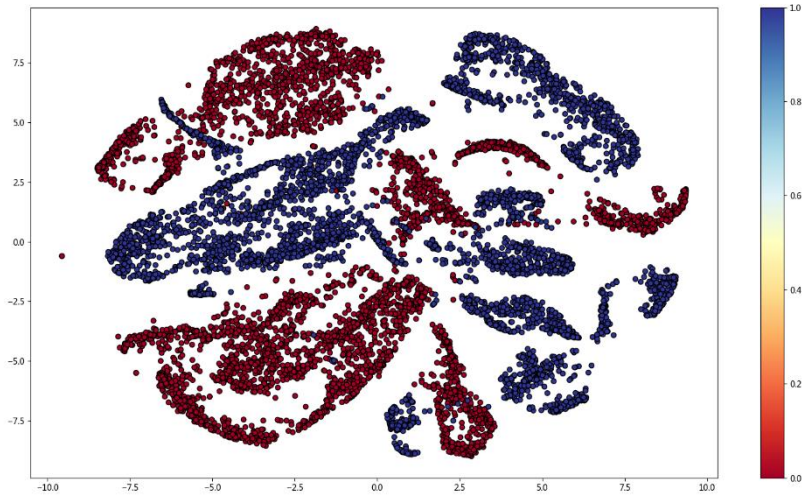
自不同 dataset。請根據這個資訊，在二維平面上視覺化 label 的分佈，接著比較和自己預測的 label 之間有何不同。(collaborators: b03901165 謝世暐)

我在同一次 TSNE 分兩群後的結果，做了兩個圖，如下:

用真實 label 作為畫圖的 label 的圖



用 predict label 作為畫圖的 label 的圖 (但標出與真實 label 不同的點，訂為 label 2，但是此處每個 predict 點都正確，故圖上看不到 label = 2 的點)



如同 C.2.以及 C.1.所述，雖然整體 140000 個 data 分群的沒到完全正確，但

是 predict 的 label 已經相當準確，跟真實的 label 相差也不多，意外的是 predict 出來的這 10000 個 label 中，前 5000 個為 class 0，後 5000 個為 class 1 (除了用看的之外，有用 code 檢查過)，代表很幸運地這 10000 筆 data **都分群對了**，其實本來分群的準確率就很高了，在 kaggle 上也快 100%，會有這樣的結果不是太意外。