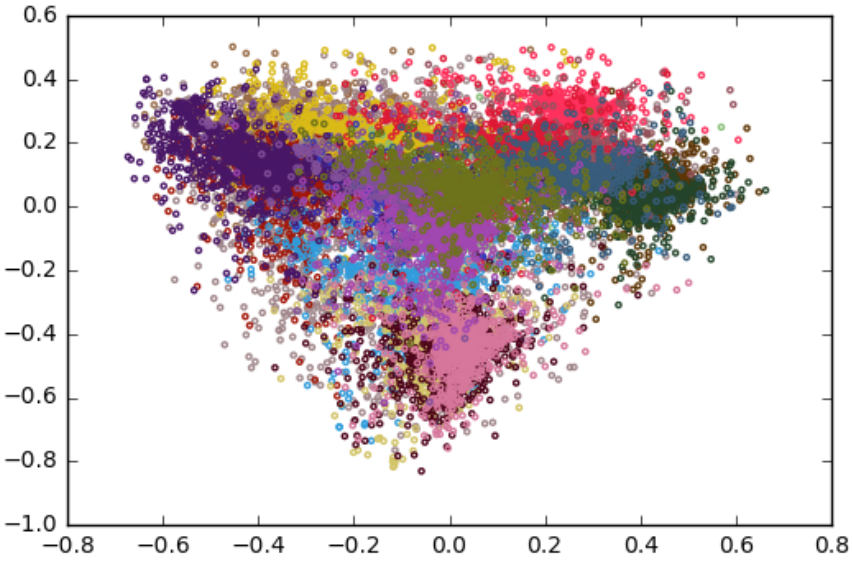
HW4

1. Analyze the most common words in the clusters

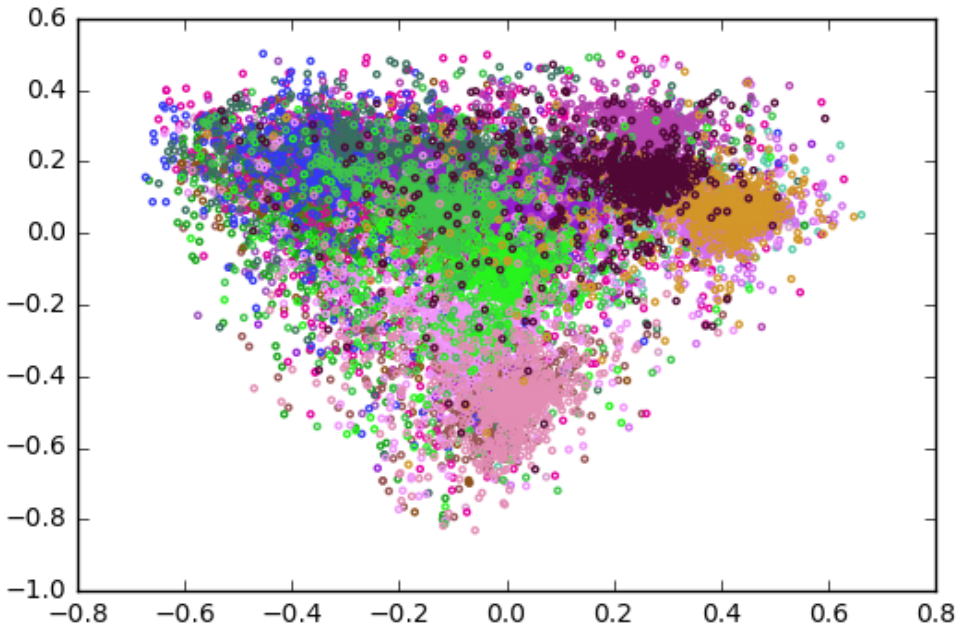
斷詞的部份我使用的是 nltk.tokenize 的 RegexpTokenizer，他可以把標點符號全部去除並斷詞，並且有使用到 nltk.corpus 的 stopwords 進行刪除基本 stopwords 的動作。為了刪除 tfidf 值較低的字，我先將斷完詞的 20000 行 title 直接使用 scikit-learn 的 TfidfVectorizer 做 tfidf 分析，產生 document-term matrix，統計每個 term 在所有文章中最大的 tfidf 值，然後把小於 0.5 的 term 去掉，做完此前處理後，準確率反而比沒做還低，可能是無意間也把一些重要的詞也去掉了，所以最後就沒採用，而是使用 TfidfVectorizer 中的參數 (max_df, min_df)，設定 max_df= 0.5 可以把出現在一半以上 documents 的 term 給去除，而設定 min_df = 2 可以把出現在少於 2 個 documents 的 term 給去除。

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

使用 TfidfVectorizer 產生 document-term matrix，再做 LSA 降維至 20，再用 kmeans(k=20)做 clustering 後，為了 project 到 2 維做視覺化，我使用 PCA 將做完 LSA 後之 20 維再縮至 2 維，下圖為分布圖，同顏色表示為同群。



而下圖為 20000 筆 titles 的原始類別同顏色代表同類別，也是做完 LSA 降至 20 維後在做 PCA 縮至 2 維。



由第二張圖所示，可見 20000 筆 titles 之原始類別的分佈經過 feature extraction 後是比較集中的，無法清楚區分各類別的分佈，可能是在 20 維的時候是分開的，經過 PCA 降至 2 維後直接壓扁分佈，或是我的 feature extraction 效果極差，沒有找出各個類別的核心 features，但是我們可以看到同一類別的 titles 大部分是密集的聚在一起只有小部分是散佈在各處，所以可以在第一張圖中找到對應的 cluster。

3. Compare different feature extraction methods.

若只是用 bag-of-words 是無法明確表示 term 於文章的重要程度，我有嘗試使用 scikitlearn 的 CountVectorizer，產生 document-term matrix，其中 matrix 的元素為 term 於 document 中的出現次數 (tf)，再做 LSA 降維至 20 得到 20000x20 的 document vector matrix，再使用 kmeans(k=20)做 clustering 後，於 private set 的分數為 0.57。

若是使用 TfidfVectorizer 產生 document-term matrix，再做 LSA 降維至 20，再用 kmeans(k=20)做 clustering 後，於 private set 的分數為 0.61，相較起來效果較好。

再來我有嘗試使用 LDA(LatentDirichletAllocation)，先使用 scikitlearn 的 CountVectorizer 產生 document-term matrix，再直接使用 scikitlearn 的 LatentDirichletAllocation，參數的設定為 : n_topics = 20，max_iter=5，learning_method='online'，learning_offset=50.，random_state=0，再做 kmeans(k=20)後，於 private set 的分數為 0.179，結果差到不行。

之後我又有嘗試 autoencoder 做為 feature extraction 的方法，使用做完 TfidfVectorizer 產生的 document-term matrix (20000x5235)做為 data，autoencoder 的結構如下圖，使用的 loss function 為 binary_crossentropy，optimizer 為 adam，np_epoch 設 20，batch_size 設 100，train，train 完後每個 documents 的 vector encode 至 20 維，在做 kmeans(k=20)後於 private set 的分數為 0.06331，跟 random 的分數差不多，可能是單純 fully connected 的結構無法 train 出什麼東西吧。

```
inputs = Input(shape=(tfidf_array.shape[1],))

x = Dense(output_dim = 1000,activation = 'relu')(inputs)
x = Dense(output_dim = 500,activation = 'relu')(x)
x = Dense(output_dim=100,activation='relu')(x)
encoded = Dense(output_dim=20,activation='relu')(x)

x = Dense(output_dim = 100 , activation = 'relu')(encoded)
x = Dense(output_dim = 500,activation='relu')(x)
x = Dense(output_dim = 1000,activation = 'relu')(x)
decoded = Dense(output_dim = tfidf_array.shape[1],activation = 'sigmoid')(x)
```

4. Try different cluster numbers and compare them.

當我使用 TfidfVectorizer 產生 document-term matrix，再做 LSA 降維至 20，再用 kmeans(k=20)做 clustering 後，於 private set 的分數為 0.61，後來將 k 增加為 25，於 private set 的分數變為 0.72，再將 k 增加至 30，於 private set 的分數變為 0.78，再將 k 增加至 50 後，於 private set 的分數變為 0.84，若再將增加至 60 後，於 private set 的分數仍是 0.84，若繼續再增加 k 會使準確率下降，至於為何 cluster number 高達 50 會使準確率提高不少，可能是判斷兩 title 是否相似的 threshold 提高，但若是 threshold 設太高了又會使得可能相似的 titles pair 被忽略。