Unsupervised Task on Individual Dataset

Guita Bianca-Oana
411

Introduction

For the second task of the Practical Machine Learning course, we were supposed to choose a unique dataset which contains at least 1000 data samples and test two different clustering models on it. Unsupervised learning is some kind of algorithm which puts emphasis on learning patterns from unlabelled data. The challenge is that It captures itself similar patterns as probability densities.

Dataset

The chosen dataset for this task, *"Emotion Classification NLP"*, consists of a total number of 7102 unique tweets expressing different emotions such as joy, fear, anger and sadness. It contains three *.csv* files as it follows:

- *emotion-labels-train.csv* : having two columns, one for 3613 unique tweets, and one for labels (joy, fear, anger, sadness)

	text	label
0	Just got back from seeing @GaryDelaney in Burs...	joy
1	Oh dear an evening of absolute hilarity I don'...	joy
2	Been waiting all week for this game ❤️❤️❤️ #ch...	joy
3	@gardiner_love : Thank you so much, Gloria! Yo...	joy
4	I feel so blessed to work with the family that...	joy
...	...	...
3608	@VivienLloyd Thank you so much! Just home - st...	sadness
3609	Just put the winter duvet on ☃️❄️🌬️☔	sadness
3610	@SilkInSide @TommyJoeRatliff that's so pretty!...	sadness
3611	@BluesfestByron second artist announcement loo...	sadness
3612	I can literally eat creamy pesto pasta topped ...	sadness

3613 rows × 2 columns

- *emotion-labels-val.csv*: validation data with 347 unique tweets and their labels

	text	label
0	@theclobra lol I thought maybe, couldn't decid...	joy
1	Nawaz Sharif is getting more funnier than @kap...	joy
2	Nawaz Sharif is getting more funnier than @kap...	joy
3	@tomderivan73 😁...I'll just people watch and e...	joy
4	I love my family so much #lucky #grateful #sma...	joy
...	...	...
342	Common app just randomly logged me out as I wa...	sadness
343	I'd rather laugh with the rarest genius, in be...	sadness
344	If you #invest in my new #film I will stop ask...	sadness
345	Just watched Django Unchained, Other people ma...	sadness
346	@KeithOlbermann depressing how despicable Trum...	sadness

347 rows × 2 columns

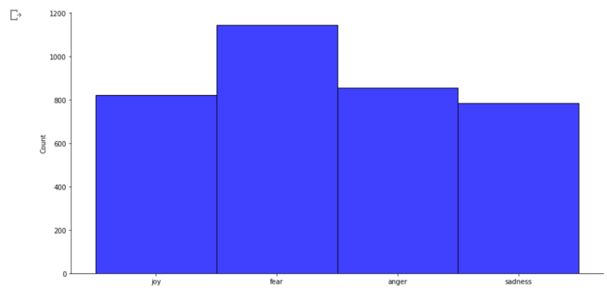
- *emotion-labels-test.csv*: test data having 3142 unique tweets and their labels

	text	label
0	You must be knowing #blithe means (adj.) Happ...	joy
1	Old saying 'A #smile shared is one gained for ...	joy
2	Bridget Jones' Baby was bloody hilarious 😅 #Br...	joy
3	@Elaminova sparkling water makes your life spa...	joy
4	I'm tired of everybody telling me to chill out...	joy
...	...	...
3137	Why does Candice constantly pout #GBBO 💄😒	sadness
3138	@redBus_in #unhappy with #redbus CC, when I ta...	sadness
3139	@AceOperative789 no pull him afew weeks ago, s...	sadness
3140	I'm buying art supplies and I'm debating how s...	sadness
3141	@sainsburys Could you ask your Chafford Hundre...	sadness

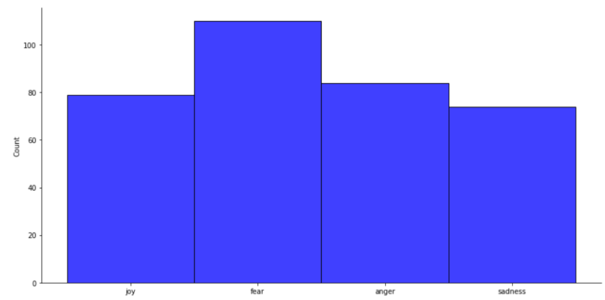
3142 rows × 2 columns

After analyzing the distribution ,using displot from seaborn library, four types of emotion in our dataset, it is noticed that there are more tweets expressing fear, followed by anger, joy and sadness for each csv.

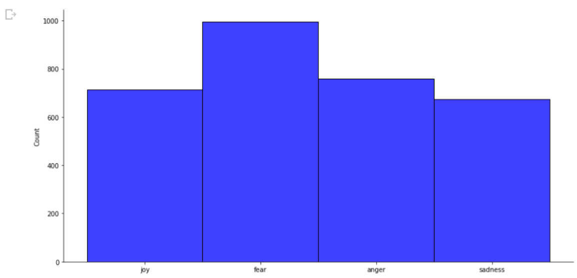
```
sns.displot(y_train,aspect=2, height=6, color='blue')
plt.show()
```




```
[10] sns.displot(y_valid,aspect=2, height=6, color='blue')
plt.show()
```




```
[13] sns.displot(y_test,aspect=2, height=6, color='blue')
plt.show()
```



Data Processing

For data processing and cleaning, the tweets were lowercased, then digits were removed as well as unwanted multiple spaces between the words, hashtags (#) and add-ons (@). It was used the emoji library to safely take away the emojis with their regular expression, as well. For the labels, I classified each emotion, giving them a number from 0 to 3, so our data would look like this:

	text	label
0	theclobra lol i thought maybe, couldn't decid...	0
1	nawaz sharif is getting more funnier than kap...	0
2	nawaz sharif is getting more funnier than kap...	0
3	tomderivan73 ...i'll just people watch and en...	0
4	i love my family so much lucky grateful sma...	0
...	...	...
342	common app just randomly logged me out as i wa...	3
343	i'd rather laugh with the rarest genius, in be...	3
344	if you invest in my new film i will stop ask...	3
345	just watched django unchained, other people ma...	3
346	keitholbermann depressing how despicable trum...	3

347 rows × 2 columns

Example of data and labels after preprocessing validation_data

Model Preparation and Features

We were supposed to use two different features for our unsupervised learning models so, CountVectorizer and TfidfVectorizer from sklearn were the main options.

CountVectorizer has the purpose of converting groups of text documents to a matrix of index counts. That being said, I applied the feature extraction on my data as it follows

```
from sklearn.feature_extraction.text import CountVectorizer
vect = CountVectorizer(lowercase=True)
# matrix=vect.fit_transform(train_data['text'])
# array=matrix.toarray()

X_tr = vect.fit_transform(train_data['text'])
X_tr=X_tr.toarray()
X_ts = vect.transform(test_data['text'])
X_ts=X_ts.toarray()
X_val = vect.transform(validation_data['text'])
X_val=X_val.toarray()
```

and obtained features with a shape of (3613, 10054), for training, (347, 10054) for validation and (3142, 10054) for test.

TfidfVectorizer transforms groups of documents that are raw to TF-IDF feature matrix. It was applied to my data in this way and obtained the same shapes for train, validation and test as I mentioned before.

```
tfidf_vectorizer = TfidfVectorizer(lowercase = True)
tfidf_representation = tfidf_vectorizer.fit(train_data['text'])

X_train = tfidf_vectorizer.transform(train_data['text'])
X_test = tfidf_vectorizer.transform(test_data['text'])
X_valid = tfidf_vectorizer.transform(validation_data['text'])

y_train = train_data['label']
y_test = test_data['label']
y_valid = validation_data['label']
```

Unsupervised Methods

Spectral Clustering is one of the algorithms chosen for my dataset. It assigns clustering to the normalized Laplacian's projection. For parameters I set a number of 4 clusters as I have 4 labels, the labels were assigned with the "discretize" strategy because it is less sensitive to random initialization. I obtained an accuracy score of 0.2616 on test and 0.2939 on validation, which were the best results after trying various parameters.

For a better understanding of what I tried regarding this model, managed to create a table having 4 columns indicating, in this order, the model used, the type of features, parameters and accuracy obtained.

Model	**Features**	**Parameters**	**Accuracy**
Spectral Clustering	TFIDF	n_clusters=4, assign_labels='kmeans', random_state=0, affinity='rbf'	validation: 0.224 test: 0.255
Spectral Clustering	TFIDF	n_clusters=4, assign_labels='discretize', eigen_solver='arpack' random_state=0, affinity='rbf'	validation: 0.261 test: 0.293
Spectral Clustering	TFIDF	n_clusters=4, assign_labels='discretize', eigen_solver='lobpcg' random_state=0, affinity='rbf'	validation: 0.233 test: 0.242

Spectral Clustering	TFIDF	n_clusters=4, assign_labels='discretize', eigen_solver='lobpcg', random_state=0, affinity='nearest_neighbors')	validation: 0.276 test: 0.225
Spectral Clustering	CountVect	n_clusters=4, assign_labels='discretize', random_state=0	validation: 0.253 test: 0.226
Spectral Clustering	CountVect	n_clusters=4, assign_labels='discretize', eigen_solver='arpack' random_state=0, affinity='rbf'	validation: 0.248 test: 0.271
Spectral Clustering	CountVect	n_clusters=4, assign_labels='discretize', eigen_solver='lobpcg' random_state=0, affinity='rbf'	validation: 0.221 test: 0.206
Spectral Clustering	CountVect	n_clusters=4, assign_labels='discretize', eigen_solver='lobpcg', random_state=0, affinity='nearest_neighbors')	validation: 0.259 test: 0.237

Gaussian Mixture was the second method used and it represents the probability distribution of Gaussian mixture model probability.

Model	**Features**	**Parameters**	**Accuracy**
GMM	TFIDF	n_components = 4, random_state = 0, covariance_type = 'spherical', tol=1e-3	test: 0.215
GMM	TFIDF	n_components = 4, random_state = 0, covariance_type = 'diag', tol=1e-4, init_params='kmeans'	test: 0.238
GMM	TFIDF	n_components = 4, random_state = 0, covariance_type = 'diag', tol=1e-4, init_params='random'	test:0.265
GMM	CountVect	n_components = 4, random_state = 0, covariance_type = 'spherical'	test:0.232

GMM	CountVect	n_components = 4, random_state = 0, covariance_type = 'diag', tol=1e-4, init_params='kmeans'	test: 0.252
GMM	CountVect	n_components = 4, random_state = 0, covariance_type = 'diag', tol=1e-4, init_params='random'	test: 0.257

As we can see in the table above, the best accuracy score was 0.257 using GMM on CountVectorizer type of features, having as parameters the covariance_type set to 'diag', so each and every component has their very own diagonal covariance matrix, the convergence threshold (tol) of 1e-4, and the weights initialization method (init_params) random.

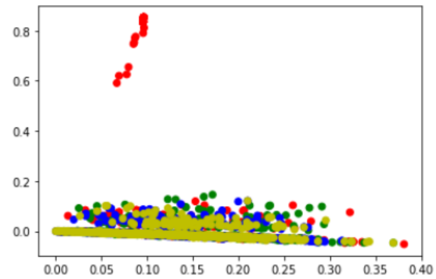
For testing purposes and out of curiosity, I also tried to apply Kmeans method on my dataset. The results were slightly better than the ones provided by GMM, but a little worse than Spectral Clustering. That being said, on TFIDF features, the best accuracy score was 0.280 having the following parameters:
n_clusters=4 (as I have 4 labels), random_state=0,init='k-means++'(which gets the initial centers of the clusters so that it can speed the convergence),tol=1e-3,algorithm='auto' (not only because it was the default form for the parameter, but also because it was the most suitable actually choosing the elkan algorithm which is good for defined clusters like we have, but it also can change if there turns out to be a better heuristic) and on CountVect it reached 0.259 with the same parameters.

Interpretation of results

Visualization of clusters on *Spectral Clustering* method:

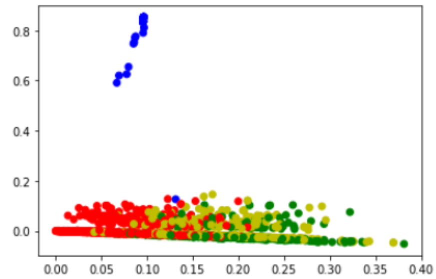
```
color = np.array(['r', 'g', 'b','y'])
plt.scatter(X_train_pca[:,0],X_train_pca[:,1], c=color[y_train.values])
```

<matplotlib.collections.PathCollection at 0x7fa732547f50>



```
[152] plt.scatter(X_train_pca[:,0],X_train_pca[:,1], c=colormap[clustering])
```

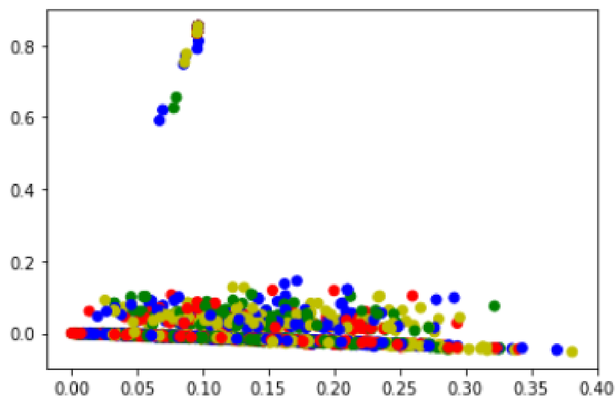
<matplotlib.collections.PathCollection at 0x7fa732547e10>



The first graph shows how labels from X_train were classified, while in the second one are presented the predictions of our model.We can observe that the clusters from Spectral are denser than the ones provided by Gaussian Mixture method, which are more mixed.

Visualization of clusters on *GMM* method:

<matplotlib.collections.PathCollection at 0x7fa732a50ed0>



Supervised Method and Random Chance

To compare the unsupervised learning methods with a supervised one, it was chosen RidgeClassifier which transforms the task into a regression problem in the multiclass case by turning target values into -1 and 1. Applied GridSearch on it in order to test multiple parameter values and get the best accuracy score.

```
# Supervised Comparison
from sklearn.linear_model import RidgeClassifier
from sklearn.model_selection import GridSearchCV
parameters = {'alpha':(1.0,0.8), 'fit_intercept': (True,False),
              'max_iter':(None, 2), 'class_weight':('dict','balanced',None),
              'tol':(1e-4, 1e-3),'solver':['auto']}
clf = RidgeClassifier()
clf = GridSearchCV(clf, parameters)
clf.fit(X_train, y_train)
sorted(clf.cv_results_.keys())
print('Train Score', clf.score(X_train, y_train))
print('Test Score', clf.score(X_test, y_test))
print('Validation Score', clf.score(X_valid, y_valid))
```

[nan nan nan nan nan nan
nan nan 0.83059505 0.83087168 0.83059505 0.83087168
0.83585016 0.83557354 0.83585016 0.83557354 0.83474673 0.8344701
0.83474673 0.8344701 0.83502335 0.83446972 0.83502335 0.83446972
nan nan nan nan nan nan
nan nan 0.83668617 0.83696318 0.83668617 0.83696318
0.83944974 0.83972636 0.83944974 0.83972636 **0.84083631** 0.8405593
0.84083631 0.8405593 0.83806661 0.83834362 0.83806661 0.83834362]

Therefore, the best accuracy scores on TFIDF features were provided by these parameters:
alpha=0.8, fit_intercept= True,max_iter= None, class_weight = 'balanced', tol=1e-4, solver = 'auto'
Train Score 0.9850539717686133
Test Score 0.8157224697644813
Validation Score 0.8443804034582133
The best accuracy scores on CountVect features:
Train Score 0.9867146415721008
Test Score 0.7905792488860598
Validation Score 0.8242074927953891

After computing the Random Chance, it is obtained a score of 0.254, which is not much worse than what we got using the unsupervised methods.

```
[35] random_chance = np.random.randint(0,4,len(test_labels))

[41] accuracy_score(y_test, random_chance)

     0.25493316359007

[36] random_chance[:100]

     array([3, 3, 0, 2, 0, 2, 0, 1, 3, 2, 1, 1, 1, 2, 1, 1, 1, 1, 2, 1, 0, 0,
            2, 3, 1, 1, 3, 3, 2, 3, 0, 0, 1, 2, 2, 2, 3, 3, 3, 2, 1, 2, 2, 1,
            3, 1, 1, 0, 3, 2, 3, 1, 1, 0, 3, 0, 3, 2, 1, 1, 3, 1, 3, 1, 0, 2,
            3, 1, 0, 3, 2, 3, 1, 0, 0, 3, 3, 3, 2, 3, 0, 1, 0, 1, 2, 0, 2, 2,
            3, 0, 2, 3, 0, 1, 0, 2, 0, 1, 3, 1])
```

Confusion Matrix and Label Permutations

For unsupervised methods, I took into consideration the problem of classifying each cluster to a label, for example, in cluster 0 we are supposed to have labels with the value of 0, cluster 1, labels with values of 1 and so on. For the predictions to be more accurate, I created a confusion matrix between the real labels and the ones generated by my models, in order to see which permutations of label-cluster values would be more suitable.

So, for the best result scored with Spectral Clustering, the confusion matrix looks like this:

```
confusion_matrix(y_test, y)

array([[139, 156, 249, 170],
       [158, 225, 378, 234],
       [114, 174, 284, 188],
       [ 87, 143, 269, 174]])
```

We can notice that the highest resemblance is classifying the clusters having the values of 1 with labels also having values of 1. A good try for permutation would be swapping the last two label values with their supposed clusters as it follows:

```
arr[y==0] =0
arr[y==1] =1
arr[y==2] =3
arr[y==3] =2
```

Calculating the accuracy score, we obtain a slightly better one (0.261 vs 0.269 on validation and 0.293 vs 0.297 on test), which proves that the permutation was a good one.

Conclusions

To conclude with, it is clear that supervised models give considerably better results on a labeled dataset than the unsupervised methods, but the challenge behind these cluster-based models should be taken into consideration since they may have uses in statistics and analytics for finding certain patterns of the datas.