clustering

May 12, 2017

1 Part 2 -- Clustering

The libraries that we are going to use:

```
In [14]: import pandas as pd
         import scipy as sc
         import numpy as np
         import nltk
         from sklearn.feature_extraction.text import TfidfVectorizer
         from sklearn.metrics import adjusted_rand_score
         from wordcloud import STOPWORDS
         from sklearn.feature_extraction.text import CountVectorizer
         from sklearn.feature_extraction.text import ENGLISH_STOP_WORDS
         from scipy.spatial.distance import cosine
         from nltk.cluster.kmeans import KMeansClusterer
         from numpy import array
         from __future__ import division
         from nltk import cluster
         from nltk.cluster import euclidean_distance
         from nltk.cluster import cosine_distance
         from sklearn.decomposition import TruncatedSVD
         from collections import Counter
```

1.1 Setting up

We read our training data:

```
In [15]: mydata = pd.read_csv('train_set.csv', sep='\t')
         mydata.head()
```

```
Out[15]:    RowNum     Id                                              Title  \
         0    9560   9561  Sam Adams founder: Beer is more than just 'col...
         1   10801  10802  Slump in oil prices could mean fall in investm...
         2    6726   6727  British Gas owner Centrica warns of higher gas...
         3   12365  12366  Ole Gunnar Solskjaer appointed manager of Card...
         4   11782  11783  Sunderland target loan signings of Kurt Zouma ...
```

```
                                              Content  Category
0  The craft beer boom, which and been attributed...  Business
1  The International Energy Agency has warned tha...  Business
2  Senior executives at British have been accused...  Business
3   is confident he will have complete control of...  Football
4  Kurt Zouma and Jack Rodwell are on Sunderland...  Football
```

We declare the stopwords that we are going to use:

```
In [16]: # declaring our stopwords
         stopwords = set(STOPWORDS) | set(ENGLISH_STOP_WORDS)
         # some additional stopwords based on our own observations
         stopwords.add('said')
         stopwords.add('say')
         stopwords.add('says')
         stopwords.add('set')
```

We declare our vectorizer, which is a *TfidfVectorizer (term-frequency times inverse document-frequency vectorizer)*, and we pass our data through him:

```
In [17]: vectorizer = TfidfVectorizer(stop_words=stopwords)
         X = vectorizer.fit_transform(mydata['Content'])
         svd = TruncatedSVD(n_components=100)
         X_lsi = svd.fit_transform(X)
         vectors = X_lsi
```

1.2 K-Means Clustering

We implement K-Means using Cosine Similarity as a distance function:

```
In [18]: # A IMPLEMENTATION OF CLUSTERING WITH KMEANS USING COSINE SIMILARITY, UTILIZING COMPONE
         clusterer = cluster.KMeansClusterer(5, cosine_distance, repeats=1)

In [19]: clusters_array = clusterer.cluster(vectors, True, trace=True) # we take a list of our c
```

```
k-means trial 0
iteration


/home/marinos/.local/lib/python3.5/site-packages/nltk/cluster/util.py:127: RuntimeWarning: inval
  return 1 - (numpy.dot(u, v) / (sqrt(numpy.dot(u, u)) * sqrt(numpy.dot(v, v))))


iteration
iteration
iteration
iteration
iteration
iteration
```

```
iteration
iteration
iteration
iteration
```

1.3 Printing Clustering Results

We will now work towards printing our Clustering's results in a nice way...

```
In [20]: our_dict = {'0':[], '1':[], '2':[], '3':[], '4':[]}

         counter = 0

         # we create a dictionary that for each cluster has the numbers of the texts that belong
         for x in clusters_array :
             our_dict[str(x)] += [counter]
             counter +=1
```

```
In [21]: cnt = Counter()
         # our categories
         categories =['Politics', 'Football', 'Business', 'Technology', 'Film']

         # we create a counter-dictionary based on the above categories
         for x in categories :
                 cnt[x] =[]

         cnt[''] =[]
         print(cnt)
```

```
Counter({'Politics': [], '': [], 'Technology': [], 'Film': [], 'Football': [], 'Business': []})
```

```
In [22]: data_categories = mydata['Category']
```

```
In [23]: # we will have 5 clusters
         clusters=['Cluster_0', 'Cluster_1', 'Cluster_2', 'Cluster_3', 'Cluster_4']
         # we have 5 categories
         categoryl = ['Politics', 'Business', 'Football', 'Film', 'Technology']
```

```
In [24]: outdict = {'':clusters, 'Technology':[], 'Politics':[], 'Business':[], 'Football':[], '

         for cluster_num in range(len(our_dict)): # for each cluster
             count = {}
             cluster_length = len(our_dict[str(cluster_num)])

             for x in our_dict[str(cluster_num)]:
                 category = data_categories[x]  # this way we take the category
                 if category in count:
```

```
            count[str(category)] += 1/cluster_length
        else:
            count[str(category)] = 1/cluster_length

    for category in categoryl: # we create our dictionary
        if str(category) in count:
            outdict[str(category)] += [count[str(category)]]
        else:
            outdict[str(category)] += [0]

print(outdict)
```

```
{'': ['Cluster_0', 'Cluster_1', 'Cluster_2', 'Cluster_3', 'Cluster_4'], 'Politics': [0.012581344
```

We create a dataframe with the above data, which we then print to a .csv file:

```
In [25]: # creating the dataframe
         out_pd = pd.DataFrame(data=outdict)
         out_pd

Out[25]:               Business      Film  Football  Politics  Technology
         0  Cluster_0  0.004338  0.960087  0.004772  0.012581    0.018221
         1  Cluster_1  0.002374  0.001018  0.993216  0.000000    0.003392
         2  Cluster_2  0.082907  0.009086  0.091993  0.016468    0.799546
         3  Cluster_3  0.901323  0.001788  0.003218  0.087236    0.006435
         4  Cluster_4  0.020774  0.001222  0.004481  0.969857    0.003666

In [26]: # creating the csv file
         out_pd = out_pd.ix[::, ['', 'Politics', 'Business', 'Football', 'Film', 'Technology']]
         out_pd.to_csv(path_or_buf='clustering_KMeans.csv', sep='\t', index=False)
```