VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

ANOMALY DETECTION IN TWEETS USING K-MEANS

WINTER SEMESTER 20-21

BY

18BIT0126- N.ROHITH REDDY

18BIT0094- D. SAI SUCHETAN

FROM

SITE SCHOOL

FOR

SOFT COMPUTING COURSE (E1+TE1 SLOT)

ABSTRACT

Anomaly/Outlier detection is one of very popular topic in ML world. It comes under 'unsupervised learning' process. Here we don't have any prior knowledge of the data patterns unlike 'supervised learning'. 'Anomaly or Outlier' is that data point which is not that much similar with other data points in our entire data set. Now a-days, we are spending the most of the time online in the social media. The example for such social media is Twitter, Facebook and Instagram. Here in our project we are gone help the users to identify the how far a tweet is true or else it was an anomaly. Steps involved are

- Data is from Kaggle Donald Tweets we use
- After getting the data we now have to select the data on which we are going to work and pre-process the data removing noise.
- After that we convert the text to vector by doc2vec function
- After that we use PCA to identify principal components
- After that we cluster the data
- After that we find the Silhouette Score and sort the scores
- Find the least scores that are nearer to zero
- Then we go for anomaly detection by basing on the Silhouette Score if Silhouette Score= 0
- Finally, we will be able to view anomaly tweets.

HARDWARE AND SOFTWARE

1. Laptop with Ram>=8 GB, i5 processor
2. Anaconda software that contains the Jupiter notebook
3. DOC2VEC is to be installed

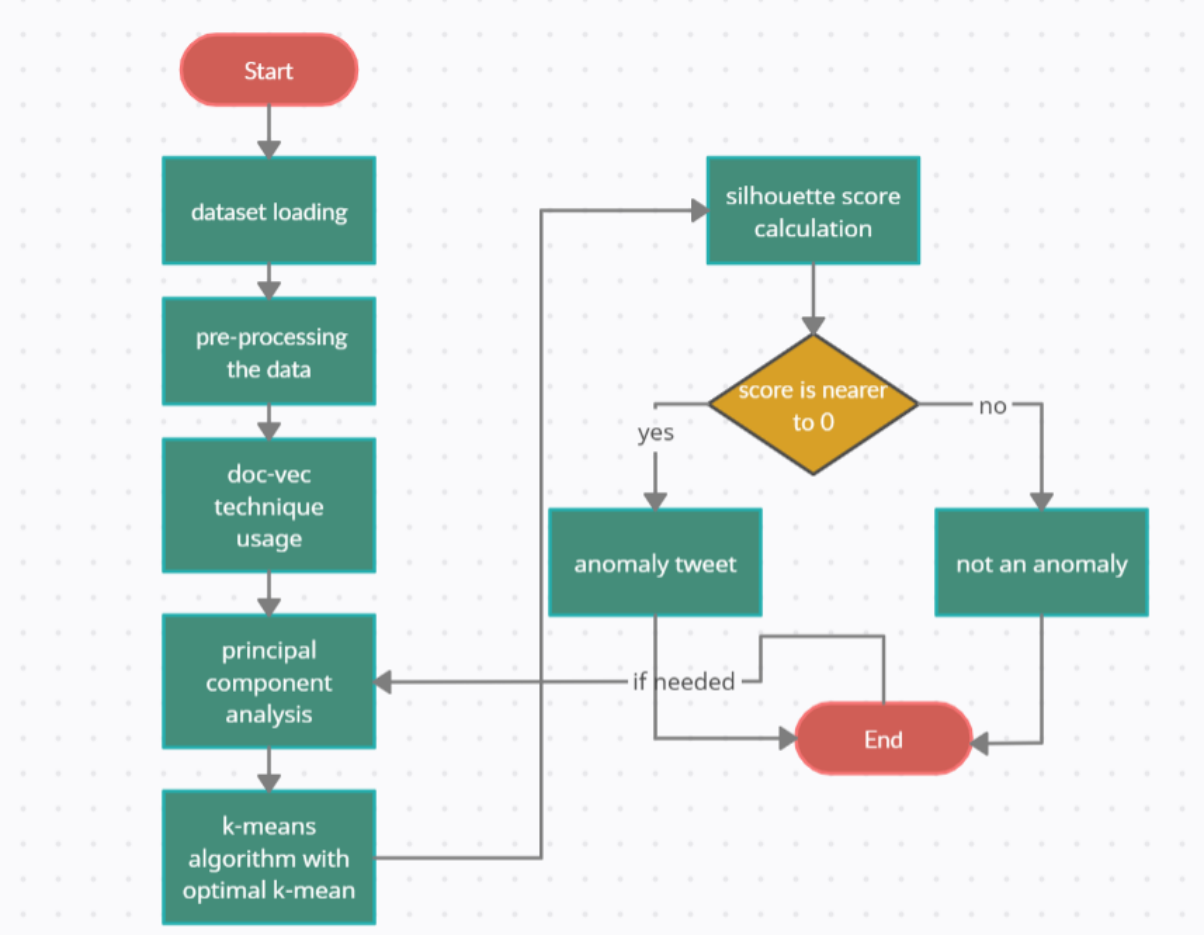
MODULES AND DESCRIPTION

- Tweet loader: loading tweet data into ide
- Doc2Vec transformer: transform tweet text to vector
- PCA: principal component analysis to find most relevant items
- Optimal k: finding optimal k for better clusters
- Silhouette score: will be the ration of inter cluster distance to intra cluster distance
- K- mean clustering: clustering the tweet data that is in vector format that has already been through PCA.
- Elbow graph to find the optimal k for k-means initialization
- Drawing the anomaly graphs from the detected anomalies

LANGUAGE AND PLATFORM

- Python language is used to implement the project
- Jupiter notebook environment used to run python project.

WORKFLOW DIAGRAM OR SYSTEM ARCHITECTURE



MODULES IMPLEMENTED

- Tweet data loading
- Tweet data transformation to the vector format
- Tweet data processing to draw elbow graph to find optimal k means
- PCA to find the principal components from the vectors
- Finding optimal k from the graph

SNAPSHOTS AND THEIR DESCRIPTION

Importing libraries and data into our ipynb socket

In [3]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy.sparse import csr_matrix
from mpl_toolkits.axes_grid1 import make_axes_locatable
from sklearn.cluster import KMeans
from sklearn.pipeline import Pipeline
from sklearn.metrics import mean_squared_error
from sklearn.base import BaseEstimator
from sklearn import utils
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_samples, silhouette_score
from gensim.models.doc2vec import TaggedDocument, Doc2Vec
from gensim.parsing.preprocessing import preprocess_string
import itertools
#here the silhouette_score helps us to effectively define the clusters
from sklearn.metrics import silhouette_samples, silhouette_score
import matplotlib.pyplot as plt
%matplotlib inline
```

In [6]:
```
#reading the csv from given location
all_tweets=pd.read_csv(r'C:\Users\asus\Desktop\Donald.csv')
#getting usable things from csv
atweets=all_tweets['Tweet_Text'];
#printing the top tweets
atweets.head()
```

Out[6]:
```
0    Today we express our deepest gratitude to all ...
1    Busy day planned in New York. Will soon be mak...
2    Love the fact that the small groups of protest...
3    Just had a very open and successful presidenti...
4    A fantastic day in D.C. Met with President Oba...
Name: Tweet_Text, dtype: object
```

DOC2Vetransformation function

In [8]:
```
import multiprocessing
from multiprocessing import *
from tqdm import tqdm
#doc2vec to convert the tweets to vectors
#training the doc2vectransform
# estimators specify all the parameters that can be set at the class level intheir __init__
#as explicit keyword arguments
#preprocessing would be converting the each text tweet into array tokens & rem oval of unwanted characters, stop words etc.
#conversion of each array of tokens corresponding to each text tweet into numerical vectors .
#Vector Space Model generation
#It is recommended to keep 'Doc2Vec' vector size from 100 to 300
class Doc2VecTransformer(BaseEstimator):
    def __init__(self, vector_size=100, learning_rate=0.02, epochs=20):
        self.learning_rate = learning_rate
        self.epochs = epochs
        self._model = None
        self.vector_size = vector_size
        self.workers =(multiprocessing.cpu_count())-1
    def fit(self, x, y=None):
        tagged_x = [TaggedDocument(preprocess_string(item), [index]) for index, item in enumerate(x)]
        model = Doc2Vec(documents=tagged_x, vector_size=self.vector_size, workers=self.workers)
        for epoch in range(self.epochs):
            #training the doc2vec
            model.train(utils.shuffle([x for x in tqdm(tagged_x)]), total_examples=len(tagged_x), epochs=1)
            model.alpha -= self.learning_rate
            model.min_alpha = model.alpha
            self._model = model
            return self
    def transform(self, x):
        arr = np.array([self._model.infer_vector(preprocess_string(item))
                        for index, item in enumerate(x)])
        return arr
```

Converting tweet text to vector forms

In [10]:
```
#Most of the algorithm needs fare amount data preprocessing.
#All these preprocessing and the actual algorithm can be configured as separate reusable steps.
#Together all these steps connected in a single entity with an inlet and an outlet is known as 'Pipeline'.
#text classification using vector modelling is used
pl = Pipeline(steps=[('doc2vec', Doc2VecTransformer())])
#fitting the tweets into array as vectors
print("THE DATA IN VECTOR FORMAT")
vectors_df = pl.fit(atweets).transform(atweets)
vectors_df
```

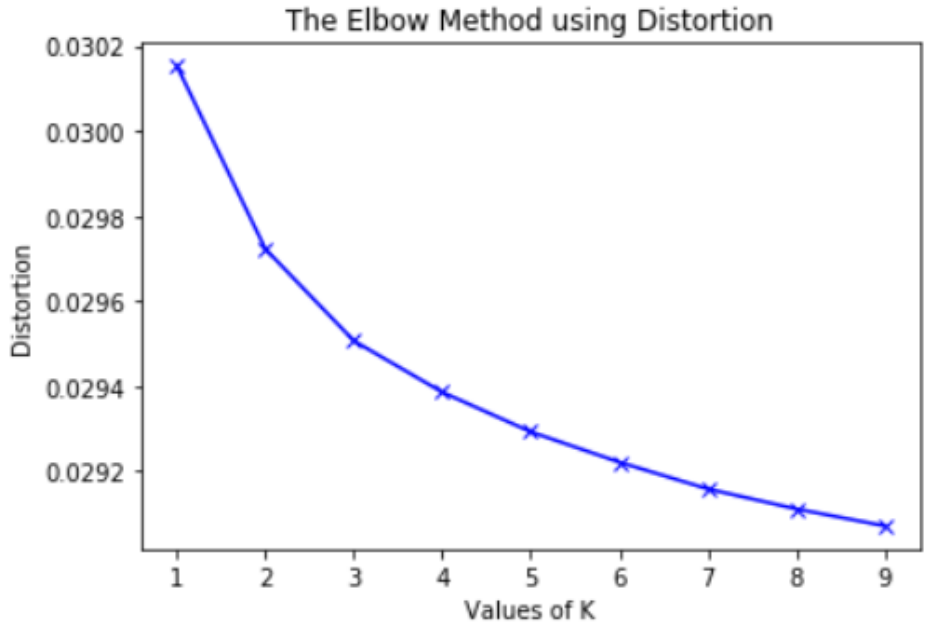
```
THE DATA IN VECTOR FORMAT
100%|██████████| 7375/7375 [00:00<00:00, 1257285.37it/s]
```

Out[10]:
```
array([[ 3.3161447e-03, -1.5800573e-03, -2.0159537e-03, ...,
        -4.2533246e-03, -3.2199034e-03,  3.6243123e-03],
       [ 1.8876344e-03,  2.8360514e-03, -2.6711379e-03, ...,
         8.1782963e-04, -1.3412925e-03, -2.2475929e-03],
       [-5.7338839e-05,  3.8820817e-04,  6.7350251e-05, ...,
         2.0729396e-03, -3.7986892e-03,  3.1896282e-03],
       ...,
       [ 3.2681543e-03,  4.9157022e-03, -3.0198712e-03, ...,
        -3.4252333e-03,  1.6677985e-03, -2.3971978e-03],
       [-1.8868389e-03,  2.7367289e-03,  1.6465839e-03, ...,
         4.2598285e-03, -3.0192579e-04,  8.3912944e-04],
       [-3.9616050e-03, -2.2417661e-03,  4.6779560e-03, ...,
         2.4034400e-03, -2.9716459e-03,  1.6145956e-03]], dtype=float32)
```

Finding optimal k

In [11]:
```
#finding the optimal k for the data
#by the elbow curve method
#Pairwise distances between observations in n-dimensional space.
from scipy.spatial.distance import cdist
distortions = []
inertias = []
mapping1 = {}
mapping2 = {}
K = range(1,10)
X=vectors_df
for k in K:
    #Building and fitting the model
    kmeanModel = KMeans(n_clusters=k).fit(X)
    kmeanModel.fit(X)
    distortions.append(sum(np.min(cdist(X, kmeanModel.cluster_centers_,'euclidean'),axis=1)) / X.shape[0])
    inertias.append(kmeanModel.inertia_)
    mapping1[k] = sum(np.min(cdist(X, kmeanModel.cluster_centers_,'euclidean'),axis=1)) / X.shape[0]
    mapping2[k] = kmeanModel.inertia_
plt.plot(K, distortions, 'bx-')
plt.xlabel('Values of K')
plt.ylabel('Distortion')
plt.title('The Elbow Method using Distortion')
plt.show()
```

The Elbow Method using Distortion



PARTIAL CONCLUSION

The project has been taken care in all the ways to sort out issues with the k-means. Anomaly detection being the leading ML system and one of the aspired model that many companies dream of to provide the customer with accurate information rather than false statements and news. Here we had made a system that convert the data into vectors and then process them using k-means clustering and based on the intra cluster distance and inter cluster distance we will be able to find the anomaly from the data.

REFERENCES

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy.sparse import csr_matrix
from mpl_toolkits.axes_grid1 import make_axes_locatable
from sklearn.cluster import KMeans
from sklearn.pipeline import Pipeline
from sklearn.metrics import mean_squared_error
from sklearn.base import BaseEstimator
from sklearn import utils
from sklearn.metrics import silhouette_samples, silhouette_score
from gensim.models.doc2vec import TaggedDocument, Doc2Vec
from gensim.parsing.preprocessing import preprocess_string
import itertools
#here the silhouette_score helps us to effectively define the clusters
from sklearn.metrics import silhouette_samples, silhouette_score
import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:
```
#reading the csv from given location
all_tweets=pd.read_csv(r'C:\Users\asus\Desktop\Donald.csv')
#getting usable things from csv
atweets=all_tweets['Tweet_Text'];
#printing the top tweets
atweets.head()
```

Out[2]:
```
0    Today we express our deepest gratitude to all ...
1    Busy day planned in New York. Will soon be mak...
2    Love the fact that the small groups of protest...
3    Just had a very open and successful presidenti...
4    A fantastic day in D.C. Met with President Oba...
Name: Tweet_Text, dtype: object
```

In [3]:
```
#prnting all the twets for reference
all_tweets.head()
```

Out[3]:

	Date	Time	Tweet_Text	Type	Media_Type	Hashtags	Tweet_Id	
0	16-11-11	15:26:37	Today we express our deepest gratitude to all ...	text	photo	ThankAVet	7.970000e+17	https://twitter.com/realD
1	16-11-11	13:33:35	Busy day planned in New York. Will soon be mak...	text	NaN	NaN	7.970000e+17	https://twitter.com/realD
2	16-11-11	11:14:20	Love the fact that the small groups of protest...	text	NaN	NaN	7.970000e+17	https://twitter.com/realD
3	16-11-11	2:19:44	Just had a very open and successful presidenti...	text	NaN	NaN	7.970000e+17	https://twitter.com/realD
4	16-11-11	2:10:46	A fantastic day in D.C. Met with President Oba...	text	NaN	NaN	7.970000e+17	https://twitter.com/realD

In [4]:
```
import multiprocessing
from multiprocessing import *
from tqdm import tqdm
#doc2vec to convert the tweets to vectors
#training the doc2vectransform
# estimators specify all the parameters that can be set at the class level int
heir __init__
#as explicit keyword arguments
#preprocessing would be converting the each text tweet into array tokens & rem
oval of unwanted characters, stop words etc.
#conversion of each array of tokens corresponding to each text tweet into nume
rical vectors .
#Vector Space Model generation
#It is recommended to keep 'Doc2Vec' vector size from 100 to 300
class Doc2VecTransformer(BaseEstimator):
    def __init__(self, vector_size=100, learning_rate=0.02, epochs=20):
        self.learning_rate = learning_rate
        self.epochs = epochs
        self._model = None
        self.vector_size = vector_size
        self.workers =(multiprocessing.cpu_count())-1
    def fit(self, x, y=None):
        tagged_x = [TaggedDocument(preprocess_string(item), [index]) for index
, item in enumerate(x)]
        model = Doc2Vec(documents=tagged_x, vector_size=self.vector_size, work
ers=self.workers)
        for epoch in range(self.epochs):
            #training the doc2vec
            model.train(utils.shuffle([x for x in tqdm(tagged_x)]), total_exam
ples=len(tagged_x), epochs=1)
            model.alpha -= self.learning_rate
            model.min_alpha = model.alpha
            self._model = model
            return self
    def transform(self, x):
        arr = np.array([self._model.infer_vector(preprocess_string(item))
                        for index, item in enumerate(x)])
        return arr
```

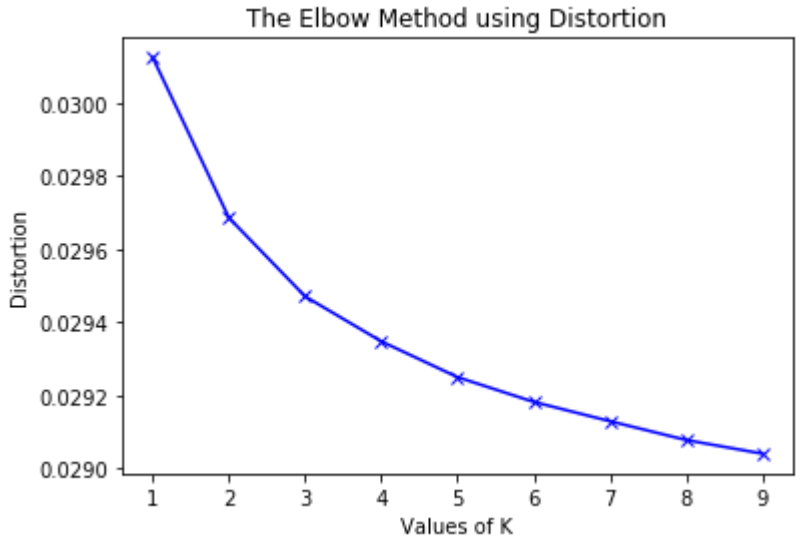
In [5]:
```
#Most of the algorithm needs fare amount data preprocessing.
#All these preprocessing and the actual algorithm can be configured as separat
e reusable steps.
#Together all these steps connected in a single entity with an inlet and an ou
tlet is known as ‘Pipeline’.
#text classification using vector modelling is used
pl = Pipeline(steps=[('doc2vec', Doc2VecTransformer())])
#fitting the tweets into array as vectors
print("THE DATA IN VECTOR FORMAT")
vectors_df = pl.fit(atweets).transform(atweets)
vectors_df
```

```
THE DATA IN VECTOR FORMAT

100%|██████████| 7375/7375 [00:00<00:00, 1903334.48it/s]
```

Out[5]:
```
array([[ 0.00470152,  0.00180077,  0.00145993, ...,  0.00064818,
         0.00254179, -0.00364475],
       [ 0.00123325, -0.00165454,  0.00377606, ..., -0.00280662,
        -0.00182993,  0.00325064],
       [ 0.00584012,  0.00193811,  0.00454637, ...,  0.00580276,
         0.00285712, -0.00300713],
       ...,
       [ 0.00244844, -0.00392837, -0.00595449, ..., -0.00284091,
         0.00108636, -0.00129191],
       [ 0.0021834 , -0.00108882,  0.00029738, ...,  0.00358891,
        -0.00179016,  0.00099751],
       [-0.00034241, -0.00041076, -0.00287249, ...,  0.00145002,
        -0.00173222, -0.00287513]], dtype=float32)
```

In [6]:
```
#finding the optimal k for the data
#by the elbow curve method
#Pairwise distances between observations in n-dimensional space.
from scipy.spatial.distance import cdist
distortions = []
inertias = []
mapping1 = {}
mapping2 = {}
K = range(1,10)
X=vectors_df
for k in K:
    #Building and fitting the model
    kmeanModel = KMeans(n_clusters=k).fit(X)
    kmeanModel.fit(X)
    distortions.append(sum(np.min(cdist(X, kmeanModel.cluster_centers_,'euclid
ean'),axis=1)) / X.shape[0])
    inertias.append(kmeanModel.inertia_)
    mapping1[k] = sum(np.min(cdist(X, kmeanModel.cluster_centers_,'euclidean'
),axis=1)) / X.shape[0]
    mapping2[k] = kmeanModel.inertia_
plt.plot(K, distortions, 'bx-')
plt.xlabel('Values of K')
plt.ylabel('Distortion')
plt.title('The Elbow Method using Distortion')
plt.show()
```



In [7]:
```
from sklearn.decomposition import PCA
def analyze_tweets_pca(n_pca_components):
    doc2vectors = Pipeline(steps=[('doc2vec', Doc2VecTransformer())]).fit(atwe
ets).transform(atweets)
    #principal components identification
    #after identification we find the variance
    #by varieance we select the best
    #there by we get the n_components
    #here we select 10 components and find the varieance
    pca = PCA(n_components=n_pca_components)
    pca_vectors = pca.fit_transform(doc2vectors)
    print('All Principal Components ..')
    print(pca_vectors)
    for index, var in enumerate(pca.explained_variance_ratio_):
        print("Explained Variance ratio by Principal Component ",(index+1), ":
", var)
analyze_tweets_pca(15)
```

```
100%|██████████| 7375/7375 [00:00<00:00, 921639.66it/s]

All Principal Components ..
[[ 0.00198247  0.00345114  0.00514515 ... -0.00022086  0.00116786
  -0.00067451]
 [ 0.0083067  -0.00267116 -0.00434863 ...  0.00062134  0.00282647
  -0.00024418]
 [ 0.00725623  0.00318805  0.00247387 ...  0.00206184  0.00329181
  -0.00037433]
 ...
 [-0.00701858 -0.00334757  0.00529489 ...  0.00325633 -0.00024097
  -0.00420547]
 [-0.00037548  0.0094165   0.00452123 ... -0.00406397  0.00156622
   0.00334806]
 [-0.00063491 -0.00453742  0.00191471 ...  0.00427892 -0.00179276
  -0.00204991]]
Explained Variance ratio by Principal Component  1 :  0.047422312
Explained Variance ratio by Principal Component  2 :  0.03204466
Explained Variance ratio by Principal Component  3 :  0.020698901
Explained Variance ratio by Principal Component  4 :  0.016689697
Explained Variance ratio by Principal Component  5 :  0.011114328
Explained Variance ratio by Principal Component  6 :  0.0109341405
Explained Variance ratio by Principal Component  7 :  0.010738972
Explained Variance ratio by Principal Component  8 :  0.010569126
Explained Variance ratio by Principal Component  9 :  0.010469232
Explained Variance ratio by Principal Component  10 :  0.01034429
Explained Variance ratio by Principal Component  11 :  0.010270455
Explained Variance ratio by Principal Component  12 :  0.010195784
Explained Variance ratio by Principal Component  13 :  0.010150746
Explained Variance ratio by Principal Component  14 :  0.010046727
Explained Variance ratio by Principal Component  15 :  0.00990078
```

In [8]:
```
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_samples, silhouette_score
#optimal kmeans clusterer
#here in the given range we have to find the optimal cluster
#from the above curve we can directly enter the optimal k
global optk;
optk=3;
class OptimalKMeansTextsClusterTransformer(BaseEstimator):
    def __init__(self, min_k, max_k):
        self.min_k = min_k
        self.max_k = max_k
    def fit(self, x, y=None):
        return self
    def transform(self, x):
        range_of_k = [x for x in range(self.min_k, self.max_k)]
        clusterer_pool = multiprocessing.Pool(processes=len(range_of_k))
        clusterer_process_responses = []
        for k in range_of_k:
            clusterer_process_responses.append(clusterer_pool.apply_async(self
._silhouette_score_with_k_, args=(x, k,)))
            optimal_k = optk
            clusterer_pool.close()
            print("Optimal k: ", optimal_k)
            optimal_clusterer = KMeansClusterer(num_means=optimal_k, distance=
cosine_distance, repeats=3)
            optimal_cluster_labels = optimal_clusterer.cluster(vectors=x, assi
gn_clusters=True, trace=False)
            return x, optimal_cluster_labels
    def _silhouette_score_with_k_(self, vectors, k):
        clusterer = KMeansClusterer(num_means=k, distance=cosine_distance, rep
eats=3)
        cluster_labels = clusterer.cluster(vectors=vectors, assign_clusters=Tr
ue, trace=False)
        silhouette_score_k = silhouette_score(X=vectors, labels=cluster_labels
, metric='cosine')
        return k, silhouette_score_k
```

In [9]:
```
def plot_tweets_k_means_clusters_with_anomalies(pca_vectors, cluster_labels, p
ca_vectors_anomalies):
    pca_vectors_anomalies_x = []
    pca_vectors_anomalies_y = []
    for pca_vectors_elem in pca_vectors_anomalies:
        pca_vectors_anomalies_x.append(pca_vectors_elem[1])
        pca_vectors_anomalies_y.append(pca_vectors_elem[0])
    plt.title('Kmeans Cluster of Tweets')
    plt.scatter(x=pca_vectors[:, 1], y=pca_vectors[:, 0], c=cluster_labels)
    plt.scatter(x=pca_vectors_anomalies_x, y=pca_vectors_anomalies_y, marker=
'^')
    plt.show()

def plot_scatter_silhouette_scores(top_n_silhouette_scores, tweets_dict, silho
uette_score_per_tweet):
    plt.close('all')
    fig, (ax1, ax2) = plt.subplots(nrows=2, ncols=1, sharex=True)
    fig.suptitle('Silhouette Scores vs Tweets')
    sub_plot_scatter_silhouette_scores(ax=ax1, top_n_silhouette_scores=top_n_s
ilhouette_scores,tweets_dict=tweets_dict,silhouette_score_per_tweet=silhouette
_score_per_tweet,with_annotation=False)

    sub_plot_scatter_silhouette_scores(ax=ax2, top_n_silhouette_scores=top_n_s
ilhouette_scores,tweets_dict=tweets_dict,silhouette_score_per_tweet=silhouette
_score_per_tweet,with_annotation=True)
    plt.show()
def sub_plot_scatter_silhouette_scores(ax,top_n_silhouette_scores, tweets_dict
, silhouette_score_per_tweet, with_annotation):
    ax.set(xlabel='Tweet Index', ylabel='Silhouette Score')
    ax.scatter(*zip(*silhouette_score_per_tweet))
    ax.scatter(*zip(*top_n_silhouette_scores), edgecolors='red')
    if with_annotation:
        for (index, score) in top_n_silhouette_scores:
            ax.annotate(tweets_dict[index], xy=(index, score), xycoords='data'
)
```

In [11]:
```
#Value close to -1 :
#The data point is wrongly put in a cluster to which ideally it should not bel
ong to. Basically it is an ‘inlier’.
#Value close to 0:
#The data point should not belong to any cluster and should be separated out.I
t is an ‘outlier’ or ‘anomaly’.
#Value close to 1:
#The data point is perfectly placed in a right cluster.
from nltk.cluster.util import VectorSpaceClusterer
from nltk.cluster import KMeansClusterer,cosine_distance
def sort_key(t):
    return t[0]
def determine_anomaly_tweets_k_means(top_n):
    print("detecting anomaly tweets please wait.........")
    tweets_dict =atweets
    tweets = atweets
    pl = Pipeline(steps=[('doc2vec', Doc2VecTransformer()),('pca', PCA(n_compo
nents=14)),('kmeans', OptimalKMeansTextsClusterTransformer(min_k=2, max_k=5
))])
    pl.fit(tweets)
    pca_vectors, cluster_labels = pl.transform(tweets)
    #sending the samples for processing
    silhouette_values = silhouette_samples(X=pca_vectors, labels=cluster_label
s, metric='cosine')
    tweet_index_silhouette_scores = []
    absolute_silhouette_scores_tweet_index = []
    for index, sh_score in enumerate(silhouette_values):
        #appending the scores
        absolute_silhouette_scores_tweet_index.append((abs(sh_score), index))
        tweet_index_silhouette_scores.append((index, sh_score))
        #sorting of the scores
    sorted_scores = sorted(absolute_silhouette_scores_tweet_index, key=sort_ke
y)
    #top anomaly scores into a array
    top_n_silhouette_scores = []
    pca_vectors_anomalies = []
    print("Top ", top_n, " anomalies")
    for i in range(top_n):
        abs_sh_score, index = sorted_scores[i]
        index_1, sh_score = tweet_index_silhouette_scores[index]
        top_n_silhouette_scores.append((index, sh_score))
        print(tweets_dict[index])
        print('PCA vector', pca_vectors[index])
        pca_vectors_anomalies.append(pca_vectors[index])
        print('Silhouette Score: ', sh_score)
        print("_____________________________________________________")
        print(" ")
    plot_tweets_k_means_clusters_with_anomalies(pca_vectors=pca_vectors, pca_v
ectors_anomalies=pca_vectors_anomalies,cluster_labels=cluster_labels)
    plot_scatter_silhouette_scores(top_n_silhouette_scores=top_n_silhouette_sc
ores,tweets_dict=tweets_dict,silhouette_score_per_tweet=tweet_index_silhouette
_scores)
determine_anomaly_tweets_k_means(10)
```

```
detecting anomaly tweets please wait.........
100%|██████████| 7375/7375 [00:00<00:00, 1476303.73it/s]
```

```
Optimal k:  3
Top  10  anomalies
It was great spending time with @joniernst yesterday. She has done a fantasti
c job for the people of Iowa and U.S. Will see her again!
PCA vector [ 0.00186537  0.00754094  0.00539339  0.00102686  0.00114347  0.00
061782
  0.00061786 -0.0038282  -0.000947    0.00057167  0.00346713  0.00350608
  0.00231843 -0.00368393]
Silhouette Score:  6.4842076e-05
```

```
"@Techn9cian1923: @ChrisCuomo It felt Like a moment of silence when U said @r
ealDonaldTrump DOESNT PLAY! Powerful Interview_G_G #Trump2016"
PCA vector [-5.3342218e-03 -1.8548984e-03  1.3894340e-02  2.6054247e-03
 -6.3420208e-03 -5.3898646e-03 -1.1555320e-03  5.8149681e-03
  6.0497341e-03  1.0760233e-03 -1.9433412e-03  4.6810722e-03
 -4.0167914e-05 -1.6371445e-03]
Silhouette Score:  -0.000107495325
```

```
Stop the assault on American values. Stand w/ Trump to #MakeAmericaGreatAgai
n!
#VotersSpeak: https://t.co/XRRJ0fMkNV https://t.co/c7EHokbLD1
PCA vector [ 2.0306739e-03  6.7232377e-03 -4.1833865e-03 -3.3214951e-03
  4.1794858e-04  4.3263403e-03  2.3507087e-03  7.8640151e-06
 -4.0603185e-04 -2.7391184e-03 -2.0260948e-03 -8.4903464e-04
  1.1898897e-03 -5.8290903e-03]
Silhouette Score:  0.00013600668
```

```
RT @Morning_Joe: .@realDonaldTrump. Tomorrow on #morningjoe http://t.co/OBzi5
pCTgP
PCA vector [ 0.00602246 -0.00837444  0.00054292  0.00319191 -0.00036352 -0.00
244041
  0.00083693 -0.00100034  0.00124703 -0.00285623  0.00701661  0.00174718
 -0.00142784 -0.00074404]
Silhouette Score:  -0.000196283
```

```
"@back2reason: @realDonaldTrump Cant wait for President Trump to put things i
n order in US. We desperately need patriot with,finally,brains
PCA vector [-0.00608795 -0.00178571  0.00205446 -0.00254428 -0.00240148  0.00
111504
 -0.00047146 -0.0016099  -0.00170187 -0.00146673  0.00155945 -0.00455487
 -0.00185753 -0.00183605]
Silhouette Score:  0.00021173724
```

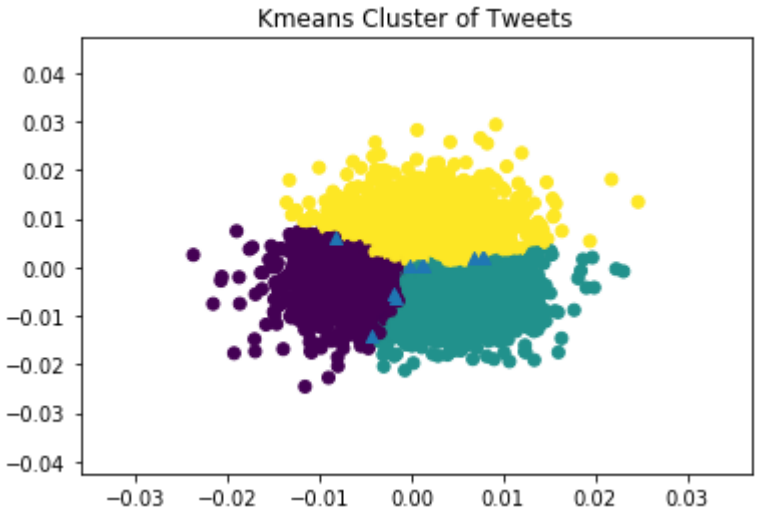
```
"@JoeNBC: Marco Rubio just criticized Ted Cruz for underperforming tonight. W
ow. #SuperTuesday"
PCA vector [ 0.00014894  0.00136492 -0.00347107  0.0071239   0.01031306  0.00
346299
  0.00369569 -0.00280159 -0.0072382  -0.00321078 -0.00626918 -0.0020376
 -0.00500365  0.00272727]
Silhouette Score:  0.00021905822
```

```
"@ObamaTax: After @hillaryclinton poor performance, waffling in debates, @Tea
msters14 should support @realDonaldTrump"
PCA vector [ 3.8930943e-04 -2.1518330e-04 -4.8990608e-03 -4.4644373e-03
  3.0614103e-03 -6.3866255e-04 -5.0695725e-03  2.9459572e-03
 -2.2295411e-03 -8.2337589e-05  7.8796525e-05 -3.6605950e-03
 -5.8224108e-03 -1.2193329e-03]
Silhouette Score:  0.0002447471
```

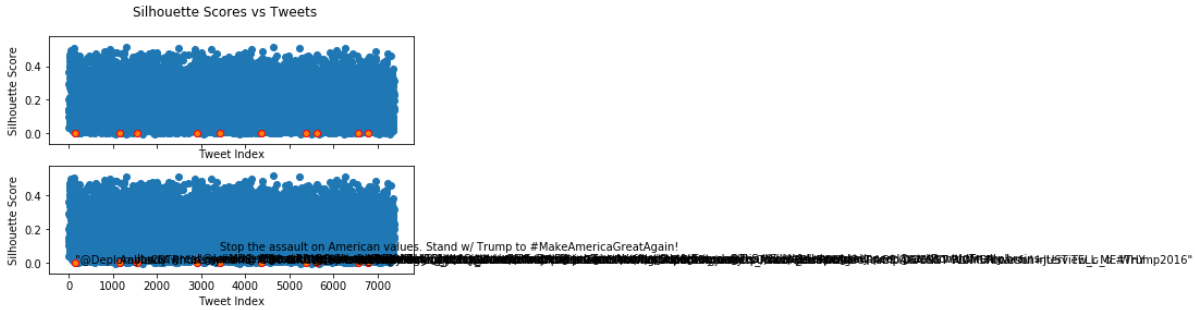
```
Anybody whose mind "SHORT CIRCUITS" is not fit to be our president! Look up t
he word "BRAINWASHED."
PCA vector [-0.01406467 -0.00428496  0.00644494 -0.00471505  0.00593114 -0.00
087606
  0.00082348  0.00071711  0.00170644  0.00224782 -0.00288273  0.00224346
 -0.00054251 -0.00520311]
Silhouette Score:  -0.00025852205
```

```
"@DeplorableCBTP: "In my mind, #DonaldTrump is the only way out of this mes
s." - #PhilRobertson of TVs #DuckDynasty"   Thank you Phil!
PCA vector [ 0.00030347  0.00116252 -0.00512332  0.00242922  0.00156919 -0.00
246156
  0.00167349  0.00600644 -0.00036886 -0.00104297 -0.00651388  0.00154924
 -0.00013761 -0.00052862]
Silhouette Score:  -0.0004012313
```

```
"@cher: Never thought Id Say_Donald Trump is a Giant Among Gop front runners_
cruz=devil  rubio=RAGE AGAINST WOMEN carson=JUST TELL ME WHY
PCA vector [ 0.00215077  0.00789872 -0.00151098  0.00101303 -0.00176074  0.00
051779
 -0.00426332  0.0041446   0.0080331  -0.00262157  0.00231771  0.00082622
 -0.00086091  0.00723814]
Silhouette Score:  0.00046437525
```



```
C:\Users\asus\anaconda3\lib\site-packages\matplotlib\backends\backend_agg.py:
211: RuntimeWarning: Glyph 1770 missing from current font.
  font.set_text(s, 0.0, flags=flags)
C:\Users\asus\anaconda3\lib\site-packages\matplotlib\backends\backend_agg.py:
180: RuntimeWarning: Glyph 1770 missing from current font.
  font.set_text(s, 0, flags=flags)
```



In [ ]: