Data Representation, Reduction and Analysis: Project Assignment

Satyam Kapoor (0534338)
Jeroen Van Soest (0517324)

Contents

1. Introduction....3
2. Pre-Processing....3
3 Feature extraction....4
3.1 TF-IDF....4
4 Method-1: *k*-means Clustering and Consensus matrix for Noise Removal....5
5 Method-2: Density-Based Spatial Clustering of Applications with Noise (DBSCAN) 6
6 Clustering....7
7 Visualisation....9
Code guideline....10
References....12

1. Introduction

This project is a part of course Data Representation, Reduction and Analysis. We've been provided with the data (2,000 tweets from London area). We have been given the instruction to preprocess the tweet followed by clustering. The preprocessing will mainly involve noise removal. In clustering, we're focusing on mainly two methods, K-means clustering and DBSCAN.

We used Jupyter notebook to work on the project. Since it is a group project, worked upon by two members, git is used as the Version Control System and all files are also present on GitHub. Link - https://github.com/satyamkapoor/project_dataanalysis (Public as from January 22, 2018).

The report is written using the task heading as provided in the project description. We divided the different steps. So that everyone provided an equal share of code.

Pre-processing: Jeroen
Feature extraction tf idf: jeroen
Noise filtering DBSCAN: Satyam
Noise filtering kmeans: Jeroen
K means clustering of the tweets: Satyam
Visualization: Satyam
Writing the report: Jeroen & Satyam

2. Pre-Processing

The provided data file is a list containing 2,000 harvested tweets from the London area. The steps are as follows:

1. Tokenization of the tweets.
2. Removal of stop words (using nltk English stop words with some custom stop words added).
3. filtering of the names starting with '@' sign.
4. filtering of the non-alphanumeric characters and numbers.
5. Removing the URL's

6. We also chose to lemmatize the words instead of stemming the words, since it provided more accurate results for some words. We did this using the Wordnet stemmer

First, we will preprocess the data. Then we will remove noise and stop words. Subsequently we have to transform the textual data to numerical features to be used in our clustering methods with TF-IDF. For the clustering we use two methods: k-Means Clustering and Consensus Matrix for Noise Removal and Density-Based Spatial Clustering of Applications with Noise (DBSCAN). In these clustering methods we have to remove the noise. Finally, we visualize the data.

3 Feature extraction

3.1 TF-IDF

Before being able to cluster the data, we need to transform them into feature vectors with TF-IDF. The TF-IDF function is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus. It is usually used as a weighting factor in searches of information retrieval, text mining and user modeling. The value of TF-IDF increases proportional to the appearance of a word in a document, but the frequency of the word in the corpus offsets it.

TF-IDF is a product of two statistics, the term frequency and the inverse document frequency.

Term frequency The term frequency tf(t,d) is the number of times the term *t* occurs in a document *d*. It is equal to the occurrence of word in a document / total number of words

$$f_{t,d} \Big/ \sum_{t' \in d} f_{t',d}$$

Inverse Document Frequency It is the measure of how much information the word provides, that is, whether the term is common or rare across all documents

$$\operatorname{idf}(t, D) = \log \frac{N}{|\{d \in D : t \in d\}|}$$

In the project, first, we create a vocabulary of words. Then we calculate the IDF of every word and then we calculate the TF-IDF of each word to get a feature matrix which we are denoting by feature_matrix.

```
: feature_matrix = tweet_tfidf_features(all_tweets)
  feature_matrix.shape

: (2001, 4480)
```

Figure 1 Dimensions of feature matrix

4 Method-1: *k*-means Clustering and Consensus matrix for Noise Removal

The TF-IDF values for the tweets are stored in a feature matrix. This matrix is used, along with the cosine distance function, to cluster the data. In a loop we go over different values of *k*, $k = [2 - 15]$.

First we wanted to use the Euclidian distance, but we encountered some problems with it. First of all many of the tweets using Euclidian distance, were all grouped into one cluster, and secondly, we had some issues with numerical stability. Therefore we opted to use the cosine distance.

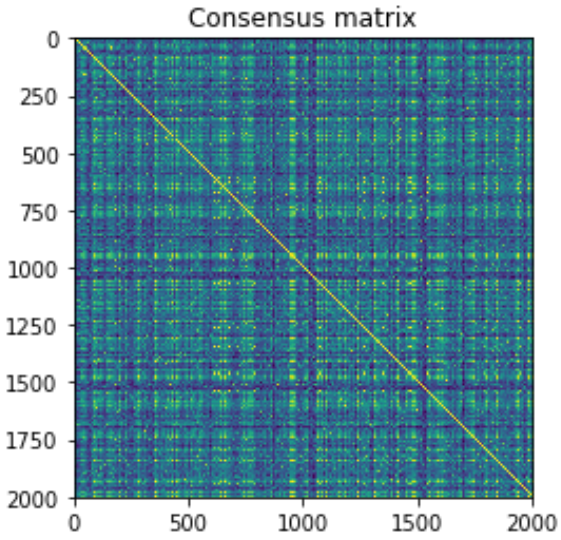


Figure 2 Consensus Matrix

For each k value we complete the consensus matrix as described in the assignment (+1 for each time two tweets are clustered together, set to 0 if not clustered for over 10% of the time.) Since we let k-Means run 2 – 15 times we decided to discard all tweets that are 1.5 rounded off to: 2 times or

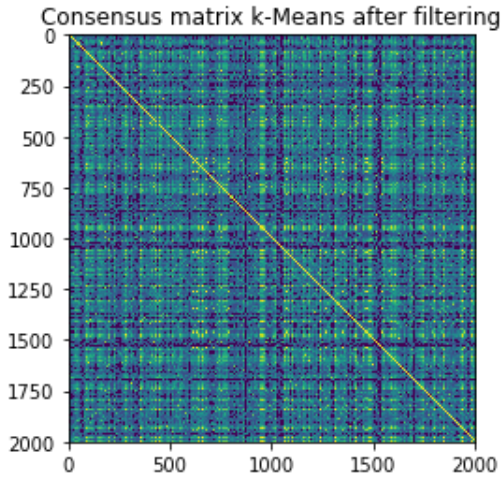


Figure 3 Consensus matrix k-means after filtering

After using k-Means for noise removal, we were left with `1019 tweets. Note that k-Means centroids are randomly initialized, thus this can vary.`

5 Method-2: Density-Based Spatial Clustering of Applications with Noise (DBSCAN)

For DBSCAN, initially we tried to go with implementation done in the Exercises session, but we were getting a lot of noise points, and very few core points, so we decided to with the Scikit Lean's DBSCAN implementation.

The parameters of DBSCAN we use in the project are:

- minPts = 5
- eps – 0.91 to 1.05 (increment of 0.03 in each step)

The results which we found after running DBSCAN 5 times with different eps are:

```
0 , eps =  0.91
3 clusters found
1 , eps =  0.94
3 clusters found
2 , eps =  0.97
3 clusters found
3 , eps =  1.0
3 clusters found
4 , eps =  1.03
3 clusters found
```

Figure 4 Iterations of DBSCAN

Thereafter, we created an n x m matrix to filter out the noise, as per the requirement of the project. Where m = the tweets, and n = the runs of the DBSCAN. We marked – 1 for the dense points and 0 for border or outlier, as for filtering we must count the number of border and outliers.

If a tweet was marked as an outlier or border point in more than 50% of the runs our DBSCAN, then we classified the tweet as noise and removed it at the end.

Thus, after noise removal, we were left with 973 tweets in total, which was almost the same result as for using k-Means for noise removal.

6 Clustering

Now we were with two datasets which we got from noise removal by K-means and DBSCAN respectively.

As per the requirement, we ran K-means from k = 2 to k = 12 and created consensus matrices for both the datasets.

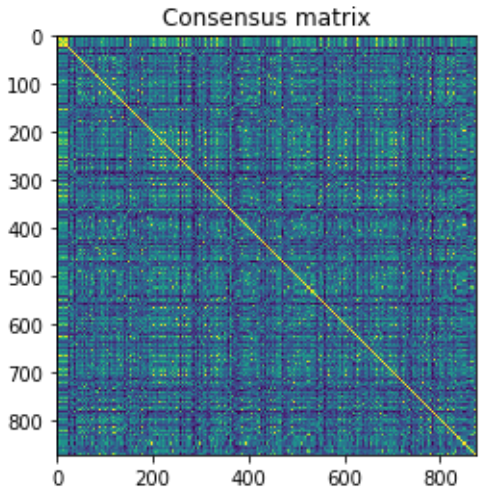


Figure 5 Consensus matrix - data set in which noise removed by K-means

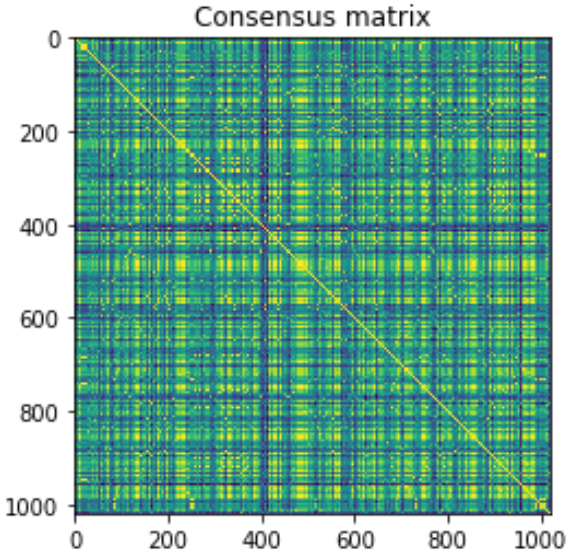


Figure 6 Consensus matrix in which noise removed by DBSCAN

After that, we implemented k-means with k-9 on the consensus matrices we created the previous step.

Once done, top ten tweets of each cluster we take out –
For DBSCAN followed by for Kmeans

Cluster 0:	Cluster 1:	Cluster 2:	Cluster 3:	Cluster 4:	Cluster 5:	Cluster 6:	Cluster 7:	Cluster 8:
nearby	berlin	Figure	year	Swear	Inhouse	Inhouse	Nipply	Blueprint
oneills	roasted	cook	behind	Life	Cold	Cold	Fish	Wtf
craigutter	dip	application	enough	Unity	Caf	Caf	Using	wasnt presser
existng via	Philip	stood	barrister	Joe	Roasted	Roasted	Kerr	tool
debate	Van	president	april	met acting	Gift	Gift	sex fitness	poxy heading waving
stickup	visited involved	without	cancer idf	puppet defeat	Global	Global	catnip	accepted basic
halloween gonna	understand deliver hearing	km	sharpish tw	global juneja	meet	meet	value	
		atrocity	simplicity	recommended	speechless	speechless	help	
		enews			gonna	gonna pre	stop	
		jimmy			pre			

Cluster 0:	Cluster 1:	Cluster 2:	Cluster 3:	Cluster 4:	Cluster 5:	Cluster 6:	Cluster 7:	Cluster 8:
half	detained	detained	france	attending	pint	tnitefor	detained	turn
down	westminster		benefit	end	cctv	announce	westminster	liked
gmt	time	westminster	wtf	tackle	gift	neighbourhood	time	goodwill
beer	sea	time	step	pub	repeating	ignored	sea	seems
legally	nationwide	sea	wouldnt	saturday	listen	yr	nationwide	music
wine	spring	nationwide	reflect	hashtag	bcg	think	spring	busted
acted	clapping	spring	st		anytime	broken	clapping	shouldve
smoothie	wildly	clapping	venue	referendum	managing	uefa	wildly	seat
bait	grownup	wildly	method	answer	kill	custody	grownup	usa
johnny	Branagh	grownup	else	jury	tbh	politically	branagh	bastille
		branagh		guardian				

7 Visualisation

We generated a matrix for visualization in Gephy.

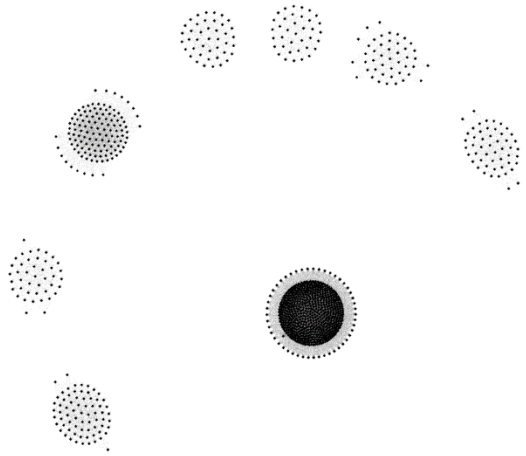
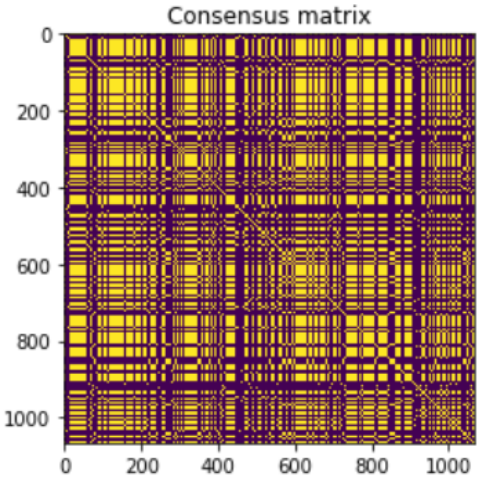


Figure 7 Visualisation for Gephy



Code guideline

The following files are included in the project:

Data Project.ipynb
Csv_helper.py
Tweets_2016London.csv
Report.pdf

All the main code is contained in a python notebook

The following packages need to be installed before running the notebook:
Nltk, numpy, sklearn, pandas, re, matplotlib, collections

Note that running the nltk part will download an English corpus.

The notebook is divided in different parts:

1 Preprocessing
2 Feature extraction with TF-idf
3 Noise removal kMeans
  3.1 Noise removal kMeans
  3.2 Noise removal kMeans
  3.3 Noise removal kMeans
4 Noise removal DBSCAN
  4.1 creating the matrix to filter out noise points
  4.2. Filter the DBSCAN noisy tweets out and produce the two datasets
5 Running K means on the two filtered datasets
6 Final results
7 Top ten of each cluster

References

[1] "Weighting word using tfidf," [Online]. Available: https://nlpforhackers.io/tf-idf/.

[2] "TFIDF explained," [Online]. Available: http://billchambers.me/tutorials/2014/12/21/tf-idf-explained-in-python.html.

[3] "scikit learn machine learning in python," [Online]. Available: http://scikit-learn.org/stable/.

[4] "clustering text documents using scikit learn," [Online]. Available: https://stackoverflow.com/questions/27889873/clustering-text-documents-using-scikit-learn-kmeans-in-python.

[5] "sklearn.cluster.DBSCAN," [Online]. Available: http://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html.

[6] "Numpy," [Online]. Available: http://www.numpy.org/.