Clustering Analysis of Text Data with K-means and DB Scan Algorithms

Hieu Nguyen, Nithya Devadoss, Ramesh Simhambhatla, Ramya Mandava Sep 13, 2018

**Abstract**

With the growth of Internet, large amount of text data is increasing, which are created by different media like social networking sites, web, and other informatics sources, etc. This data is in unstructured format which makes it tedious to analyze it, so we need methods and algorithms which can be used with various types of text formats. Clustering is an important part of the data mining. Clustering is the process of dividing the large &similar type of text into the same class. Clustering is widely used in many applications like medical, biology, signal processing, etc. This paper briefly covers the various kinds of text clustering algorithm, present scenario of the text clustering algorithm, analysis and comparison of various aspects which contain sensitivity, stability. Algorithm contains traditional clustering like hierarchal clustering, density based clustering and self-organized map clustering.

Text Clustering is an important part of the data mining and clustering of input data. Clustering is the process of separating the similar type of samples into the same class. It has applications in automatic document organization, topic extraction and fast information retrieval or filtering. This paper briefly discusses 2 kinds of text clustering algorithms – K Means and DB Scan – with clustering of documents based on the word frequency provided from those documents, analysis and comparison with testing of different cluster sizes, and findings from our testing.

1 Introduction

With the large use of computer and rapid development of Internet technology, a large amount of unstructured data is generated by various devices & application. This data is stored in various systems. This system has a lot of information and to extract the appropriate information from this system manually is an impossible task for any organization. Clustering can be very useful to remove the above problem and to bind similar documents together. Our corpus has 1500 documents and 12420 unique words on which, we would like to try K-means and DBSCAN clustering techniques which will be further explained in section 2.

However, in order to perform clustering, the unstructured data will have to be quantified. This can be achieved by looking at the words that make up the document and calculating the corresponding term frequency – inverse document frequency(tf-idf) scores.

tf is a measure of how important a word can be within a given document and it is calculated by looking at a word’s frequency in a given document and is calculated as shown in formula 1, where count is the frequency of a word per document. It is important to note that a document can have highly frequent but insignificant words such as ‘in’, ‘the’, ‘is’ etc., which are also called stop words that might have to be removed to better understand the content and context of the document. On the other hand, idf (Inverse document frequency) is the inverse of how many times a word repeats across all the documents within a corpus. Idf decreases the score for most commonly used words and increases the weight for those that are not very common across the corpus and is calculated as shown in formula 2, where N = total number of documents in the corpus and n= number of documents a given word appears in.

In our approach we would like to use the tf-idf scores which is a combination of tf and idf and as defined in formula 3.

**(1)**

(2)

(3)

tf and idf formulas are slightly modified to normalize the data and also to remove ‘undefined’ values arising due to sparse representations. Once the tf-idf scores are ready, these values are then assigned to a matrix called tfidf of dimensions: ***i by j***where, ***i*** is the number of documents and ***j*** is the total number of unique words or vocab size of the entire corpus. This newly derived matrix is then ready for clustering in order club similar documents together.

2 Methods

Distance measure is needed to define similarity between documents, and then a criterion function to compute the quality of our clusters and finally a criterion optimization algorithm. Distance based partitioned clustering is widely used in database literature for achieving distance measure defined above. Partitioned clustering divides the data set into k disjoint clusters by using the distance measurements. Each k cluster contains the homogeneous data. One of the most widely used distance based partitioned clustering algorithm is k-means [15].

2.1 Clustering with k-mean algorithm

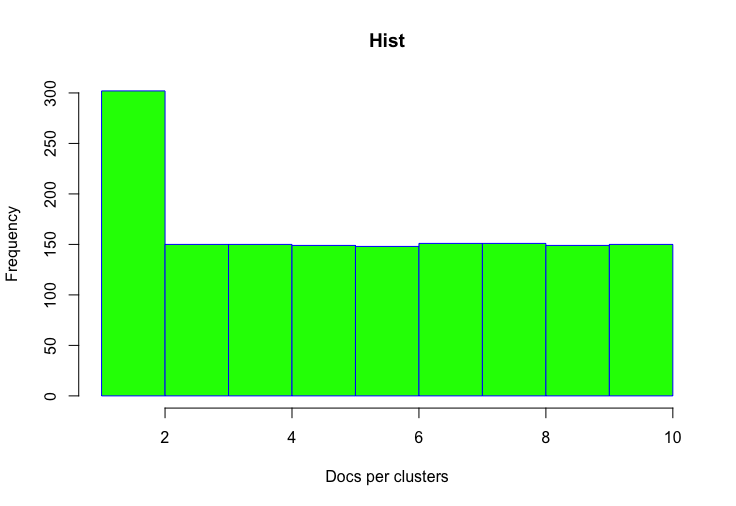
The goal of k-means algorithm is based on the input parameters k, in which the data set is divided into k clusters. Algorithm uses iterative update in each round, based on k point of references which were grouped around k clusters. Each cluster centroid will be used as a reference point for next round of iteration. Iteration makes the selected reference point closer to the true cluster centroid, so the clustering effect gets better.

K-means has been applied on tfidf matrix calculated in section 1, for various cluster sizes (5, 10, 20) and similar documents are plotted against their clusters as shown in fig 2.1.

|  |  |  |
| --- | --- | --- |
|  |  |  |
| **Documents/cluster using k-means with 5 clusters** | **Documents/cluster using k-means with 10 clusters** | **Documents/cluster using k-means with 20 clusters** |

**Figure 2.1**

We have noticed that the data is better partitioned with about 10 clusters. K=20 yielded few overlapping clusters and splitting some of similar documents across different clusters. So, our final model for K-mean clustering uses a value of 10 for the number of clusters. And the below histogram (figure 2.2) shows the frequency of documents across all 10 clusters.



**Figure 2.2: Histogram of Documents/cluster using k-means with 10 clusters**

**K-means Usage:**

kmeans(x, centers, iter.max = 10, nstart = 1, algorithm = c("Hartigan-Wong", "Lloyd", "Forgy", "MacQueen"), trace=FALSE)

### Key Arguments used in the test:

|  |  |
| --- | --- |
| x | numeric matrix of data, or an object that can be coerced to such a matrix (such as a numeric vector or a data frame with all numeric columns). |
| centers | either the number of clusters, say *k*, or a set of initial (distinct) cluster centers. If a number, a random set of (distinct) rows in x is chosen as the initial centers. |

2.2 Clustering with Density Based Clustering (DBScan)

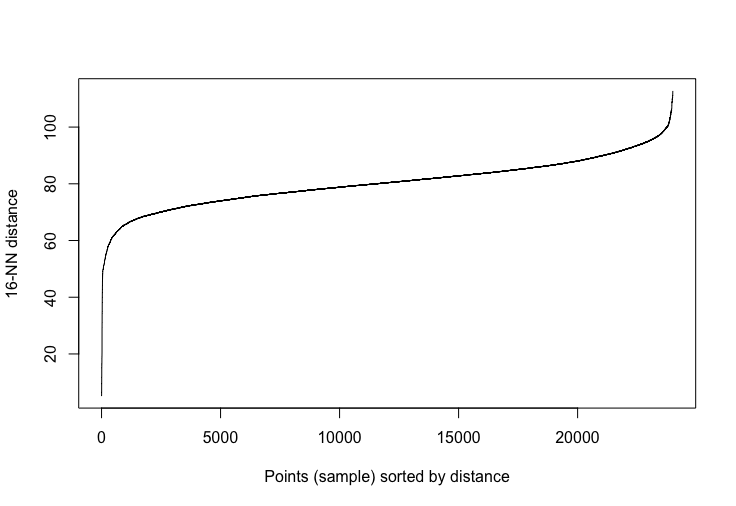
It is very difficult to cluster when data is in nonlinear shape means in different size, density and shape, so k-mean algorithm can’t be applied on this type of data. Density based clustering algorithm is very useful to classify this type of data, in density-based clustering algorithm cluster is defined as area of higher density that is the remainder of the dataset. Density based clustering considers density and boundary area of the cluster. One of the most widely used density based spatial clustering with the application of noise is DBSCAN.

DBSCAN has been applied on ***tfidf*** matrix calculated in section 1, for various [eps and min points] combinations ([65,25], [70,16], [77,16], [90,16]) and similar documents are plotted against their clusters as shown in fig 2.3.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| DBSCAN with eps=65,min points= 25 | DBSCAN with eps=70, min points= 16 | **DBSCAN with eps=77, min points= 25** | DBSCAN with eps=90, min points= 25 |

**Figure 2.3**

***Due to lack of domain knowledge on our corpus, KNNdistplot has been used with k= log (number of data points) in order to obtain the optimal eps and min points values to be used for DBSCAN algorithm. Best value for eps has been identified to be around 77 (knee point) as shown in figure 2.4 and min points to be 16(which is log (number of data points)). A knee corresponds to a threshold where a sharp change occurs along the k-distance curve.***



**Figure 2.4 kNN dist plot**

We have noticed that the data is better partitioned with about 10 clusters. So, our final model for DBSCAN clustering uses eps=77 and min points =16.

**DB Scan Usage:**

dbscan(x, eps, minPts = 5, weights = NULL, borderPoints = TRUE, ...)

### Key Arguments used in the test:

|  |  |
| --- | --- |
| x | a data matrix or a dist object. |
| eps | size of the epsilon neighborhood |
| minPts | number of minimum points in the eps region (for core points). Default is 5 points. |

3 Results

k-means algorithm has been applied initially with an initial choice of 5, 10 and 20 clusters. After visual exploration of the results as shown in section 2, k-means with 10 clusters yielded best results – clusters are more evenly distributed, and no overlapping has been observed.

We further tested our data with DBSCAN across a range of eps and min-point values and the combination of [77, 16] yielded the best results which also gave us 10 clusters as can be seen in figure 2.3 from section 2. ***In all the DBSCAN results, we have noticed several data points within cluster 0 which are regarded as noise points.***

After running our data through k-means and DBSCAN algorithms, k-means is a better fit for our data, as we observed that there are too many noise data points in DBSCAN, which means they are not properly classified. This is also the main disadvantage of DBSCAN algorithm where it can’t cluster data with large density. However, K-means tends to assume that our clusters will be simple, due to its partitional approach: it tries to decompose the dataset into non-overlapping subsets. k-means has a good effect on the convex cluster and is a good fit between the two for our data (corpus).

In order to further understand the context of each cluster, word clouds have been constructed using clusters from k-means algorithm with k=10.

|  |  |
| --- | --- |
| Cluster 1: Word Cloud  Based on the word cloud, we could infer that cluster 1 has documents related to neural networks and have references to ***MIT papers.*** |  |
| Cluster 2: Word Cloud  Based on the word cloud, we could infer that cluster 2 has documents related to neural networks and have references to ***IEEE papers.*** |  |
| Cluster 3: Word Cloud  Based on the word cloud, we could infer that cluster 3 has documents related to neural networks and ***data modeling*** and have references to ***MIT papers.*** |  |
| Cluster 4: Word Cloud  Based on the word cloud, we could infer that cluster 4 has documents related to neural networks and have references to ***MIT papers.*** |  |
| Cluster 5: Word Cloud  Based on the word cloud, we could infer that cluster 5 has documents related to neural networks and have references to ***nature*** journals and ***MIT*** papers. Also, looks like these documents refer to many articles by Vincent st. Amour. His significant work is in image processing. |  |
| Cluster 6: Word Cloud  Based on the word cloud, we could infer that cluster 6 has documents related to neural networks and have references to IEEE papers. |  |
| Cluster 7: Word Cloud  Based on the word cloud, we could infer that cluster 7 has documents related to neural networks and have references to MIT papers. |  |
| Cluster 8: Word Cloud  Based on the word cloud, we could infer that cluster 8 has documents related to neural networks and have references to IEEE papers and MIT papers. |  |
| Cluster 9: Word Cloud  Based on the word cloud, we could infer that cluster 9 has documents related to neural networks. |  |
| Cluster 10: Word Cloud  Based on the word cloud, we could infer that cluster 8 has documents related to neural networks and have references to MIT papers. |  |

4 Conclusion

Our corpus of 1500 documents has been analyzed and the associated tf-idf scores have been calculated. On which, we further performed clustering with k-means and DBSCAN algorithms.

K-Means is useful when you have an intuitive idea of how many clusters actually exists in your space. Its main benefit is its speed. There is a relationship between attributes and the number of observations in your dataset.

We observed that k-means clustering with k=10 gave us best results with proper segregation and grouping of documents while DBSCAN still showed several noise points even when tested across range of eps and min point combinations.

The quality of DBSCAN depends on the distance measure used in the function. Most commonly used is Euclidean. Especially for high-dimensional data, this metric can be rendered almost useless due to the so-called "Curse of dimensionality", making it difficult to find an appropriate value. DBSCAN cannot cluster data sets well with large differences in densities, since the minPts-\var epsilon combination cannot then be chosen appropriately for all clusters.

**References:**

* Prof Slater’s Jupyter Notebook sample and class material