ISSN No. 0976-5697

Volume 4, No. 9, July-August 2013

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

# A Critical Review of K Means Text Clustering Algorithms

Francis Musembi Kwale (Lecturer),
Department of Mathematics & Computer Science,
University of Eldoret,
P.O. Box 1125-30100,
ELDORET,
KENYA.
Cell Phone: 0254-0722471697
Email: fmkwale@yahoo.com, fmkwale@gmail.com

**Abstract:** Text clustering is a text mining technique used to group text documents into groups (or clusters) based on similarity of content. This organization (i.e. clustering) is so as to make documents more understandable and easier to search the relevant information, easier to process, and even more efficient in utilizing communication bandwidth and storage space. An example is clustering results of a web search engine operation into groups of similar documents. Many text clustering algorithms have been developed using different approaches, but none can be said to be the best. The choice of a particular algorithm is a big issue to text clustering system developers. K Means is arguably the most popular text clustering algorithm. However, just like the others, it must be having its own weaknesses. In this paper, we explore the K Means algorithm as well as its variants and discuss their appropriateness in text clustering. We describe the characteristics of the algorithms accompanied by some examples and illustrations in an attempt to discover the strengths and weaknesses. The paper thus gives an in depth view of the K Means algorithms, discusses the appropriateness of the algorithms, and also gives guidance to researchers of text mining concerning the choice of K Means for text clustering.

**Keywords**: text mining, text clustering, clusters, and K Means.

## 1. INTRODUCTION

### *1.1: Text Clustering*

**Data mining** (DM) can be defined as extraction of useful information from large structured data sets. A structured data set is one with well defined arrangement, e.g. a database table with well defined fields, sizes of fields, etc. By observing large data sets over a period of time, we can deduce previously-unknown and useful information concerning patterns, models, trends, and rules in the area of application. For example, a careful analysis of a retail database can deduce that item **x** goes with item **y**. However, since the most natural form of storing data is in form unstructured data (e.g. text documents, web documents - which have no well defined arrangements), we must apply text mining to extract useful information. 'Traditional data mining assumes that the information to be "mined" is already in the form of a relational database' [26].

**Text mining** (TM) refers to the process of extracting useful and non-trivial patterns or knowledge from unstructured text. TM can be applied to detect patterns, models, trends, or rules from unstructured data. It is more complex task than data mining since it deals with text data that are inherently unstructured, ambiguous and fuzzy.

**Text clustering** is a DM or TM technique used to group data sets with similar content. 'Normally, documents within a cluster are more similar to each other than documents lying in other clusters' [33]. For example, electronic text messages that discuss a related topic will form a cluster.

### *1.2: Text Clustering Algorithms*

Many text clustering techniques (i.e. algorithms) exist, and can be classified into various types including distance-based algorithms, frequent sequence algorithms, feature selection and extraction algorithms, density-based algorithms, probability-based algorithms, grid-based algorithms, ontology-based algorithms, and neural networks algorithms.

It's important to note that none of the algorithms is fully sufficient to cluster text documents, and none is dominant over the others. Each has its own strengths and limitations based on the following criteria. The criteria are also the desirable characteristics of a text clustering algorithm.

- **Document representation model**: It's important to represent the unstructured text documents using an appropriate structured representation.
- **Defining the similarity measure**: The effectiveness of a clustering algorithm depends on the definition of "similarity", as documents' similarity is hard to define.
- **Dimension reduction:** Because of the usual high dimension of textual data, it's important to reduce this size (e.g. by removing words that are irrelevant to the topic) to improve efficiency of operations. This is however not easy to achieve.
- **Clusters labels**: Obtaining appropriate label/topic name for each cluster is appropriate but also difficult. Note that text clustering is unsupervised and so cluster labeling is not user's activity.
- **Number of clusters**: Deciding the number of clusters is also important. It is difficult to specify a reasonable number of clusters for a data set when you have little information about it.
- **Overlapping of clusters**: The algorithms should allow for overlapping of document clusters. Some documents may each concern several topics.
- **Scalability**: An algorithm should be scalable so as to cater for huge data sizes.
- **Flexibility**: An algorithm should be flexible. It should be able to deal with different types of attributes, clusters with arbitrary shapes, and noise.

## 2. THE TEXT CLUSTERING APPROACH

K Means is a distance-based algorithm and so the clustering approach discussed here is distance-based.

Applying DM techniques (e.g. data clustering) is simple and straight forward since the data is structured. But when dealing with unstructured text documents, we can't apply the traditional DM clustering straight. But if we could get a way of converting the unstructured text documents into a structured form, we could then simply apply the traditional DM clustering on the resulting structure. This is the usual approach in text clustering, and it contains three key tasks, i.e.

(i) ***Document representation***: Convert the text documents into a structured form so that we can apply the traditional DM clustering.

(ii) ***Definition of similarity measure***: Define how a document (using the structured form) is similar to another (so that similar documents go to the same clusters).

(iii) ***Clustering logic*****:** Define the logic (i.e. algorithm) of determining exactly how the documents are assigned to their clusters based on the above similarity measure, and using the structured representations of the documents.

### *2.1: Document Representation*

The typical model used by distance-based algorithms is the vector space model (VSM), and the simplest implementation of the VSM is the **Boolean model**, whereby a document is regarded simply as a "bag of words" (i.e. a set of words). 'In mathematics, a bag, also called a multiset, is a set with duplicates allowed' [30]. Here, a collection of **n** documents containing **m** terms (or words) is represented using a matrix of **m** rows and **n** columns, whereby the rows represent the terms (or words) and the columns represent the documents. In other words, the rows are term vectors while the columns are document vectors. The **ij**$^{th}$ entry in the matrix is either a 1 (if the **i**$^{th}$ term is present in the **j**$^{th}$ document), or a 0 (if not). Thus, this 'term-document' matrix is said to be a "bag of words" since it contains repetition of values 1 and 0. In space-based view, a document will be a data point in a high dimensional space, whereby each term is an axis of the space.

***Example***

Assume the following three documents with underlined identifiable key terms. Obviously, the underlined qualify to be the key terms to form the basis of identifying what a document talks about, and thus do clustering.

D1: Eating fruit improves health.
D2: Give your infant fruit regularly, for the infant to have good health.
D3: Regular exercise improves your health.

We construct a term-dictionary as T1: fruit, T2: health, T3: infant, T4: exercise. We can then form a term-document matrix as

$$A = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Here, the first row is the vector (1, 1, 0) representing the first term (fruit), showing that the term occurs in the first and the second document, but not in the third. Similarly, the vector (1, 1, 0, 0) represents the first document (that contains the first and the second terms, but not the third and the fourth terms). And entry $\mathbf{A_{42}}$ is 0, showing that the fourth term (exercise) is not present in the second document.

***The space-based view***

In space-based view, the documents will be represented using four dimensions (because of four terms), and the three documents will be the points (1, 1, 0, 0), (1, 1, 1, 0), and (0, 1, 0, 1) on the space.

***Frequency-based VSM***

The Boolean model is simple and straight forward since it immediately matches the computer-based Boolean algebra. However, the Boolean model is limited in that the relevance of a term in a document is a binary decision (i.e. either term occurs or not). It doesn't cater for the level of importance of the term in a document, e.g. more frequent terms in a document may be more important.

Thus, we usually modify the model to use word frequencies. In this case, the **ij**$^{th}$ entry in the term-document matrix represents the frequency of the **i**$^{th}$ term in the **j**$^{th}$ document. This provides more information about terms. For example, using the immediate above example, the term-document matrix using frequencies is

$$A = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

The difference is that the third term (infant) occurs twice in the second document.

### *2.2: Definition of Similarity Measure*

We have seen from above that in VSM's space-based view, a document will be a data point in a high dimensional space, whereby each term is an axis of the space. Consequently in a distance-based approach, the distance between two points in the space represents the measure of (dis)similarity between the two documents. This means the length of the straight line between the two points, i.e. the **Euclidean measure.**

Alternatively, we can use the **cosine measure**, whereby the similarity between documents **x** and **y** can be expressed as the cosine of the angle between the two document vectors. Thus, the distance measure is

$$\mathbf{cos(x,y)=(x.y) / (||x|| ||y||)}$$

whereby **(x.y)** is the dot product of the two vectors **x** and **y**, and **||x||** is the length of vector **x**.

## 3. CLUSTERING USING K MEANS

### *3.1: The Traditional K Means Algorithm*

The K-Means algorithm is among the few most popular clustering algorithms, and was developed by J. MacQueen in 1967. It's a distance-based algorithm. It's a flat-type (or partitioning) clustering algorithm, meaning that the produced clusters are one-level (i.e. un-hierarchical). The above approach (in section 2) is implemented by K Means as;

- ***Document representation***: K Means converts the text documents into a VSM (structured) form.
- ***Definition of similarity measure***: It measures similarity between two text documents as the Euclidean measure or the cosine measure of their two points in the VSM.

- ***The clustering logic (or algorithm)***: Is as follows.
  1. Choose the number of clusters, **k**.
  2. Randomly generate **k** clusters and determine the cluster centers (centroids), where a cluster's centroid is the mean of all points in the cluster.
  3. Repeat the following until no object moves (i.e. no object changes its cluster)
     (i) Determine the Euclidean distance of each object to all centroids.
     (ii) Assign each point to the nearest centroid.
     (iii) Re-compute the new cluster centroids.

Thus, according to [29], the K Means algorithm assigns each point to a cluster whose center (also called centroid) is nearest. The centroid of a cluster is the average of all the points in the cluster based on the Euclidian distance measure.

Thus, in each loop of step 3 above, the algorithm aims at minimizing the following function for **k** clusters and **n** data points.

$$J=\sum_{j=1}^{k}\sum_{i=1}^{n}\|x_i-c_j\|^2 \quad (1)$$

where $\|x_i-c_j\|$ is a choosen distance measure (e.g. Euclidean measure) between data point $x_i$ from cluster $c_j$.

***(i) Example***

We could apply the K Means algorithm on the VSM produced in section 2.1 above, but we won't be able to illustrate the clustering graphically since there are four terms in the matrix and hence four dimensions in the space-based view. Therefore for simplicity, let's use another example that contains only two terms, so that we illustrate the clustering graphically on the **xy** plane.

Assume four documents containing two terms, whereby the first term occurs with frequencies 1, 0, 4, and 6 respectively in the documents, while the second term occurs with frequencies 2, 2, 1, 0 respectively. We represent the documents in VSM using the term-document matrix

$$A=\begin{pmatrix}1 & 0 & 4 & 6\\ 2 & 2 & 1 & 0\end{pmatrix}$$

We choose initial number of clusters **k**=2, and the first two points (1, 2), (0, 2) as the initial first and second centroids.

***(a) First loop***

We compute the distance matrix (containing distance of each point from each centroid) to be

$$D^1=\begin{pmatrix}0 & 1 & 3.16 & 5.39\\ 1 & 0 & 4.12 & 6.33\end{pmatrix}$$

The first row of **D** shows the distance of each point from the first centroid, and the second row shows the distance of each point from the second centroid.

Here, the point (1, 2) has distance $((1-1)^2+(2-2)^2)^{1/2}=0$) from centroid (1, 2), and distance $((1-0)^2+(2-2)^2)^{1/2}=1$) from centroid (0, 2).

The point (0, 2) has distance $((0-1)^2+(2-2)^2)^{1/2}=1$) from centroid (1, 2), and distance $((0-0)^2+(2-2)^2)^{1/2}=0$) from centroid (0, 2).

The point (4, 1) has distance $((4-1)^2+(1-2)^2)^{1/2}=3.16$) from centroid (1, 2), and distance $((4-0)^2+(1-2)^2)^{1/2}=4.12$) from centroid (0, 2).

The point (6, 0) has distance $((6-1)^2+(0-2)^2)^{1/2}=5.39$) from centroid (1, 2), and distance $((6-0)^2+(0-2)^2)^{1/2}=6.33$) from centroid (0, 2).

We then form the clusters by assigning each point to its nearest centroid. We form the group matrix **G** by assigning each point value 1 (if it should belong to that cluster), and value 0 if not. Note that first row represents the first cluster, and second row the second cluster. E.g. the third point (4, 1) has distance 3.16 from the first centroid, and distance 4.12 from the second centroid, meaning it's nearer to the first centroid. So we set the third column of **G** below to (1, 0).

Thus,

$$G^1=\begin{pmatrix}1 & 0 & 1 & 1\\ 0 & 1 & 0 & 0\end{pmatrix}$$

This shows that first, third and fourth points belong to the first cluster, while the second point to the second cluster.

We then recompute the centroid of each cluster as the average of the points in that cluster. Thus, first centroid is ((1+4+6)/3, (2+1+0)/3) which is (3.67, 1), while the second centroid is (0, 2).

***(b) Second loop***

We start the second loop of the algorithm and compute **D** to be

$$D^2=\begin{pmatrix}2.85 & 3.80 & 0.33 & 2.54\\ 1 & 0 & 4.12 & 6.33\end{pmatrix}$$

Here, the point (1, 2) has distance $((1-3.67)^2+(2-1)^2)^{1/2}=2.85$) from centroid (3.67, 1), and distance $((1-0)^2+(2-2)^2)^{1/2}=1$) from centroid (0, 2).

The point (0, 2) has distance $((0-3.67)^2+(2-1)^2)^{1/2}=3.80$) from centroid (3.67, 1), and distance $((0-0)^2+(2-2)^2)^{1/2}=0$) from centroid (0, 2).

The point (4, 1) has distance $((4-3.67)^2+(1-1)^2)^{1/2}=0.33$) from centroid (3.67, 1), and distance $((4-0)^2+(1-2)^2)^{1/2}=4.12$) from centroid (0, 2).

The point (6, 0) has distance $((6-3.67)^2+(0-1)^2)^{1/2}=2.54$) from centroid (3.67, 1), and distance $((6-0)^2+(0-2)^2)^{1/2}=6.33$) from centroid (0, 2).

Thus, the first point changes into the second cluster since it's now distance 2.85 from the first centroid (3.67, 1) compared to distance 1 from the second centroid (0, 2). We therefore compute the new group matrix to be

$$G^2=\begin{pmatrix}0 & 0 & 1 & 1\\ 1 & 1 & 0 & 0\end{pmatrix}$$

We then recompute the centroid of each cluster as the average of the points in that cluster. Thus, first centroid is ((4+6)/2, (1+0)/2) which is (5, 0.5), while the second centroid is ((1+0)/2, (2+2)/2) which is (0.5, 2).

***(c) Third loop***

We start the third loop of the algorithm and compute **D** to be

$$D^3=\begin{pmatrix}4.27 & 5.22 & 1.11 & 1.11\\ 0.5 & 0.5 & 3.64 & 5.85\end{pmatrix}$$

Here, the point (1, 2) has distance $((1-5)^2+(2-0.5)^2)^{1/2}=4.27$) from centroid (5, 0.5), and distance $((1-0.5)^2+(2-2)^2)^{1/2}=0.5$) from centroid (0.5, 2).

The point (0, 2) has distance $((0-5)^2+(2-0.5)^2)^{1/2}=5.22$) from centroid (5, 0.5), and distance $((0-0.5)^2+(2-2)^2)^{1/2}=0.5$) from centroid (0.5, 2).

The point (4, 1) has distance $((4-5)^2+(1-0.5)^2)^{1/2}=1.11$) from centroid (5, 0.5), and distance $((4-0.5)^2+(1-2)^2)^{1/2}=3.64$) from centroid (0.5, 2).

The point (6, 0) has distance $((6-5)^2+(0-0.5)^2)^{1/2}=1.11$) from centroid (5, 0.5), and distance $((6-0.5)^2+(0-2)^2)^{1/2}=5.85$) from centroid (0.5, 2). Thus,

$$G^3 = \begin{pmatrix} 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{pmatrix}$$

And so there is no change of the clusters' grouping, and so we stop.

***(d) Conclusion***

The first two points (thus documents) i.e. (1, 2), (0, 2) are in the second cluster while the last two points (documents) i.e. (4, 1), and (6, 0) are in the first cluster.

***(e) Illustration of the clustering using space-based view***

Our original data was (1, 2), (0, 2), (4, 1), and (6, 0), i.e. with term-document matrix

$$A = \begin{pmatrix} 1 & 0 & 4 & 6 \\ 2 & 2 & 1 & 0 \end{pmatrix}$$

Since there are two terms, we have a two dimensional space whereby the **x** axis represents the first term while the **y** axis represents the second term. Each document is a point on the **xy** space. Note that;

- Document points are shown using ☆
- Centroids are shown using ○ or ☆ (if they are also data points)
- Points inside a cluster are enclosed using

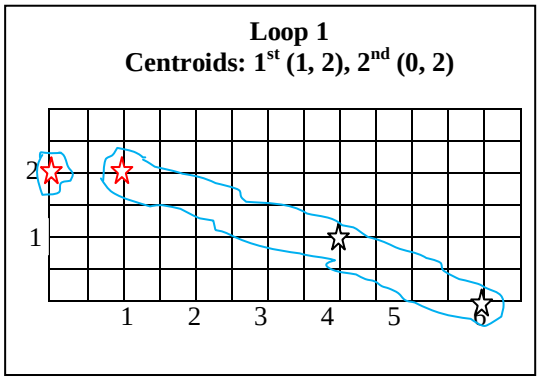



Figure. 1 First loop of the clustering

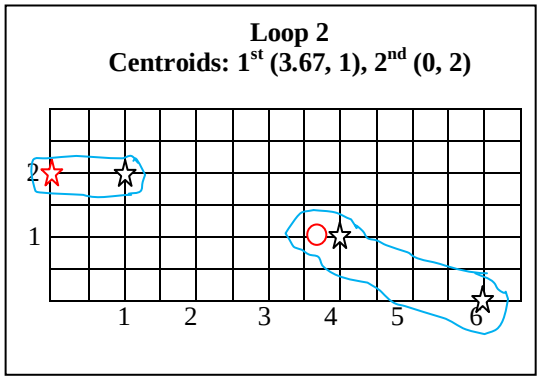



Figure. 2 Second loop of the clustering

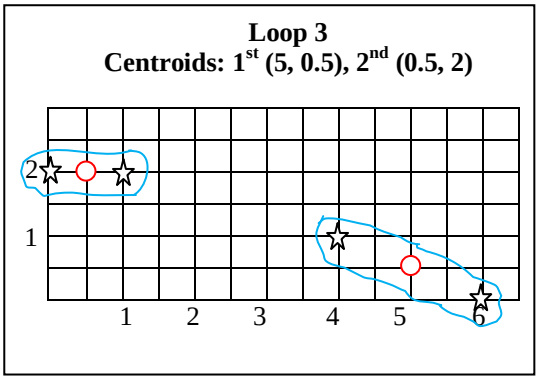


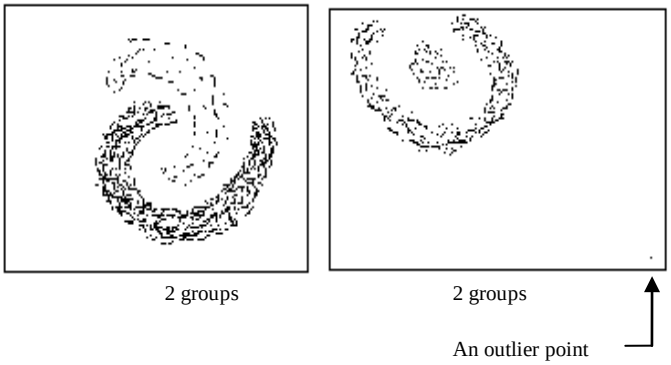

Figure. 3 Third loop of the clustering

***(ii) Strengths and Limitations of the K Means Algorithm***

The above example helps us see clearly how the K Means algorithm works, so that we can easily observe its strengths and limitations. The advantage of the K Means algorithm is that it's simple to understand and implement, just as the above example uses straight forward Euclidean calculations and elementary data storage using matrices. Secondly, the K-means algorithm is efficient in memory requirements since the system only needs to store the data points and their distances from centroids (using matrices A and D), the membership of data points to the clusters (using matrix G), and some variables to hold centroids. According to [10], K Means is fast for small document sizes. According to [32], the time complexity of the K Means algorithm is O(knI), where **k** is the number of clusters, **n** the number of objects and **I** the number of iterations (which is dependent on the stopping criterion). Thus, the algorithm is very efficient. It's also scalable.

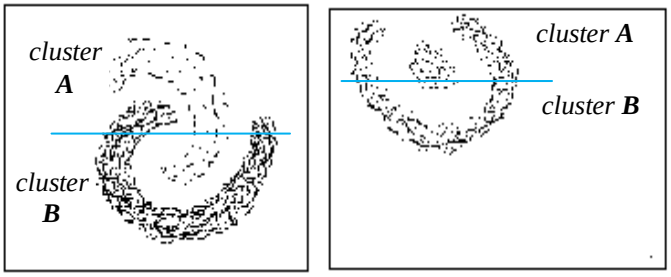
Its limitation is that the user must specify the initial number of clusters before clustering. It is not trivial for the user to determine a reasonable number of clusters depending on the number of documents. For instance in the previous example, we were forced to choose randomly, **k**=2. 'A major problem with partitioning algorithms is selecting an appropriate number of output clusters' [19]. Secondly, it doesn't include dimension reduction. Our example had only four documents and two terms. But practical applications can involve hundreds of web documents, with tens of thousands of terms, meaning huge matrices. Thirdly, according to [22], there is no description about the cluster's contents (i.e. labels of clusters), so the contents can't be utilized more efficiently. Following the above examples (in sections 2 and 3), it's hard to determine clear-cut topics of the obtained clusters. Fourth, according to [35], the K Means algorithm is too sensitive to outliers (i.e. unusual data values e.g. too big values as a result of errors). The outliers will substantially distort the distribution of data (i.e. affects the mean a lot). Also, when the data points are few, it's more likely to get different clustering for different initial centroids. Lastly, according to [17], K Means algorithm may not give good results when features are more, and generally, the results vary quite a bit from one run to another. And generally, K Means doesn't perform well when clusters are of different sizes, densities, and irregular shapes as illustrated below.

Illustration. 1 Groups of different shapes, densities (hard to detect in K Means) and outlier




The two obvious groups of points in each pair are of different shapes and densities. Thus, some points in a group could be nearer (in distance) to the center (or centroid) of the other group (or cluster) rather than theirs, based on the K Means algorithm. Thus, it's very hard for the K Means algorithm to detect the clusters since it's purely based on distance measurements. And so, wrong clustering could happen as illustrated below.

Illustration. 2 Wrong clustering by K Means




Also, some outlier points as the one shown at the bottom right corner of the second pair of clusters makes calculations of the mean distances of points from their centroids distorted.

In an attempt to improve the K Means algorithm, other similar algorithms have been developed. All these can be considered as the K Means family. They include the following.

### *3.2. The K Medians Algorithm*

The K Medians algorithm works just like the K Means algorithm, except that we compute the median instead of the mean of each cluster as the centroid.

As a result, there is less effect of extreme values (i.e. outliers). But, the algorithm suffers from the other limitations of the K Means algorithm.

### *3.3. The Bisecting K Means Algorithm*

Whereas the K Means algorithm splits a cluster into **k** sub clusters, the Bisecting K Means algorithm splits a cluster into two sub clusters in a divisive hierarchical manner, but using the K Means-type of clustering. In other words, Bisecting K Means works just like K Means, except that **k**=2, and it clusters in a divisive-hierarchical manner. The documents are initially partitioned into two clusters. The algorithm iteratively selects and bisects each one of the leaf clusters until **k** clusters are reached. Thus, the algorithm is basically as follows.

1. Receive the documents set as a cluster.
2. Select a cluster to split.
3. Perform K Means algorithm on the selected cluster with k=2.
4. Go back to step 2 and continue, stop when there are **k** clusters.

According to [22], this algorithm is more accurate and efficient than the traditional K Means algorithm. Also, according to [32], the time complexity of the Bisecting K Means algorithm is O(nI log k), which is less than the K Means algorithm since it does not compare all objects to all cluster centroids. 'If the number of clusters is large, then bisecting K-means is more efficient than the regular K-means algorithm. Hence, Bisecting K-means gives better results for larger data sets' [17].

However, it still suffers the limitations of the K Means algorithm, i.e. specifying the initial number of clusters, lack of dimension reduction, and lack of description about the cluster's contents.

### *3.4. The K Medoids Algorithm*

This is an extension of the K-means algorithm, and is still a partioning-based algorithm. As opposed to the K Means algorithm whose centers (or centroids) may not be data points (centroids are obtained as the mean value of all data points in a cluster, thus centroids may not be data points), the K Medoids algorithm chooses data points as centers (medoids). I.e., each cluster is represented by one of the objects in the cluster (i.e. the representative object). A medoid is a data point that is the most centrally located in a cluster, meaning that its average dissimilarity to all other objects in the cluster is the minimum (compared to the average of the other points). This makes the K Medoids algorithm be less sensitive to outliers than the K Means algorithm. [13] explains that the algorithm works as

> We first set a random representative object for each clustering to form **k** clustering of **n** data. Then according to the principle of minimum distance, other data will be distributed to corresponding clustering according to the distance from the representative objects.

According to [35], instead of taking the mean value of the objects in a cluster as a reference point, we take the most centrally located object in a cluster as the medoid.

The K Medoids algorithm is more accurate than the K Means algorithm, because of the deeper computations of the representative objects. According to [9], the K Medoids algorithm is more robust than K Means algorithm. According to [41], the K Medoids algorithm uses representative objects as reference points instead of using the mean value of the objects in each cluster, and this makes the algorithm less sensitive to outliers. According to [44], the K Medoids algorithm has a higher accuracy of pattern matching than the K Means algorithm.

However, this algorithm is less efficient than the K Means algorithm (the step of computing medoids – specifically their dissimilarity to all other objects is harder than computing means), and also retains other limitations of K Means (except the problem of outliers). According to [9], although K Medoids algorithm is more robust than K Means algorithm, it is less efficient than K Means. According to [44], the limitations of the K Means algorithm are retained by K Medoids (except the problem of outliers), and in addition, the adoption of new particle computing rules by K Medoid's increases the computation time to $(k(n-k)^2)$. From the experiment done by [41], it was observed that the K-Means

algorithm is more efficient for smaller data sets than the K-Medoids algorithm. From the experiment done by [40], it was concluded that the efficiency of the K Means algorithm is better than the one of the K Medoids algorithm.

### *(i) The Partitioning Around Medoids (PAM) Algorithm*

This is a 'realization' of the K Medoids algorithm. It was developed by Kaufmann and Rousseeuw in 1987. 'The process operates by swapping one of the medoids with one of the objects iteratively such that the total distance between non-selected objects and their medoid is reduced' [8]. The computational complexity of the algorithm according to [8] is $O((1 + \beta)k(T- k)^2)$ - based on the number of partitions per object, or $O((1+\beta)k^2(T -k)^2)$ - based on the number of distance calculations, (one partition per object is equivalent to **k** distances calculations). Here, **k**, **β** and **T** are number of medoids, the number of successful swaps for all samples tested and the total number of objects, respectively.

It can thus, be summarized as follows.

1. Select by random **k** medoids from the given **n** data points.
2. Assign each data point to the closest medoid using a measurement metrix, e.g. the Euclidean measurement.
3. For each medoid **m**
   For each non-medoid data point **c**
   Swap **m** and **c** and compute the total cost of the configuration.
   Select the configuration with the lowest cost (i.e. lowest dissimilarity).
4. Repeat steps 2 to 3 until there is no change in the medoids.

***Example***

Consider repeating the above clustering example using PAM instead of K Means. Remember the original data was (1, 2), (0, 2), (4, 1), and (6, 0), i.e. with term-document matrix

$$A = \begin{pmatrix} 1 & 0 & 4 & 6 \\ 2 & 2 & 1 & 0 \end{pmatrix}$$

The K Means method produced the centroids (1, 2), (0, 2) originally, (3.67, 1), (0, 2) in the first loop, and (5, 0.5), (0.5, 2) in the second loop. And centroids (3.67, 1) and (5, 0.5), (0.5, 2) are not data points.

However using PAM, each centroid (medoid) in each loop must be a data point, i.e. the only possible medoids are (1, 2) or (0, 2) or (4, 1) or (6, 0). The clustering of these four data points will be as follows.

1. Assume **k**=2, and let initial medoids be (1, 2), (0, 2).
2. Assign the points (4, 1), (6, 0) to their nearest medoid.
3. For medoid (1, 2): Swap (1, 2) with (4, 1), then with (6, 0), and get the best configuration.
   For medoid (0, 2): Swap (0, 2) with (4, 1), then with (6, 0), and get the best configuration.
4. Assign the resulting non-medoid data points to their medoids (from the above best configurations). E.g. assuming the best configuration results to medoids (4, 1) and (0, 2), we assign points (1, 2), (6, 0) to either of the two medoids.
5. Iterate steps (iii), (iv) using the new medoids (resulting every time) until there is no change in medoids.

Though PAM improves the K Medoids algorithm, it's clearly inefficient from the above-mentioned complexity (out of the nested loops in steps 3, 4 above).

### *(ii) CLARA (Clustering LARge Applications)*

This is yet another realization of the K Medoids algorithm, and it's a modification of PAM.

It was developed by Kaufmann and Rousseeuw in 1990. 'CLARA reduces the computational complexity by drawing multiple samples of the objects and applying the PAM algorithm on each sample' [8]. According to [4], instead of taking the whole set of data into consideration, a small portion of the actual data is taken to represent the whole data, and medoids are then taken from this sample. The assumption here is that if the sample is representative (or taken in fairly random manner), then the medoids of the sample should approximate the medoids of the entire dataset (i.e. the sample should closely represent the original data). But to improve the approximation, multiple samples are taken. The clustering accuracy is measured by the average dissimilarity of the objects.

The steps according to [8] are:

Step 1: Call the PAM algorithm with a random sample, s objects from the original set of T objects.

Step 2: Partition the T objects based on the k medoids obtained from previous step. Update the better medoids based on the average distance of the partition.

The computational complexity of the CLARA algorithm according to [8], is $O(q(ks^2 + (T- k)) + \beta ks^2)$ based on the number of partitions per object or $O(q(k^2s^2 + k(T - k)) + \beta k^2s^2)$ based on the number of distance calculations, where **q**, **s**, **k**, **β** and **T** are the number of samples, object size per sample, number of medoids, the number of successful swaps for all samples tested and the total number of objects, respectively. 'Clearly, the CLARA algorithm can deal with a larger number objects than can PAM if s<= T' [8]. And generally, CLARA is more efficient than PAM in large number of objects. According to [27], CLARA is more efficient than PAM in large data sets.

Thus, the advantage of CLARA is that it improves efficiency. But the limitation is that its effectiveness depends with the sample size (the size should be relatively high for good effectiveness). But high sample sizes reduce the efficiency of the algorithm (i.e. tradeoff between effectiveness and efficiency). According to [4], CLARA can't find the best clustering if any sampled medoid is not among the best **k** medoids. Thus, in order to address this tradeoff issue, the CLARANS algorithm was developed.

### *(iii) CLARANS (Clustering Large Applications based on RANdomized Search)*

CLARANS is a modification of CLARA. It was developed by [27] in 1994. According to [4], CLARANS combines the sampling technique with PAM.

The CLARANS research considers PAM and CLARA clustering as a search graph, and tries to improve the clustering graphically (thus obtaining CLARANS). According to [27], a set of **k** objects in the graph (i.e. a set of medoids) is a node. All nodes in the graph will represent the set of all possible medoids (each data object is a possible medoid). And two nodes will be said to be neighbors if their sets differ by only one medoid. Consequently, each node will have **k(n-k)** neighbors, whereby **n** is the total number of objects. Each

node can be considered as a clustering since it contains **k** medoids, and so can be assigned a cost which is the total dissimilarity between every object and the medoid of its cluster.

To illustrate this, assume clustering the data 2, 6, 4, 8 into two clusters. Here, n=4, k=2. The set of all possible medoids (or the graph's nodes) is (2, 6), (2, 4), (2, 8), (6, 4), (6, 8), (4, 8). The neighbors of (2, 6) are: (2, 4), (2, 8), (6, 4), (6, 8). Neighbors of (2, 4) are: (2, 6), (2, 8), (6, 4), (4, 8). Each of these two nodes have 4 neighbors which using the above formula is **k(n-k)**=2(4-2)=4, and so should be the other four nodes above. The idea with this formula is that a node has **k** objects (drawn from the **n** objects), e.g. (2, 6) has 2 objects drawn from the 4 given objects (i.e. 2, 6, 4, 8). And since a neighbor of a node is the one with only one different object, the neighbors of a node can be gotten by combining each of the **k** objects in the node with each of the other objects not in this node, i.e. the **n-k** other objects (thus, neighbors of (2, 6) are gotten by combining each of 2 and 6 (**k** objects) with each of 4 and 8 (**n-k** objects), i.e (2, 4), (2, 8), (6, 4), (6, 8)).

Another illustration is clustering 2, 6, 4, 8, 9, 7 into 3 clusters. The nodes are (2, 6, 4), (2, 6, 8), (2, 6, 9), (2, 6, 7), (2, 4, 8), (2, 4, 9), (2, 4, 7), (2, 8, 9), (2, 8, 7), (2, 9, 7), (6, 4, 8), (6, 4, 9), (6, 4, 7), (6, 8, 9), (6, 8, 7), (6, 9, 7), (4, 8, 9), (4, 8, 7), (4, 9, 7), (8, 9, 7). The neighbors of (2, 6, 4) are (2, 6, 8), (2, 6, 9), (2, 6, 7), (2, 4, 8), (2, 4, 9), (2, 4, 7), (6, 4, 8), (6, 4, 9), and (6, 4, 7). I.e. node (2, 6, 4) has 9 neighbors, which can also be gotten as **k(n-k)**=3(6-3)=9 neighbors.

Here, PAM is viewed as a search for a minimum in this graph whereby, at each step, all neighbors of the current node are examined. The neighbor which corresponds to the deepest descent in cost is chosen as the next solution. CLARA's clustering is thus, viewed as a modification of this so that we don't search all neighbors of a node, but a sample of them. CLARANS further modifies this, such that instead of drawing a sample of nodes at the beginning of a search (as in CLARA), we draw a sample of neighbors in each step of a search, i.e. CLARANS draws a sample of neighbors dynamically. The benefit is that we do not confine a search to a localized area. As a result CLARANS produces higher quality clustering than CLARA, and is also more efficient.

### *3.5. The Kernel K Means Algorithm*

The Kernel K Means algorithm is an extension of the K Means algorithm meant to address the limitation of the K Means algorithm of not being able to cluster irregular shapes (or non-linear data). The idea behind the algorithm is to map the original data to a feature space by a non-linear transformation, then apply the K Means algorithm on the resulting space. According to [39],

> this means that nonlinearly separated clusters in input space can be obtained, overcoming the limitation of the K Means of finding only linearly separable clusters. This results in linear separators in feature space which correspond to nonlinear separators in input space.

The algorithm applies the same idea as the K Means algorithm, but instead of using the Euclidean distance measurements as in K Means, it uses the kernel method.

The strength of the Kernel K Means algorithm is that it's able to identify the non-linear structures. Consequently, it's suitable in clustering real life data.

A limitation of the algorithm is that it's less efficient than the K Means algorithm.

## 4. CONCLUSIONS

The K Means algorithm and its variants are simple and efficient to use, despite some weaknesses. It is recommended that an approach combining the K Means (or its variant) with another approach of text clustering be considered so as to take the advantages of both.

## 5. REFERENCES


[1] Alelyani, S, Tang, J, Liu, H, "Feature selection for clustering: A review", Online notes, unpublished.

[2] Andrews, N & Fox, E, "Recent developments in document clustering", Technical Report, Department of Computer Science, Virginia Tech, viewed 31 January 2013, <http://eprints.cs.vt.edu/archive/00001000/01/docclust.pdf> unpublished.

[3] Bharathi, G &Venkatesan, D, "Study of ontology or thesaurus based document clustering and information retrieval", Journal of Theoretical and Applied Information Technology, vol. 40, no. 1, 2012

[4] Boomija, M, "Comparison of partition based clustering algorithms", Journal of Computer Applications, vol. 1, no. 4, 2008.

[5] Chen, C, Tseng, F &Liang, T, "Mining fuzzy frequent item sets for hierarchical document clustering", Information Processing and Management, vol. 46, no. 2, pp. 193–211, 2010

[6] Chifu, E, "Self organizing maps in web mining and semantic web", PhD Thesis, Technical University of Cluj-Napoca, 2010.

[7] Chitsaz, E, Taheri, M, Katebi S et al., "An improved fuzzy feature clustering and selection based on chi-squared test", Proceedings of the International MultiConference of Engineers and Computer Scientists 2009, vol. I, IMECS 2009, March 18 - 20, 2009, Hong Kong, viewed 14 July 2013, <http://www.iaeng.org/publication/IMECS2009/IMECS2009_pp35-40.pdf>

[8] Chu, S, Roddick, J, Pan, J, "Improved search strategies and extensions to K-medoids-based algorithms", Technical Report KDM-02-005, School of Informatics and Engineering Flinders University of South Australia, viewed 24 June 2013, <http://kdm.first.flinders.edu.au/KDMTR/KDM02005.pdf> unpublished.

[9] Fung, B, "Hierarchical document clustering using frequent item sets", MSc Thesis, Simon Fraser University, 1999.

[10] Geraci, F, "Fast clustering for web information retrieval", PhD Thesis, Universit' A Degli Studi Di Siena, 2008.

[11] Gruber, T, "Toward principles for the design of ontologies used for knowledge sharing", International Journal Human-Computer Studies, vol. 43, nos. 5-6, pp. 907-928, 1995.

[12] Guduru, N, "Text mining with support vector machines and non-negative matrix factorization algorithms", MSc Thesis, University of Rhode Island, 2006.

[13] Hao, Z, "A new text clustering method based on KGA", Journal of Software, vol. 7, no. 5, pp. 1-5, 2012.
[14] Hotho, A, Maedche, A & Staab, S, "Ontology-based text document clustering", Proceedings of the Workshop "Text Learning: Beyond Supervision" at IJCAI 2001 Seattle WA USA, August 6, 2001, viewed 05 February 2013, <http://www.aifb.uni-karlsruhe.de/WBS> unpublished.
[15] Jayabharathy, J, Kanmani, S & Parveen A, "A survey of document clustering algorithms with topic discovery", Journal of Computing, vol. 3, no. 2, pp. 1-3, 2011.
[16] Khan, L, "Ontology-based information selection", PhD Thesis, University of Southern California, 2000.
[17] Krishna B, et al, "Comparative study of K-means and Bisecting K-means techniques in Wordnet-based document clustering", International Journal of Engineering and Advanced Technology, vol 3, no 2, pp 1-4, 2012.
[18] Langville, A & Meyer, C, "Text mining using the nonnegative matrix factorization", SIAM-SEAS–Charleston, 2005, unpublished.
[19] Lasek, P, "Efficient density-based clustering", PhD Thesis, Warsaw University of Technology, 2011.
[20] Lee, S, Song, J & Kim, Y, "An empirical comparison of four text mining methods", Journal of Computer Information Systems, 2010, unpublished.
[21] Liu, T, Liu, S, Chen, Z & Ma, Z, "An evaluation on feature selection for text clustering", Paper presented at proceedings of the Twentieth International Conference on Machine Learning (ICML-2003), Washington DC, 2003.
[22] Li, Y, "High performance text document clustering", PhD Thesis, Wright State University, 2007.
[23] Li Y, et al., "Text clustering with feature selection by using statistical data", IEEE Transactions on Knowledge and Data Engineering, vol. XX, no. YY, 2008.
[24] Magatti, D, "Graphical models for text mining: knowledge extraction and performance estimation", PhD Thesis, UNIVERSITÀ DEGLI STUDI DI MILANO – BICOCCA, 2010.
[25] Moldovan, D, Novischi, A, "Word sense disambiguation of WordNet glosses", Elsevier Ltd, 2004, viewed 16 June, 2013, <www.hlt.utdallas.edu/~moldovan/newpapers/j04dman.pdf> unpublished.
[26] Mooney, R, Nahm, U, "Text mining with information extraction", paper presented at the Proceeding of the 4th International MIDP Colloquim, Bloemfontein, South Africa, pp.141-160, September 2003, viewed 29 January 2013, <http://www.cs.utexas.edu/users/ml/papers/discotex-melm-03.pdf>
[27] Ng, R &Han, J, "CLARANS: A method for clustering objects for spatial data mining", IEEE Transactions on Knowledge and Data Engineering, vol. 14, no. 5, 2002.
[28] Ning, W, "Text mining and organization in large corpus", MSc Thesis, Technical University of Denmark (DTU), 2005.
[29] Punitha, S & Punithavalli M, "A comparative study to find a suitable method for text document clustering", IJCSNS International Journal of Computer Science and Network Security, vol. 12, no. 10, 2012.
[30] Rehurek, R, "Scalability of semantic analysis in natural language processing", PhD Thesis, Masaryk University, 2011.
[31] Rai, P, "A survey of clustering techniques", International Journal of Computer Applications, vol. 7, no 12, 2010.
[32] Rosell, M, "Clustering exploration: Swedish text representation and clustering results unraveled", PhD Thesis, Stockholm, Sweden, 2009.
[33] Sharma, S & Gupta, V, "Recent development in text clustering techniques", International Journal of Computer Applications (0975 – 8887), vol. 37, no. 6, pp. 1-5, 2012.
[34] Sree K, Murthy J, "Clustering based on cosine similarity measure", International Journal of Engineering Science & Advanced Technology, vol 2, no 3, pp 1-2, 2012.
[35] Stefanowski, J, "Data mining clustering, online lecture notes", 2009, viewed 10 June 2013, <http://www.cs.put.poznan.pl/jstefanowski/sed/DM-7clusteringnew.pdf.> unpublished.
[36] Steinbach, M, Karypis, G & Kumar, V, "A comparison of document clustering techniques", Technical Report, Department of Computer Science and Engineering, University of Minnesota, 2000, viewed 30 July 2012, <http://www.cs.umn.edu/tech_reports_upload/tr2000/00-034.pdf> unpublished.
[37] Tar, H & Nyunt, T, "Ontology-based concept weighting for text documents", Paper presented at the 2011 International Conference on Information Communication and Management IPCSIT, IACSIT Press, Singapore, vol.16, 2011, viewed 05 February 2013, <http://www.ipcsit.com/vol16/31-ICICM2011M2004.pdf>
[38] Teknomo, K, "K- Mean clustering tutorial", Online notes, 2007, viewed 12 October 2012, <http:\\people.revoledu.com\kardi\tutorial\kMean\index.html> unpublished.
[39] Tzortzis, G & Likas, A, "The global Kernel K-means algorithm for clustering in feature space", IEEE Transactions on Neural Networks, vol. 20, no. 7, 2009.
[40] Velmurugan, T, "Efficiency of K-Means and K-Medoids algorithms for clustering arbitrary data points", International Journal of Computer Technology & Applications, vol 3, no. 5, 2012.
[41] Velmurugan, T & Santhanam, T, "Computational complexity between K-Means and K-Medoids clustering algorithms for normal and uniform distributions of data points", Journal of Computer Science, vol. 6, no. 3, pp. 1-6, 2010.
[42] Wanner, L, "Introduction to clustering techniques", Online notes, 2004, viewed 10 June, 2013, <http://www.iula.upf.edu/materials/040701wanner.pdf> unpublished.
[43] Weis, D, "Descriptive clustering as a method for exploring text collections", PhD Thesis, Pozna´n University of Technology, 2006.
[44] Wentian, J & Qingju, G, "Improved K-medoids clustering algorithm under semantic web", paper presented at the proceedings of the 2nd International Conference on Computer Science and Electronics Engineering (ICCSEE 2013), in press.