A Clustering Accuracy Comparison Framework

Kwale F. M.[1*], Wagacha P.W. [2] and Kahonge A. M. [2]
[1]Department of Mathematics & Computer Science, University of Eldoret, P.O. Box 1125-30100, Eldoret-Kenya
[2]School of Computing & Informatics, University of Nairobi, P.O. Box 30197-00100, Nairobi-Kenya

***Corresponding Author's Email Address:** fmkwale@yahoo.com

Abstract

Clustering is a data mining problem of dividing documents into groups, such that documents in one group are more similar than those in other groups. The aim of this study is to propose a framework for comparing the accuracy of clustering algorithms. The study applies qualitative research through document analysis to review previous clustering algorithms' comparisons so as to obtain the issues/problems with such previous comparisons. We then deduce appropriate comparisons framework that addresses the problems. The study obtained the following comparison issues: Nature of comparison, nature of data, size of data, source of data, evaluation metrics, and parameter settings. Consequently, the study proposed the rules, formulae, and procedures needed to be used in a comparison. It is recommended that applying this framework will ensure that such evaluations and comparisons are done using formal procedures that will yield dependable results. This study suggests a further study to be done to apply this framework and do a comprehensive comparison of some clustering algorithms.

Keywords: Clustering, algorithms, metrics, parameters.

INTRODUCTION

Clustering is an area currently receiving a lot of attention. Many researchers have done comparative study of various clustering algorithms in an attempt to recommend the well performing ones. Others have produced their own algorithms and compared them with other existing algorithms. However, review done in this research suggests much weakness in the comparisons. According to Chen (2005), most authors of newly-proposed algorithms claim success of their algorithms based on comparisons with older algorithms using a few hand-crafted examples, showing only where the old fails and the new one succeeds.

It is important to note that the process of comparing some products (including clustering algorithms) includes evaluating them. Therefore, we include both the aspects of evaluation and comparisons of clustering algorithms. Evaluating a product (e.g. an algorithm) can simply be understood to be the process of ascertaining how good (or well performing) the product is. Consequently, comparing two or more products is simply checking which one evaluates higher either based on a particular performance criterion or in overall.

METHODOLOGY

The study applies qualitative research through document analysis to review previous clustering algorithms' evaluations/comparisons and noting the methods and procedure followed. This is aimed at obtaining the issues/problems with the previous comparisons.

We then deduce appropriate comparisons framework that addresses the problems. The framework is evaluated by:

- Use of examples: Examples are designed to include the various possibilities of the results of a comparison of assumed algorithms, and consequently test the fairness of applying the various ideas, formulae and procedures of the framework.
- Use of observations: During the development of the framework, actual runs of chosen algorithms are carried out. This is specifically during the derivation of the recommended parameter values.
- Logical arguments from previous research on clustering evaluations. Research findings and theories from other researchers concerning requirements of good evaluations/comparisons are quoted and used to justify various aspects of the framework in the process of the framework's development.

RESULTS AND DISCUSSION

The Clustering Comparison Framework

Issues with Previous Clustering Comparisons

From review of the previous comparisons, it can be concluded that the previously conducted comparisons/evaluation studies were not done using an appropriate framework, thus causing lack of information on algorithms' performance, and consequently making it difficult to choose algorithms for implementing clustering solutions. The many clustering comparative studies reviewed haven't followed a particular comparison framework resulting in their being limited in scope/inclusivity (or failing to solve some issues of concern). For example, it is observed that an algorithm may be better than another one using data sets with particular sizes but poorer than other data sets, e.g. using data sets with only two classes will of course favor KMeans algorithm since it clusters data (by default) into two clusters (i.e. has default parameter value K=number of clusters=2). Thus, there has been no justification of the data sizes used. And this is related to parameter values used. For example, KMeans (with default parameter K=2) will be favored when the data points are arranged such that there are around two classes, and otherwise not disfavored. And in this case, using the default parameter value (K=2) in a comparison will definitely favor KMeans. Thus, there has been no careful study to determine the best set of parameter values or data sizes to use in a comparison based, among other possible issues that may affect comparisons results and that require a careful study to formally identify them. The summary of those issues is as follows.

***(i) Nature of evaluations/comparisons*:** The problem here is whether or not empirical comparison (which has been mainly used in previous comparisons) is sufficient. From theory, some aspects of any algorithm can be compared descriptively but not empirically. An example is the complexity of running time of an algorithm. Most researchers have compared algorithm only empirically. However, we note that descriptive comparison is also very important in distinguishing algorithms. Source?

***(ii) Nature of data*:** The nature of documents can vary based on various factors. These include shapes of clusters as well as the noise level (unusual values including erroneous data/missing values, outliers, or unknown values). It has been noted the performance of a clustering algorithm may vary by using data of different nature (or characteristics). An algorithm may perform the best in usual regular-shaped data but perform the least with irregular shaped data which is typical in many applications. According to Chen (2005), the choice of the datasets used determines the comparison results.

***(iii) Sizes of data*:** It has also been noted the performance of a clustering algorithm may vary by using data of different sizes. Sets of data documents may vary in their sizes (with respect to clustering) based on various criteria. We can identify these as the total number of text documents, the total number of attributes (or terms), and the number of groups/topics (or clusters).

***(iv) Sources of data*:** The possible sources of clustering data are generated (or synthetic) data sets, benchmark (or real world) data sets, and personally prepared data sets. Each source/type has its strength and limitation. The use of synthetic data sets permits us to explore the space as much as possible, thus test the different characteristics of data including different levels of shapes (irregularities of shapes), noise, and sizes. However unlike the other two, it is obviously not possible to be sure that it is realistic i.e. if it represents the true real-world data whose clustering algorithms are intended for. However, the latter are more detailed, tested and standard, and so are advisable to be used.

***(v) Evaluation metrics*:** According to Chen (2005), the choice of evaluation indices determines the comparison results. There should thus be a justification of the use of the metrics. The various accuracy metrics include the F measure, the entropy, purity, rand index, the within-cluster sum of square, etc.

***(vi) Parameter settings*:** Different algorithms have different parameters, e.g. *K* for K Means algorithm. It has been observed that using different parameter values for an algorithm will yield different evaluation/comparison results. For example from the concept of densities, using a too big value of parameter **Minpts** in DBSCAN will cause failure to detect existing clusters (thus lowering accuracy), while a too small value of the parameter will lead to too much detection of non-existing clusters (thus lowering efficiency and also accuracy). And it is also evident that it's hard to decide an optimal value for each parameter. According to Jiang (2003), it is difficult to stipulate the suitable parameters for complex data set. We therefore need to justify what parameter values will be used for each algorithm to ensure fairness in the comparison.

Clustering Comparison Domains

In the real sense, a clustering evaluation/comparison framework that accommodates the different clustering approaches is very difficult to propose. This is because different approaches have different ways of perceiving clusters of data sets, different similarity measurement methods, as well as different parameter requirements. Prelic (2006, p. 1) agrees with this by saying "Comparing clustering methods in general is difficult as the formalization in terms of an optimization problem strongly depends on the scenario under consideration and accordingly varies for different approaches". However, we aim at proposing common comparison settings that can be a compromise of the different approaches.

Many previous clustering algorithms' evaluations have involved comparing the algorithms by having a very limited description of the various behaviors that distinguish the algorithms, and generally, without following any formal theory. This leaves a lot to be desired. We consequently propose a formal way of looking at the process of evaluating clustering algorithms.

We can evaluate an algorithm's clustering accuracy empirically (i.e. by running the algorithms on sample data sets and observing the results), or descriptively (i.e. by describing the various behaviors of an algorithm, e.g. the number of parameters). We call these the comparison domains. For the descriptive domain, we borrow from Shtern (2010)) who gives

the various attributes that should be included in a descriptive comparison as assumptions, goals, process, and characteristics. Each algorithm makes assumptions concerning what constitutes a cluster. For example to a density-based algorithm, a cluster is assumed to be a group of concentrated VSM points surrounded by empty space. Each algorithm is also meant to achieve particular goal(s). For example the goal of DBSCAN algorithm (density-based) is to find groups of VSM points, each with a minimum of *Minpts* points within the radius *Eps*. Thirdly, each algorithm follows a unique exact process of achieving the goal. The process of the DBSCAN algorithm for example, involves finding all neighbor points within distance *Eps* of a starting point *p*, and either forming a cluster (if the number of the neighbors is at least *Minpts*), or else considering *p* as noise. If a cluster is formed, then the point *p* and its neighbors are added to this cluster and then *p* is marked as visited. The algorithm then repeats the evaluation process for all neighbors recursively. Lastly, each algorithm has some characteristics that describe its behavior. For example, one characteristic of DBSCAN algorithm is that it has two parameters, i.e. *Eps* and *Minpts*.

The Proposed Clustering Comparison Framework

***Descriptive domain*:** We give the assumptions, goals, process, and characteristics of each algorithm. This is important because it gives an insight of the strengths and weaknesses of the algorithms.

***Empirical domain*:** We need to address each of the above six clustering comparison issues.

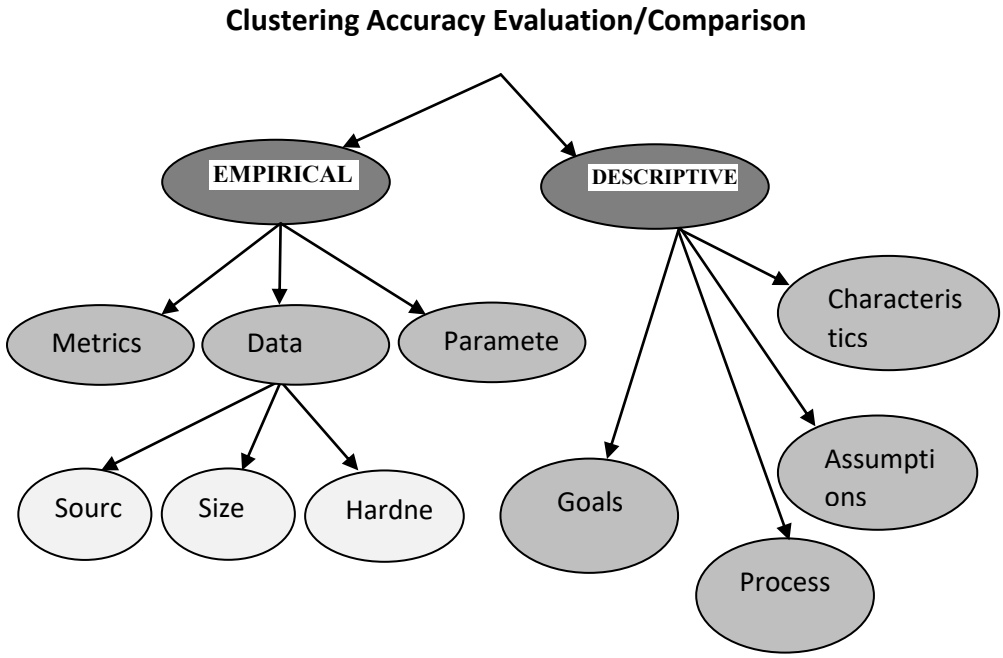

Figure 1: Framework for Clustering Accuracy Evaluation/Comparison

Comparing Accuracy

An algorithm's accuracy is mostly measured empirically i.e. under empirical domain. But we explain that the descriptive domain should also be applied to give the characteristics of an algorithm. These characteristics should explain the observed results (from the empirical measures). Concerning metrics, the accuracy of a clustering algorithm may be empirically

measured using internal measures, external measures, or visual inspection. Internal evaluation measures are depended on the type (or approach) of the algorithm used. According to (Ghosh, cited in Greene, 2007), a key drawback of internal measures is that valuable comparisons are only possible between clusterings that use the same data model and similarity metric. The limitation of the visual inspection method is that it is not suitable for large data sets since humans are prone to errors with large data, and that it is appropriate mainly on two-dimensional data i.e. the **x-y** plane. External measures are not based on a particular approach since they use only the final clustered data. Greene (2007) agrees with this by saying that the external measures are independent on the data representation method and metrics used during the text clustering process. This seems a major advantage of external metrics. Various external measures include entropy, F Measure, Purity, Rand index, and Jaccard measure. But the problem then is which of these measures to use in comparing algorithms. We therefore need to evaluate the measures and choose one among them. We do so by identifying some constraints that a good metric needs to meet, and checking which metric is better based on the constraints.

From the work of Amigo (2009), we can identify the following constraints for external evaluation measures: Homogeneity and completeness. For a metric to satisfy homogeneity, it should prefer clusterings where items in a cluster are from one class. According to Amigo (2009), this means that if **A** is a clustering with a particular cluster **C** containing items from two classes $\mathbf{c_1}$ and $\mathbf{c_2}$, while **B** is a clustering same as **A** except that items in **C** are instead in two clusters corresponding to classes $\mathbf{c_1}$ and $\mathbf{c_2}$, then a metric should yield better value for distribution **B** for it to be said to be homogenous. For a metric to satisfy completeness, it should prefer clusterings where items belonging to a class are clustered together. According to the same work, this means that if **A** is a clustering where two clusters $\mathbf{C_1}$ and $\mathbf{C_2}$ contain items of the same class **c**, while B is a clustering same as A except that clusters $\mathbf{C_1}$ and $\mathbf{C_2}$ are combined into a single cluster **C**, then a metric should yield better value for distribution **B** for it to be said to be complete.

We therefore propose evaluating an algorithm's accuracy from the simulation domain by clustering the documents using the algorithm on each of particular data sets and observing a particular metric's value for each data set. The chosen metric should satisfy both homogeneity and completeness requirements. Also, since the choice of metric has an effect on the performance of an algorithm, we should settle at some metric(s) as the standard for all the algorithms under comparison. Similarly, we should settle at particular data sizes, data source, data hardness, and parameter values for all algorithms under comparison, so as to have a standard and so avoid the causal effect of these on the performances of the algorithms. Concerning data sizes, one should use almost-near data sizes (typically small or medium sized data sets). This avoids testing scalability instead of accuracy (i.e. the data sizes usually have a causal effect on the accuracy of an algorithm). Regarding the data source, one needs to combine synthetic and benchmark data sets so as to take the benefits of each of the two types. The issue of data hardness should be similarly addressed by not varying the data hardness, unless one is testing robustness performance criteria. The data sets chosen should be used for each algorithm in the comparison.

Concerning parameters, it should be noted that from theory, these have more causal effect on the accuracy of algorithms more than the other factors under the empirical domain (i.e. data sets sizes, source, hardness, and metrics). An algorithm may produce very different results using different parameters. Also, unlike the other factors, it's hard to determine the best set of parameter values to use in a comparison, since different data sets have different distributions of data points. For example in DBSCAN, there is no ideal radius (*Eps*) that should be used for all data sets because different data sets will have different distances

between the data points. Similarly, there is no ideal value of parameter *K* for KMeans since different data sets will have different number of classes. But we need to settle at some values for the parameters of each algorithm in order to have a fair comparison. Thus, though developers of most algorithms have implemented default values for their algorithms, they are not sufficient for all data sets. The problem is therefore deciding on what parameter values to use. And as explained above, this parameter values issue is not a problem when evaluating one algorithm as long as constant parameters values are used for all data sets. What matters is when comparing algorithms. We need to justify what parameter values will be used for each algorithm to ensure fairness. We may simply opt for the default parameter values for each algorithm. However, this may be unfair to some algorithms when the data sets used are of particular sizes/nature. For example, KMeans will be favored when the documents contain around two classes (since default value for parameter *K*=number of clusters to generate is 2 for KMeans) and disfavored when most documents have many classes. Similarly, DBSCAN will be favored when most documents' approximate distances between their data points is around 0.9 and disfavored in the otherwise situation.

To solve this issue, we can use settings that give each algorithm in the comparison a particular level of chance, e.g. the best chance. And this is of course based on the properties of the algorithms (typically the parameters) and of the data sets (sizes and nature – the distributions of the data points). For centroid-based algorithms (or those with parameter k), the only major parameter is *K*=number of clusters to generate, and it's the k's value that determines the accuracy irrespective of the sizes and nature of data points. And consequently, such algorithms' best performance is when the produced clusters matches (and so equals) the classes. Thus, we argue that the best chance (compared to using default parameters) of centroid-based algorithms (or those with parameter *K*) is *K*=number of classes. Similarly, we argue that the best chance for density-based algorithms can be obtained from the algorithms' parameters and the data sets nature and sizes. And this is by assigning the parameters values corresponding to the distributions of the data points, specifically, setting *Eps* to equal the approximated average distance between the data points and their group center, and ensuring that the *MinPts* value used is not larger than the average class size. And the points' distances from a center of each group/class can be measured from the sum of squared error (SSE) formula, which measures the total squared distances between each object x_i and the center of its class (referred to as C_u) as

$$\text{SSE} = \sum_{c=1}^{k} \sum_{x_i \in Cc} \text{d}(x_i - C_u)^2 \qquad [1]$$

This means that we can approximate the average distance between the data points and their centers as

$$\text{Approximate points distance} = \sqrt{(\text{SSE/number of instances})} \qquad [2]$$

And so since DBSCAN tries to find groups of points containing at least *MinPts* points within distance *Eps,* we can argue that setting parameter *Eps*= √ (SSE/number of instances) could yield the best chance for DBSCAN. AS for parameter *MinPts*, if we use data sets whereby (no. of instances / no. of classes) >= 6 (which is true with most data sets in use for comparisons), we can comfortably retain parameter *MinPts* with default value 6, since this value represents the minimum number of objects in a group, and so is not affected by larger groups.

We thus compare algorithms' accuracy by running each algorithm on each data set and record the metric value for each run. The chosen metric should satisfy both homogeneity

and completeness requirements. The data sets used should be randomly sampled with good sample size, of almost-near data sizes (typically small or medium sized data sets), with no hardness added, and from both synthetic and benchmark sources. Random sampling is important in ensuring that the wide range of data sets available is represented (a random sample is usually a good representation of the population), and so the data biasness is avoided. The parameter values used should be *K*=number of classes (for centroid-based algorithms or those with parameter *K*) and *Eps*=√(SSE/number of instances), *MinPts*=default value (for density-based algorithms), but we ensure that the average class size for each data set (i.e. no. of instances / no. of classes) is at least 6. We then calculate the average accuracy for each algorithm out of the accuracies in all the data sets. We then rank the algorithms such that those with the best average accuracy win.

Also where appropriate, we apply the descriptive domain to describe how the assumptions, goals, process, or characteristics of an algorithm contribute to its observed accuracy (either high accuracy or low accuracy). For example from the literature, the cell adjacency problem of cell-based algorithms (their assumption is that a cluster is a group of neighboring cells) theoretically makes them have relatively lower accuracy as compared to density-based algorithms.

Addressing issue of un-clustered data sets: But an issue with calculating average accuracies and ranking the algorithms based on the average accuracies is the un-clustered data sets. From experience, an algorithm may not cluster at all some data set during the performance comparison (e.g. it may fail after a short time giving fail message, or may just hang, or may take indefinitely very long time to cluster without stopping). The question is what accuracy value to assign that algorithm for that data set? Ordinarily, we assign accuracy value of 0 for that algorithm on that data set such that the average accuracy value for that algorithm is lowered.

But the problem here is that some accuracy metrics have the best algorithms with the lower values (e.g. within cluster sum of square, percentage of wrongly clustered instances, etc), and so giving such algorithm value 0 will erroneously mean it's the most accurate in that data set, and the average is also lowered (improved). And we call such a metric a low-value metric, and the opposite a high-value metric. For example, assume two algorithms (A1 and A2) obtaining the following accuracy values (using a low-value metric) for three data sets A, B, and C respectively: (2, 4) and (2, 4, 3) i.e. both algorithms have the same accuracies for data A and B but A1 does not cluster data C. If we assign 0 for the un-clustered data set, the comparison will be as follows.

Table 1: Wrong accuracy comparison of two algorithms

Data set	A1	A2
A	2	2
B	4	4
C	0	3
Average	2	3
Rank	1	2

So algorithm A1 is unfairly judged as the most accurate in data set C, and also as the most accurate in accuracy ranking because of its lowered average. Another problem here is that we won't be able to distinguish between 0's occurring as a result of un-clustered data and those occurring as a result of actual accuracies of value 0.

So we alternative use the style of just leaving missing values for the un-clustered data sets (and computing averages excluding the un-clustered data). But the problem here is that still, an algorithm with missing values may get better average accuracy as a result (since the average is computed excluding the un-clustered data sets).

For example, repeating the immediate above example to have missing values rather than zeros for un-clustered data sets, algorithm A1 will have a blank value for data set C. The average of A1 will thus be 3 (i.e. $(2 + 4)/2$), while for A2 will remain 3. So both A1 and A2 will be ranked equal yet A1 cannot cluster at all data C unlike A2! Thus, we need a formula to cater for such un-clustered data.

In clustering, the issue of an algorithm failing to cluster some data sets should be addressed well during comparisons.

We therefore propose a simple and fair formula for un-clustered accuracies. We consider the accuracy of such un-clustered data sets as missing values. We then calculate the average accuracy for that algorithm out of all clustered data sets (excluding the un-clustered data sets), but then increase this average value proportionally to the number of un-clustered data sets. We do so by incrementing the average value by the ratio of the number of un-clustered data sets over the total data sets multiplied by the average value, i.e.

Accuracy = average accuracy + (average accuracy * unclustered data sets / total data sets) [3]

This means that for the algorithms that cluster all data sets, the formula is

Accuracy = average accuracy [4]

But in case there is an algorithm with no clustered data set at all, since it has missing values for all data sets, we simply don't apply the formula nor do we compute even the average value. We simply rank it (or them) last. Repeating the immediate above example to use this formula, we get

Table 2: Accuracy comparison of two algorithms using low-value metric

Data set	A1	A2
A	2	2
B	4	4
C		3
Average	3	3
Measure	4	3
Rank	2	1

Here, the accuracy measure of A1 is computed as $(3 + (3 * \frac{1}{3})) = 4$. Thus, A1's average is increased proportionally to the number of un-clustered data sets.

We also explain that this idea of having missing values should also happen when using a high-value metric. In this case, an algorithm with missing values should have lower average rather than higher one as a result. So the formula becomes

Accuracy = average accuracy - (average accuracy * unclustered data sets / total data sets) [5]

Which is logically the same as

Accuracy = average accuracy * clustered data sets / total data sets [6]

For example, assuming the immediate above example was based on a high-value metric, the comparison will become

Table 3: Accuracy comparison of two algorithms using high-value metric

Data set	A1	A2
A	2	2
B	4	4
C		3
Average	3	3
Measure	2	3
Rank	2	1

Here, the accuracy measure of A1 is computed as $(3 - (3 * \frac{1}{3})) = 2$ or $(3 * (\frac{2}{3})) = 2$. Thus, A1 average accuracy is reduced proportionally to the number of un-clustered data sets.

***Empirically evaluating the best chance parameter values*:** Here, some empirical study is done to test the correctness of the above recommended parameters values representing the best chance. These are *K*=number of classes (for centroid-based algorithms or those with parameter *K*) and *Eps*=√(SSE/number of instances), *MinPts*=default value (for density-based algorithms), but ensuring that the average class size for each data set (i.e. no. of instances / no. of classes) is at least 6. As explained above, the only other chance is using the default parameters values. We thus investigate if these recommended parameter values yield better performance than the use of default parameters values.

Therefore, a hypothesis is formed as follows.
Ho (null hypothesis): The recommended parameter values do not yield better performance than the use of default parameters values
H1 (alternative hypothesis): The recommended parameter values do yield better performance than the use of default parameters values

The following is the framework and results of the tests.

(i) Research type, framework
The observational research method is applied in this sub-study to observe behaviors of algorithms, specifically the accuracy, when run on data sets. This is by following the empirical domain of Figure 1 (Framework for Clustering Evaluation or Comparison) which dictates determining the parameter values, metrics, data sets sources, sizes, and nature used in the study. But since the study aims as determining if some parameter values give the best chance, parameter values are not determined. The rest are explained below.

(ii) Algorithms
The KMeans and DBSCAN algorithm are used to represent the algorithms that require parameter *K* and parameters (*Eps, MinPts)* respectively.

(iii) Software platform

The WEKA data mining software tool version 3.6.12 was used in the tests. The WEKA software tool was selected because of various reasons. First, all the above algorithms are available on the WEKA tool. Secondly, WEKA is widely used in data mining applications, and so is well tested. Thirdly, it is very easy to learn and use WEKA besides its friendly graphical user interface that allows for visualization of clustering results. Fourth, WEKA is open source software, and thus is freely available, modifiable, and so has a more guaranteed quality.

(iv) Data sets

The total number of data sets used for DBSCAN was 30 (15 benchmark and 15 synthetic), while for KMeans was 30 (12 benchmark and 18 synthetic). Using both benchmark and generated data sets is for the purpose of having varieties of data sets. Since we are not comparing algorithms but analyzing behaviors of same algorithm for many runs, the sizes and hardness of data used is not important. However, the average class size for each data set (i.e. no. of instances / no. of classes) is determined to be at least 6 (for DBSCAN data sets) for the purpose of testing DBSCAN parameter value with *MinPts* =6. Also, the class size used for KMeans was of course not equal to 2 (since there wouldn't be any difference in results here when *K*= default value i.e. 2 and *K*=no. of classes=2).

The benchmark data sets selected based from the popular UCI machine learning repository. The choice of this repository was on various criteria. First, the previously tested and popularly-used data sets were preferred. These are more likely to be standard and of good quality. Secondly, the data sets should be freely available and permissions for academic use granted by its authors. Thirdly, only preprocessed data sets which are in the format of the preferred WEKA software tool i.e. .arff or .csv formats were used. Using the already preprocessed data saves comparison time. Fourth, only the data sets with already pre-assigned classes were selected as required by our framework. This is so as to be able to directly apply external evaluation measures during evaluation of clusterings. The UCI data sets were selected randomly to ensure that the selected ones are actual representation of all the data sets available, and so there was varied number of instances, attributes classes, except in checking that the obtained data sets met the conditions (no. of instances / no. of classes)>=6 (for DBSCAN data sets) and (no. of classes doesn't equal 2) for KMeans data sets.

Some synthetic data sets were generated by the following WEKA generation tools: RanodmRBF, RDG1, LED24, and Agrawal. One WEKA tool was excluded, i.e. BayesNet. This is because it generated non-numeric data, which is not desirable in our study. Other synthetic data sets used were previously generated by a researcher, Muller (2009), and made available for academic use. The data sets are described below. Also given is the calculated SSE measure for the sake of testing DBSCAN's best chance parameter values.

Table 4: Data sets for testing DBSCAN parameter values (showing data set description, number of instances, attributes, and classes, SSE measure, and value √(SSE/number of instances)

Data set Ref	Description	No. of inst.	No. of attr.	No. of classes	SSE	Sqrt(SSE/ inst.)
1	Hepatitis	155	20	2	430.2	1.7
2	Tae	151	6	3	48.4	0.6
3	Credit-g	1000	21	2	5366	2.3
4	Vehicle	846	19	4	223.6	0.5
5	Iris	150	5	3	7.0	0.2
6	Diabetes	768	9	2	121.3	0.4
7	Vote	435	17	2	1449	1.8
8	Anneal	898	39	6	2401.8	1.6
9	Livers_disorders.	345	7	2	30.0	0.3
10	Sick	3772	30	2	6611.2	1.3
11	Waveform-5000	5000	41	3	3405.2	0.8
12	Multi-feature Digit	2000	48	10	1502.6	0.9
13	Pendigits	10992	17	10	5079.9	0.7
14	Cmc	1473	10	3	2953.2	1.4
15	Segment-challenge	1500	20	7	263.8	0.4
16	Dataset generated by RandomRBF WEKA tool	1000	32	8	811.2	0.9
17	Dataset generated by RDG1 WEKA tool	1000	32	8	2310.2	1.5
18	Dataset generated by LED24 WEKA tool	1000	25	10	6908.0	2.6
19	Dataset generated by Agrawal WEKA tool	1000	10	2	2770.4	1.7
20	S1500: Dataset from Muller, E et al. (2009)	1595	21	12	1074.6	0.8
21	D05: Dataset from Muller, E et al. (2009)	1595	6	12	132.1	0.3
22	Dataset generated by RandomRBF WEKA tool	500	16	4	90.4	0.4
23	Dataset generated by RDG1 WEKA tool	500	16	4	552.8	1.1
24	Dataset generated by RandomRBF WEKA tool	1000	31	2	220.0	0.5
25	Dataset generated by RDG1 WEKA tool	250	8	2	131.7	0.7
26	S2500: Dataset from Muller, E et al. (2009)	2658	21	12	1544.7	0.8
27	D10: Dataset from Muller, E et al. (2009)	1595	11	12	446.7	0.5
28	S3500: Dataset from Muller, E et al. (2009)	3722	21	13	2136.5	0.8
29	S4500: Dataset from Muller, E et al. (2009)	4785	21	12	2766.0	0.8
30	S5500: Dataset from Muller, E et al. (2009)	5848	21	12	3578.9	0.8

Table 5: Data sets for testing KMeans parameter values (showing data set description, number of instances, attributes, and classes)

Data set Ref	Description	No. of inst.	No. of attr.	No. of classes
1	**Tae**	151	6	3
2	Vehicle	846	19	4
3	Iris	150	5	3
4	Anneal	898	39	6
5	Waveform-5000 dataset.	5000	41	3
6	Multi-feature Digit dataset.	2000	48	10
7	Pendigits	10992	17	10
8	Cmc	1473	10	3
9	Segment-challenge	1500	20	7
10	SensorDiscrimination	2212	13	3
11	LandformIdentification	300	7	15
12	Ecoli: contains protein data	336	8	8
13	Dataset generated by RandomRBF WEKA tool	1000	32	8
14	Dataset generated by RDG1 WEKA tool	1000	32	8
15	Dataset generated by LED24 WEKA tool	1000	25	10
16	S1500: Dataset from Muller, E et al. (2009)	1595	21	12
17	D05: Dataset from Muller, E et al. (2009)	1595	6	12
18	Dataset generated by RandomRBF WEKA tool	500	16	4
19	Dataset generated by RDG1 WEKA tool	500	16	4
20	S2500: Dataset from Muller, E et al. (2009)	2658	21	12
21	D10: Dataset from Muller, E et al. (2009)	1595	11	12
22	S3500: Dataset from Muller, E et al. (2009)	3722	21	13
23	S4500: Dataset from Muller, E et al. (2009)	4785	21	12
24	S5500: Dataset from Muller, E et al. (2009)	5848	21	12
25	Generated by LED24 generator of WEKA with 0% noise added	100	25	10
26	Generated by LED24 generator of WEKA with 15% noise added	100	25	10
27	Generated by LED24 generator of WEKA with 30% noise added	100	25	10
28	Dataset generated by RandomRBF WEKA tool	250	8	6
29	Dataset generated by RandomRBF WEKA tool	250	8	10
30	Dataset generated by RDG1 WEKA tool	250	8	6

(v) Metrics

We note the accuracy using the metric Error Rate. The metric is chosen because it has been found to meet the constraints of homogeneity and completeness. Besides, its popularity is unquestionable since it is included in the popular WEKA tool and some researchers who used it for comparison studies include KGA (Hao 2012) and Verma (2012). Unless the data is added hardness, it means the total inaccuracy of the algorithm will be the sum of the two WEKA metrics (failure to cluster data is inaccuracy unless noise is added). We then express (by calculation) this sum as a percentage of the total number of instances (i.e. the percentage accuracy) as

(Unclustered Instances + Incorrectly Clustered Instances) / total instances *100 [7]

This is especially because the data sets don't include hardness (i.e. outlier points) which would otherwise be validly interpreted as un-clustered data.

(vi) Method

(1) Tests using DBSCAN algorithm

For each of the 30 data sets, the algorithm is run twice, i.e.

(I) Using default parameter values of *Eps*=0.9, *MinPts*=6

(II) Using the parameter values *Eps*=√(SSE/number of instances), *MinPts*=default value (6).

The percentage accuracy value is recorded in each run using a table. The best run of the two is then noted and recorded for each data set. We finally calculate the number of times that each of the two types of runs (i.e. (I) and (II)) has better accuracy (i.e. lower percentage accuracy value). If (II) has highest number, then the alternative hypothesis is adopted, otherwise the null hypothesis. Another confirmation is done using calculated average percentage accuracy value for each of the two types of runs i.e. (I) and (II).

(2) Tests using KMeans algorithm

For each of the 30 data sets, the algorithm is run twice, i.e.

(I) Using default parameter value for *K*=2

(II) Using the parameter value *K*=number of classes

The percentage accuracy value is recorded in each run using a table. The best run of the two is then noted and recorded for each data set. We finally calculate the number of times that each of the two types of runs (i.e. (I) and (II)) has better accuracy (i.e. lower percentage accuracy value). If (II) has highest number, then the alternative hypothesis is adopted, otherwise the null hypothesis. Another confirmation is done using calculated average percentage accuracy value for each of the two types of runs i.e. (I) and (II).

(vii) Results for tests using DBSCAN algorithm

Observations

- Runs (II) i.e. using parameter *Eps*=√(SSE/inst.) is better 21 times as compared to 4 times for runs (I) i.e. using default value *Eps*=0.9, and this is clearly a significant difference.
- The lower accuracy of runs (I) is also manifested from the three failed runs (i.e. the algorithm could not cluster at all) as opposed to no failed run for (II).
- In addition, the differences in accuracy values between runs (I) and (II) is quite high in most cases, and as such, the calculated average percentage accuracy values are (we assign percentage accuracy=100 where the algorithm failed to cluster);
  (I)=66.3 %, (II)= 44.2%

The number of times runs (I) and (II) are better is as follows.

	Using benchmark data	**Using synthetic data**	**Total**
Runs (I)	4	0	4
Runs (II)	9	12	21

Table 6: Parameter values test results for DBSCAN (using percentage accuracy values)

Data set		Sqrt(SSE/ inst.)	(I) Using default Eps, minpts	(II) Using *Eps*=√(SSE/inst.)	Better
Ref	Description				
1	Hepatitis	1.7	88.4	23.2	II
2	Tae	0.6	59.6	59.6	None
3	Credit-g	2.3	Fail	30.1	II
4	Vehicle	0.5	74.2	74.4	I
5	Iris	0.2	66.7	34.0	II
6	Diabetes	0.4	34.9	35.6	I
7	Vote	1.8	93.8	39.1	II
8	Anneal	1.6	93.8	23.6	II
9	Livers_disorders	0.3	42.0	45.8	I
10	Sick.	1.3	83.9	6.4	II
11	Waveform-5000	0.8	67.0	77.7	I
12	Multi-feature Digit	0.9	90.0	90.0	None
13	Pendigits	0.7	89.6	89.4	II
14	Cmc	1.4	59.5	57.1	II
15	Segment-challenge	0.4	84.3	53.0	II
16	Dataset generated by RandomRBF WEKA tool	0.9	23.1	23.1	None
17	Dataset generated by RDG1 WEKA tool	1.5	Fail	91.2	II
18	Dataset generated by LED24 WEKA tool	2.6	Fail	88.1	II
19	Dataset generated by Agrawal WEKA tool	1.7	99.6	39.1	II
20	S1500**:** Dataset from Muller, E et al. (2009)	0.8	27.1	26.8	II
21	D05**:** Dataset from Muller, E et al. (2009)	0.3	90.2	80.3	II
22	Dataset generated by RandomRBF WEKA tool	0.4	66.2	17.8	II
23	Dataset generated by RDG1 WEKA tool	1.1	88.0	30.8	II
24	Dataset generated by RandomRBF WEKA tool	0.5	33.0	19.8	II
25	Dataset generated by RDG1 WEKA tool	0.7	21.2	21.2	None
26	S2500: Dataset from Muller, E et al. (2009)	0.8	27.5	27.5	None
27	D10: Dataset from Muller, E et al. (2009)	0.5	90.4	55.6	II
28	S3500: Dataset from Muller, E et al. (2009)	0.8	19.2	19.0	II
29	S4500: Dataset from Muller, E et al. (2009)	0.8	38.1	18.9	II
30	S5500: Dataset from Muller, E et al. (2009)	0.8	37.6	27.9	II
	Totals				I: 4 II: 21 None: 5

It can be seen that runs (II) are better using both benchmark data and synthetic data (thus in any type of data), and in overall. In addition, the average accuracy values (in percentage accuracy) for both types of runs are;

(I)=66.3%, (II)=44.2%

This is clearly a significantly big accuracy difference.

Therefore, we adopt the alternative hypothesis and conclude that using the approximated parameter value *Eps*=√(SSE/inst.) is a better accuracy chance for DBSCAN instead of using default value *Eps*=0.9

(viii) Results for tests using KMeans algorithm

Table 7: Parameter values test results for KMeans (using percentage accuracy values)

Data set Ref	Description	No. of classes	(I) Using default K=2	(II) Using K= classes	Better
1	Tae	3	57.0	48.3	II
2	Vehicle	4	62.9	63.0	I
3	Iris	3	33.3	11.3	II
4	Anneal	6	49.6	57.8	I
5	Waveform-5000	3	37.7	48.9	I
6	Multi-feature Digit	10	80.2	37.8	II
7	Pendigits	10	79.4	33.8	II
8	Cmc	3	60.8	61.1	I
9	Segment-challenge	7	71.5	33.4	II
10	SensorDiscrimination	3	55.2	29.4	II
11	LandformIdentification	15	86.3	32.7	II
12	Ecoli	8	37.8	38.7	I
13	Dataset generated by RandomRBF WEKA tool	8	36.0	26.3	II
14	Dataset generated by RDG1 WEKA tool	8	70.5	80.9	I
15	Dataset generated by LED24 WEKA tool	10	88.0	65.4	II
16	S1500: Dataset from Muller, E et al. (2009)	12	81.0	33.3	II
17	D05: Dataset from Muller, E et al. (2009)	12	80.7	53.8	II
18	Dataset generated by RandomRBF WEKA tool	4	33.6	11.2	II
19	Dataset generated by RDG1 WEKA tool	4	50.4	65.6	I
20	S2500: Dataset from Muller, E et al. (2009)	12	80.9	14.5	II
21	D10: Dataset from Muller, E et al. (2009)	12	80.9	38.3	II
22	S3500: Dataset from Muller, E et al. (2009)	13	81.0	27.4	II
23	S4500: Dataset from Muller, E et al. (2009)	12	81.1	22.5	II
24	S5500: Dataset from Muller, E et al. (2009)	12	80.8	34.3	II
25	Generated by LED24 generator of WEKA with 0% noise added	10	81.0	52.0	II
26	Generated by LED24 generator of WEKA with 15% noise added	10	77.0	68.0	II
27	Generated by LED24 generator of WEKA with 30% noise added	10	84.0	74.0	II
28	Dataset generated by RandomRBF WEKA tool	6	28.4	21.2	II
29	Dataset generated by RandomRBF WEKA tool	10	54.8	37.2	II
30	Dataset generated by RDG1 WEKA tool	6	56.8	72.0	I
	Totals				I: 8 II: 22 None: 0

Observations from table 7
The number of times runs (I) and (II) are better is as follows.

	Using benchmark data	Using synthetic data	Total
Runs (I)	5	3	8
Runs (II)	7	15	22

It can be seen that runs (II) are better using both benchmark data and synthetic data (thus in any type of data), and in overall. In addition, the average accuracy values (in percentage accuracy) for both types of runs are;

(I)=64.6%, (II)=43.1%

This is clearly a significantly big accuracy difference.

Therefore, we adopt the alternative hypothesis and conclude that using the approximated parameter value *K*=number of classes gives a better accuracy chance for KMeans instead of using default value *K*=2.
Based on the findings in Tables 6 and 7, we therefore adopt the alternative hypothesis that the recommended parameter values do yield better performance than the use of default parameters values.

CONCLUSION

It is concluded that clustering algorithms' evaluation and comparison has not been previously done satisfactory. It is recommended that applying this framework will ensure that such evaluations and comparisons are done using formal procedures that will yield dependable results. It is suggested that another study should be done to apply this framework and do a comprehensive comparison of some clustering algorithms.

REFERENCES

Amigo, E., Gonzalo, J., Artiles, J. &Verdejo, F. (2009). A comparison of Extrinsic Clustering Evaluation Metrics based on Formal Constraints. Technical Report, Departamento de Lenguajes y SistemasInformaticos, UNED, Madrid, Spain, viewed 19 January 2015, http://nlp.uned.es/docs/amigo2007a.pdf.

Chen, J. (2005). *Comparison of Clustering Algorithms and its Application to Document Clustering.* PhD Thesis. Princeton University.

Chen, Y. Qin, B. Liu, T. Liu, Y. & Li, S (2010). The Comparison of SOM and K-means for Text Clustering. *International Journal of Computer and Information Science, 3(2)*.

Greene, D. (2007). *A State-of-the-Art Toolkit for Document Clustering.* PhD Thesis. University of Dublin.

Hao, Z. (2012). A New Text Clustering Method Based on KGA. *Journal of Software*, 7(5), pp. 1-5.

Jiang, D., Pei, J. & Zhang, A. (2003). DHC: A Density-based Hierarchical Clustering Method for Time Series Gene Expression Data. *Proceedings of Third IEEE Symposium on Bioinformatics and Bioengineering 10-12 March 2003, pp. 393 – 400, print ISBN: 0-7695-1907-5*.

Müller E., Günnemann S., Assent I., Seidl T. (2009). Evaluating Clustering in Subspace Projections of High Dimensional Data http://dme.rwth-aachen.de/OpenSubspace/. In Proc. 35th International Conference on Very Large Data Bases (VLDB 2009), Lyon, France.

Prelic, A., Bleuler, S., Zimmermann, P., Wille, A., Buhlmann, P., Gruissem, W., Hennig, L., Thiele, L., & Zitzler, E. (2006). A systematic comparison and evaluation of biclustering methods for gene expression data. *Oxford University Press, 22(9)*.

Shtern, M. (2010). *Methods for Evaluating, Selecting And Improving Software Clustering Algorithms*. PhD Thesis, York University.

Verma, M., Srivastava, M., Chack, N., Diswar, A. & Gupta, N. (2012). A Comparative Study of Various Clustering Algorithms in Data Mining. *International Journal of Engineering Research and Applications (IJERA)*, 2(3).