A Text Clustering Comparison Methodology

F.M. Kwale
(Lecturer),
Dept. of Mathematics & Computer Science,
University of Eldoret,
P.O. Box 1125-30100,
ELDORET-KENYA.

P.W. Wagacha
(Associate Professor),
School of Computing & Informatics,
University of Nairobi,
P.O. Box 30197-00100,
NAIROBI-KENYA.

A. Mwaura
(Senior Lecturer),
School of Computing & Informatics,
University of Nairobi,
P.O. Box 30197-00100,
NAIROBI-KENYA.

ABSTRACT
Text Clustering is a problem of dividing text documents into groups, such that documents in one group are more similar than those in other groups. Although comparisons of the different algorithms have been done in an attempt to choose some over the others, such comparisons have been found to be either too limited or inadequate. In such comparisons, either the researchers (who are usually the authors of the algorithms being compared with others) did not apply a formal comparison methodology, or the comparisons were based on inadequate data, metrics and procedures.Also, the comparisons always focus on only the aspects where their algorithms are superior to the other algorithms. The few algorithms being compared with theirs obviously seem to be carefully selected such that they are the ones performing lesser than theirs on those aspects.Thus, there is still a large gap on the most suitable methodology for comparing the algorithms.

In this paper, a methodology for fairly comparing text clustering algorithms is proposed.

Keywords
Clustering, Text Clustering, Metrics.

1. INTRODUCTION
Text clustering (TC) is an area currently receiving a lot of attention. Previous comparisons of TC algorithms are clearly very limited. According to [1], 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 (e.g. TC algorithms) includes evaluating them. Therefore, we include both the aspects of evaluation and comparisons of TC 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. Note that the performance of a product has various criteria. Consequently, comparing two or more products is simply checking which one evaluates higher either based on a particular performance criteria or in overall.

Evaluating and comparing algorithms in TC is still a big challenge nowadays. Various identified reasons for this are;

- **Nature of comparison**: Most researchers have compared algorithm only empirically. But it's evident that some aspects of algorithms need to be compared descriptively and not empirically, for example the complexity of running time of an algorithm. And an algorithm may be better than another descriptively and not empirically, and so which of the two comparison types to use is a challenge.

- **Nature of data**: The nature (or hardness) of text documents can vary based on various factors. This study identifiedthese to be the irregularities of shapes of clusters, irregularities of densities of points in a group, as well as the noise level (unusual values including erroneous data/missing values, outliers, or unknown values). And it has been noted that the performance of a TC algorithm may vary by using data of different hardness. And in that case, it's important to have either a standard data or data sets with varying hardness for fair comparisons.

- **Sizes of data**: The performance of a TC algorithm, and consequently the comparison results may vary with varying data sizes. It's therefore important to consider a standard size of the data sets for all the algorithms. And the various aspects of the data sizes were identified as the total number of text documents, the total number of attributes (or terms), and the number of groups/topics (or clusters).

- **Source of data**: Sources of data include personally generated data sets or benchmark data sets (of real-world data sets that are readily available and have been grouped into clusters, so an algorithms clustering is compared with the already known clusters). It is still not agreed which of the three types of sources is more appropriate to use in evaluating TC algorithms. However, it is noted that the source of data has an impact on the conclusion of a TC algorithms comparison. Each of the two types of sources has advantages and disadvantages. Thus during a comparison, we need to combine the two sources where possible to ensure a fair comparison.

- **Evaluation criteria and metrics:**Many researchers have commended on the performance of TC algorithms based on unclear or limited criteria. For example, some claim superiority of an algorithm due to lesser running time only, and others on accuracy (and using unjustified metric). In such research, there is no comprehensive look at the various criteria for the performance of a TC algorithm. Also, there is no justification of the use of the metrics (e.g. purity, rand index, etc). It has been noted from literature review that using a different metric in a comparison may in many cases alter the conclusions. Therefore, the choice of the evaluation criteria and metrics should be justified.

- **Parameter settings:**Different algorithms have different parameters. For example, K Means algorithm has one parameter i.e. **k** (i.e. the number of desired clusters). From this research, using different values of each algorithm's parameters during a TC comparison makes each algorithm perform very differently andchanges the comparison conclusions alot. And it is also evident that it's hard to

decide an optimal value for each parameter. [2]agrees with this for K Means algorithm by saying that the clustering results may be different if the different value of parameter **k** is used.Therefore, it is proposed that one should settle at particular parameter values for a fair comparison

2. THE METHODOLOGY

In the real sense, a TC comparison methodology that accommodates the different TC 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. As [3] 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, this studydoes propose common comparison settings that can be a compromise of the different approaches.

2.1 TC Comparison Domains

This study proposes a formal way of looking at the process of comparing TC algorithms.From literature, the various objectives in clustering can be identified as appropriate document representation, accuracy, efficiency, robustness, scalability, interpretability and cluster labeling, avoiding input parameters, simplicity, and applicability. Consequently, the basis (or criteria) of evaluating or comparing algorithms is derivable from these objectives. And one needs to exhaust these objectives as much as possible in fair comparisons. Also, it can be observed that some of these objectives can be determined empirically, e.g. accuracy. This is by running the algorithms on sample data sets and observing the results empirically to determine an algorithm's expected performance. Others can only be done descriptively, by describing the various behaviors of an algorithm, e.g. the number of parameters. Other objectives can be determined in both ways (i.e. empirically and descriptively). For example, the accuracy of an algorithm can be measured not only empirically, but by studying the algorithm's characteristics and concluding possibilities of weaknesses of effectively clustering the data. But other objectives can be determined by doing some calculations. For example, scalability of an algorithm or software is usually done by calculating the expected number of steps from the algorithm's statements and logic depending on the data inputs, and consequently expressed this using the big O notation. This is generalized as analytical comparison. And these are the various approaches that one can evaluate a TC algorithm. Therefore, the identified three domains (or approaches) for evaluating and comparing TC clustering algorithms are descriptive domain, analytical domain and empirical domain.

2.1.1: Descriptive domain

This study proposes comparing algorithms descriptively using the attributes **assumptions**, **goals**, **process**, and **characteristics**. These are important since one is able to know the exact nature of each algorithm under comparison. Each algorithm makes assumptions concerning what constitutes a cluster (e.g. to a density-based algorithm, a cluster is assumed to be a group of concentrated VSM points surrounded by empty space). And each algorithm is meant to achieve particular goal(s), e.g. 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. 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*. And the various characteristics of an algorithm can be derived from the objectives in clustering (given immediately above).

2.1.2: Analytical domain

TC algorithms should also be compared by analyzing their expected performanceusing some mathematical model(s). For example, the expected running time of an algorithm can be approximated from the known process by calculating the best case and the worst case of an algorithm (i.e. the least possible running time and the highest possible running time respectively). Also,the big O notation should be given to express the scalability of an algorithm,e.g. an algorithm of order 2 i.e. O(2) has the highest power being 2.

2.1.3: Empirical domain

Finally, TC algorithms should be compared or evaluated based on observed results based on some scenario. The algorithm is run using the scenario and using appropriate settings, and the results empirically observed. This is done using appropriate metrics and data sets. The measures that can be observed under this domain include accuracy, efficiency, scalability, and robustness.

2.2 Performance Criteria

Thus, TC algorithms should be compared using the above criteria (in section 2.1).Butit's obvious that some criteria have higher priorities than others. For example, it's obvious that the key-most requirement for any process is its ability to accomplish what it's expected to do with sufficient accuracy. But this should be within reasonable time. Some algorithms have been found to take too long time to cluster some data set (e.g. 10 hours instead of few seconds), such that it's completely not practical for most applications. Thus, accuracy and efficiency are primary performance factors. But two other criteria that are related to accuracy and efficiency are scalability and robustness. The size of data sets affects a lot the performance of algorithms. For example, an algorithm may take a few seconds to cluster a small data size, and yet take even a full day to cluster a data set of typical larger size, i.e. being less scalable. Alternatively the larger data size may make the algorithm produce wrong results, i.e. be less accurate. Another algorithm may even fail entirely to cluster such larger data. And in such cases, it may not be important to consider such an algorithm as accurate or efficient only when dealing with small data sizes, and not with larger sizes which are typical in text mining. Similarly, the hardness of data (either irregularity of shapes, irregularities of densities or level of noise) affects algorithms' accuracy and efficiency. An accurate or efficient algorithm on normal data may either produce wrong results when dealing with harder data or be less efficient in such data. Other criteria (interpretability and cluster labeling, usability, and applicability) are clearly secondary to the above four.Also, it is important to note that each of the criteria can be done under one or more of the above domains. For example, one can measure the salability of an algorithm analytically using the big O notation, or by using a simulation and observing the time taken by the algorithm on particular scenario. Also, interpretability of clusters of an algorithm can be described as a behavior of the

algorithm (e.g. either the algorithm produces cluster labels or not) i.e. under descriptive domain. This algorithm's behavior can also be empirically observed i.e. under empirical domain.

2.3 Comparing Performance

Here, this study proposes the exact method of evaluating/comparing the performance of any given algorithms' for each of the above criteria. For each performance criterion, it specifies the domain(s) under which the comparison needs to be done (i.e. to address the issue of the type of comparison), then describes how to evaluate or compare the performance.For any criterion under thedescriptive domain, the specified are the assumptions, goals, process, and characteristics of each algorithm. For any criterion under the analytical domain, the study describes the mathematical model that needs to be used in approximating the performance, typically the big O notation. For a criteria under the empirical domain, the study addresses each of the six TC comparison (mainly empirical) issues discussed in part 1 (Introduction) i.e. the nature of comparisons, the source of data, the size of data, the nature (or hardness) of data, the metric used, and the parameter values used. However, the nature of comparison is already addressed by the domain used, and so is excluded here. The criteria that can be measured under this domain are accuracy, efficiency, scalability, and robustness. However, it's still possible to measure other criteria here, e.g. one can also measure usability by running an algorithm and observing the number of parameters it requires (instead of just stating this – the number of parameters of most algorithms is already known). But for the purpose of this methodology being proposed, the study limits the criteria here to the above four.The figure below illustrates the comparison rules.

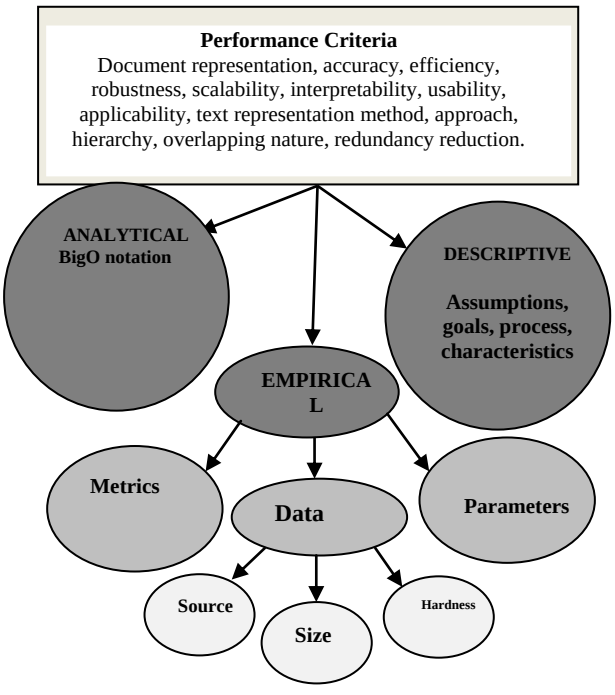

Figure 1: Framework for TC Evaluation/Comparison

It is noted that the items under the empirical domain (i.e. metrics, parameters, data sets size, data sets source, and data sets hardness) are those identified as the key issues in algorithms' empirical comparisons in part 1 above. This implies that they have causal effect on the performance of an algorithm, and thus the comparison results. The following is the proposedmethod of evaluating or comparingeach criterion.

2.3.1: Accuracy

Algorithms' accuracy should be evaluated or compared from the simulation domain by clustering the documents using each algorithm on each of particular data sets and observing a particular metric's value for each data set. Concerning the chosen metric, the study settles at external metrics because they are independent of the approach used, unlike the internal measures. [4]agrees with this by saying that the external measures are independent on the text representation method and metrics used during the text clustering process. And to justify which of the external measures to use, the study identifies the following constraints: Homogeneity and completeness. For a metric to satisfy homogeneity, it should prefer clusterings whereby items in a cluster are much as possible from one class. According to [5], 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 whereby items belonging to a class are as much as possible be clustered together. According to [5], this means that if **A** is a clustering whereby 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. Thus, the chosen metric should satisfy both homogeneity and completeness requirements. Concerning data sizes, the study suggests the use of 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, synthetic and benchmark data sets should be combined so as to take the benefits of each of the two types. The data sets chosen should be used for each algorithm in the comparison. Concerning parameters, it is 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.It is noted that though the default parameter values are not the best for all data sets, if we can use different data sets with vrying data distributions in terms of the distances among the points, the default parameter values (or K=number of classes for distance-based algorithms) represent good approximation of the most appropriate parameter values on average. And a good metric that tells of the distances among the data points is the SSE. For example, data points that are very far from one another

have higher SSE values and vice versa. Alternatively rather than using default parameters with data of varying distributions, one can comfortably set appropriate parameters to correspond to the actual SSE approximated value for each data set. For example for DBSCAN, one should set *Eps*=(SSE/number of instances)$^{0.5}$, which corresponds to the approximated inter-points distance.Andthe calculation of the average accuracy for each algorithm out of the accuracies in all the data sets should be done. This should be followed by ranking the algorithms such that those with the best average accuracy win.Also where appropriate, the descriptive domain is applied to describe how the assumptions, goals, process, or characteristics of an algorithm contribute to its observed accuracy (either high accuracy or low accuracy).

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, one assigns 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 this studyrefers to such a metric as a low-value metric, and the opposite a high-value metric.So alternatively, one can 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).This study therefore proposes a simple and fair formula for un-clustered accuracies. The accuracy of such un-clustered data sets should be consideredas missing values. One should 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. This is done by incrementing the average value by the ratio of the number of un-clustered data sets over the total number of the data sets multiplied by the average value, i.e.

> Accuracy = average accuracy + (average accuracy * unclustered data sets / total data sets)

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

> Accuracy = average accuracy

But in case there is an algorithm with no clustered data set at all, since it has missing values for all data sets, the formula simply shouldn't be appliednor computing even the average value. One should simply rank it (or them) last. Also, 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)

Which is logically the same as

> Accuracy = average accuracy * clustered data sets / total data sets

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. Using a high-value metric, the comparison will be

Table1: Sample accuracy comparison of2 algorithms

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*1/3))=2 or (3*2/3)=2. Thus, A1 average accuracy is reduced proportionally to the number of un-clustered data.

2.3.2: Efficiency

The efficiency of an algorithm is its ability to process using minimum computer resources, mainly the CPU and the memory. These are empirically measured using the CPU's time taken to process the algorithm's task and the memory size used by the algorithm respectively. However, the execution time is the most crucial metric for determining algorithms' efficiency comparison and evaluation. And many researchers have computed the efficiency of algorithms by running them using various data sets of any size, then computing the average running times, and observing the algorithm with the least average as the winner. Also, many have confused efficiency with scalability by measuring running times as data sets increase, such that what they actually measure is scalability. This study explains that the efficiency test of an algorithm should include only the normal-sized data sets. Just like in testing accuracy where the factors of empirical domain are maintained constants for all algorithms under comparison (including metrics, data sets sizes, and data sources), the same should be done here.Another aspect that lacks with previous research is lack of considerations of the correlation between accuracy and efficiency during efficiency comparisons of algorithms, leading to wrong conclusions. An algorithm may take lesser time to cluster a data set not because of its high efficient nature, but because of its inability to cluster the data fully (detecting more groups/clusters of data obviously requires more time).

i.e. low accuracy of an algorithm on some given data may make it take less time to cluster the data. And an algorithm may obviously take more time to produce better clusters.

For example, the comparative study results of three classification algorithms by [6] concluded that there is a tradeoff between accuracy and efficiency such that the most accurate algorithms in their study had the highest time complexity (lowest efficiency) and vice versa. Consequently, even the efficiency will vary with different parameter values,

since efficiency is correlated to accuracy. And consequently, one should at least talk of an algorithm's efficiency at a particular accuracy level (or at particular values of parameters). But measuring the efficiency of each algorithm at a particular level of accuracy (e.g. 90%)requires that one sets the factors (mainly the parameters) of each algorithm that will yield that accuracy level. Problems here are that even with all possible values of parameters, some algorithms may fail to yield the required accuracy level, and that one needshuge number of runs searching for the required level of accuracy using different parameter values. Alternatively, since the efficiency is correlated to accuracy, this studysuggests doing the accuracy and efficiency measures together using the same accuracy comparison runs explained above. This means using data with different SSE values, and consequently considering the default parameter values to be a good approximation of the best parameter settings for all the data sets.

Thus, this study proposescomparing an algorithm's efficiency from the simulation domain by observing the time taken (in seconds) to fully cluster the documents using the algorithm. One should use the exact arrangement as in accuracy comparison i.e. the same data sets of small to medium sizes with varying SSE levels, but with no noise added, of both synthetic and benchmark types, and default parameter values for the algorithms (or K=number of classes for distance-based algorithms). One should run each algorithm on each data set and record the time taken to cluster in each run, and then calculate the average efficiency for each algorithm out of the efficiencies in all the data sets. And the missing values issue should be addressed similar to the accuracy comparison above, whereby the efficiency metric (running time) is a low-value metric. In this case, one should calculate the efficiency measure as

> Efficiency = average efficiency + (average efficiency * unclustered data sets / total data sets)

Then the algorithms should be ranked such that those with the best average efficiency (i.e. the lowest average running times) win.Also where appropriate, one should apply the descriptive domain to describe how the assumptions, goals, process, or characteristics of an algorithm contribute to its observed efficiency (either high efficiency or low efficiency).

2.3.3: Scalability

The scalability of an algorithm as defined by some researchers has ambiguities,e.g. the ability of a TC algorithm to cluster documents of increasing sizes. [7]for example, defines the scalability of a model-building algorithm as follows: 'This is the ability to construct the model efficiently even for a large amount of high dimensional data'. From literature, the testing and comparisons of algorithms' performance has been weak, limited and unclear. First, most researchers have tested scalability of their newly-proposed algorithm empirically by running the algorithm and others being compared with using data sets of different sizes (mainly the number of documents). They then represented this using a graph, and visually observed how the running time of the algorithm varies with the different data sets sizes. If the graph of the proposed algorithm 'seems' to be better than the ones for the other algorithms, then its considered to be more scalable. But this aspect of a graph appearing "better" than others has not been defined, and clearly can't be used to conclude which algorithm is more scalable than which other. Secondly, it is not only the running time that can be affected by huge data sizes, but also any other performance criterion, usually the accuracy. Third just like explained above, oneshould vary the various aspects of data sizes i.e. the number of text documents, the number of attributes (or terms), and the number of groups/topics (or classes). Fourth, it's important for scalability to be measured also from the analytical domain by approximating an algorithms run time depending on a given data size from its logic.Fifth, is the ambiguity of the usual scalability definition i.e. "the ability to cluster documents of increasing sizes without affecting performance". The question is "is the most scalable algorithm the one with the least rate of decrement of performance as data sizes increase or the one with the highest (average) performance as the data sizes increase"?And if one is to combine the two aspects, what would be the formula? Last and just like in accuracy, it is possible to have un-clustered data sets. But the difference here is that in the case of scalability, failure to cluster larger-sized data sets should lead to lesser scalability, i.e. the data sets' sizes should be considered in the formula for un-clustered data.These last two ambiguities are illustrated below by assuming running seven algorithms (named **A1**, **A2**, **A3**, **A4**, **A5**, **A6**, and **A7**) on four data sets of increasing sizes (named **A**, **B**, **C**, and **D**), and obtaining the accuracy, average accuracy, and standard deviation (stdev) values shown below (using a high-value metric).

Table 2: Sample scalability of 4 algorithms

ACCURACIES OF 7 ALGORITHMS ON 4 DATA SETS OF INCREASING SIZES (size(A)<size(B)<size(C)<size(D))						
Algorithm	**A**	**B**	**C**	**D**	**Avg**	**Stdev**
A1	47	43	39	35	41	5.163978
A2	40	40	40	40	40	0
A3	80	70	60	50	65	12.90994
A4	60	59	58	57	58.5	1.290994
A5	62	62	62	62	62	0
A6	82	81	80		81	1
A7	5	4	3	2	3.5	1.290994

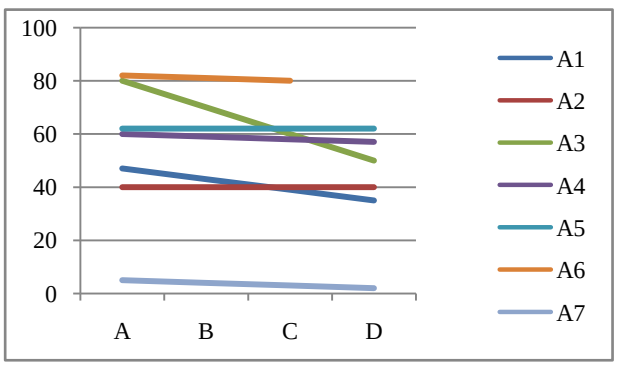

Figure 2: Visualizing scalability of 4 algorithms

It can be seen that algorithms with the least rate of decrease of accuracy are **A2** and **A5** (**stdev**=0). But it would be clearly unfair to rank them equal with the same scalability because algorithm (**A5**) has higher values of accuracy (i.e. 62). Also, both **A3** and **A4** have higher values of accuracy (thus higher average accuracy) than **A2** despite that they have higher rate of reduction of accuracy than **A2**. It's clearly unfair to consider **A2** as more scalable than **A3**, **A4** just because it has a lower rate of reduction of accuracy as data sizes increase. Still, it would be clearly unfair to decide that **A3** has higher average accuracy than **A5** and so is more scalable, when visual inspection of the graph shows that **A3**'s performance is clearly affected more by large data sizes (especially if we could include more data with higher sizes than **D**'s size). Thus, a fair measure of scalability should include both aspects of the rate of decrease in performance and the average performance. Still another issue is if **A6** is more scalable than many of the other algorithms since it has the highest average accuracy and also a lower rate of reduction of accuracy (or lower **stdev**) than **A1**, **A3**, **A4**, and **A7**(because **A6** is not able to cluster at all data set **D**,i.e. it can't cluster data of very large sizes - the most desirable for scalability). The question is should it be completely ruled out of scalability comparison or ranked as the least scalable? And in both cases, is it really fair to rank it lower than **A7** just because it has a missing value for high data size when it can cluster very well medium data sizes? The question is what formula or exact criteria would address these issues.

Solving the ambiguities for scalability on accuracy

Thus, oneshould combine the average performance and the rate of reduction of the as the data sizes increase (e.g. using stdev). Here, both the algorithms with the highest averages and those with the least standard deviationshould be favored by getting the difference of the later from the former, i.e.

Scalability on accuracy = (average accuracy - stdev)

For low-value metrics, this mathematically means

Scalability on accuracy = (average accuracy + stdev)

Unless otherwise stated, a high-value metric is assumed henceforth. And this formula assumes that the scalability of algorithms decreases with increasing size. Yes it's strange to see an opposite case. But in case such a case occurs (i.e. meaning an algorithm's scalability increases with increasing data size, implying it's highly scalable), we will add the standard deviation to the accuracy as

Scalability on accuracy = (average accuracy + stdev)

(because stdev measures are always positive whether the observed values are increasing or decreasing), or as follows for a low-value metric.

Scalability on accuracy = (average accuracy - standard deviation)

The issue of un-clustered data is addressedby incrementing the average value just like in accuracy and efficiency comparison, but proportionally to the sizes of the un-clustered sets, rather than their number,as

Scalability on accuracy = (average + stdev) + ((average + stdev) * unclustered data sizes / total data sizes)

which can be simplified to

Scalability on accuracy = (average + stdev) * (1+ (unclustered data sizes / total data sizes))

(for a low-value metric) and

Scalability on accuracy = (average - stdev) - ((average - stdev) * unclustered data sizes / total data sizes)

(for a high-value metric), which is logically the same as

Scalability on accuracy = (average - stdev) * (clustered data sizes / total data sizes)

But in case there is an algorithm with no clustered data set at all, since it has missing values for all data sets, the formula is not applied nor is the computation of average value. It is simply rank last. Also, an algorithm that fails to cluster many data sets but clusters only one should be disqualified from the formula because it's stdev will be unfairly low i.e. 0 (unfairly making it the algorithm with the least rate of reduction of performance, yet this stdev value of 0 is as a result of its inability to cluster data sets rather than good performance). And in these formulae, one can either use the exact data size values or rank levels of the sizes of the data sets (e.g. rank 1 for **A**, 2 for **B**, 3 for **C**, and 4 for **D**). But in the later case, the data sizes should be chosen such that the sizes increase in a near-uniform manner (e.g. size(**A**)=500, size(**B**)=1000, size(**C**)=1500, size(**D**)=2000). And consequently, ranking of the algorithms is done such that those with the highest scalability measures (for a high-value metric) or with the lowest scalability measures (for a low-value metric) win.

Solving the ambiguities for scalability on efficiency

Similar procedure, recordings and calculations as above (computing scalability on accuracy) are done, this time computing scalability on efficiency. In this case, the metric is the running time (which is a low-value metric).

Getting overall scalability ranking

The overall scalability ranking of the algorithms is gotten by summing the two ranks of each algorithm (i.e. in accuracy and efficiency). The algorithms with the least sums of ranks are the overall winners.

Summary of comparing scalability

This study therefore proposesevaluating an algorithm's scalability fromthe empirical domain,whereby one uses the same arrangement as in accuracy comparison except for data sizes i.e. use data sets with varying sizes, with varying SSE values, of both synthetic and benchmark types, and with no hardness added. Default parameter values for the algorithms (or K=number of classes for distance-based algorithms) are used. The data sets should be of varying sizes in the number of documents, the number of attributes, and the number of classes. Let the sizes increase in almost-uniform manner in each of the three aspects. Run each algorithm on each data set, and record the accuracy and the efficiency (running time) each time. Apply statistical measures to obtain both the average and the rate of reduction (e.g. using stdev) of both the accuracy and the efficiency for each algorithm as the data sizes increase. Obtain the scalability measure of each algorithm based on accuracy using the above formulae.Note that the clustered size is the total size (or total ranks) of the clustered data sets while total size is the total size (or the total ranks) of the data sets. The ranks of the data sets are either values that are directly proportional to the sizes (or the

ranges of sizes) of the data sets or the actual sizes. For example, rank 1 may refer to size 500, rank 2 to size 1000, rank 3 to size 3000, etc. Alternatively, rank 1 may refer to any size of between 1and 500, rank 2 to sizes 501 to 1000, etc. Referring to data sets' sizes in terms of their ranges may be specifically easier in comparing algorithms. Then rank the algorithms based on scalability on accuracy such that the most scalable algorithm (ranked 1) has the best scalability measure – lowest (for a low-value metric) or highest (for a high-value metric). In case some algorithms have missing values in either all the data sets or all the data sets except one, do not apply the formulae on them, but simply rank them the last.Then repeat the above procedure and formulae of obtaining scalability measure, but in this case based on efficiency rather than accuracy, and using a low-value metric (i.e. the running time). Rank the algorithms based on scalability on efficiency such that the most scalable algorithm (ranked 1) has the least scalability measure (meaning the least running time).

Finally, obtain the overall scalability comparison by adding the two ranks for each algorithm (i.e. scalability based on accuracy rank and scalability based on efficiency rank). The most scalable algorithm has the least sum. In case of a tie, for each of the tying algorithms, compute the algorithm's percentage of the scalability measure out of the total measure of the tying algorithms. This is done for both accuracy and efficiency. Then get the difference of the percentages for both accuracy and efficiency. Note the aspect with the higher difference (out of accuracy and efficiency), and pick the algorithm with the best rank for that aspect as the overall winner, since this means that that algorithm beats the other with a higher margin, than the other algorithm beats this winning algorithm.

Also, apply the analytical domain and express the expected running time using the big O notation, as well as the descriptive domaintodescribe how the assumptions, goals, process, or characteristics of an algorithm contribute to its observed scalability.

2.3.4 Flexibility (or robustness)

Apart from testing the accuracy, efficiency, and scalability of algorithms, it's important to test if the performance of the algorithms is affected by data of different hardness.This is one criterion that has not been included in the research of most researchers (or has not been appropriately measured), yet is also important because an algorithm may perform the best in ordinary-type data yet perform poorly if the data has high hardness levels. This study identifies the various aspects of hardness as irregularities of shapes, irregularities of densities, and noise levels. And in those few researches that have included this criterion, only the change in efficiency of algorithms has been approximated. But we explain that just like in scalability, we need also to measure the rate of change of accuracy. Many algorithms will have much lower accuracy with harder data, such that it's important to consider how much the accuracy is affected for different algorithms, so as to make a better decision since many applications encounter hardness of data. And the same issues as in scalability apply here, i.e.combining averages with the rate of change of performance as data hardness increase, and missing values.

Thus, the study proposes comparing algorithms' robustness empirically just like in scalability except that it is the data hardness rather than size that is varied. Butit is noted that it may be hard to obtain data of varying levels in each of the three aspects of hardness (the available data may only be in terms of noise levels). And most data generators allow setting hardness levels in only one aspect, i.e. noise. In this case, one should compare the other aspects of hardness under the descriptive domain. One should run each algorithm using data of increasing levels of hardness and measure the accuracy and efficiency each time, and then calculatethe average accuracy and the average efficiency, as well as their standard deviations, and then calculate the robustness on accuracy as

> Robustness on accuracy = (average + stdev) + ((average + stdev) * unclustered data sizes / total data sizes)

(for a low-value metric) and

> Robustness on accuracy = (average - stdev) * (clustered data sizes / total data sizes)

(for a high-value metric). The robustness on efficiency should also be calculated using the same formula (for low-value metric). Then rank the algorithms just like in scalability ranking.Also, from the descriptive domain, deduce the algorithm's flexibility in all the three aspects .

2.3.5: Overall empirical comparison

Finally, have a final empirical ranking of the algorithms based on the total rank value for each algorithm based on the four aspects of empirical comparisons i.e. accuracy, efficiency, scalability, and robustness.In case of a tie, use procedure similar to scalability/robustness ranking above for tying algorithms.

2.3.6: Other performance criteria

Other performance criteria are to be evaluated or compared under the descriptive domain. This is by describing the characteristics of the algorithms. They are;

- **The approach**: By stating the algorithm's approach.
- **Text representation model**: By stating the text representation model used by an algorithm.
- **Usability**: By describing the number of input parameters. The best algorithms have the least number of parameters.
- **Robustness**: By explaining the algorithm's ability to cluster data of irregular shapes and irregular densities.
- **Interpretability**: By stating whether or not an algorithm produces cluster labels. Those algorithms that produce cluster labels are better in this aspect.
- **Applicability**: By stating whether or not an algorithm can produce overlapping clusters. It is better for an algorithm to produce labels to make the produced clusters more understandable.
- **The hierarchy**: By stating if the algorithm is hierarchical or partitioning in terms of the organizational structure of the produced clusters. And if hierarchical, we state if it's divisive or agglomerative in nature.
- **The redundancy reduction**: By stating whether or not an algorithm does redundancy reduction or not, and if so, we state the method used. An algorithm that does redundancy reduction is more efficient, and so performs better.

3. CONCLUSIONS

It is concluded that TC algorithms' evaluation and comparison has not been previously done satisfactory. It is suggested that applying this methodology will ensure that

such evaluations and comparisons are done using formal procedures that will yield dependable results. Another study will be done to apply this methodology and do a comprehensive comparison of some TC algorithms.

4. REFERENCES

[1] Chen, J 2005, *Comparison of Clustering Algorithms and its Application to Document Clustering*, PhD Thesis, Princeton University.

[2] 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, vol. 3, no. 2.

[3] 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 methodsfor gene expression data', *Oxford University Press*, vol. 22, no. 9.

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

[5] 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.

[6] Akinola, S &Oyabugbe O 2015, 'Accuracies and Training Times of Data Mining Classsifications Algorithms: An Empirical Comparative Study', Journal of software Engineering and Applications, vol. 8, 470-477.

[7] Shahzad, W 2010, *Classification and Associative Classification Rule Discovery Using Ant Colony Optimization*, PhD Thesis, FAST National University of Computer & Emerging Sciences.