Title: A SYSTEM AND METHOD FOR A GREEDY PAIRWISE CLUSTERING

Abstract: This disclosure teaches a clustering system. The clustering system includes an initializer, a merger, a clustering assignment selector and an assigner database. The initializer adapted to perform at least one of: update of self-similarity values and calculation of an initial score for an optional clustering assignment. The merger is adapted to perform at least one of: calculate the delta values for the potential merger of any of two clusters, and update the pairwise similarity as a result of merging of two clusters with the highest score in each round of calculations. The clustering assignment selector is adapted to determine the clustering assignment with the highest overall score. The assigner database is adapted to store and retrieve at least: pairwise similarity and updates thereof, and scores of potential clustering assignment determined at each round of calculations.
A System and Method for a Greedy Pairwise Clustering

I. Description

I.A. Claim for Priority

This application claims priority of U.S. Provisional Patent Application serial No. 60/226,128 filed on August 18, 2000, incorporated in its entirety by reference herein. This application claims priority of U.S. Provisional Patent Application serial No. 60/259,575 filed on January 4, 2001, incorporated in its entirety by reference herein.

I.B. Field

This disclosure teaches techniques for pairwise clustering of data elements. More specifically, the teachings include, but are not limited to systems and methods for the implementation of a greedy pairwise clustering of data elements and and iteratively merging the best elements and/or clusters.

I.C. Background and Related Art

The following documents provide background for a better understanding of the disclosed teaching, and to that extent, they are incorporated herein by reference.

1. US Patents

- 4,181,821 Jan 1980 Pirz et al.
- 4,837,831 Jun 1989 Gillick et al.
- 5,040,133 Aug 1991 Feintuch et al.
- 5,202,840 Apr 1993 Wong
- 5,276,766 Jan 1994 Bahl et al.
5,389,936 Feb 1995 Alcock
5,404,561 Apr 1995 Castlelaz
5,519,789 May 1996 Etoh
5,537,491 Jul 1996 Mahoney et al.
5,566,078 Oct 1996 Ding et al.
5,682,321 Oct 1997 Ding et al.
5,703,964 Dec 1997 Menon et al.
5,706,503 Jan 1998 Poppen et al.
5,764,824 Jun 1998 Kurtzberg et al.
5,806,030 Sep 1998 Junqua
5,832,182 Nov 1998 Zhang et al.
5,857,169 Jan 1999 Seide
5,940,832 Aug 1999 Hamada
6,021,383 Feb 2000 Dornay et al.
6,026,397 Feb 2000 Sheppard
6,031,979 Feb 2000 Hachiya
6,038,340 Mar 2000 Ancin et al.
6,134,541 Oct 2000 Castell et al.
6,144,838 Nov 2000 Sheehan

2. Other References


Hofmann, T. & Buhmann, J.M., "Inferring Hierarchical Clustering Structures by Deterministic Annealing"

I.D. Definitions

To better understand this disclosure and its teachings, the following terms are described asunder:

Clustering Assignment – a specific case that represents an allocation of elements to one or more clusters. A clustering assignment indicates which element belongs to each cluster.
Cluster - a set of one or more elements

Element - a specific data point in the domain under consideration

Merge - the operation of combining two clusters, including a cluster containing a single element, to form a merged cluster such that all the elements contained in each cluster become part of the merged cluster.

Score - a real number representing a quantitative and qualitative indication of how good a cluster assignment is.

Conventionally, clustering assignment problems are considered a subset of pattern recognition problems. Solutions to clustering assignment problems are well-known. In such clustering assignment problems, the goal is to partition a set of data points into multiple sets of data points. These multiple sets of data points are commonly referred to as clusters. Clearly, the partitioning (or clustering) of data points is influenced by assumptions regarding the origin of these data points that form the cluster. The particular field of application in which the clusters will be used also influences the partitioning.

Given N data points that need to be clustered, it can be easily determined that 1 to N clusters can be formed in a given clustering assignment. FIGs. 1A and 1B are examples of two possible clustering assignments for elements "X_1" through "X_{13}" organized in a two dimensional projection. The first clustering assignment is shown in FIG. 1A where there are three clusters 10, 11, and 12 containing \{X_1, X_2, X_3, X_4, X_5\}, \{X_6, X_7, X_8, X_9\}, and \{X_{10}, X_{11}, X_{12}, X_{13}\} respectively. FIG.1B shows a second possible clustering assignment, using the same data points. In this clustering assignment, there are four clusters 15, 16,
17, and 18 containing \( \{X_1, X_2, X_3, X_5\}, \{X_4, X_9\}, \{X_6, X_7, X_8\}, \) and \( \{X_{10}, X_{11}, X_{12}, X_{13}\} \) respectively.

Such conventional clustering assignment techniques have been used, for example, in pattern and image processing (US patents 5,519,789, 5,537,491, 5,703,964, 5,857,169, 6,038,340), speech analysis (US patents 4,181,821, 4,837,831, 5,276,766, 5,806,030), placement (US patents 5,202,840, 5,566,078, 5,682,321), data clustering in databases (US Patents 5,706,503, 5,832,182, 5,940,832), and general clustering and clustering analysis (US patents 5,040,133, 5,389,936, 5,404,561, 5,764,824, 6,021,383, 6,026,397, 6,031,979, 6,134,541).

These conventional techniques allegedly solve certain clustering and scoring problems. However, they constrain the data elements by imposing certain conditions on the data elements. It will be advantageous to develop efficient clustering techniques that require as few assumptions as possible.

II. Summary

To help realize the advantages mentioned above, the disclosed teachings provide a clustering system. The system comprises, an initializer, a merger, and a selector. All three units are further connected to an assigner database. The initializer is adapted to accept as an input pairwise similarity data and provide the merger and the assigner database with the initialized data ready for the clustering assignment process. The merger is adapted to perform the repetitive task of calculating delta values, determining a clustering assignment with a highest score, obtained from the highest delta, from all pairwise clustering
options in each round of calculations, and updating the assigner database. The selector is adapted to determine the clustering assignment having the highest overall score. The assigner database is adapted to hold the initialized pairwise similarity data and updates thereof, such updates are made by the merger, as well as other data structures as may be required for the operation of the system.

Specifically, the initial similarity value for said first one of an element or a cluster to itself is an average of a similarity of said first one of an element or a cluster to all other elements or clusters. More specifically, initial similarity value for said first one of an element or a cluster to itself is an average of all other elements or clusters to said first one of an element or cluster.

Specifically, the initial similarity values are organized in a two-dimensional table.

More specifically, the initial similarity value for said first one of an element or a cluster to itself is an average of the similarity of said first one of an element or cluster to all other elements in a same row in the table.

More specifically, the initial similarity value for said first one of an element or a cluster to itself is an average of the similarity of said first one of an element or cluster to all other elements in a same row in the table.

Specifically, the score is a sum of multiple sub-scores. More specifically, at least one sub-score is based on scores related to all elements within a cluster. Still more specifically, an initial overall score is a sum of the similarity of each element to itself. Even more specifically, each subsequent overall scores, other than said initial overall score, is calculated as a differential value from an immediately preceding score. Even more specifically, each score for a potential clustering assignment is determined by adding an immediately preceding overall
score to said differential value. Even more specifically, only a highest of said
differential values in each round of calculations is added to the immediately
preceding score.

More specifically, updating of similarity matrix is achieved by adding rows
corresponding to the merged clusters in the table and adding columns
corresponding to the merged clusters.

More specifically, the differential values are placed in a priority queue. More
specifically, the priority queue is implemented by a heap. Still more specifically,
a back-annotation of a differential value and its position in the heap is
maintained. Even more specifically, the updates of similarity matrix result in
removal of all delta values in the heap corresponding to said merged clusters.
Even more specifically, the system is adapted to replace said removed value with
the value at the bottom of the heap, reduce the heap length by one, swap value
with a parent if said value is larger then the parent or, heapify the heap starting
from a node corresponding to the value if the value is smaller than at least one
of its child nodes, and update said back-annotation to reflect changes in the
heap.

More specifically, at least one new differential value is inserted into the heap.

Another aspect of the disclosed teachings is a computer system that
implements the system summarized above.

Yet another aspect of the disclosed teachings is a method for the indirect
calculation of a series of greedy pairwise IN scores. The method comprises
inputting pairwise similarity between elements; replacing a self-similarity of each
element to itself as an average of said element’s similarity to all other elements
and all other elements to said element; calculating an initial IN score as the sum
of said self similarities; calculating a delta value corresponding to each case of merging two elements into a cluster; determining the highest of said delta values and accordingly selecting the corresponding merge of the two elements having the highest delta score for a clustering assignment; calculating the new IN score as the sum of the most recent IN score calculated and said highest delta value; updating the pairwise similarity by combining rows corresponding to said merged clusters and combining columns corresponding to said merged clusters; repeating until all elements are merged into a single cluster; selecting the clustering assignment which has the highest IN score.

Specifically, the pairwise similarities are organized in a similarity matrix. Specifically, the initial similarity value for a first element to itself is replaced by an average of the similarity of said first element to all other elements in a same row.

Specifically, the initial similarity value for a first element to itself is replaced by an average of the similarity of said first element to all other elements in a same column.

Specifically, the delta values for merging two clusters are computed using a method adding values of similarity between a first cluster and a second cluster and similarity between a second cluster and a first cluster; subtracting the self-similarity of first cluster multiplied by the ratio between the number of elements in said second cluster and said first cluster; subtracting the self similarity of second cluster multiplied by the ratio between the number of elements in said first cluster and said second cluster; and dividing by a total number of elements to be merged in said potential clustering assignment. Specifically, the
repetitive loop is stopped when a new IN score is smaller then an immediately preceding IN score.

Yet another aspect of the disclosed teachings is a method for clustering elements comprising accessing initial similarity values in a similarity matrix, said similarity values corresponding to pairs of clusters wherein each of said pair of clusters could also be an individual element; calculating a score for an optional merger of a pair of clusters; determining a highest score from all possible cluster pairs; updating said similarity matrix as a result of merging said clusters with said highest score; determining the clustering assignment with the highest overall score.

Specifically, after said accessing of initial similarity value, the initial similarity value of a first element to itself is replaced by an average of a similarity of said first one of an element or a cluster to all other elements, and all other elements to said first one of an element or a cluster.

Specifically, the initial similarity values are organized in a two-dimensional table.

More specifically, the initial similarity value for a for said first one of an element or a cluster to itself is an average of the similarity of said first one of an element or cluster to all other elements in a same row in the table.

More specifically, the initial similarity value for a for said first one of an element or a cluster to itself is an average of the similarity of said first one of an element or cluster to all other elements in a same row in the table.

Specifically, the score is a sum of multiple sub-scores. More specifically, the sum is a weighted sum. More specifically, at least one sub-score is based on scores related to all elements within a cluster. Still more specifically, an initial
overall score is a sum of the similarity of each element to itself. Even more specifically, each subsequent overall scores, other than said initial overall score, is calculated as a differential value from an immediately preceding score. Even more specifically, each score for a potential clustering assignment is determined by adding an immediately preceding overall score to said differential value. Even more specifically, only a highest of said differential values in each round of calculations is added to the immediately preceding score.

More specifically, the updating of similarity matrix is achieved by adding rows corresponding to the merged clusters in the table and adding columns corresponding to the merged clusters.

More specifically, the differential values are placed in a priority queue. More specifically the priority queue is implemented by a heap. Still more specifically, a back-annotation of a differential value and its position in the heap is maintained. Even more specifically, updates of similarity matrix result in removal of all delta values in said heap corresponding to said merged clusters.

More specifically, the method comprises replacing the removed said value with the value at the bottom of the heap; reducing the heap length by one; if said value is larger then the parent then swapping places and continuing said swap while a child is larger the a parent; if said value is smaller then at least one of its respective child nodes then heapifying the heap from said value’s node; and updating the said back-annotation to reflect the changes in the heap.

More specifically, at least one new differential value is inserted into the heap.

Yet another aspect of the disclosed teachings is a method for sorting and removal of data in a heap, the heap including nodes. The method comprises placing values in a heap; removing at least one node in the heap and updating
the heap to conform with the heap property; and updating a back-annotation of said values and their position in said heap.

Specifically, the removal of a value is comprised of replacing the removed value with a value corresponding to a bottom of the heap; reducing a length of the heap by one; if said value is larger then a parent, then swapping places and continuing said swap while a child is larger than a parent; if said value is smaller then at least one of its respective child nodes, then heapifying the heap from said value’s node.

More specifically, at least one new value is inserted into the heap.

Another aspect of the disclosed teachings is a method for the indirect calculation of a series of greedy pairwise IN-OUT scores. The method comprises inputting a cluster pairwise similarity between pairs of clusters; replacing a self-similarity of each element to itself with an average of said element’s similarity to all other elements and all other elements to said element; creating a vector of values where each value in said vector corresponds to an element and where the content of said value is the sum of the similarities of said element to all other elements; setting an initial IN-OUT score to zero; calculating IN-OUT delta values of each case of merging two clusters based on the data in said pairwise similarity and said vector; determining a highest of said delta values and accordingly selecting a corresponding clustering assignment; calculating a new IN-OUT score as a sum of the most recent IN-OUT score calculated and said highest delta value; updating said similarity matrix by combining rows of said merged clusters and combining columns of said merged clusters; updating said vector by adding values corresponding to each of the merged clusters,

subtracting similarity of first cluster to second cluster, and, subtracting similarity
of second cluster to first cluster; replacing said values with the newly calculated value; repeating until only two clusters remain; selecting a clustering assignment which has a highest IN-OUT score.

Specifically, the initial similarity value for a first element to itself is replaced by the average of the similarity of said first element to all the other elements in the same row.

Specifically, the initial similarity value for a first element to itself is replaced by the average of the similarity of said first element to all the other elements in the same column.

Specifically, the combined IN-OUT score is determined by assigning a first weight to the IN score and a second weight to the OUT score and subtracting a weighted OUT score from a weighted IN score.

Specifically, the IN delta values is computed by a sub-process comprising:

adding values of similarity between a first cluster and a second cluster and similarity between a second cluster and a first cluster; subtracting the self-similarity of first cluster multiplied by the ratio between the number of elements in said second cluster and said first cluster; subtracting the self similarity of second cluster multiplied by the ratio between the number of elements in said first cluster and said second cluster; and dividing by a total number of elements to be merged in said potential clustering assignment.

Specifically, the OUT delta values is computed by a sub-process comprising multiplying an aggregation of similarities of a first cluster to all clusters but itself by a ratio of a number of elements in a second cluster to a difference of a total
number of elements and a number of elements of the first cluster; adding an
aggregation of similarities of the second cluster to all clusters but itself by a ratio
of a number of elements in the first cluster to a difference of a total number of
elements and a number of elements of the second cluster; subtracting a sum of
values of similarity between a first cluster and a second cluster and similarity
between a second cluster and a first cluster; dividing by a difference of a total
number of elements and a sum of the number of elements of the first cluster and
second cluster.

Specifically, the repetitive loop is stopped when a new IN-OUT score is
smaller than the immediately preceding IN-OUT score.

The above summaries are merely meant to provide a guidance for a better
understanding of the disclosed teachings and are not intended to be limiting the
scope of the claims in any manner.

III. Brief Description of the Drawings

The disclosed teachings and techniques are described in more details using
embodiments thereof with reference to the attached drawings in which:

FIGs. 1A-B represent conventional illustration of two possible clustering
assignments.

FIGs. 2A-B represent conventional illustration of a heap.

FIG. 3 – Is a general similarity matrix.

FIGs. 4A-E – depict an exemplary similarity matrix input and various stages in
the calculation of the IN Score.

FIG. 5 – Is a graph of IN scores.
FIGs. 6A-D depict an exemplary similarity matrix and various stages in the calculation of the IN-OUT Score.

FIG. 7 – is a graph of IN-OUT scores.

FIGs. 8A-E show diagrams of a heap used for the purpose of determining scores for greedy based clustering assignments.

FIG. 9 – is an exemplary block diagram of the disclosed system

IV. Detailed Description of the Disclosed Techniques

Initially, understanding of some basic techniques is required for understanding implementations of the disclosed techniques.

It will be clear that a set of data points can be clustered in various clustering assignments. It is important to know which clustering assignment is the best among the various clustering assignments possible for the same set of data points. In other instances, it may be important to know if a given clustering assignment is good, thereby satisfying a minimum quality threshold. This is done by allocating a global measure, or score, to each of the clustering assignments. The quality of the clustering assignment is then determined based on this score. Finding a good clustering assignment of the data is based on finding a global measure, or score, for the quality of each clustering assignment. Techniques are then determined to obtain a good clustering assignment (or the best among several potential clustering assignments) based on this score.

In many data clustering problems, a matrix of pairwise similarity or dissimilarity measures is often used to calculate the overall score. The pairwise
similarity (or dissimilarity) measure may be a measure, based on the specific application, of the similarity (or dissimilarity) between two data points. However, it should be noted that, this similarity need not be symmetric. In other words, the similarity of element X₁ to element X₂ could be different from the similarity of element X₂ to element X₁. For instance, if the elements represent persons in a workplace, it is possible that a first employee is highly dependent on the work done by a second employee (hence similarity of employee one to employee two is high), while the second employee is much less dependant on the work done by the first employee (hence the similarity of employee two to employee one is low). Likewise, driving from point A to point B may be a shorter distance then driving from point B to point A, as a result of various access constraints such as one way streets and other road conditions. The disclosed data clustering algorithms use pairwise similarity or dissimilarity only, without making any further assumptions apart from the similarity or dissimilarity measure itself. This makes the disclosed data clustering algorithms belong to a class of statistical non-parametric techniques.

Generally, a clustering assignment where the average pairwise similarity within clusters is higher than another clustering assignment will have a better score. It should be noted that any qualitative assessments like “better,” “best,” “good,” etc, made between (or among) clustering assignments is relative to the pairwise relationship measure (similarity, dissimilarity, proximity, etc) and the scoring method used. A different similarity measure and/or scoring method could lead to different results.

FIGs. 1A-B provide examples of clustering assignments. In the example of FIGs.1A-B, similarity is related to the physical proximity. It may be difficult to
visualize by human eye and evaluate the clustering assignment of FIG. 1A and compare it against the clustering assignment of FIG. 1B. On the other hand, a score gives a concrete measure indicating the better choice of clustering assignment relative to this score. The score itself, as is noted earlier, is based on a predefined measure. Therefore, a score is calculated for each of the potential clustering assignments. The clustering assignment having the better overall score is selected.

A technique for providing a score of a certain clustering assignment is based on quantitatively determining the relationship of the elements within each cluster and combining them to calculate a score for each cluster and combining the score for each cluster within a potential clustering assignment for an overall score for the specific potential clustering assignment. This is referred to herein as the "IN" score. This IN score, thus, provides an indication of the average intra-cluster proximity of elements within a clustering assignment. Clearly, the higher the proximity of elements to other elements within their cluster, the higher is the overall score. One of the techniques that is useful in the implementation of the disclosed teaching is a priority queue. The priority queue is designed to help in finding an item with the highest priority efficiently over a series of operations. Some of the basic operations that are required for implementing a priority queue are: insert, find-minimum (or maximum), and delete-minimum (or maximum). In addition, some implementations also support joining of two priority queues efficiently. Such a joining operation is called melding. Still further, deleting an arbitrary item, increasing (or decreasing) the priority of an item, etc. are supported efficiently.
A priority queue can be implemented using a heap. A heap is a tree where every node is associated with a key. Further a key satisfies the condition that for every node, the key of the parent is greater than or equal to the key of a child. Alternately, for every node, the key of the parent could be less than or equal to the key for a child. The heap data structure is well-known in the art and is further described in Cormen et al. “Introduction to Algorithms”, pages 140-151. The heap data structure is organized as a binary tree, as shown in the example depicted in FIG. 2A. In this tree, each node I has at most two child nodes, a left child, Left(I), and a right child, Right(I). Heaps are required to satisfy the “heap” property. According to the heap property, a parent is always greater than or equal to the value of its children according to some predefined criteria. An advantage of using heaps for sorting data is its efficiency in ensuring that the node with the largest value will always be at the top of the heap. In some implementations of a heap the smallest value will always be at the top of the heap. Further, the implementation of basic operations on a heap require a processing time proportional to, at most, the height of the binary tree created by the heap. Such a height of a heap is of the order of the logarithm of N, where N is the number of elements within the heap.

Heaps can be organized in a memory array, as illustrated using an example in FIG. 2B. It should be noted that the address of the left child of a node in a heap is always two times the address of the parent. Accordingly, if the parent is placed in location “n”, the left child of the parent will be placed in location “2*n”. It should be further noted that the right child of the parent is located in address “n” will be placed in location with the address “2*n + 1”. The heap is further defined by the heap size that determines the maximum number
of nodes within the heap. It is further possible to attach additional data that can be used in conjunction with the node value, to the node.

Several computational operations are required to create, maintain and utilize heaps efficiently. The first is the build-heap operation that creates a heap from an unsorted input of data. Second is the "heapify" operation that ensures that every node in the heap obeys the heap property; assuming that individual heaps under the node also obey the heap property. In the case where the left child and right child of a heap obeys the heap property but the parent does not, thereby the parent being smaller than at least one of its children, the heapify operation performs a swap. A parent is swapped with the child that has the largest value.

However, as a result of the swap, the heap property of the child may now be corrupted. It should be noted that only the children along the branch from the root to the leaf that belong to the swapping sequence may be corrupted. Therefore, an update has to be recursively performed until such point where the parent is larger than, or equal to, both its children. The heapify operation may be performed starting at any node of the heap affecting all the children nodes thereof.

Third, the heap-extract-max operation performs at least the following: a) extracts the element at the top of the heap; b) removes that element from the top of the heap; c) replaces the extracted element from the heap with the element removed from the bottom of the heap; d) shortens the length of the heap by one; and e) uses the heapify operation to ensure that after the changes made by the above operations, the heap property is still maintained.
Fourth is heap-insert, used for adding a new element to the heap. New elements are added at the bottom of the heap, as a new node. The heap-insert operation performs at least the following: a) increases the heap length by one; b) inserts the new element in the new location created in the heap; c) checks if the new element is larger than the parent. If in step c) the child is larger than the parent, the content of the nodes is swapped and due to the nature of the heap the parent now is in conformance with the heap property. However, it is possible that the parent, possibly a child of another parent, is in violation of the heap property and hence, the operation is recursively repeated until such time where no further swapping is required. At the end of this operation, the heap satisfies the heap property after the insertion.

The disclosed teachings provide new systems and methods for clustering of elements, including the establishment of new methods for the efficient calculation of clustering assignment scores. One possible implementation of the disclosed teachings is in a computer system having as an input a two dimensional similarity matrix generally of the format described in FIG. 3. However, it should be noted that the similarity between two elements might be provided in several other forms, including a list. The similarity matrix defines the similarity between each pair of input elements and therefore respective data is organized in a two dimensional matrix. The similarity of an element to itself is naturally considered to be very high and denoted by the value "1". Other similarities may have a different degree of similarity ranging from "0" to "1". The lower the value in this matrix, lower is the similarity between the corresponding elements. The similarity of \( X_1 \) to \( X_2 \) is denoted by the value \( V_{12} \) and the similarity of \( X_2 \) to \( X_1 \) is denoted by the value \( V_{21} \).

As explained earlier \( V_{ij} \) is not necessarily equal to \( V_{ji} \). It should be noted that the
above range of [0,1] is only illustrative, and any convenient range [x,y] can be used without deviating from the scope of the disclosed teachings.

In the implemented technique, a "greedy" approach is used. The term "agglomerative" is also used in the art to denote this approach. In such a greedy approach, in every step, a current best score is calculated and used thereon, rather than calculating the entire universe of possible cases. Calculating all the possible cases may provide a more accurate result. However, this is not feasible especially where there are significantly large number of elements to be clustered.

Hence, in order to entice pairs of elements into merging initially, the disclosed teachings contemplate preferably replacing the self-similarity \( V_i = 1 \), i.e., the similarity of an element to itself, by another value more conducive to allowing the merge of elements to take place. A skilled artisan will note that leaving the self-similarity at a value of "1", though not incorrect, will result in increased "resistance" for an element to cluster with other elements.

To further improve the chance of merging, the disclosed teachings contemplates using a modified values in the similarity matrix. This is done by replacing the self-similarity value, in each row of the similarity matrix, with the average similarity to the other elements in the row. Therefore the modified self-similarity value \( V_{11} \) for \( x_1 \) shall be the result of the sum of the elements \( V_{12} \) through \( V_{1n} \) divided by \( n-1 \). While in this implementation the averaging is performed by rows, without deviating from the spirit of the disclosed teachings, columns, combination of columns and rows, as well as a weighted combination of columns and rows could be used as a basis for averaging.
FIG. 4A shows a non-limiting exemplary similarity matrix for five elements. As a first step, the self-similarity values $V_{11}$, $V_{22}$, $V_{33}$, $V_{44}$, and $V_{55}$ are replaced with the respective average for each row and hence:

$$V_{11} = (V_{12} + V_{13} + V_{14} + V_{15})/4 = (0.714 + 0.286 + 0.429 + 0.429)/4 = 0.464$$

Similarly

$$V_{22} = (V_{21} + V_{23} + V_{24} + V_{25})/4 = (0.833 + 0.167 + 0.5 + 0.333)/4 = 0.458$$

After completing all the calculations, replacing the original self-similarities with the new averages, the modified similarity matrix has the values shown in FIG. 4B.

A repetitive procedure is used for the purpose of identifying the best two clusters to be merged in a specific similarity matrix. The pair having a highest score is chosen to be merged. It should be noted that the pair could consist of two elements, two clusters, or a cluster and an element, where an element is considered to be in a cluster containing a single element, otherwise known in the art as a singleton. The similarity matrix is then recalculated by adding the merged clusters’ respective rows and columns. By this operation the similarity matrix could become a non-normalized similarity matrix thereon. For simplicity, this non-normalized similarity matrix will continue to be referred herein as a similarity matrix. The process continues until all the elements are grouped in one cluster.

This repetitive procedure is discussed in detail subsequently with reference to FIGs. 4B-4E explaining this repetitive procedure.

A maximum score will correspond to one of the selected clustering
assignments of each round of calculation. The clustering assignment corresponding to the maximum score is then chosen. Relative to the relationship measure and the scoring technique used, this chosen clustering assignment can be characterized as the best clustering assignment. As noted earlier, a different relationship measure and a different scoring technique could yield a different result.

The disclosed teachings further note that once the clustering assignment score drops for the first time no further calculations are necessary. This is because, the immediately preceding clustering assignment has at least a local maximum score and hence can be chosen as the clustering assignment of choice.

In another implementation, instead of directly calculating the score in each of the iterations, the difference between consecutive score values, herein referred to as the delta or delta values is used. These delta values are added to the immediately preceding score, resulting in the determination of the next score. Using this technique, the processing time for score calculations is reduced further.

As mentioned above, a score is determined based on the pairwise proximity or similarity, that is, it measures the proximity or similarity of elements within a cluster. However, in alternate implementations, multiple scores may be calculated. Each such score may be denoted as \( m_1 \) - \( m_n \). A final score may be obtained by adding the multiple scores to one another to get a combined score. The multiple scores may also be weighted using a variety of weights, \( \alpha, \beta, \gamma \), etc, such that more important of the multiple scores may influence the final combined score more than others. Hence, in a general form the combined score for the clustering assignment would be calculated as follows:
Combined score = $\alpha m_1 + \beta m_2 + \ldots$

In an implementation of the disclosed teachings, a measure relative to the average similarity (and/or the score noted above) of each element of a cluster to all other elements within that cluster is used. This is known as the "IN" score. The initial "IN" value is calculated for a clustering assignment wherein each cluster consists of one element, with each element having its own cluster. Therefore, the initial "IN" score is the summation of the $V_{ii}$ values starting at $i=1$ through $i=n$. Hence, the IN(5) score for the example in FIG. 4B, i.e., the IN score where there are five separate elements each belonging to its own cluster, has the following value:

$$IN(5) = V_{11} + V_{22} + V_{33} + V_{44} + V_{55} = 0.464 + 0.458 + 0.5 + 0.35 + 0.389 = 2.162$$

This IN score represents the case where each element remains in its own cluster. It should be noted that this IN score is also the highest score possible for this case. Each time a cluster assignment is tested this calculation must be repeated.

However, the disclosed teachings also contemplate using a simpler method of efficiently selecting a clustering assignment, as well as an efficient technique for calculating the IN of each clustering option. According to this technique, instead of calculating the IN score for all possible clustering assignments (ten in the case described in FIG. 4B), only delta values from the immediately previous IN score need to be calculated. The new overall score is a sum of the immediately previous IN score and the delta value. This method of calculation of IN(i-1) through a delta value from IN(i), which is simpler and faster, can be easily proven to be equally
effective by a skilled artisan.

The alternative would be to create ten new non-normalized similarity matrices and calculate the IN score from there. The non-normalized similarity is the simple sum of the similarities between elements.

The next step is to calculate the delta value that would represent the results of merging any two clusters into one cluster, while leaving all other clusters intact. The disclosed teachings contemplate calculating delta using the following formula:

\[ \Delta_m(i,j) = \frac{1}{N_i + N_j} \left( V_{ii} + V_{jj} - \frac{N_i}{N_i} V_{ii} - \frac{N_j}{N_j} V_{jj} \right) \]

where the notations have the following meanings:

\( \Delta_m(i,j) \) - is the incremental value to the score of merging cluster i and cluster j

\( N_i \) - is the number of elements in the ith cluster

\( V_{ij} \) - is the non-normalized similarity value of cluster i to cluster j

A new IN score is calculated based on the best clustering assignment, which is the one where the \( \Delta_m(i,j) \) is the highest. In FIG. 4B it is possible to identify ten optional clustering assignments for merging two of the five clusters and hence ten delta values have to be calculated. For example, the value calculated for merging the cluster containing \( X_1 \) with the cluster containing \( X_2 \) into one cluster is calculated as follows:
\[ \Delta_{in}(1,2) = \frac{1}{1+1} \left( 0.714 + 0.833 - \frac{1}{1} \ast 0.464 - \frac{1}{1} \ast 0.458 \right) \]

The result of the calculation is that \( \Delta_{in}(1,2) \) is equal to 0.3125. In another example the value calculated for merging the cluster containing \( X_3 \) with the cluster containing \( X_5 \) into one cluster is calculated as follows:

\[ \Delta_{in}(3,5) = \frac{1}{1+1} \left( 0.75 + 0.333 - \frac{1}{1} \ast 0.5 - \frac{1}{1} \ast 0.389 \right) \]

The result of the calculation is that \( \Delta_{in}(3,5) \) is equal to 0.09722. This sequence is repeated for all ten possible merges and the respective results are presented in FIG. 4B. The best score for IN(4) can now be easily calculated as it is defined as the value of IN(5) plus the highest of the delta values calculated in this step. In this case the highest delta value is that of \( \Delta_{in}(1,2) \) and hence the value of IN(4) is determined to be 2.474.

The next step of the technique recalculates the similarity matrix of FIG. 4B such that it takes into account that two clusters have been merged. In each case where there is an effect of the merge of clusters the recalculation must take place. Initially, in this recalculation, the rows respective to the clusters to be merged are added to each other followed by the same process for the columns of the clusters to be merged. FIG. 4C shows the similarity matrix after this recalculation. It can be easily seen that the values for clusters that have not merged in this step have remained the same. For example, the value for \( V_{34} \) that has the value of 0.5 in FIG. 4B has the same value in FIG. 4C. In comparison, the value \( V_{15} \) no longer exists in FIG. 4C as the clusters containing elements \( X_1 \) and \( X_2 \) were merged into one cluster. The new value is the sum of the values of \( V_{15} \) and \( V_{25} \), now \( V_{12,5} \),
which is 0.762. In the case of the value of $V_{12,12}$ the values $V_{11}, V_{12}, V_{21},$ and $V_{22},$
are summed up resulting in a value of 2.47.

- FIG.s 4C, 4D, and 4E depicts the repetition of this process that enables the
calculation of all possible IN scores using the disclosed teachings. At each stage,
the delta values are calculated and the highest delta score selected. The clusters
respective to that delta value are merged and the similarity matrix updated
accordingly. This process continues until all elements are merged into one cluster.

The results of IN(5)=2.162, IN(4)=2.474, IN(3)=2.7379, IN(2)=2.6647,
and IN(1)=2.1615, are depicted in graphical form in FIG. 5. It can be easily seen
that the IN score raises when there are four clusters in comparison to five clusters,
and raises even more when there are three clusters in comparison to four clusters.
However, the IN score begins to decline when two clusters are created and
continues to decline when all elements are combined into one cluster.

Therefore, according to the disclosed teachings, it can be concluded that by
maximizing the IN score, a “best” clustering assignment relative to the relationship
measure and the scoring technique (the IN score computation technique in this
case) is obtained. Accordingly, the case where there are three clusters, namely
$\{X_4, X_5\} \cup \{X_3\} \cup \{X_1, X_2\}$ should be chosen.

In another implementation of the disclosed teachings, the calculations of
deltas will cease when the first decline of IN scores is detected.

In yet another implementation of the disclosed teachings, multiple scoring
techniques are used, each designated its own weight factor. A graph similar to the
graph shown in FIG. 5 is drawn. However, in this case, the weighted overall score
is used rather than the IN scores alone. The highest overall score is used to
determine the best clustering assignment possible.

In still another implementation, the calculation ceases when the first decline
in score values is observed.

For calculations related to the modified similarity matrix, such as the
exemplary implementation described in FIG. 4B, another quality measurement,
the IN-OUT score, could be used. The IN-OUT score is a variation of previously
explained IN score, the variation being based on a new score measurement named
OUT. The OUT score is reflective of the average proximity between a cluster and
all other clusters. It should be noted that proximity between two clusters is the
aggregate (or average) of the proximity between all possible pairwise
combinations of members of the first cluster and the second cluster. More
specifically, the use of IN-OUT score is aimed at merging two clusters having a
strong proximity to each other but a weak proximity to all other clusters.

The initial OUT value is identical to that of the initial IN score explained
above, as each cluster contains only one element, hence IN(5)=OUT(5) of the
example. It should be further noted, that when all elements are clustered into one
cluster, the OUT score has no meaning as there is no proximity/similarity to
another cluster to compare with, and hence the score calculation ceases at two
clusters. It should be noted that the meaning of delta in the context of an OUT
score is the same as before. The delta value for an OUT score, from the
immediately preceding OUT score is calculated as follows:

\[ \Delta_{OUT}(i,j) = \frac{1}{N-N_i+N_j} \left( -V_i - V_j + \frac{N_j}{N-N_i} U_i + \frac{N_i}{N-N_j} U_j \right) \]
where the notations have the following meanings:

N – is the total number of elements to be clustered.

U – is the aggregation of similarities to all clusters but itself

All other designators have the meaning explained above in reference to the IN score.

A score may now be calculated based on both $\Delta_{IN}$ and $\Delta_{OUT}$ value as follows:

$$\Delta_{IN-OUT} = \alpha \cdot \Delta_{IN} - \beta \cdot \Delta_{OUT}$$

The modified similarity matrix using the IN-OUT score is now depicted in FIG. 6A with the corresponding delta values $\Delta_{IN-OUT}$ being calculated for each possible clustering assignment. The similarity matrix is updated and the highest overall score is selected in the same way as previously described. The vector representing all the $U$ values is updated using the following formula:

$$U_{merge(i,j)} = U_i + U_j - V_{ij} - V_{ji}$$

$U_{merge(i,j)}$ represents the value of $U$ for the merged cluster formed by merging of clusters $i$ and $j$.

Assuming an $\alpha$ and $\beta$ value each equal to "1" the results may be viewed in FIG.s 6B through 6D. The IN-OUT scores for IN-OUT(5), IN-OUT(4), IN-OUT(3), and IN-OUT(2) are 0, 0.5208, 0.9606, and 0.9977 respectively. These scores are plotted in FIG. 7 and it can be easily seen that in this case the highest IN-OUT
score is achieved when there are only two clusters. The selected cluster assignment is therefore \{X1,X2\} \{X3,X4,X5\}. The IN-OUT score for leaving each element in its own cluster is always zero, as explained above. Different \( \alpha \) and \( \beta \) values may result in a different cluster assignment.

In order to facilitate an efficient implementation of the disclosed teachings, an efficient sorting of the delta values calculated, is required. This is desirable as this clustering method may be employed for clustering a large number of elements and hence efficient sorting is required for practical implementation using the large number of elements. For such a sorting operation, the present implementation uses a heap for priority queue sorting. However, it should be noted that it is not an absolute requirement for implementing the disclosed teachings to use a heap.

However, it is advantageous to use a heap-based technique, as the heap guarantees that the largest number will always be at the top of the heap. However, certain modifications to the standard heap implementation are required to handle the fact that data stored in nodes of the heap may be changed or removed as a result of the merging of elements. It is undesirable to create the heap each time some of the values change, as this will be a time consuming task.

Therefore, a back-annotation that includes the cluster location within the heap, is used for easy update of the heap. This enables fast updates of the heap, including size reduction of the heap, without requiring recreation of the heap after every update of the table. Therefore, a remove function is added to the functions for managing the heap. The remove function allows for the removal of certain node information from the heap, as may be desired, and consequently updating the heap such that it retains the heap property. The removal is required as a result
of the merging of elements into a cluster as explained above.

The remove operation performs the following when receiving the index of a node to be removed: a) replaces the content of the node to be removed with the content of the node at the bottom of the heap; b) reduces the length of the heap by one; c) compares the content of the node with the content of its parent and performs the following: i) if the content of the element is larger then that of its parent, then a swap-up operation, as explained below, takes place; otherwise ii) a heapify operation beginning at that node takes place.

The swap-up operation replaces the content of a parent and child if the content of the child is higher than that of the parent and continues recursively as long as a swap can take place. This is repeated, when necessary, all the way to the top of the heap. By doing this, it is guaranteed that once performed, the heap in its entirety will maintain the heap property.

The operation of this technique is best explained through the example described in FIG. 8. The delta values are the same values that were calculated in the first example. The first step is to place the calculated delta values in heap 800, in no specific order, as shown in FIG. 8A. The delta values were taken from FIG. 4B using the order from $\Delta_{bn}(1,2)$ through $\Delta_{bn}(4,5)$. In addition a back-annotation table (BAT) 850, mapping the location of each delta value in the heap, is also shown in FIG. 8A. It can be easily seen that $\Delta_{bn}(1,2)$ equals to 0.3125 is placed at the location “1” of the heap in node 801. Similarly $\Delta_{bn}(1,3)$ that is equal to -0.0893 is placed at the location “2” of the heap in node 802. This continues until the last delta value is placed in location “10” at node 810. For each delta value placed in a node, the corresponding node number is placed in BAT 850. Hence, $\Delta_{bn}(1,2)$ has a
designator "1" in the table as it is placed in node 801, which is the first location of the heap. In addition to the value of each node, the origin of the delta value is also included, resulting in an immediate indication of the potential merge of elements or clusters.

At this time, it should be noted that the heap does not satisfy the heap property. That is, presently, a parent node is not always equal to, or larger than, the child nodes. Therefore a build-heap operation is required. However a modified build-heap(MBH) operation is used, because, an update of BAT 850 is also necessary as delta values are moved around heap 800. Hence, in addition to the normal operation of the build-heap operation, the corresponding update of BAT 850 takes place.

In FIG. 8B the heap 800 and BAT 850 are shown after the heap was updated in order to conform to the heap property. At the top of the heap is the largest delta value and the indication that elements "1" and "2" are the candidates for a merge. As the elements "1" and "2" are to be merged, there will be implications on the delta values respective to the delta values where either element "1" or element "2" appear. All such delta values need to be updated. The procedure of removing the top of the heap is conventionally known as explained above. After applying the heap-extract-max, the heap will have the structure described in FIG. 8C. The remove operation is now used to remove elements "1,3", "1,4", "1,5", "2,3", "2,4", and "2,5" from the heap and the result is shown in FIG. 8D. The operation can be performed due to the data stored in BAT 850 that provides for the back-annotation of the elements or clusters to the nodes.

At this stage, three new nodes will be added representing the possible
merge between the cluster 1, 2 and elements 3, 4, and 5 respectively. The corresponding delta values can be found in FIG. 4C and the newly created heap is shown in FIG. 8E. It should be noted that this technique allowed for the reduction of the size of the heap, and hence, as the operation continues a smaller heap is to be processed. BAT 850 is updated to allow for the back annotation corresponding to the changes in the heap. It should be noted that from the two merged elements or clusters, the row with more data cells in BAT 850 is replaced by "-1" to denote an invalid value. The index to the row containing less data cells is updated to correspond to a designator indicating that it is a merged case. The process is now repeated until all elements/clusters are merged into one cluster.

In another implementation, this repetitive procedure will cease once the new IN score drops for the first time from the immediately preceding IN score.

While the above discussion was for the case of delta values for the IN score it can be used for the IN-OUT score, or other scores or combination of scores. A person skilled in the art could easily implement, instead of a heap-based sorting mechanism, a binary search tree (BST) based sorting mechanism. The first step for the BST is somewhat different from that of the heap, however, the rest of the techniques used in the disclosed teachings apply.

Furthermore, a skilled artisan art could easily implement the system for use with dissimilarity measurements instead of similarity between elements. However, in such a case it will be necessary to seek minimum values instead of the maximum values.

It should be clear to a skilled artisan that the disclosed teaching can be
implemented using hardware, software or a combination thereof. An exemplary
block diagram of such a clustering system 900 is described in FIG. 9. System
900 comprises, an initializer 910, a merger 920, a selector 930 and an assigner
database 940. Initializer 910 receives the pairwise similarity data and performs
the functions required to allow the iterative processing disclosed in this
invention. Initialization functions may include at least the update of the self-
similarity values and the calculation of the initial scores. Merger 920 is adapted
to perform the iterative function of finding the best two clusters to be merged at
each step of the calculations, and providing a score for the selected clustering
assignment. It is further adapted to perform the calculation of the scores by
calculating delta values to be added to the immediately preceding score. It is
further adapted to perform the iterative process of repeating this process until all
clusters are clustered into a single cluster, or until selector 930 determines that
the highest score possible using the method disclosed was achieved. Selector
930 selects the best clustering assignment from all the potential clustering
assignments it receives. In some embodiments it will select the clustering
assignment upon first indication of a decrease in the score of the immediately
proceeding potential clustering assignment. Assigner database 940 contains all
the data necessary for the operation of merger 920 and is a source for data
provided to selector 930. Assigner database may hold the similarity data and its
updates, the delta values possibly in a heap structure. Any block described
hereinabove can be implemented in hardware, software, or combination of
hardware and software. The system can also be implemented using any type of
computer including PCs, workstations and mainframes. The disclosed teaching
also includes computer program products that includes a computer readable
medium with instructions. These instructions can include, but not limited to, source code, object code and executable code and can be in any higher level language, assembly language and machine language. The computer readable medium is not restricted to any particular type of medium but can include, but not limited to, RAMS, ROMs, hard disks, floppies, tapes and internet downloads.

Other modifications and variations to the disclosed teachings will be apparent to those skilled in the art from the foregoing disclosure and teachings. Thus, while only certain embodiments using the disclosed teachings have been specifically described herein, it will be apparent that numerous modifications may be made thereto without departing from the spirit and scope of the disclosed teachings.
What is claimed is:

1. A clustering system comprising:
   an initializer adapted to perform at least one of: update of self-similarity values and calculation of an initial score for an optional clustering assignment; and
   a mergerer adapted to perform at least one of: calculate the delta values for the potential merger of any of two clusters, and update the pairwise similarity as a result of merging of two clusters with the highest score in each round of calculations; and
   a clustering assignment selector adapted to determine the clustering assignment with the highest overall score; and
   an assigner database adapted to store and retrieve at least: pairwise similarity and updates thereof, and scores of potential clustering assignment determined at each round of calculations.

2. The clustering system of claim 1, wherein said initial similarity value for said first one of an element or a cluster to itself is an average of a similarity of said first one of an element or a cluster to all other elements.

3. The clustering system of claim 1, wherein said initial similarity values are organized in a two-dimensional table.

4. The clustering system of claim 3, wherein said initial similarity value for said first one of an element or a cluster to itself is an average of the similarity of said first one of an element or cluster to all other elements in a same row in said table.
5. The clustering system of claim 3, wherein said initial similarity value for said first one of an element or a cluster to itself is an average of the similarity of said first one of an element or cluster to all other elements in a same row in said table.

6. The clustering system of claim 1, wherein said score is a sum of multiple sub-scores.

7. The clustering system of claim 6, wherein said sum is a weighted sum.

8. The clustering system of claim 6, wherein at least one sub-score is based on scores related to all elements within a cluster.

9. The clustering system of claim 8, wherein an initial overall score is a sum of the similarity of each element to itself.

10. The clustering system of claim 9, wherein each subsequent overall scores, other than said initial overall score, is calculated as a differential value from an immediately preceding score.

11. The clustering system of claim 10, wherein each score for a potential clustering assignment is determined by adding an immediately preceding overall score to said differential value.
12. The clustering system of claim 11, wherein only a highest of said differential values in each round of calculations is added to the immediately preceding score.

13. The clustering system of claim 3, wherein said updating of similarity matrix is achieved by adding rows corresponding to the merged clusters in said table and adding columns corresponding to the merged clusters.

14. The clustering system of claim 10, wherein the differential values are placed in a priority queue.

15. The clustering system of claim 14, wherein the priority queue is implemented by a heap.

16. The clustering system of claim 15, wherein a back-annotation of a differential value and its position in the heap is maintained.

17. The clustering system of claim 16, wherein said updates of similarity matrix result in removal of all delta values in said heap corresponding to said merged clusters.

18. The clustering system of claim 17, wherein said system is adapted to replace said removed value with the value at the bottom of the heap, reduce the heap length by one, swap value with a parent if said value is larger then the parent or, heapify the heap starting from a node corresponding to the value if the value is smaller than at least one of its
child nodes, and update said back-annotation to reflect changes in the heap.

19. The clustering system of claim 17, wherein at least one new differential value is inserted into the heap.

20. A computer system with a memory and a central processing unit, the memory comprising instructions, said instructions being capable of enabling the computer to implement clustering, said instructions including instruction for implementing:

   an initializer adapted to perform at least one of: update of self-similarity values and calculation of an initial score for an optional clustering assignment;

   a merger adapted to perform at least one of: calculate the delta values for the potential merger of any of two clusters, and update the pairwise similarity as a result of merging of two clusters with the highest score in each round of calculations; and

   a clustering assignment selector adapted to determine the clustering assignment with the highest overall score; and

   an assigner database adapted to store and retrieve at least: pairwise similarity and updates thereof, and scores of potential clustering assignment determined at each round of calculations.

21. The computing system of claim 20, wherein said initial similarity value for said first one of an element or a cluster to itself is an average of a
similarity of said first one of an element or a cluster to all other elements, 
and all other elements to said first one of an element or a cluster.

22. The computing system of claim 20, wherein said initial similarity 
values are organized in a two-dimensional table.

23. The computing system of claim 22, wherein said initial similarity value 
for a for said first one of an element or a cluster to itself is an average of 
the similarity of said first one of an element or cluster to all other 
elements in a same row in said table.

24. The computing system of claim 22, wherein said initial similarity value 
for a for said first one of an element or a cluster to itself is an average of 
the similarity of said first one of an element or cluster to all other 
elements in a same row in said table.

25. The computing system of claim 20, wherein said score is a sum of 
multiple sub-scores.

26. The computing system of claim 25, wherein said sum is a weighted 
sum.

27. The computing system of claim 25, wherein at least one sub-score is 
based on scores related to all elements within a cluster.
28. The computing system of claim 25, wherein an initial overall score is a sum of the similarity of each element to itself.

29. The computing system of claim 28, wherein each subsequent overall scores, other than said initial overall score, is calculated as a differential value from an immediately preceding score.

30. The computing system of claim 29, wherein each score for a potential clustering assignment is determined by adding an immediately preceding overall score to said differential value.

31. The computing system of claim 30, wherein only the highest of said differential values in each round of calculations is added to the immediately preceding score.

32. The computing system of claim 22, wherein said updating of similarity matrix is achieved by adding rows corresponding to the merged clusters in the said table and adding columns corresponding to the merged clusters.

33. The computing system of claim 29, wherein the differential values are placed in a priority queue.

34. The computing system of claim 33, wherein said priority queue is implemented by a heap.
35. The computing system of claim 34, wherein a back-annotation of a
differential value and its position in the heap is maintained.

36. The computing system of claim 35, wherein said updates of similarity
matrix result in removal of all delta values in said heap corresponding to
said merged clusters.

37. The computing system of claim 36, wherein said system is adapted to
replace the removed value with the value at the bottom of the heap,
reduce the heap length by one, swap value with a parent if said value is
larger then the parent and swap value with a child if the child is larger the
parent, heapify the heap starting from a node corresponding to the
value if the value is smaller than at least one of its child nodes, and
update said back-annotation to reflect changes in the heap.

38. The clustering system of claim 36, wherein at least one new
differential value is inserted into the heap.

39. The computing system of claim 20, wherein said central processing
unit includes a plurality of processing units.

40. The computing system of claim 20, wherein said central processing
unit includes a plurality of distributed processing units.
41. A computer program product for clustering elements, the computer program including computer-readable media, said instructions including instructions for enabling a computer to implement operations comprising:

a) accessing initial similarity values, said similarity values corresponding to a pair of clusters wherein each of said pair of clusters could also be an individual element;

b) calculating a delta score for an optional merger of a pair of clusters;

c) determining a highest score from all possible cluster pairs' scores;

e) updating said similarity values as a result of merging said clusters with said highest score;

f) determining the clustering assignment with the highest overall score.

42. The computer program product of claim 41, wherein said similarity values are organized in a similarity matrix.

43. The computer program product of claim 41, wherein after said accessing of initial similarity value, the initial similarity value of a first element to itself is replaced by an average of a similarity of said first one of an element or a cluster to all other elements, and all other elements to said first one of an element or a cluster.

44. The computer program product of claim 41, wherein said initial similarity values are organized in a two-dimensional table.

45. The computer program product of claim 44, wherein said initial similarity value for a for said first one of an element or a cluster to itself is
an average of the similarity of said first one of an element or cluster to all
other elements in a same row in said table.

46. The computer program product of claim 44, wherein said initial
similarity value for a for said first one of an element or a cluster to itself is
an average of the similarity of said first one of an element or cluster to all
other elements in a same row in said table.

47. The computer program product of claim 41, wherein said score is a
sum of multiple sub-scores.

48. The computer program product of claim 47, wherein said sum is a
weighted sum.

49. The computer program product of claim 47, wherein at least one sub-
score is based on scores related to all elements within a cluster.

50. The computer program product of claim 49, wherein an initial overall
score is a sum of the similarity of each element to itself.

51. The computer program product of claim 50, wherein each subsequent
overall score, other than said initial overall score, is calculated as a
differential value from an immediately preceding score.
52. The computer program product of claim 51, wherein each score for a potential clustering assignment is determined by adding an immediately preceding overall score to said differential value.

53. The computer program product of claim 52, wherein only a highest of said differential values in each round of calculations is added to the immediately preceding score.

54. The computer program product of claim 44, wherein said updating of similarity matrix is achieved by adding rows corresponding to the merged clusters in said table and adding columns corresponding to the merged clusters.

55. The computer program product of claim 51, wherein the differential values are placed in a priority queue.

56. The computer program product of claim 55, wherein said priority queue is implemented by a heap.

57. The computer program product of claim 56, wherein a back-annotation of a differential value and its position in the heap is maintained.

58. The computer program product of claim 57, wherein said updates of similarity matrix result in removal of all delta values in said heap corresponding to said merged clusters.
59. The computer program product of claim 58, wherein said instruction further comprise instructions for:

   g) replacing the removed said value with the value at the bottom of the heap;
   h) reducing the heap length by one;
   i) if said value is larger then the parent then swapping places and continuing said swap while a child is larger the a parent;
   j) if said value is smaller then at least one of its respective child nodes then heapifying the heap from said value’s node; and
   k) updating the said back-annotation to reflect the changes in the heap.

60. The computer program product of claim 59, wherein at least one new differential value is inserted into the heap.

61. A method for the indirect calculation of a series of greedy pairwise IN scores, the method comprising:

   a) inputting pairwise similarity between elements;
   b) replacing a self-similarity of each element to itself as an average of said element’s similarity to all other elements and all other elements to said element;
   c) calculating an initial IN score as the sum of said self similarities;
   d) calculating a delta value corresponding to each case of merging two elements into a cluster;
e) determining the highest of said delta values and accordingly selecting the corresponding clustering assignment;

f) calculating the new IN score as the sum of the most recent IN score calculated and said highest delta value;

g) updating the pairwise similarity by combining rows corresponding to said merged clusters and combining columns corresponding to said merged clusters;

h) repeating steps d) through g) until all elements are merged into a single cluster;

i) selecting the clustering assignment which has the highest IN score.

62. The method of claim 61, wherein said pairwise similarities are organized in a similarity matrix.

63. The method of claim 61, wherein in step b) said initial similarity value for a first element to itself is replaced by an average of the similarity of said first element to all other elements in a same row.

64. The method of claim 61, wherein in step b) said initial similarity value for a first element to itself is replaced by an average of the similarity of said first element to all other elements in a same column.

65. The method of claim 61, wherein said delta values for merging two clusters are computed using a method comprising:

i) adding values of similarity between a first cluster and a second cluster and similarity between a second cluster and a first cluster;
ii) subtracting from the result of step i) the self-similarity of first cluster multiplied by the ratio between the number of elements in said second cluster and said first cluster;

iii) subtracting from the result of step ii) the self similarity of second cluster multiplied by the ratio between the number of elements in said first cluster and said second cluster; and

iv) dividing the result of step iii) by a total number of elements to be merged in said potential clustering assignment ..

66. The method of claim 61, wherein the repetitive loop is stopped when a new IN score is smaller then an immediately preceding IN score.

67. A method for clustering elements comprising:

   a) accessing initial similarity values in a similarity matrix, said similarity values corresponding to pairs of clusters wherein each of said pair of clusters could also be an individual element;

   b) calculating a score for an optional merger of a pair of clusters;

   c) determining a highest score from all possible cluster pairs;

   e) updating said similarity matrix as a result of merging said clusters with said highest score;

   f) determining the clustering assignment with the highest overall score.

68. The method of claim 67, wherein after said accessing of initial similarity value, the initial similarity value of a first element to itself is replaced by an average of a similarity of said first one of an element or a
cluster to all other elements, and all other elements to said first one of an
element or a cluster.

69. The method of claim 67, wherein said initial similarity values are
organized in a two-dimensional table.

70. The method of claim 69, wherein said initial similarity value for a for
said first one of an element or a cluster to itself is an average of the
similarity of said first one of an element or cluster to all other elements in
a same row in said table.

71. The method of claim 69, wherein said initial similarity value for a for
said first one of an element or a cluster to itself is an average of the
similarity of said first one of an element or cluster to all other elements in
a same row in said table.

72. The method of claim 67, wherein said score is a sum of multiple sub-
scores.

73. The method of claim 72, wherein said sum is a weighted sum.

74. The method of claim 72, wherein at least one sub-score is based on
scores related to all elements within a cluster.

75. The method of claim 74, wherein an initial overall score is a sum of the
similarity of each element to itself.
76. The method of claim 75, wherein each subsequent overall scores, other than said initial overall score, is calculated as a differential value from an immediately preceding score.

77. The method of claim 76, wherein each score for a potential clustering assignment is determined by adding an immediately preceding overall score to said differential value.

78. The method of claim 77, wherein only a highest of said differential values in each round of calculations is added to the immediately preceding score.

79. The method of claim 69, wherein said updating of similarity matrix is achieved by adding rows corresponding to the merged clusters in said table and adding columns corresponding to the merged clusters.

80. The method of claim 76, wherein the differential values are placed in a priority queue.

81. The method of claim 80, wherein said priority queue is implemented by a heap.

82. The method of claim 81, wherein a back-annotation of a differential value and its position in the heap is maintained.
83. The method of claim 82, wherein said updates of similarity matrix result in removal of all delta values in said heap corresponding to said merged clusters.

84. The method of claim 83, further comprising:
   g) replacing the removed said value with the value at the bottom of the heap;
   h) reducing the heap length by one;
   i) if said value is larger then the parent then swapping places and continuing said swap while a child is larger the a parent;
   j) if said value is smaller then at least one of its respective child nodes then heapifying the heap from said value’s node; and
   k) updating the said back-annotation to reflect the changes in the heap.

85. The method of claim 84, wherein at least one new differential value is inserted into the heap.

86. A method for sorting and removal of data in a heap, the heap including nodes, the method comprising:
   a) placing values in a heap;
   b) removing at least one node in the heap and updating the heap to conform with the heap property; and
   c) updating a back-annotation of said values and their position in said heap.
87. The method of claim 86, wherein said removal of a value comprises:

b1) replacing the removed value with the a value corresponding to a bottom of the heap;

b2) reducing a length of the heap by one;

b3) if said value is larger then a parent, then swapping places and continuing said swap while a child is larger than a parent; and

b4) if said value is smaller then at least one of its respective child nodes, then heapifying the heap from said value’s node.

88. The method of claim 87, wherein at least one new value is inserted into the heap.

89. A method for the indirect calculation of a series of greedy pairwise IN-OUT scores, the method comprising:

a) inputting a cluster pairwise similarity between said pairs of clusters;

b) replacing a self-similarity of each element to itself with an average of said element’s similarity to all other elements and all other elements to said element;

c) creating a vector of values where each value in said vector corresponds to an element and where the content of said value is the sum of the similarities of said element to all other elements;

d) setting an Initial IN-OUT score to zero;

e) calculating IN-OUT delta values of each case of merging two clusters based on the data in said pairwise similarity and said vector;

f) determining a highest of said delta values and accordingly selecting a corresponding clustering assignment;
g) calculating a new IN-OUT score as a sum of the most recent IN-OUT score calculated and said highest delta value;

h) updating said similarity matrix by combining rows of said merged clusters and combining columns of said merged clusters;

i) updating said vector by:

   1) adding values corresponding to each of the merged clusters;
   2) subtracting similarity of first cluster to second cluster; and,
   3) subtracting similarity of second cluster to first cluster;
   4) replacing said values with the newly calculated value;

j) repeating steps d) through i) until only two clusters remain; and

k) selecting a clustering assignment which has a highest IN-OUT score.

90. The method of claim 89, wherein said pairwise similarity is organized in a similarity matrix.

91. The method of claim 89, wherein in step b) said initial similarity value for a first element to itself is replaced by the average of the similarity of said first element to all the other elements in the same row.

92. The method of claim 89, wherein in step b) said initial similarity value for a first element to itself is replaced by the average of the similarity of said first element to all the other elements in the same column.

93. The method of claim 89, wherein the combined IN-OUT score is determined by assigning a first weight to the IN score and a second
weight to the OUT score and subtracting a weighted OUT score from a
weighted IN score.

94. The method of claim 93, wherein the IN delta values is computed by a
sub-process comprising:

   i) adding values of similarity between a first cluster and a second
cluster and similarity between a second cluster and a first cluster;

   ii) subtracting from the result of step i) the self-similarity of first
cluster multiplied by the ratio between the number of elements in said
second cluster and said first cluster;

   iii) subtracting from the result of step ii) the self similarity of second
cluster multiplied by the ratio between the number of elements in said first
cluster and said second cluster; and

   iv) dividing the result of step iii) by a total number of elements to be
merged in said potential clustering assignment.

95. The method of claim 93, wherein the OUT delta values is computed by
a sub-process comprising:

   i) multiplying an aggregation of similarities of a first cluster to all
clusters but itself by a ratio of a number of elements in a second cluster to
a difference of a total number of elements and a number of elements of
the first cluster;

   ii) adding to result of i) an aggregation of similarities of the seconf
cluster to all clusters but itself by a ratio of a number of elements in the
first cluster to a difference of a total number of elements and a number of
elements of the second cluster;
iii) subtracting from result of ii) a sum of values of similarity between a first cluster and a second cluster and similarity between a second cluster and a first cluster;

iv) dividing the result of iii) by a difference of a total number of elements and a sum of the number of elements of the first cluster and second cluster.

96. The method of claim 89, wherein the repetitive loop is stopped when a new IN-OUT score is smaller than the immediately preceding IN-OUT score.

97. The clustering system of claim 1, wherein said initial similarity value for said first one of an element or a cluster to itself is an average of a similarity of all other elements to said first one of an element or a cluster.
FIG. 1A

FIG. 1B

FIG. 2A
FIG. 2B

\[
\begin{array}{|c|c|c|c|}
\hline
\text{Address} & \text{Content} & \text{Parent} & \text{Left Child} & \text{Right Child} \\
\hline
1 & 20 & \text{root} & 2 & 3 \\
2 & 10 & 1 & 4 & 5 \\
3 & 3 & 1 & 6 & 7 \\
4 & 9 & 2 & 8 & 9 \\
5 & 6 & 2 & 10 & \text{n/a} \\
6 & 2 & 3 & \text{n/a} & \text{n/a} \\
7 & 1 & 3 & \text{n/a} & \text{n/a} \\
8 & 8 & 4 & \text{n/a} & \text{n/a} \\
9 & 7 & 4 & \text{n/a} & \text{n/a} \\
10 & 4 & 5 & \text{n/a} & \text{n/a} \\
\hline
\end{array}
\]

FIG. 3
<table>
<thead>
<tr>
<th></th>
<th>$X_1$</th>
<th>$X_2$</th>
<th>$X_3$</th>
<th>$X_4$</th>
<th>$X_5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X_1$</td>
<td>1.000</td>
<td>0.714</td>
<td>0.286</td>
<td>0.429</td>
<td>0.429</td>
</tr>
<tr>
<td>$X_2$</td>
<td>0.833</td>
<td>1.000</td>
<td>0.167</td>
<td>0.500</td>
<td>0.333</td>
</tr>
<tr>
<td>$X_3$</td>
<td>0.500</td>
<td>0.250</td>
<td>1.000</td>
<td>0.500</td>
<td>0.750</td>
</tr>
<tr>
<td>$X_4$</td>
<td>0.300</td>
<td>0.300</td>
<td>0.200</td>
<td>1.000</td>
<td>0.600</td>
</tr>
<tr>
<td>$X_5$</td>
<td>0.333</td>
<td>0.222</td>
<td>0.333</td>
<td>0.667</td>
<td>1.000</td>
</tr>
</tbody>
</table>

**FIG. 4A**

|   | $X_1$ | $X_2$ | $X_3$ | $X_4$ | $X_5$ | $\Delta_{N}(i|j)$ | $\Delta_{D}(i|j)$ |
|---|-------|-------|-------|-------|-------|-------------------|-------------------|
| $X_1$ | 0.464 | 0.714 | 0.286 | 0.429 | 0.429 | $X_1$ $X_2$ | 0.3125 |
| $X_2$ | 0.833 | 0.458 | 0.167 | 0.500 | 0.333 | $X_1$ $X_3$ | -0.0893 |
| $X_3$ | 0.500 | 0.250 | 1.000 | 0.500 | 0.750 | $X_1$ $X_4$ | -0.0429 |
| $X_4$ | 0.300 | 0.300 | 0.200 | 1.000 | 0.600 | $X_1$ $X_5$ | -0.0456 |
| $X_5$ | 0.333 | 0.222 | 0.333 | 0.667 | 0.389 | $X_2$ $X_3$ | -0.2708 |

**FIG. 4B**

|   | $X_{12}$ | $X_3$ | $X_4$ | $X_5$ | $\Delta_{N}(i|j)$ | $\Delta_{D}(i|j)$ |
|---|---------|-------|-------|-------|-------------------|-------------------|
| $X_{12}$ | 2.470 | 0.452 | 0.929 | 0.762 | | $X_{12}$ $X_3$ | -0.34425 |
| $X_3$ | 0.750 | 0.500 | 0.500 | 0.750 | | $X_{12}$ $X_4$ | -0.13552 |
| $X_4$ | 0.600 | 0.200 | 0.350 | 0.600 | | $X_{12}$ $X_5$ | -0.23181 |
| $X_5$ | 0.556 | 0.333 | 0.667 | 0.389 | | $X_3$ $X_4$ | -0.075 |

**FIG. 4C**

|   | $X_{45}$ | $X_{12}$ | $X_3$ | $\Delta_{N}(i|j)$ |
|---|----------|----------|-------|-------------------|
| $X_{45}$ | 2.006 | 1.156 | 0.533 | $X_{45}$ $X_{12}$ | -0.40744 |
| $X_{12}$ | 1.690 | 2.470 | 0.452 | $X_{45}$ $X_3$ | -0.07315 |
| $X_3$ | 1.250 | 0.750 | 0.500 | $X_{12}$ $X_3$ | -0.34425 |

**FIG. 4D**

|   | $X_{453}$ | $X_{12}$ | $\Delta_{N}(i|j)$ |
|---|----------|----------|-------------------|
| $X_{453}$ | 4.289 | 1.906 | $X_{453}$ $X_{12}$ | -0.5032 |
| $X_{12}$ | 2.143 | 2.470 | |

**FIG. 4E**

![Graph showing the relationship between Number of Clusters and a varying value.](Image)

**FIG. 5**
FIG. 6A

\[
\begin{array}{cccccc}
X_1 & X_2 & X_3 & X_4 & X_5 & U \text{ Matrix} \\
X_1 & 0.464 & 0.714 & 0.286 & 0.429 & U_1 & 1.857 \\
X_2 & 0.833 & 0.458 & 0.167 & 0.500 & U_2 & 1.833 \\
X_3 & 0.500 & 0.250 & 0.500 & 0.750 & U_3 & 2.000 \\
X_4 & 0.300 & 0.300 & 0.200 & 0.350 & U_4 & 1.400 \\
X_5 & 0.333 & 0.222 & 0.333 & 0.667 & U_5 & 1.556 \\
\end{array}
\]

\[
\begin{array}{cccc}
i & j & \Delta_{\text{IN-OUT}(i,j)} & i & j & \Delta_{\text{IN-OUT}(i,j)} \\
X_1 & X_2 & 0.520833 & X_2 & X_4 & -0.00694 \\
X_1 & X_3 & -0.14881 & X_2 & X_5 & -0.24306 \\
X_1 & X_4 & -0.07143 & X_3 & X_4 & -0.125 \\
X_1 & X_5 & -0.07606 & X_3 & X_5 & 0.162037 \\
X_2 & X_3 & -0.45139 & X_4 & X_5 & 0.439815 \\
\end{array}
\]

FIG. 6B

\[
\begin{array}{cccccc}
X_{12} & X_2 & X_3 & X_4 & X_5 & U \text{ Matrix} \\
X_{12} & 2.470 & 0.452 & 0.929 & 0.762 & U_{12} & 2.143 \\
X_3 & 0.750 & 0.500 & 0.500 & 0.750 & U_3 & 2.000 \\
X_4 & 0.600 & 0.200 & 0.350 & 0.600 & U_4 & 1.400 \\
X_5 & 0.556 & 0.333 & 0.667 & 0.389 & U_5 & 1.556 \\
\end{array}
\]

\[
\begin{array}{cccc}
i & j & \Delta_{\text{IN-OUT}(i,j)} & i & j & \Delta_{\text{IN-OUT}(i,j)} \\
X_{12} & X_3 & -0.60020 & X_3 & X_5 & 0.162037 \\
X_{12} & X_4 & -0.07837 & X_4 & X_5 & 0.439815 \\
X_{12} & X_5 & -0.31911 & X_3 & X_4 & -0.125 \\
\end{array}
\]

FIG. 6C

\[
\begin{array}{cccc}
X_{45} & X_{12} & X_3 & U \text{ Matrix} \\
X_{45} & 2.006 & 1.156 & 0.533 & U_{45} & 1.689 \\
X_{12} & 1.690 & 2.470 & 0.452 & U_{12} & 2.143 \\
X_3 & 1.250 & 0.750 & 0.500 & U_3 & 2.000 \\
\end{array}
\]

\[
\begin{array}{cccc}
i & j & \Delta_{\text{IN-OUT}(i,j)} \\
X_{45} & X_{12} & -0.11591 \\
X_{45} & X_3 & 0.037037 \\
X_{12} & X_3 & -0.60020 \\
\end{array}
\]

FIG. 6D

![Graph showing the relationship between number of clusters and a parameter](image)

FIG. 7
FIG. 9