An Empirical Study on Similarity between Documents in Nptel Application Using Clustering Techniques

S.Appavu Alias Balamurugan
Department of Information & Communication Engineering, Professor, KLNCIT

N.Kalpana
Department of Computer Science & Engineering, Assistant Professor, PSNA College of Engineering & Technology

Abstract
This chapter presents a tutorial overview of the main clustering methods used in Data Mining. The goal is to provide a self-contained review of the concepts and the similarity underlying clustering techniques. The chapter begins by providing measures and criteria that are used for determining whether two objects are similar or dissimilar. Then the clustering methods are presented, divided into: hierarchical, partitioning, density-based, model-based, grid-based, and soft-computing methods. Following the methods, the challenges of performing clustering in large data sets are discussed. Finally, the chapter presents how to determine the number of clusters.

Keywords:
Clustering, K-means, Intra-cluster homogeneity, Inter-cluster separability,

1. Introduction

Clustering and classification are both fundamental tasks in Data Mining. Classification is used mostly as a supervised learning method, clustering for unsupervised learning (some clustering models are for both). The goal of clustering is descriptive, that of classification is predictive (Veyssieres and Plant, 1998). Since the goal of clustering is to discover a new set of categories, the new groups are of interest in themselves, and their assessment is intrinsic. In classification tasks, however, an important part of the assessment is extrinsic, since the groups must reflect some reference set of classes. *"Understanding*
322 *DATA MINING AND KNOWLEDGE DISCOVERY HANDBOOK our world requires conceptualizing the similarities and differences between the entities that compose it"* (Tyron and Bailey, 1970).
Clustering groups data instances into subsets in such a manner that similar instances are grouped together, while different instances belong to different groups. The instances are thereby organized into an efficient representation that characterizes the population being sampled. Formally, the clustering structure is represented as a set of subsets $C = C1; : : : ;Ck$ of S, such that:
$S = Sk$
$i=1$
Ci and $Ci \setminus Cj = ;$
for $i 6= j$.
Consequently, any instance in S belongs to exactly one and only one subset. Clustering of objects is as ancient as the human need for describing the salient characteristics of men and objects and identifying them with a type. Therefore, it embraces various scientific disciplines: from mathematics and statistics to biology and genetics, each of which uses different terms to describe the topologies formed using this analysis. From biological "taxonomies", to medical "syndromes" and genetic "genotypes" to anufacturing "group technology"— the problem is identical: forming categories of entities and assigning individuals to the proper groups within it.

2. Distance Measures

Since clustering is the grouping of similar instances/objects, some sort of measure that can determine whether two objects are similar or dissimilar is required. There are two main type of measures used to estimate this relation: distance measures and similarity measures. Many clustering methods use distance measures to determine the similarity or dissimilarity between any pair of objects. It is useful to denote the distance between two instances xi and xj as: $d(xi,xj)$. A valid distance measure should be symmetric and obtains its minimum value (usually zero) in case of identical vectors. The distance measure is called a metric distance measure if it also
satisfies the following properties:
1. Triangle inequality $d(xi,xk) \cdot d(xi,xj) + d(xj ,xk)$ $8xi,xj ,xk 2 S$.
2. $d(xi,xj)= 0) xi = xj 8xi ,xj 2 S$.

Manuscript received December 5, 2013
Manuscript revised December 20, 2013

2.1 Minkowski: Distance Measures for Numeric Attributes

Given two *p*-dimensional instances, *xi* = (*xi*1*; xi*2*; : : : ; xip*) and *xj* =(*xj*1*; xj*2*; : : : ; xjp*),
The distance between the two data instances can be calculated using the Minkowski metric (Han and Kamber, 2001):*d*(*xi; xj*) = (*jxi*1 ¡ *xj*1*jg* + *jxi*2 ¡ *xj*2*jg* + *: : : :* + *jxip* ¡ *xjpjg*)1=*g*
Clustering Methods 323
The commonly used Euclidean distance between two objects is achieved when *g* = 2. Given *g* = 1, the sum of absolute paraxial distances (Manhattanmetric) is obtained, and with *g=1* one gets the greatest of the paraxial distances (Chebychev metric). The measurement unit used can affect the clustering analysis. To avoid the dependence on the choice of measurement units, the data should be standardized.
Standardizing measurements attempts to give all variables an equal weight. However, if each variable is assigned with a weight according to its importance, then the weighted distance can be computed as:
d(*xi; xj*) = (*w*1 *jxi*1 ¡ *xj*1*jg* + *w*2 *jxi*2 ¡ *xj*2*jg* + *: : : :* + *wp jxip* ¡ *xjpjg*)1=*g* where *wi 2* [0*;1*)

2.2 Distance Measures for Binary Attributes

The distance measure described in the last section may be easily computed for continuous-valued attributes. In the case of instances described by categorical, binary, ordinal or mixed type attributes, the distance measure should be revised. In the case of binary attributes, the distance between objects may be calculated based on a contingency table. A binary attribute is symmetric if both of its states are equally valuable. In that case, using the simple matching coefficient can assess dissimilarity between two objects:
d(*xi; xj*) = *r* + *s*
q + *r* + *s* + *t*
where *q* is the number of attributes that equal 1 for both objects; *t* is the number of attributes that equal 0 for both objects; and *s* and *r* are the number of attributes that are unequal for both objects. A binary attribute is asymmetric, if its states are not equally important (usually the positive outcome is considered more important). In this case, the denominator ignores the unimportant negative matches (t). This is called the
Jaccard coefficient:
d(*xi; xj*) = *r* + *s*
q + *r* + *s*

2.3 Distance Measures for Nominal Attributes

When the attributes are *nominal*, two main approaches may be used:

1. Simple matching:
d(*xi; xj*) = *p* ¡ *m p*

where *p* is the total number of attributes and*m*is the number of matches.

324 *DATA MINING AND KNOWLEDGE DISCOVERY HANDBOOK*
2. Creating a binary attribute for each state of each nominal attribute and computing their dissimilarity as described above.

2.4 Distance Metrics for Ordinal Attributes

When the attributes are *ordinal*, the sequence of the values is meaningful.
In such cases, the attributes can be treated as numeric ones after mapping their range onto [0,1]. Such mapping may be carried out as follows:
zi;n = *ri;n* ¡ 1
Mn ¡ 1
where *zi;n* is the standardized value of attribute *an* of object *i*. *ri;n* is that value before standardization, and*Mn* is the upper limit of the domain of attribute *an* (assuming the lower limit is 1).

2.5 Distance Metrics for Mixed-Type Attributes

In the cases where the instances are characterized by attributes of *mixedtype*, one may calculate the distance by combining the methods mentioned above. For instance, when calculating the distance between instances *i* and *j* using a metric such as the Euclidean distance, one may calculate the difference
between nominal and binary attributes as 0 or 1 (“match” or “mismatch”, respectively), and the difference between numeric attributes as the difference between their normalized values. The square of each such difference will be added to the total distance. Such calculation is employed in many clustering
algorithms presented below.

The dissimilarity *d*(*xi; xj*) between two instances, containing *p* attributes of
mixed types, is defined as:
d(*xi; xj*) =P*p*
n=1
±(*n*)
ij d(*n*)
ij
P*p*
n=1
±(*n*)
ij
where the indicator ±(*n*)

ij =0 if one of the values is missing. The contribution of attribute *n* to the distance between the two objects *d*(*n*)(*xi;xj*) is computed according to its type:
If the attribute is binary or categorical, *d*(*n*)(*xi; xj*) = 0 if *xin = xjn* ,
otherwise *d*(*n*)(*xi; xj*)=1.
If the attribute is continuous-valued, *d*(*n*)
ij = jxin¡xjnj
max*h xhn¡*min*h xhn*
, where *h*
runs over all non-missing objects for attribute *n*.
Clustering Methods 325
If the attribute is ordinal, the standardized values of the attribute are computed first and then, *zi;n* is treated as continuous-valued.

3. Similarity Functions

An alternative concept to that of the distance is the similarity function *s*(*xi; xj*) that compares the two vectors *xi* and *xj* (Duda *et al*., 2001). This function should be symmetrical (namely *s*(*xi; xj*) = *s*(*xj ; xi*)) and have a large value when *xi* and *xj* are somehow "similar" and constitute the largest value for identical vectors. A similarity function where the target range is [0,1] is called a dichotomous similarity function. In fact, the methods described in the previous sections for calculating the "distances" in the case of binary and nominal attributes may be considered as similarity functions, rather than distances.

3.1 Cosine Measure

When the angle between the two vectors is a meaningful measure of their similarity, the normalized inner product may be an appropriate similarity measure:
s(*xi; xj*) = *xTi*
¢ xj
kxik ¢ kxjk

3.2 Pearson Correlation Measure

The normalized Pearson correlation is defined as:
s(*xi; xj*) =(*xi ¡ ¹xi*)*T ¢* (*xj ¡ ¹xj*)
kxi ¡ ¹xik ¢ kxj ¡ ¹xjk
where *¹xi* denotes the average feature value of *x* over all dimensions.

3.3 Extended Jaccard Measure

The extended Jaccard measure was presented by (Strehl and Ghosh, 2000)
and it is defined as:
s(*xi; xj*) = *xTi*
¢ xj
*kxik*2 + *kxjk*2 *¡ xTi*
¢ xj

3.4 Dice Coefficient Measure

The dice coefficient measure is similar to the extended Jaccard measure and
it is defined as:
s(*xi; xj*) =2*xTi*
¢ xj
*kxik*2 + *kxjk*2
326 *DATA MINING AND KNOWLEDGE DISCOVERY HANDBOOK*

4. Evaluation Criteria Measures

Evaluating if a certain clustering is good or not is a problematic and controversial issue. In fact Bonner (1964) was the first to argue that there is no universal definition for what is a good clustering. The evaluation remains mostly in the eye of the beholder. Nevertheless, several evaluation criteria have been developed in the literature. These criteria are usually divided into two categories:
Internal and External.

4.1 Internal Quality Criteria

Internal quality metrics usually measure the compactness of the clusters using some similarity measure. It usually measures the intra-cluster homogeneity, the inter-cluster separability or a combination of these two. It does not use any external information beside the data itself.

4.1.1 Sum of Squared Error (SSE).

SSE is the simplest and most widely used criterion measure for clustering. It is calculated as:
SSE = X*K*
k=1
X
8xi2Ck
*kxi ¡ ¹kk*2
where *Ck* is the set of instances in cluster *k*; *¹k* is the vector mean of cluster
k. The components of *¹k* are calculated as:
¹k;j =1
Nk
X
8xi2Ck
xi;j
where *Nk = jCkj* is the number of instances belonging to cluster *k*.
Clustering methods that minimize the SSE criterion are often called minimum variance partitions, since by simple

algebraic manipulation the SSE criterion may be written as:
SSE = 1 2
X*K*
k=1
Nk ¹ *Sk*
where:
¹ *Sk* =1
*N*2
k
X
xi;xj2Ck
*kxi ¡ xjk*2
(*Ck*=cluster k)
The SSE criterion function is suitable for cases in which the clusters formcompact clouds that are well separated from one another (Duda *et al*., 2001).
Clustering Methods 327

4.1.2 Other Minimum Variance Criteria.

Additional minimum criteria to SSE may be produced by replacing the value of *Sk* with expressions
such as:
¹ *Sk* =1
*N*2
k
X
xi;xj2Ck
s(*xi; xj*)
or:
¹ *Sk* = min
xi;xj2Ck
s(*xi; xj*)

user authentication, and the base software environment, rather than implementing the platform themselves.

5. Clustering Methods

In this section we describe the most well-known clustering algorithms. The main reason for having many clustering methods is the fact that the notion of "cluster" is not precisely defined (Estivill-Castro, 2000). Consequently many clustering methods have been developed, each of which uses a different induction
principle. Farley and Raftery (1998) suggest dividing the clustering methods into two main groups: hierarchical and partitioning methods. Han and Kamber (2001) suggest categorizing the methods into additional three main categories: *density-based methods*, *model-based clustering* and *gridbased*
methods. An alternative categorization based on the induction principle of the various clustering methods is presented in (Estivill-Castro, 2000).

5.1 Hierarchical Methods

These methods construct the clusters by recursively partitioning the instances in either a top-down or bottom-up fashion. These methods can be subdivided as following:
Agglomerative hierarchical clustering—Each object initially represents a cluster of its own. Then clusters are successively merged until the desired cluster structure is obtained.
Clustering Methods 331
Divisive hierarchical clustering — All objects initially belong to one cluster. Then the cluster is divided into sub-clusters, which are successively divided into their own sub-clusters. This process continues until
the desired cluster structure is obtained. The result of the hierarchical methods is a dendrogram, representing the nested grouping of objects and similarity levels at which groupings change. A clustering
of the data objects is obtained by cutting the dendrogram at the desired similarity level. The merging or division of clusters is performed according to some similarity measure, chosen so as to optimize some criterion (such as a sum of squares). The hierarchical clustering methods could be further divided according to the manner that the similarity measure is calculated (Jain *et al*., 1999):
Single-link clustering (also called the connectedness, the minimum method or the nearest neighbor method) — methods that consider the distance between two clusters to be equal to the shortest distance from any member of one cluster to any member of the other cluster. If the data consist of similarities, the similarity between a pair of clusters is considered to be equal to the greatest similarity from any member of one cluster to any member of the other cluster (Sneath and Sokal, 1973).
Complete-link clustering (also called the diameter, the maximum method or the furthest neighbor method) - methods that consider the distance between two clusters to be equal to the longest distance from any member of one cluster to any member of the other cluster (King, 1967).
Average-link clustering (also called minimum variance method) – methods that consider the distance between two clusters to be equal to the average distance from any member of one cluster to any member of the other cluster. Such clustering algorithms may be found in (Ward, 1963) and (Murtagh, 1984).
The disadvantages of the single-link clustering and the average-link clustering can be summarized as follows (Guha *et al*., 1998): Single-link clustering has a drawback known as the "chaining effect": A
few points that form a bridge between two clusters cause the single-link clustering to unify these two clusters into one. Average-link clustering may cause elongated clusters to split and for portions
of neighboring elongated clusters to merge. The complete-link clustering methods usually produce more compact

clusters and more useful hierarchies than the single-link clustering methods, yet the 332 *DATA MINING AND KNOWLEDGE DISCOVERY HANDBOOK* single-link methods are more versatile. Generally, hierarchical methods are characterized with the following strengths:
Versatility — The single-link methods, for example, maintain good performance on data sets containing non-isotropic clusters, including well separated, chain-like and concentric clusters. Multiple partitions — hierarchical methods produce not one partition, but multiple nested partitions, which allow different users to choose different partitions, according to the desired similarity level. The hierarchical partition is presented using the dendrogram. The main disadvantages of the hierarchical methods are:
Inability to scale well—The time complexity of hierarchical algorithms is at least $O(m2)$ (where m is the total number of instances), which is non-linear with the number of objects. Clustering a large number of
objects using a hierarchical algorithm is also characterized by huge I/O costs. Hierarchical methods can never undo what was done previously. Namely there is no back-tracking capability.

5.2 Partitioning Methods

Partitioning methods relocate instances by moving them from one cluster to another, starting from an initial partitioning. Such methods typically require that the number of clusters will be pre-set by the user. To achieve global optimality in partitioned-based clustering, an exhaustive enumeration process of
all possible partitions is required. Because this is not feasible, certain greedy heuristics are used in the form of iterative optimization. Namely, a relocation method iteratively relocates points between the k clusters. The following subsections present various types of partitioning methods.

5.2.1 Error Minimization Algorithms.

These algorithms, which tend to work well with isolated and compact clusters, are the most intuitive and frequently used methods. The basic idea is to find a clustering structure that minimizes a certain error criterion which measures the "distance" of each instance to its representative value. The most well-known criterion is the Sum of Squared Error (SSE), which measures the total squared Euclidian distance of instances to their representative values. SSE may be globally optimized by exhaustively enumerating all partitions, which is very time-consuming, or by giving an approximate solution (not necessarily leading to a global minimum) using heuristics. The latter option is the most common alternative. *Clustering Methods* 333
The simplest and most commonly used algorithm, employing a squared error criterion is the *K*-means algorithm. This algorithm partitions the data into K clusters ($C1;C2; : : : ;CK$), represented by their centers or means. The center of each cluster is calculated as the mean of all the instances belonging to
that cluster. Figure 15.1 presents the pseudo-code of the *K*-means algorithm. The algorithm
starts with an initial set of cluster centers, chosen at random or according to some heuristic procedure. In each iteration, each instance is assigned to its nearest cluster center according to the Euclidean distance between the two. Then the cluster centers are re-calculated. The center of each cluster is calculated as the mean of all the instances belonging to that cluster:

$$^{1}k = \frac{1}{Nk} \sum_{q=1}^{Nk} xq$$

where Nk is the number of instances belonging to cluster k and ^{1}k is the mean of the cluster k.
A number of convergence conditions are possible. For example, the search may stop when the partitioning error is not reduced by the relocation of the centers. This indicates that the present partition is locally optimal. Other stopping criteria can be used also such as exceeding a pre-defined number of iterations.

Input: S (instance set), K (number of cluster)
Output: clusters
1: Initialize K cluster centers.
2: **while** termination condition is not satisfied **do**
3: Assign instances to the closest cluster center.
4: Update cluster centers based on the assignment.
5: **end while**
Figure 15.1. K-means Algorithm.
The *K*-means algorithm may be viewed as a gradient-decent procedure,which begins with an initial set of K cluster-centers and iteratively updates it so as to decrease the error function.
A rigorous proof of the finite convergence of the *K*-means type algorithms is given in (Selim and Ismail, 1984). The complexity of T iterations of the *K*-means algorithm performed on a sample size of m instances, each characterized by N attributes, is: $O(T ¤ K ¤ m ¤ N)$. This linear complexity is one of the reasons for the popularity of the *K*- means algorithms. Even if the number of instances is substantially large (which often is the case nowadays), this algorithm is computationally attractive. Thus, the *K*-means algorithm has an advantage in comparison to other clustering .

Related Work

5.4.1 Decision Trees.

In decision trees, the data is represented by a hierarchical tree, where each leaf refers to a concept and contains a probabilistic *Clustering Methods* 337 description of that concept. Several algorithms produce classification trees for representing the unlabelled data. The most well-known algorithms are: COBWEB—This algorithm assumes that all attributes are independent (an often too naive assumption). Its aim is to achieve high predictability of nominal variable values, given a cluster. This algorithm is not suitable for clustering large database data (Fisher, 1987). CLASSIT, an extension of COBWEB for continuous-valued data, unfortunately has similar problems as the COBWEB algorithm.

5.6 Soft-computing Methods

Section 5.4.2 described the usage of neural networks in clustering tasks. This section further discusses the important usefulness of other soft-computing methods in clustering tasks.

5.6.1 Fuzzy Clustering.

Traditional clustering approaches generate partitions; in a partition, each instance belongs to one and only one cluster. Hence, the clusters in a hard clustering are disjointed. Fuzzy clustering (see 338 *DATA MINING AND KNOWLEDGE DISCOVERY HANDBOOK* for instance (Hoppner, 2005)) extends this notion and suggests a *soft clustering* schema. In this case, each pattern is associated with every cluster using some sort of membership function, namely, each cluster is a fuzzy set of all the patterns. Larger membership values indicate higher confidence in the assignment of the pattern to the cluster. A hard clustering can be obtained from a fuzzy partition by using a threshold of the membership value. The most popular fuzzy clustering algorithm is the fuzzy *c*-means (FCM) algorithm. Even though it is better than the hard *K*-means algorithm at avoiding local minima, FCM can still converge to local minima of the squared error criterion. The design of membership functions is the most important problem in fuzzy clustering; different choices include those based on similarity decomposition and centroids of clusters. A generalization of the FCM algorithm has been proposed through a family of objective functions. A fuzzy *c*-shell algorithm and an adaptive variant for detecting circular and elliptical boundaries have been presented.

5.6.2 Evolutionary Approaches for Clustering.

Evolutionary techniques are stochastic general purpose methods for solving optimization problems. Since clustering problem can be defined as an optimization problem, evolutionary approaches may be appropriate here. The idea is to use evolutionary operators and a population of clustering structures to converge into a globally optimal clustering. Candidate clustering are encoded as chromosomes. The most commonly used evolutionary operators are: selection, recombination, and mutation. A fitness function evaluated on a chromosome determines a chromosome's likelihood of surviving into the next generation. The most frequently used evolutionary technique in clustering problems is genetic algorithms (GAs). Figure 15.2 presents a high-level pseudo-code of a typical GA for clustering. A fitness value is associated with each clusters structure. A higher fitness value indicates a better cluster structure. A suitable fitness function is the inverse of the squared error value. Cluster structures with a small squared error will have a larger fitness value.

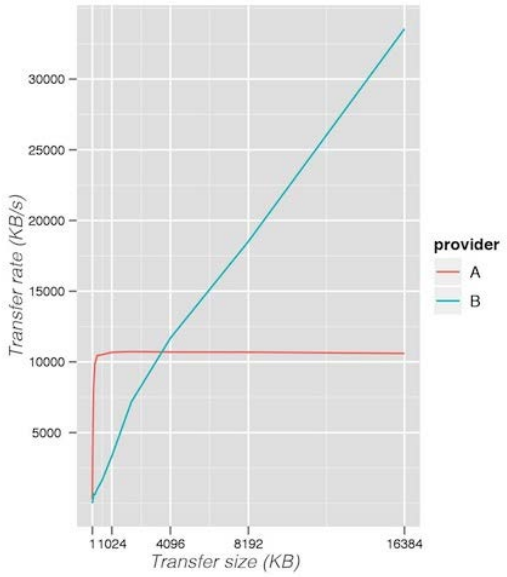
Implementation of Data Framework

6. Clustering Large Data Sets

There are several applications where it is necessary to cluster a large collection of patterns. The definition of 'large' is vague. In document retrieval, millions of instances with a dimensionality of more than 100 have to be clustered to achieve data abstraction. A majority of the approaches and algorithms proposed
in the literature cannot handle such large data sets. Approaches based on genetic algorithms, tabu search and simulated annealing are optimization techniques and are restricted to reasonably small data sets. Implementations of conceptual clustering optimize some criterion functions and are typically computationally expensive. The convergent *K*-means algorithm and its ANN equivalent, the Kohonen
net, have been used to cluster large data sets. The reasons behind the popularity
of the *K*-means algorithm are:
1. Its time complexity is $O(mkl)$, where m is the number of instances; k is the number of clusters; and l is the number of iterations taken by the algorithm to converge. Typically, k and l are fixed in advance and so the algorithm has linear time complexity in the size of the data set.
2. Its space complexity is $O(k+m)$. It requires additional space to store the data matrix. It is possible to store the data matrix in a secondary memory and access each pattern based on need. However, this scheme requires a

huge access time because of the iterative nature of the algorithm. As a
consequence, processing time increases enormously.
3. It is order-independent. For a given initial seed set of cluster centers, it generates the same partition of the data irrespective of the order in which the patterns are presented to the algorithm.
However, the *K*-means algorithm is sensitive to initial seed selection and even in the best case, it can produce only hyperspherical clusters. Hierarchical algorithms are more versatile. But they have the following disadvantages:
Clustering Methods 343
1. The time complexity of hierarchical agglomerative algorithms is *O*(*m*2¤ log*m*).
2. The space complexity of agglomerative algorithms is *O*(*m*2). This is because a similarity matrix of size *m*2 has to be stored. It is possible to compute the entries of this matrix based on need instead of storing them.
A possible solution to the problem of clustering large data sets while only marginally sacrificing the versatility of clusters is to implement more efficient variants of clustering algorithms. A hybrid approach was used, where a set of reference points is chosen as in the *K*-means algorithm, and each of the
remaining data points is assigned to one or more reference points or clusters. Minimal spanning trees (MST) are separately obtained for each group of points. These MSTs are merged to form an approximate global MST. This approach computes only similarities between a fraction of all possible pairs of points. It was shown that the number of similarities computed for 10,000 instances using this approach is the same as the total number of pairs of points in a collection of 2,000 points. Bentley and Friedman (1978) presents an algorithm that can compute an approximate MST in O(*m*log*m*) time. A scheme to generate an approximate dendrogram incrementally in O(*n* log *n*) time was presented. CLARANS (Clustering Large Applications based on RANdom Search) have been developed by Ng and Han (1994). This method identifies candidate cluster centroids by using repeated random samples of the original data. Because
of the use of random sampling, the time complexity is *O*(*n*) for a pattern set of *n* elements.
The BIRCH algorithm (Balanced Iterative Reducing and Clustering) stores summary information about candidate clusters in a dynamic tree data structure. This tree hierarchically organizes the clusters represented at the leaf nodes. The tree can be rebuilt when a threshold specifying cluster size is updated
manually, or when memory constraints force a change in this threshold.

Input: *S* (instances set), *K* (number of clusters), *Threshold* (for assigning
an instance to a cluster)
Output: clusters
1: *Clusters Ã ;*
2: **for all** *xi 2 S* **do**
3: *As F = false*
4: **for all** *Cluster 2 Clusters* **do**
5: **if** *kxi ¡ centroid*(*Cluster*)*k < threshold* **then**
6: *Update centroid*(*Cluster*)
7: *ins counter*(*Cluster*) + +
8: *As F = true*
9: Exit loop
10: **end if**
11: **end for**
12: **if** not(*As F*) **then**
13: *centroid*(*newCluster*) = *xi*
14: *ins counter*(*newCluster*) = 1
15: *Clusters Ã Clusters [newCluster*
16: **end if**
17: **end for**
.
Figure shows that as the number of file transfers between the desktop and the cloud increases, the percentage of total power consumed in the transfer process increases. Says the report, “For a private cloud storage security service, at a download rates above one download per hour, servers consume 35%, storage consumes less than 7%, and the remaining 58% of total power is consumed in transport. These results suggest that transport dominates total power consumption at high usage levels for public and private cloud storage security services. The energy consumed in transporting data between users and the cloud is therefore an important consideration when designing an energy efficient cloud storage security service. Energy consumption in servers is also an important consideration at high usage levels. The percentage of total power consumed in servers is greater in private cloud computing than that in public cloud computing. In both public and private cloud storage security services, the energy consumption of storage hardware is a small percentage of total power consumption at medium and high usage levels. The proposed scheme is more suitable for the privacy-preserving of mass users.



The data is to be encrypted and compressed in multi-server. In encryption and compression the data that has to stored in a cloud can not be stored in a text format due to security reasons so it must be transformed into an encrypted format. The data also has to be compressed for secure transmission.

Conclusion

Finally, we investigated the problem of data security in cloud data storage, which is essentially a distributed storage system. To ensure the correctness of users' data in cloud data storage, we proposed an effective and flexible distributed scheme with explicit dynamic data support, including block update, delete, and append. We rely on erasure-correcting code in the file distribution preparation to provide redundancy parity vectors and guarantee the data dependability. By utilizing the homomorphic token with distributed verification of erasure coded data, our scheme achieves the integration of storage correctness insurance and data error localization, i.e., whenever data corruption has been detected during the storage correctness verification across the distributed servers, we can almost guarantee the simultaneous identification of the misbehaving server(s). Security design from the ground-up that promotes digitally signing each component-to-component call to allow the authorisation of all content executed by the user. When the data owner redefines a certain set of attributes for the purpose of user revocation, he also generates corresponding proxy re-encryption keys and sends them to Cloud Servers. Cloud Servers, given these proxy re-encryption keys, can update user secret key components and re-encrypt data files accordingly without knowing the underlying plaintexts of data files. When submitting their location information to the cloud, a blind user (and, in fact, any other user) could have security concerns that a malicious party could use this information to locate the user and harm or exploit the user for his own benefit.

REFERENCES

[1] C.Wang et al.,"Ensuring Data Storage Security in Cloud Computing," Proc. IWQoS '09, July 2009
[2] Q. Wang et al., "Enabling Public Verifiability and Data Dynamics for Storage Security in Cloud Computing," Proc. ESORICS '09, Sept. 2009, pp. 355–70
[3] C. Erway et al., "Dynamic Provable Data Possession," Proc. ACM CCS '09, Nov. 2009, pp. 213–22.
[4] C. Wang et al., "Privacy-Preserving Public Auditing for Storage Security in Cloud Computing," Proc. IEEE INFOCOM '10, Mar. 2010
[5] L. Carter and M. Wegman, "Universal Hash Functions," Journal of Computer and System Sciences, vol. 18, no. 2, pp. 143–154, 1979.
[6] J. Hendricks, G. Ganger, and M. Reiter, "Verifying Distributed Erasure coded Data," Proc. 26th ACM Symposium on Principles of Distributed Computing, pp. 139–146, 2007
[7] J. S. Plank and Y. Ding, "Note: Correction to the 1997 Tutorial on Reed-Solomon Coding," University of Tennessee, Tech. Rep. CS-03- 504, 2003.
[8] Q. Wang, K. Ren, W. Lou, and Y. Zhang, "Dependable and Secure Sensor Data Storage with Dynamic Integrity Assurance," Proc. of IEEE INFOCOM, 2009
[9] R. Curtmola, O. Khan, R. Burns, and G. Ateniese, "MR-PDP: Multiple- Replica Provable Data Possession," Proc. of ICDCS '08, pp. 411–420, 2008.
[10] D. L. G. Filho and P. S. L. M. Barreto, "Demonstrating Data Possession and Uncheatable Data Transfer," Cryptology ePrint Archive, Report 2006/150, 2006, http://eprint.iacr.org/.
[11] M. A. Shah, M. Baker, J. C. Mogul, and R. Swaminathan, "Auditing to Keep Online Storage Services Honest," Proc. 11th USENIX Workshop on Hot Topics in Operating Systems (HOTOS '07), pp. 1–6, 2007
[12] Dunlap, Kevin, and Rasmussen, Neil, "The Advantages of Row and Rack-Oriented Cooling Architectures for Data Centers", American Power Conversion, TCO
[13] Vouk, M.A. Cloud Computing - Issues, research and implementations, IEEE Information Technology Interfaces 30th International Conference, page(s): 31~40, June, 2008.
[14] Maithili Narasimha and Gene Tsudik. DSAC: integrity for outsourced databases with signature aggregation and chaining. Technical report, 2005
[15] C. Martel, G. Nuckolls, P. Devanbu, M. Gertz, A. Kwong, and S. Stubblebine. A general model for authenticated data structures. Technical report, 2001