Document Clustering for Forensic Analysis: An Approach for Improving Computer Inspection

Luís Filipe da Cruz Nassif and Eduardo Raul Hruschka

***Abstract*—In computer forensic analysis, hundreds of thousands of files are usually examined. Much of the data in those files consists of unstructured text, whose analysis by computer examiners is difficult to be performed. In this context, automated methods of analysis are of great interest. In particular, algorithms for clustering documents can facilitate the discovery of new and useful knowledge from the documents under analysis. We present an approach that applies document clustering algorithms to forensic analysis of computers seized in police investigations. We illustrate the proposed approach by carrying out extensive experimentation with six well-known clustering algorithms (K-means, K-medoids, Single Link, Complete Link, Average Link, and CSPA) applied to five real-world datasets obtained from computers seized in real-world investigations. Experiments have been performed with different combinations of parameters, resulting in 16 different instantiations of algorithms. In addition, two relative validity indexes were used to automatically estimate the number of clusters. Related studies in the literature are significantly more limited than our study. Our experiments show that the Average Link and Complete Link algorithms provide the best results for our application domain. If suitably initialized, partitional algorithms (K-means and K-medoids) can also yield to very good results. Finally, we also present and discuss several practical results that can be useful for researchers and practitioners of forensic computing.**

***Index Terms*—Clustering, forensic computing, text mining.**

I. Introduction

IT IS estimated that the volume of data in the digital world increased from 161 hexabytes in 2006 to 988 hexabytes in 2010 [1]—about 18 times the amount of information present in all the books ever written—and it continues to grow exponentially. This large amount of data has a direct impact in *Computer Forensics*, which can be broadly defined as the discipline that combines elements of law and computer science to collect and analyze data from computer systems in a way that is admissible as evidence in a court of law. In our particular application domain, it usually involves examining hundreds of thousands of files per computer. This activity exceeds the expert's ability of analysis and interpretation of data. Therefore, methods for automated data analysis, like those widely used for machine learning and data mining, are of paramount importance. In particular, algorithms for pattern recognition from the information present in text documents are promising, as it will hopefully become evident later in the paper.

Clustering algorithms are typically used for exploratory data analysis, where there is little or no prior knowledge about the data [2], [3]. This is precisely the case in several applications of *Computer Forensics*, including the one addressed in our work. From a more technical viewpoint, our datasets consist of unlabeled objects—the classes or categories of documents that can be found are *a priori* unknown. Moreover, even assuming that labeled datasets could be available from previous analyses, there is almost no hope that the same classes (possibly learned earlier by a classifier in a supervised learning setting) would be still valid for the upcoming data, obtained from other computers and associated to different investigation processes. More precisely, it is likely that the new data sample would come from a different population. In this context, the use of clustering algorithms, which are capable of finding latent patterns from text documents found in seized computers, can enhance the analysis performed by the expert examiner.

The rationale behind clustering algorithms is that objects within a valid cluster are more similar to each other than they are to objects belonging to a different cluster [2], [3]. Thus, once a data partition has been induced from data, the expert examiner might initially focus on reviewing representative documents from the obtained set of clusters. Then, after this preliminary analysis, (s)he may eventually decide to scrutinize other documents from each cluster. By doing so, one can avoid the hard task of examining all the documents (individually) but, even if so desired, it still could be done.

In a more practical and realistic scenario, domain experts (e.g., forensic examiners) are scarce and have limited time available for performing examinations. Thus, it is reasonable to assume that, after finding a relevant document, the examiner could prioritize the analysis of other documents belonging to the cluster of interest, because it is likely that these are also relevant to the investigation. Such an approach, based on document clustering, can indeed improve the analysis of seized computers, as it will be discussed in more detail later.

Clustering algorithms have been studied for decades, and the literature on the subject is huge. Therefore, we decided to choose a set of (six) representative algorithms in order to show the potential of the proposed approach, namely: the partitional K-means [3] and K-medoids [4], the hierarchical

Manuscript received April 10, 2012; revised July 24, 2012; accepted September 11, 2012. Date of publication October 09, 2012; date of current version December 26, 2012. The work of L. F. da Cruz Nassif was supported by the Brazilian Federal Police Department. The work of E. R. Hruschka was supported by the Brazilian Research Agencies CNPq and FAPESP. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Teddy Furon.

L. F. da Cruz Nassif is with the Brazilian Federal Police Department, São Paulo 05038-090, Brazil, and also with the University of Brasilia (UnB), Brasilia, 70910-900, Brazil (E-mail: nassif.lfcn@dpf.gov.br).

E. R. Hruschka is with the University of São Paulo (USP) at São Carlos, São Carlos 13566-590, Brazil, and also with the University of Texas at Austin, Austin, TX 78731 USA (e-mail: erh@icmc.usp.br).

Digital Object Identifier 10.1109/TIFS.2012.2223679

TABLE I
SUMMARY OF ALGORITHMS AND THEIR PARAMETERS

Acronym	Algorithm	Attributes	Distance	Initialization	K-estimate
Kms	*K-means*	Cont. (all)	Cosine	Random	Simp. Sil.
Kms100	*K-means*	100 > TV	Cosine	Random	Simp. Sil.
Kms100*	*K-means*	100 > TV	Cosine	[18]	Simp. Sil.
KmsT100*	*K-means*	100 > TV	Cosine	[18]	Silhouette
KmsS	*K-means*	Cont. (all)	Cosine	Random	Rec. Sil.
Kms100S	*K-means*	100 > TV	Cosine	Random	Rec. Sil.
Kmd100	*K-medoids*	100 > TV	Cosine	Random	Silhouette
Kmd100*	*K-medoids*	100 > TV	Cosine	[18]	Silhouette
KmdLev	*K-medoids*	Name	Lev.	Random	Silhouette
KmdLevS	*K-medoids*	Name	Lev.	Random	Rec. Sil.
AL100	*AverageLink*	100 > TV	Cosine	-	Silhouette
CL100	*CompleteLink*	100 > TV	Cosine	-	Silhouette
SL100	*SingleLink*	100 > TV	Cosine	-	Silhouette
NC	CSPA	Name, Cont. (all)	CSPA	Random	Simp. Sil.
NC100	CSPA	Name,100 > TV	CSPA	Random	Simp. Sil.
E100	CSPA	Cont.100 random	CSPA	Random	Simp. Sil.

100 > TV: 100 attributes (words) that have the greatest variance over the documents
Cont. 100 random: 100 randomly chosen atributes from document content
Cont. (all): all features from document content
Lev.: Levenshtein distance
Simp. Sil.: Simplified Silhpuette
Rec. Sil.: "Recursive" Silhouette
*: Initialization on distant objects
Name: file name

Single/Complete/Average Link [5], and the cluster ensemble algorithm known as CSPA [6]. These algorithms were run with different combinations of their parameters, resulting in sixteen different algorithmic instantiations, as shown in Table I. Thus, as a contribution of our work, we compare their relative performances on the studied application domain—using five real-world investigation cases conducted by the Brazilian Federal Police Department. In order to make the comparative analysis of the algorithms more realistic, two relative validity indexes (Silhouette [4] and its simplified version [7]) have been used to estimate the number of clusters automatically from data.

It is well-known that the number of clusters is a critical parameter of many algorithms and it is usually *a priori* unknown. As far as we know, however, the automatic estimation of the number of clusters has not been investigated in the *Computer Forensics* literature. Actually, we could not even locate one work that is reasonably close in its application domain and that reports the use of algorithms capable of estimating the number of clusters. Perhaps even more surprising is the lack of studies on hierarchical clustering algorithms, which date back to the sixties. Our study considers such classical algorithms, as well as recent developments in clustering, such as the use of consensus partitions [6]. The present paper extends our previous work [23], where nine different instantiations of algorithms were analyzed. As previously mentioned, in our current work we employ sixteen instantiations of algorithms. In addition, we provide more insightful quantitative and qualitative analyses of their experimental results in our application domain.

The remainder of this paper is organized as follows. Section II presents related work. Section III briefly addresses the adopted clustering algorithms and preprocessing steps. Section IV reports our experimental results, and Section V addresses some limitations of our study. Finally, Section VI concludes the paper.

II. RELATED WORK

There are only a few studies reporting the use of clustering algorithms in the *Computer Forensics* field. Essentially, most of the studies describe the use of classic algorithms for clustering data—e.g., Expectation-Maximization (EM) for unsupervised learning of Gaussian Mixture Models, K-means, Fuzzy C-means (FCM), and Self-Organizing Maps (SOM). These algorithms have well-known properties and are widely used in practice. For instance, K-means and FCM can be seen as particular cases of EM [21]. Algorithms like SOM [22], in their turn, generally have inductive biases similar to K-means, but are usually less computationally efficient.

In [8], SOM-based algorithms were used for clustering files with the aim of making the decision-making process performed by the examiners more efficient. The files were clustered by taking into account their creation dates/times and their extensions. This kind of algorithm has also been used in [9] in order to cluster the results from keyword searches. The underlying assumption is that the clustered results can increase the information retrieval efficiency, because it would not be necessary to review all the documents found by the user anymore.

An integrated environment for mining e-mails for forensic analysis, using classification and clustering algorithms, was presented in [10]. In a related application domain, e-mails are grouped by using lexical, syntactic, structural, and domain-specific features [11]. Three clustering algorithms (K-means, Bisecting K-means and EM) were used. The problem of clustering e-mails for forensic analysis was also addressed in [12], where a Kernel-based variant of K-means was applied. The obtained results were analyzed subjectively, and the authors concluded that they are interesting and useful from an investigation perspective. More recently [13], a FCM-based method for mining association rules from forensic data was described.

The literature on *Computer Forensics* only reports the use of algorithms that assume that the number of clusters is known and fixed *a priori* by the user. Aimed at relaxing this assumption, which is often unrealistic in practical applications, a common approach in other domains involves estimating the number of clusters from data. Essentially, one induces different data partitions (with different numbers of clusters) and then assesses them with a relative validity index in order to estimate the best value for the number of clusters [2], [3], [14]. This work makes use of such methods, thus potentially facilitating the work of the expert examiner—who in practice would hardly know the number of clusters *a priori*.

III. CLUSTERING ALGORITHMS AND PREPROCESSING

A. Pre-Processing Steps

Before running clustering algorithms on text datasets, we performed some preprocessing steps. In particular, *stopwords* (prepositions, pronouns, articles, and irrelevant document metadata) have been removed. Also, the Snowball *stemming* algorithm for Portuguese words has been used. Then, we adopted a traditional statistical approach for text mining, in which documents are represented in a vector space model [15]. In this model, each document is represented by a vector containing the frequencies of occurrences of words, which are defined as delimited alphabetic strings, whose number of characters is between 4 and 25. We also used a dimensionality reduction technique known as Term Variance (TV) [16] that

can increase both the effectiveness and efficiency of clustering algorithms. TV selects a number of attributes (in our case 100 words) that have the greatest variances over the documents. In order to compute distances between documents, two measures have been used, namely: cosine-based distance [15] and Levenshtein-based distance [17]. The later has been used to calculate distances between file (document) names only.

B. Estimating the Number of Clusters From Data

In order to estimate the number of clusters, a widely used approach consists of getting a set of data partitions with different numbers of clusters and then selecting that particular partition that provides the best result according to a specific quality criterion (e.g., a relative validity index [2]–[5]). Such a set of partitions may result directly from a hierarchical clustering dendrogram or, alternatively, from multiple runs of a partitional algorithm (e.g., K-means) starting from different numbers and initial positions of the cluster prototypes (e.g., see [14] and references therein).

For the moment, let us assume that a set of data partitions with different numbers of clusters is available, from which we want to choose the best one—according to some relative validity criterion. Note that, by choosing such a data partition, we are performing model selection and, as an intrinsic part of this process, we are also estimating the number of clusters.

A widely used relative validity index is the so-called *silhouette* [4], which has also been adopted as a component of the algorithms employed in our work. Therefore, it is helpful to define it even before we address the clustering algorithms used in our study.

Let us consider an object i belonging to cluster $\mathbf{A}$. The average dissimilarity of i to all other objects of $\mathbf{A}$ is denoted by $a(i)$. Now let us take into account cluster $\mathbf{C}$. The average dissimilarity of i to all objects of $\mathbf{C}$ will be called $d(i, \mathbf{C})$. After computing $d(i, \mathbf{C})$ for all clusters $\mathbf{C} \neq \mathbf{A}$, the smallest one is selected, i.e., $b(i) = \min d(i, \mathbf{C}), \mathbf{C} \neq \mathbf{A}$. This value represents the dissimilarity of i to its neighbor cluster, and the silhouette for a give object, $s(i)$, is given by:

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}} \tag{1}$$

It can be verified that $-1 \leq s(i) \leq 1$. Thus, the higher $s(i)$ the better the assignment of object i to a given cluster. In addition, if $s(i)$ is equal to zero, then it is not clear whether the object should have been assigned to its current cluster or to a neighboring one [4]. Finally, if cluster $\mathbf{A}$ is a singleton, then $s(i)$ is not defined and the most neutral choice is to set $s(i) = 0$ [4]. Once we have computed $s(i)$ over $i = 1, 2, \ldots, N$, where N is the number of objects in the dataset, we take the average over these values, and the resulting value is then a quantitative measure of the data partition in hand. Thus, the best clustering corresponds to the data partition that has the maximum average silhouette.

The average silhouette just addressed depends on the computation of all distances among all objects. In order to come up with a more computationally efficient criterion, called *simplified* silhouette, one can compute only the distances among the objects and the centroids of the clusters. The term $a(i)$ of (1) now corresponds to the dissimilarity of object i to its corresponding cluster ($\mathbf{A}$) centroid. Thus, it is necessary to compute only one distance to get the $a(i)$ value, instead of calculating all the distances between i and the other objects of $\mathbf{A}$. Similarly, instead of computing $d(i, \mathbf{C})$ as the average dissimilarity of i to all objects of $\mathbf{C}$, $\mathbf{C} \neq \mathbf{A}$, we can now compute the distances between i and the centroid of $\mathbf{C}$.

Note that the computation of the original silhouette [4], as well as of its simplified version [7], [14], depends only on the achieved partition and not on the adopted clustering algorithm. Thus, these silhouettes can be applied to assess partitions (taking into account the number of clusters) obtained by several clustering algorithms, as the ones employed in our study and addressed in the sequel.

C. Clustering Algorithms

The clustering algorithms adopted in our study—the partitional K-means [2] and K-medoids [4], the hierarchical Single/Complete/Average Link [5], and the cluster ensemble based algorithm known as CSPA [6]—are popular in the machine learning and data mining fields, and therefore they have been used in our study. Nevertheless, some of our choices regarding their use deserve further comments. For instance, K-medoids [4] is similar to K-means. However, instead of computing centroids, it uses medoids, which are the representative objects of the clusters. This property makes it particularly interesting for applications in which (i) centroids cannot be computed; and (ii) distances between pairs of objects are available, as for computing dissimilarities between names of documents with the Levenshtein distance [17].

Considering the partitional algorithms, it is widely known that both K-means and K-medoids are sensitive to initialization and usually converge to solutions that represent local minima. Trying to minimize these problems, we used a nonrandom initialization in which distant objects from each other are chosen as starting prototypes [18]. Unlike the partitional algorithms such as K-means/medoids, hierarchical algorithms such as Single/Complete/Average Link provide a hierarchical set of nested partitions [3], usually represented in the form of a dendrogram, from which the *best* number of clusters can be estimated. In particular, one can assess the quality of every partition represented by the dendrogram, subsequently choosing the one that provides the best results [14].

The CSPA algorithm [6] essentially finds a consensus clustering from a cluster ensemble formed by a set of different data partitions. More precisely, after applying c clustering algorithms to the data, a similarity (coassociation) matrix [19] is computed. Each element of this matrix represents pair-wise similarities between objects. The similarity between two objects is simply the fraction of the c clustering solutions in which those two objects lie in the same cluster. Later, this similarity measure is used by a clustering algorithm that can process a proximity matrix—e.g., K-medoids—to produce the final consensus clustering. The sets of data partitions (clusterings) were generated in two different ways: (a) by running K-means 100 times with different subsets of attributes (in this case CSPA processes 100 data partitions); and (b) by using only two data partitions, namely: one obtained by K-medoids from the dissimilarities between the file

names, and another partition achieved with K-means from the vector space model. In this case, each partition can have different weights, which have been varied between 0 and 1 (in increments of 0.1 and keeping their sum equals to 1).

For the hierarchical algorithms (Single/Complete/Average Link), we simply run them and then assess every partition from the resulting dendrogram by means of the silhouette [4]. Then, the best partition (elected according to the relative validity index) is taken as the result of the clustering process. For each partitional algorithm (K-means/medoids), we execute it repeatedly for an increasing number of clusters. For each value of K, a number of partitions achieved from different initializations are assessed in order to choose the best value of K and its corresponding data partition, using the Silhouette [4] and its simplified version [7], which showed good results in [14] and is more computationally efficient. In our experiments, we assessed all possible values of K in the interval $[2, N]$, where N is the number of objects to be clustered.

D. Dealing With Outliers

We assess a simple approach to remove *outliers*. This approach makes recursive use of the *silhouette*. Fundamentally, if the best partition chosen by the silhouette has singletons (i.e., clusters formed by a single object only), these are removed. Then, the clustering process is repeated over and over again—until a partition without singletons is found. At the end of the process, all singletons are incorporated into the resulting data partition (for evaluation purposes) as single clusters. Table I summarizes the clustering algorithms used in our work and their main characteristics.

IV. Experimental Evaluation

A. Datasets

Sets of documents that appear in computer forensic analysis applications are quite diversified. In particular, any kind of content that is digitally compliant can be subject to investigation. In the datasets assessed in our study, for instance, there are textual documents written in different languages (Portuguese and English). Such documents have been originally created in different file formats, and some of them have been corrupted or are actually incomplete in the sense that they have been (partially) recovered from deleted data.

We used five datasets obtained from real-world investigation cases conducted by the Brazilian Federal Police Department. Each dataset was obtained from a different hard drive, being selected all the nonduplicate documents with extensions "doc", "docx", and "odt". Subsequently, those documents were converted into plain text format and preprocessed as described in Section III-A. The obtained data partitions were evaluated by taking into account that we have a *reference partition* (ground truth) for every dataset. Such reference partitions have been provided by an expert examiner from the Brazilian Federal Police Department, who previously inspected every document from our collections. The datasets contain varying amounts of documents (N), groups (K), attributes (D), singletons (S), and number of documents per group (#), as reported in Table II.

TABLE II
Dataset Characteristics[1]

Dataset	N	K	D	S	# Largest cluster
A	37	23	1744	12	3
B	111	49	7894	28	12
C	68	40	2699	24	8
D	74	38	5095	26	17
E	131	51	4861	31	44

B. Evaluation Measure

From a scientific perspective, the use of *reference partitions* for evaluating data clustering algorithms is considered a principled approach. In controlled experimental settings, reference partitions are usually obtained from data generated synthetically according to some probability distributions. From a practical standpoint, reference partitions are usually obtained in a different way, but they are still employed to choose a particular clustering algorithm that is more appropriate for a given application, or to calibrate its parameters. In our case, reference partitions were constructed by a domain expert and reflects the expectations that (s)he has about the clusters that should be found in the datasets. In this sense, the evaluation method that we used to assess the obtained data partitions is based on the Adjusted Rand Index [3], [20], which measures the agreement between a partition $\boldsymbol{P}$, obtained from running a clustering algorithm, and the reference partition $\boldsymbol{R}$ given by the expert examiner. More specifically, $\mathrm{ARI} \in [0, 1]$ and the greater its value the better the agreement between $\boldsymbol{P}$ and $\boldsymbol{R}$.

C. Results and Discussions

Table III summarizes the obtained ARI results for the algorithms listed in Table I. In general, AL100 (Average Link algorithm using the 100 terms with the greatest variances, cosine-based similarity, and silhouette criterion) provided the best results with respect to both the average and the standard deviation, thus suggesting great accuracy and stability. Note also that an ARI value close to 1.00 indicates that the respective partition is very consistent with the reference partition—this is precisely the case here. In this table, we only report the best obtained results for the algorithms that search for a consensus partition between file name and content (NC100 and NC)—i.e., partitions whose weights for name/content resulted in the greatest ARI value. The ARI values for CL100 are similar to those found by AL100. Single Link (SL100), by its turn, presented worse results than its hierarchical counterparts—especially for datasets A and B. This result can be explained by the presence of *outliers*, whose chain effect is known to impact Single Link performance [2].

The results achieved by $\mathrm{Kmd100}^*$ and $\mathrm{KmsT100}^*$ were also very good and competitive to the best hierarchical algorithms (AL100 and CL100). We note that, as expected, a

[1]The relative frequencies of words per document for the employed datasets are available upon request. Due to privacy issues, we cannot provide detailed information about the document names and their respective bag of words.

TABLE III
ADJUSTED RAND INDEX (ARI) RESULTS

Alg./Dataset	A	B	C	D	E	Mean	σ
AL100	0.94	0.83	0.89	0.99	0.90	0.91	0.06
CL100	0.94	0.76	0.89	0.98	0.90	0.89	0.08
KmsT100*	0.81	0.76	0.89	0.97	0.94	0.88	0.09
Kmd100*	0.81	0.76	0.89	0.96	0.93	0.87	0.08
SL100	0.54	0.63	0.90	0.98	0.88	0.79	0.19
NC100	0.66	0.64	0.78	0.74	0.72	0.71	0.06
Kms	0.61	0.60	0.69	0.79	0.84	0.71	0.11
NC	0.61	0.60	0.69	0.79	0.84	0.71	0.11
Kms100*	0.53	0.63	0.63	0.68	0.93	0.68	0.15
Kmd100	0.81	0.58	0.72	0.25	0.79	0.63	0.23
Kms100	0.64	0.64	0.78	0.29	0.72	0.62	0.19
KmsS	0.47	0.11	0.75	0.80	0.82	0.59	0.30
Kms100S	0.60	0.54	0.74	0.20	0.69	0.55	0.21
E100	0.61	0.10	0.29	0.76	0.08	0.37	0.31
KmdLevS	0.62	0.23	0.37	0.55	0.05	0.36	0.23
KmdLev	0.46	0.16	0.32	0.74	0.08	0.35	0.26

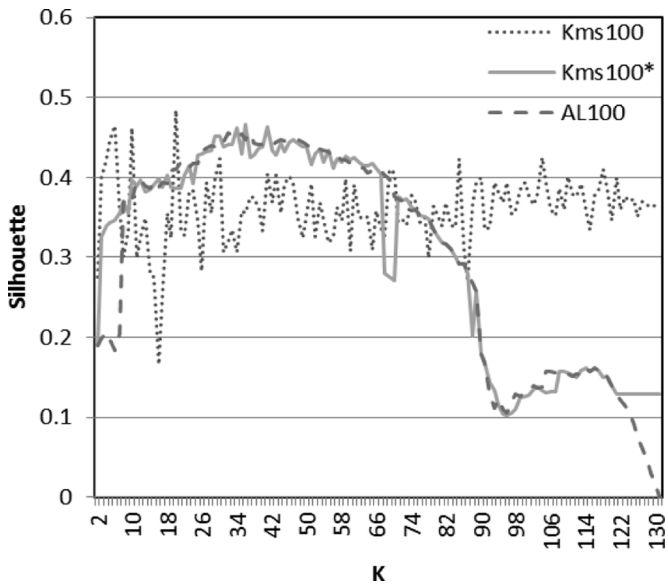

Fig. 1. Silhouettes for Kms100, AL100, and Kms100* (dataset E).

good initialization procedure such as the one described in [18] provides the best results. Particularly, the initialization on distant objects can minimize the K-means/medoids problems with respect to local minima. To illustrate this aspect, Fig. 1 shows the Silhouette values as a function of K for three algorithms (Kms100, $\mathrm{Kms100}^*$, and AL100). One can observe that Kms100 (with random initialization of prototypes), presents more local maxima for the Silhouette (recall that these were obtained from local minima of K-means), yielding to less stable results. Conversely, $\mathrm{Kms100}^*$ (initialization on distant objects [18]) has fewer local maxima, being more stable. This trend has also been observed in the other datasets. Surprisingly, $\mathrm{Kms100}^*$ has curves similar to those of AL100, especially for higher values of K. This fact can be explained, in part, because both algorithms tend to separate outliers.

It can also be observed that $\mathrm{Kms100}^*$ got slightly better results than its variant with random initialization of prototypes (Kms100). However, the ARI values for $\mathrm{Kms100}^*$, which uses

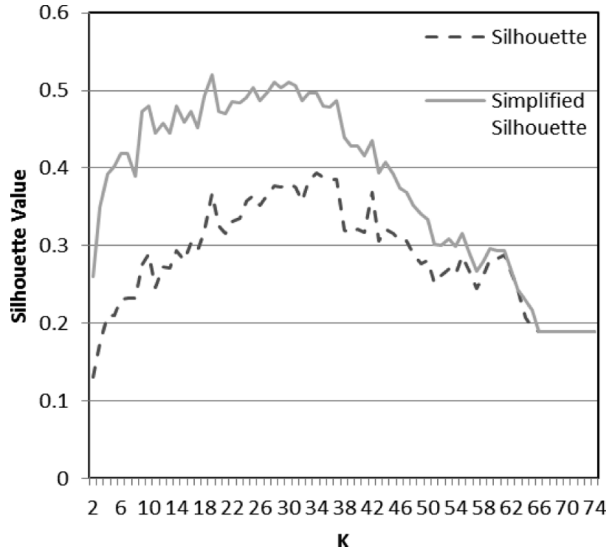

Fig. 2. Silhouettes—$\mathrm{Kms100}^*$ (Dataset D).

the simplified Silhouette to estimate K, are worse than those obtained for $\mathrm{KmsT100}^*$, $\mathrm{Kmd100}^*$, and the hierarchical algorithms. Recall that these algorithms use the traditional version of the silhouette. This observation suggests that the simplified silhouette provided worse estimates for the values of K. This can be explained from the fact that the simplified silhouette essentially computes distances between objects and centroids, whereas the traditional silhouette takes all the pair-wise distances between objects into account in order to estimate K. In other words, there is some information loss in the computation of the simplified silhouette, which indeed provided smaller values for K than both the silhouette and the reference partition in four out of five datasets. We also point out that both the silhouette and its simplified version estimate the number of clusters by taking into account two concepts: cluster compactness (average intracluster dissimilarity) and cluster separability (inter-cluster dissimilarity). Both are materialized by computing average distances. We observed that the average distance from a given object to all the objects of a cluster tends to be greater than the distance of that object to the cluster's centroid. Moreover, such a difference tends to be greater when computing intracluster distances (compactness)—compared to calculating intercluster distances (separability). In other words, it is more likely that the simplified silhouette underestimates the intracluster distances. As a consequence, in the trade-off relationship between cluster compactness and cluster separability, the simplified silhouette value tends to be higher than the one find by silhouette for a given data partition. Also, the higher the number of clusters the better the simplified silhouette's estimate for the compactness, approaching the value estimated by silhouette and becoming equal in the extreme case $(K = N)$, as shown in Fig. 2 (Dataset D). Similar figures have been observed for the other datasets. Thus, the simplified silhouette favors lower values for K than the silhouette.

The use of the file names to compute dissimilarities between documents in principle seemed to be interesting, because one usually chooses meaningful file names. However, it turns out that, empirically, the use of only the file names to compute

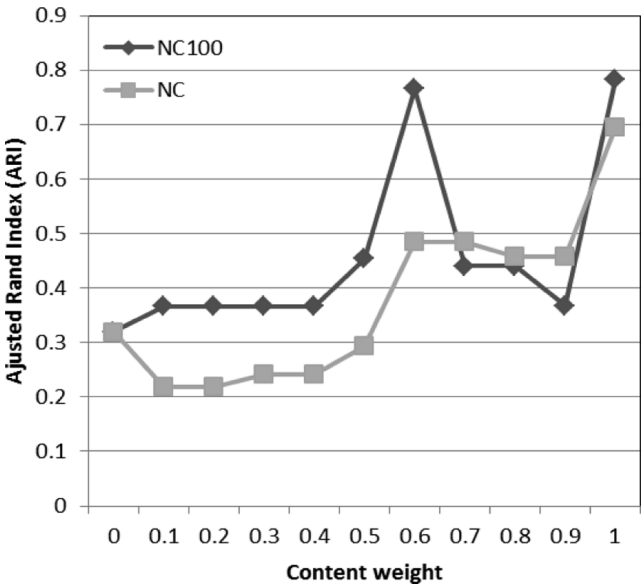

Fig. 3. ARI values—NC and NC100 algorithms (Dataset C).

TABLE IV
EXAMPLE OF THE INFORMATION FOUND IN THE CLUSTERS

Cluster	Information
C_1	3 blank documents
C_2	4 financial transactions
C_3	2 maternity payments
C_4	2 grocery lists
C_5	1 foreign exchange transaction warning 1 list of documents for registration information
C_6	2 documents from foreign exchange operations
C_7	1 registration form from a brokerage company 1 contract template from the broker
C_8	1 investment club status 1 agreement for joining the club
C_9	2 models for handling cash greater than R$ 100k
C_{10}	8 receipts of foreign exchange insurance transactions
C_{11}	2 warnings about foreign brokerage business hours
C_{12}	3 label designs of a brokerage company
C_{13}	1 notice about working hours 1 check receipt
C_{14}	2 daily reports from buying/selling exchanges
C_{15}	2 sample documents from office application

the dissimilarity between documents did not bring good results in general—e.g., see results for KmdLev and KmdLevS in Table III. On the other hand, these results may not be considered surprising because the name of the file provides less information than the file content. However, there is an exception to this general behavior that can be observed from the relatively good results of these algorithms on Dataset D, thus suggesting that the file name, although less informative than the file content, can help the clustering process. In addition, let us now analyze Fig. 3 (for Dataset C), which shows the ARI values for the algorithms that search for a consensus partition between those formed by name and content. The ARI values are shown for different weights that capture the (imposed) relative importance of *name* and *content*. Some particular combinations of values for the weights, like when the content weight is comprised between 0.1 and 0.5, provided worse results compared to the standalone use of either name or content, thus suggesting that mixing information of different nature may be prejudicial to the clustering process. However, one can see that NC100 shows a (secondary) peak of the ARI value (for content weight equals to 0.6). Although the primary peak suggests using only the data partition found from content information, it seems that adding information about the file name may indeed be useful. To wrap up this discussion, the use of the file name can help the clustering process, but it seems to be highly data dependent.

Let us now focus on the performance of E100. Recall from Table I that this is a particular instantiation of the CSPA [6] algorithm. More specifically, it obtains a consensus partition from a set of partitions generated by K-means—by randomly selecting 100 different attributes. Because of both the high sparsity of the data matrix (common to text datasets) and the random attribute selection, many documents are represented by very sparse vectors. Consequently, such documents have been grouped into a single cluster, whose centroid is also very sparse and with component values close to zero. Such centroids induce misleading partitions, which are the inputs for the computation of the coassociation matrix. Thus, a misleading consensus clustering is obtained. Therefore, the choice of random sets of attributes to generate partitions for consensus clustering algorithms seems to be an inappropriate approach for such text data.

Considering the algorithms that recursively apply the Silhouette for removing singletons (KmsS and Kms100S), Table III shows that their results are relatively worse when compared to the related versions that do not remove singletons (Kms and Kms100). However, KmdLevS, which is based on the similarities between file names, presented similar results to those found by its related version that does not remove singletons (KmdLev). In principle, one could expect that the removal of outliers, identified from carefully analyzing the singletons, could yield to better clustering results. However, we have not observed this potentially good property in our experiments and, as expected, this aspect is rather data-dependent. As such, these algorithms may be potentially helpful to deal with other datasets.

As far as the adopted dimensionality reduction technique is concerned—Term Variance (TV) [16]—we observed that the selection of the 100 attributes (words) that have the greatest variance over the documents provided best results than using all the attributes in three out of five datasets (see Table III). Compared to Kms and KmsS, the worse results obtained from feature selection by Kms100 and Kms100S, especially in the dataset D, are likely due to k-means convergence to local optima from bad initialization. By considering all the results obtained from feature selection, we believe that it should be further studied mainly because of the potentially advantageous computational efficiency gains.

Finally, from a practical viewpoint, a variety of relevant findings emerged from our study. It is worth stressing that, for all the investigated datasets, the best data partitions are formed by clusters containing either relevant or irrelevant documents. For example, in dataset C, the algorithm AL100 obtained a data partition formed by some singletons and by other 15 clusters $(C_1, C_2, \ldots, C_{15})$ whose information are listed in Table IV.

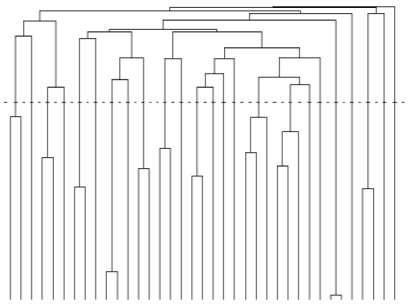

Fig. 4. Dendrogram obtained by AL100—Dataset A.

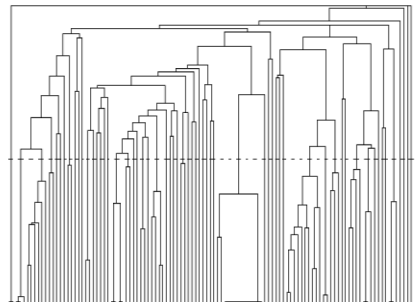

Fig. 5. Dendrogram obtained by AL100—Dataset B.

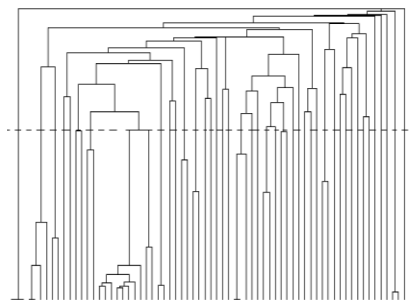

Fig. 6. Dendrogram obtained by AL100—Dataset C.

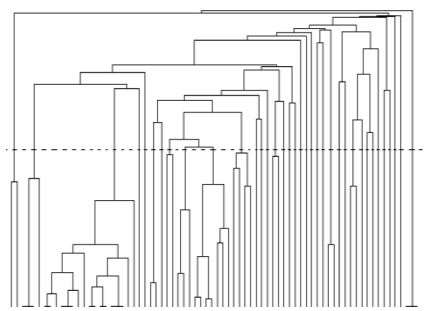

Fig. 7. Dendrogram obtained by AL100—Dataset D.

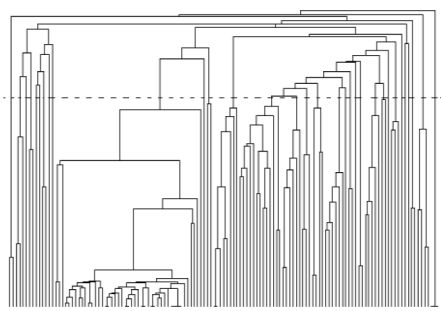

Fig. 8. Dendrogram obtained by AL100—Dataset E.

For obvious reasons, we cannot reveal more detailed, confidential information. However, we can mention that, on this real-world financial crime investigation, clusters of relevant documents have been obtained, such as C_{10}, C_{14}, and C_2, which contain financial or exchange transaction information. Also, other obtained clusters have only irrelevant documents—e.g., C_{12}, which contains label designs. These clusters of either relevant or irrelevant documents can help computer examiners to efficiently focus on the most relevant documents without having to inspect them all. In summary, document clustering has great potential to be useful for computer inspection.

As a final remark, a desirable feature of hierarchical algorithms that make them particularly interesting for expert examiners is the summarized view of the dataset in the form of a dendrogram, which is a tree diagram that illustrates the arrangement of the clusters. The root node of the dendrogram represents the whole data set (as a single cluster formed by all objects), and each leaf node represents a particular object (as a singleton cluster). The intermediate nodes, by their turn, represent clusters merged hierarchically. The height of an intermediate node is proportional to the distance between the clusters it merges. This representation provides very informative descriptions and visualization for the potential data clustering structures [5], thus being helpful tools for forensic examiners that analyze textual documents from seized computers. As we already discussed, the ultimate clustering results can be obtained by cutting the dendrogram at different levels—e.g., by using a relative validity criterion like the silhouette.

For the sake of illustration, Figs. 4–8 show examples of dendrograms obtained by AL100. The dendrograms were cut horizontally at the height corresponding to the number of clusters estimated by the silhouette (dashed line). Roughly speaking, subtrees with low height and large width represent both cohesive and large clusters. These clusters are good candidates for a starting point inspection. Moreover, the forensic examiner can, after finding a cluster of relevant documents, inspect the cluster most similar to the one just found, because it is likely that it is also a relevant cluster. This can be done by taking advantage of the tree diagram.

V. Limitations

It is well-known that the success of any clustering algorithm is data dependent, but for the assessed datasets some of our adaptations of existing algorithms have shown to be good enough. Scalability may be an issue, however. In order to deal with this issue, a number of sampling and other techniques can be used—e.g., see [24]–[29]. Also, algorithms such as bisecting k-means and related approaches can be used. These algorithms can also induce dendrograms, and have a similar inductive bias with respect to hierarchical methods considered in our work. More precisely, and aimed at circumventing computational difficulties, partitional clustering algorithms can be used to compute a hierarchical clustering solution by using repeated cluster bisectioning approaches. For instance, bisecting k-means has relatively low computational requirements, i.e.,

it is $O(N \cdot \log N)$, versus the overall time complexity of $O(N^2 \cdot \log N)$ for agglomerative methods. Since the inductive biases of bisecting k-means and the hierarchical algorithms used in our work are similar, we believe that, if the number of documents is prohibitively high for running agglomerative algorithms, then bisecting k-means and related approaches can be used.

Considering the computational cost of estimating the number of clusters, the silhouette proposed in [4] depends on the computation of all distances between objects, leading to an estimated computational cost of $O(N^2 \cdot D)$, where N is the number of objects in the dataset and D is the number of attributes, respectively. As already mentioned in the paper, to alleviate this potential difficulty, especially when dealing with very large datasets, a simplified silhouette [7] can be used. The simplified silhouette is based on the computation of distances between objects and cluster centroids, thus making it possible to reduce the computational cost from $O(N^2 \cdot D)$ to $O(k \cdot N \cdot D)$, where k, the number of clusters, is usually significantly less than N. It is also worth mentioning that there are several different relative validity criteria that can be used in place of the silhouettes adopted in our work. As discussed in [14], such criteria are endowed with particular features that may make each of them to outperform others in specific classes of problems. Also, they present different computational requirements. In this context, in practice one can try different criteria to estimate the number of clusters by taking into account both the quality of the obtained data partitions and the associated computational cost.

Finally, as a cautionary note, we would like to mention that, in practice, it is not always of paramount importance to have scalable methods. In our particular application scenario, there are no hard time constraints to get data partitions (like those present when analyzing streaming data with online algorithms). Instead, domain experts can usually spend months analyzing their data before reaching a conclusion.

VI. Final Remarks

We presented an approach that applies document clustering methods to forensic analysis of computers seized in police investigations. Also, we reported and discussed several practical results that can be very useful for researchers and practitioners of forensic computing. More specifically, in our experiments the hierarchical algorithms known as Average Link and Complete Link presented the best results. Despite their usually high computational costs, we have shown that they are particularly suitable for the studied application domain because the dendrograms that they provide offer summarized views of the documents being inspected, thus being helpful tools for forensic examiners that analyze textual documents from seized computers. As already observed in other application domains, dendrograms provide very informative descriptions and visualization capabilities of data clustering structures [5].

The partitional K-means and K-medoids algorithms also achieved good results when properly initialized. Considering the approaches for estimating the number of clusters, the relative validity criterion known as *silhouette* has shown to be more accurate than its (more computationally efficient) simplified version. In addition, some of our results suggest that using the file names along with the document content information may be useful for cluster ensemble algorithms. Most importantly, we observed that clustering algorithms indeed tend to induce clusters formed by either relevant or irrelevant documents, thus contributing to enhance the expert examiner's job. Furthermore, our evaluation of the proposed approach in five real-world applications show that it has the potential to speed up the computer inspection process.

Aimed at further leveraging the use of data clustering algorithms in similar applications, a promising venue for future work involves investigating automatic approaches for cluster labeling. The assignment of labels to clusters may enable the expert examiner to identify the semantic content of each cluster more quickly—eventually even before examining their contents. Finally, the study of algorithms that induce overlapping partitions (e.g., Fuzzy C-Means and Expectation-Maximization for Gaussian Mixture Models) is worth of investigation.

Acknowledgment

L. F. da Cruz Nassif would like to thank the Brazilian Federal Police Department for providing the datasets used in the experiments.

References

[1] J. F. Gantz, D. Reinsel, C. Chute, W. Schlichting, J. McArthur, S. Minton, I. Xheneti, A. Toncheva, and A. Manfrediz, "The expanding digital universe: A forecast of worldwide information growth through 2010," *Inf. Data*, vol. 1, pp. 1–21, 2007.

[2] B. S. Everitt, S. Landau, and M. Leese, *Cluster Analysis*. London, U.K.: Arnold, 2001.

[3] A. K. Jain and R. C. Dubes, *Algorithms for Clustering Data*. Englewood Cliffs, NJ: Prentice-Hall, 1988.

[4] L. Kaufman and P. Rousseeuw, *Finding Groups in Gata: An Introduction to Cluster Analysis*. Hoboken, NJ: Wiley-Interscience, 1990.

[5] R. Xu and D. C. Wunsch, II, *Clustering*. Hoboken, NJ: Wiley/IEEE Press, 2009.

[6] A. Strehl and J. Ghosh, "Cluster ensembles: A knowledge reuse framework for combining multiple partitions," *J. Mach. Learning Res.*, vol. 3, pp. 583–617, 2002.

[7] E. R. Hruschka, R. J. G. B. Campello, and L. N. de Castro, "Evolving clusters in gene-expression data," *Inf. Sci.*, vol. 176, pp. 1898–1927, 2006.

[8] B. K. L. Fei, J. H. P. Eloff, H. S. Venter, and M. S. Oliver, "Exploring forensic data with self-organizing maps," in *Proc. IFIP Int. Conf. Digital Forensics*, 2005, pp. 113–123.

[9] N. L. Beebe and J. G. Clark, "Digital forensic text string searching: Improving information retrieval effectiveness by thematically clustering search results," *Digital Investigation, Elsevier*, vol. 4, no. 1, pp. 49–54, 2007.

[10] R. Hadjidj, M. Debbabi, H. Lounis, F. Iqbal, A. Szporer, and D. Benredjem, "Towards an integrated e-mail forensic analysis framework," *Digital Investigation, Elsevier*, vol. 5, no. 3–4, pp. 124–137, 2009.

[11] F. Iqbal, H. Binsalleeh, B. C. M. Fung, and M. Debbabi, "Mining writeprints from anonymous e-mails for forensic investigation," *Digital Investigation, Elsevier*, vol. 7, no. 1–2, pp. 56–64, 2010.

[12] S. Decherchi, S. Tacconi, J. Redi, A. Leoncini, F. Sangiacomo, and R. Zunino, "Text clustering for digital forensics analysis," *Computat. Intell. Security Inf. Syst.*, vol. 63, pp. 29–36, 2009.

[13] K. Stoffel, P. Cotofrei, and D. Han, "Fuzzy methods for forensic data analysis," in *Proc. IEEE Int. Conf. Soft Computing and Pattern Recognition*, 2010, pp. 23–28.

[14] L. Vendramin, R. J. G. B. Campello, and E. R. Hruschka, "Relative clustering validity criteria: A comparative overview," *Statist. Anal. Data Mining*, vol. 3, pp. 209–235, 2010.

[15] G. Salton and C. Buckley, "Term weighting approaches in automatic text retrieval," *Inf. Process. Manage.*, vol. 24, no. 5, pp. 513–523, 1988.

[16] L. Liu, J. Kang, J. Yu, and Z. Wang, "A comparative study on unsupervised feature selection methods for text clustering," in *Proc. IEEE Int. Conf. Natural Language Processing and Knowledge Engineering*, 2005, pp. 597–601.
[17] V. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," *Soviet Physics Doklady*, vol. 10, pp. 707–710, 1966.
[18] B. Mirkin, *Clustering for Data Mining: A Data Recovery Approach*. London, U.K.: Chapman & Hall, 2005.
[19] A. L. N. Fred and A. K. Jain, "Combining multiple clusterings using evidence accumulation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 6, pp. 835–850, Jun. 2005.
[20] L. Hubert and P. Arabie, "Comparing partitions," *J. Classification*, vol. 2, pp. 193–218, 1985.
[21] C. M. Bishop, *Pattern Recognition and Machine Learning*. New York: Springer-Verlag, 2006.
[22] S. Haykin, *Neural Networks: A Comprehensive Foundation*. Englewood Cliffs, NJ: Prentice-Hall, 1998.
[23] L. F. Nassif and E. R. Hruschka, "Document clustering for forensic computing: An approach for improving computer inspection," in *Proc. Tenth Int. Conf. Machine Learning and Applications (ICMLA)*, 2011, vol. 1, pp. 265–268, IEEE Press.
[24] , Aggarwal, C. C. Charu, and C. X. Zhai, Eds., "Chapter 4: A Survey of Text Clustering Algorithms," in *Mining Text Data*. New York: Springer, 2012.
[25] Y. Zhao, G. Karypis, and U. M. Fayyad, "Hierarchical clustering algorithms for document datasets," *Data Min. Knowl. Discov.*, vol. 10, no. 2, pp. 141–168, 2005.
[26] Y. Zhao and G. Karypis, "Evaluation of hierarchical clustering algorithms for document datasets," in *Proc. CIKM*, 2002, pp. 515–524.
[27] S. Nassar, J. Sander, and C. Cheng, "Incremental and effective data summarization for dynamic hierarchical clustering," in *Proc. 2004 ACM SIGMOD Int. Conf. Management of Data (SIGMOD '04)*, 2004, pp. 467–478.
[28] K. Kishida, "High-speed rough clustering for very large document collections," *J. Amer. Soc. Inf. Sci.*, vol. 61, pp. 1092–1104, 2010, doi: 10.1002/asi.2131.
[29] Y. Loewenstein, E. Portugaly, M. Fromer, and M. Linial, "Effcient algorithms for exact hierarchical clustering of huge datasets: Tackling the entire protein space," *Bioinformatics*, vol. 24, no. 13, pp. i41–i49, 2008.

Luís Filipe da Cruz Nassif received the B.Sc. degree in computer engineering from the Military Institute of Engineering (IME), Brazil, in 2005, and the M.Sc. degree in electrical engineering from the University of Brasilia (UnB), Brazil, in 2011.

Since 2006, he has been working as a computer forensic examiner with the Brazilian Federal Police Department, São Paulo, Brazil. His main research interests are in computer forensics and data mining.

Eduardo Raul Hruschka received the Ph.D. degree in computational systems from COPPE/Federal University, Rio de Janeiro, Brazil, in 2001.

He is with the Computer Science Department (ICMC), University of São Paulo (USP) at São Carlos, Brazil. From 2010 to 2012, he was a visiting researcher at the University of Texas at Austin. His primary research interests are in data mining and machine learning. He has authored or coauthored more than 50 research publications in peer-reviewed reputed journals and conference proceedings.

Dr. Hruschka is associate editor of *Information Sciences* (Elsevier). He has also been a reviewer for several journals such as IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNTICS, IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, IEEE TRANSACTIONS ON FUZZY SYSTEMS, IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, as well as a member of the Program Committee of several international conferences such as IEEE ICDM and ACM KDD.