IJoFCS (2012) 1, 24-41

e-forensic PRESS

The International Journal of
FORENSIC COMPUTER SCIENCE

www.IJoFCS.org

DOI: 10.5769/J201201003 or http://dx.doi.org/10.5769/J201201003

Using ART1 Neural Networks for Clustering Computer Forensics Documents

Georger Rommel Ferreira de Araújo[(1)] and Célia Ghedini Ralha[(2)]

(1) Technical-Scientific Unit, Federal Police Department, Juazeiro do Norte, Brazil, rommel dot grfa at dpf dot gov dot br
(2) Computer Science Department, University of Brasília (UnB), Brasília, Brazil, ghedini at cic dot unb dot br

Abstract - Computer forensic text corpora are usually very heterogeneous and easily surpass the terabyte range. Classification methods should be an aid in the exploration of such corpora, but they do not help in the task of thematically grouping together documents. In this paper, we propose the use of Adaptive Resonance Theory (ART), applying the ART1 algorithm, to help in the task of thematically grouping together computer forensics documents into clusters. For the clustering approach we present the defined conceptual model and the software package implemented, in which a modified version of the ART1 algorithm was developed to improve the running time. Furthermore, real world forensic experiments were carried out to validate the model using a two-fold approach with a quantitative and a qualitative analysis method. The results demonstrate that our approach can generate good clusters when compared to the gold standard defined by domain area experts, with one clear advantage over other clustering methods (e.g. SOM and k-means) since there is no need to supply parameters beforehand such as the number of clusters.

Keywords - computer forensics; document clustering; artificial neural networks; adaptive reasonance theory; ART1 algorithm

1. INTRODUCTION

Computer forensics comes to the acquisition, preservation, analysis and presentation of electronic evidence. After evidence is seized or acquired, examination begins. A computer forensic examination is a procedure that is both broad and deep, performed by a specialist that is responsible and legally accountable for handling the evidence, keeping the chain of custody and writing a report with his findings. Depending on the size and complexity of data, an examination may take a long time to complete. Two techniques commonly used to speed up examinations are data reduction [2] and text indexing [20]. Data reduction consists of applying filters during pre-processing to exclude certain portions of data that are known to be safely ig-

Paper submitted on February 16th, 2012.

norable, usually by means of querying a hash database, while text indexing consists of building a text index so that computer forensic artifacts can be searched quickly after the pre-processing phase.

The current process of text string searching using computer forensic tools is basically the same: the forensic examiner inputs search strings and the application returns a number of search results, or hits, for the examiner to peruse. It is not uncommon for text searches to return hundreds to thousands of hits. All hits must be returned, because the application cannot distinguish the relevant ones from the irrelevant. The task of reviewing the hits is subjective in nature and belongs to the domain specialist – the examiner. Thus, it becomes critically important for the application to present hits in such a manner that the examiner may quickly and efficiently review them.

Unfortunately, current computer forensic applications employ a process not much more sophisticated than running the tool *grep* on the image file [3] and displaying a simplistic visualization of its output, and at most categorize hits by file type. As storage capacity of seized computers and media begins to surpass the terabyte range, the need for better ways of displaying and navigating search hits becomes even more apparent. Despite the importance of the research subject, relatively little research has been done to develop efficient searching mechanisms with adequate presentation methods for the exploration of computer forensic text corpora [4,5,6,7,8,9].

As stated, text is very important in forensic examinations. Computer forensic text corpora often consist of a very heterogeneous mix of artifacts, including but not limited to: office suite documents and spreadsheets, PDF files, email, web browser and instant messaging chat logs, and a huge list of text strings extracted from unallocated and slack space. It is very common to find all of these types of forensic artifacts in computers and discrete media seized by law enforcement. Finding, organizing and analyzing evidence from this diverse mix of artifacts often takes more time and effort than the resources available, especially considering the immense growth of storage capacity [1].

In this direction, this paper applies the Adaptive Resonance Theory (ART) with the ART1 algorithm (ART with binary input vectors) to thematically cluster digital documents returned from query tools used with forensic text corpora. Documents that would previously be presented in a disorganized and often long list are thematically clustered, giving the examiner a faster and effective way of obtaining a general picture of document content during forensic examinations. Our experimental results are expressive to validate our approach, achieving good agreement between the clustering solution processed with our software package and the gold standard defined by domain area experts.

The rest of this paper is organized as follows: in Section 2 we present the background of the research work; in Section 3 we describe our proposed approach to deal with clustering in the computer forensics domain; in Section 4 we detail the experiments and analyze the results; and finally, in Section 5 we present the conclusions and future work.

2. BACKGROUND

In this section we provide an overview of the literature associated with document clustering and ART in order to make the reader aware of the issues related to our proposed approach for clustering computer forensics documents. We also present a general view of some of the major limitations being conducted in text clustering for computer forensics analysis.

Text classification or text categorization is the process of assigning text documents to predefined classes or labels. As the classes and the training instances that belong to them are known beforehand, the classification problem is a task of supervised learning [10]. The process of document clustering aims to discover natural groupings (clusters) so that documents within a cluster should be as similar as possible, while documents in one cluster should be as dissimilar as possible from documents in other clusters [11]. Clustering differs from classification in that it assumes no prior knowledge of categories or the existence of a training set, and as such it is a task of unsupervised learning. Clustering may be applied in scenarios where classification is inadequate or impossible. This kind of scenario is commonplace in computer forensic examinations, where the knowledge obtained from text searches and stored in bookmarks (logical containers created by the examiner that reference files and other artifacts considered relevant to the case) in a case usually cannot be reused in other unrelated cases.

In literature we find many clustering algorithms that differ in several aspects, among which one may cite computational complexity, cluster quality, and sensitivity to the order of presentation of the input patterns. While it is desirable to build the best possible clusters, there is no way to define the best clusters, since there could be many correct, context-dependent ways to arrange the documents for a given corpus. Clustering quality may be evaluated by means of internal and external criteria. An internal criterion measures intra-cluster similarity (documents belonging to the same cluster should be similar) and inter-cluster similarity (documents belonging to different clusters should be dissimilar). While good scores on an internal criterion are desirable, they do not necessarily translate into good effectiveness in an application [11]. An external criterion, on the other hand, measures how well the clustering solution matches a set of classes produced by human judges, or *gold standard*. Thus, it is desirable for a clustering algorithm to produce clusters that approximate the gold standard classes.

Many clustering algorithms take vectors as input. The *vector space model* can be used to represent text documents as vectors of terms [12]. The document collection is represented as a matrix where each document is a row and each term is a column representing a different dimension. In this way, a document collection containing N documents and M terms can be represented by the $N \times M$ matrix presented in Table I, where a_{nm} is an attribute that represents the term frequency of term t_m in document d_n. This is also known as the *bag-of-words* model, or simply BOW. For the purpose of the present work, binary input vectors will be used. The matrix will be modeled as a binary incidence matrix, where a_{nm} is 1 if term t_m occurs in document d_n and is 0 otherwise.

TABLE I. Bag-Of-Words model

	t_1	t_2	$\cdots$	t_M
d_1	a_{11}	a_{12}	$\cdots$	a_{1M}
d_2	a_{21}	a_{22}	$\cdots$	a_{2M}
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
d_N	a_{N1}	a_{N2}	$\cdots$	a_{NM}

Even moderately sized document collections (with just a few hundreds of documents) often have several tens of thousands of different terms, with each document containing relatively few terms. The matrix is thus very sparse. This poses a problem for clustering algorithms because considerable resources are spent to represent and perform calculations on a large volume of data that is mostly empty except for a relatively small percentage of terms in each document. This and a number of other problems arise when analyzing high-dimensional data spaces; these problems are collectively referred to as the "curse of dimensionality" [13]. Several dimensionality reduction techniques have been proposed to deal with it in the domain of document clustering. These techniques aim to obtain a reduced representation of the original data set that is smaller in volume, yet closely maintains the integrity of the original data [14].

2.1. Search results

Document clustering can also be applied to a subset of the document collection. Search result clustering can be used to cluster the documents that were returned in response to a query submitted to an Information Retrieval (IR) system.

A common representation of search results is a simple list the user must scan from top to bottom to find the information that addresses the specific information needs. In [37] we find the standard model of information access processes, which is presented in Fig. 1.

0 Start with an information need.
1. Select a system and collections to search on.
2. Formulate a query.
3. Send the query to the system.
4. Receive the results in the form of information items.
5. Scan, evaluate, and interpret the results.
6. Either stop, or,
7. Reformulate the query and go to step 4.

Figure 1. Standard model of the information access process [37]

Related to the search results, Baeza-Yates and Ribeiro-Neto in [37] state that *"... many users dislike being confronted with a long disorganized list of retrieval results that do not directly address their information needs."* In the context of computer forensics examinations where each search request may return hundreds to thousands of results, it can be confusing and time-consuming to wade through all of them. Enhancement in the presentation of search hits could help examiners save time and deliver improved results.

In [15] van Rijsbergen presents the *cluster hypothesis* which states that *"Closely associated documents tend to be relevant to the same requests."* In other words, if there is a document from a cluster that is relevant to a search request, then it is likely that other documents from the same cluster are also relevant. This is because clustering puts together documents that share many terms [11]. Thus, if the search results are clustered, similar documents will appear together in coherent groups, which are arguably easier to scan than a disorganized list. If the cluster hypothesis holds, when the examiner finds a cluster where the first few documents are relevant to the query then the whole cluster can be flagged for more detailed analysis; conversely, a cluster where the first few documents are judged irrelevant can be discarded right away.

Search result clustering has been found to offer promising results, both in traditional text-based and in web-based information retrieval [16,17,18]. This paper focuses on the use of the ART1 algorithm to cluster search results returned from queries to computer forensic text corpora. The scenario investigated is that of hard clustering, where each document can be a member of exactly one cluster.

2.2. Adaptive Resonance Theory

ART, which was inspired by human cognitive information processing, was introduced by Grossberg in 1976 [25]. It describes a number of self-organizing artificial neural network architectures that employ an unsupervised learning process, and one of its main goals is to overcome the stability/plasticity dilemma. The stability/plasticity dilemma lies in that a system is desired which is able to learn new patterns without forgetting what it has already learned. A number of other previous neural network models are not plastic because they cannot learn from new patterns after the network is first trained. They are also not stable because even though they can be retrained from scratch to process new patterns, they do so at the cost of quickly forgetting old knowledge.

ART neural networks, on the other hand, are plastic because they can dynamically learn new patterns even after the network has stabilized, and are also stable since they preserve knowledge about past input patterns as new ones are presented. Each output neuron represents a cluster, and the algorithm creates and updates them on-the-fly as required by the input data, meaning the network is self-organizing. These properties make ART neural networks suitable for incremental clustering of data. The present work focuses on document clustering with ART1, an ART neural network architecture designed to work with binary input vectors. The foundations of ART and the neural network architectures it describes are laid out in detail in a number of publications, together with ART1 and Fuzzy ART [26,27].

2.2.1. ART1 architecture

Fig. 2 presents a typical ART1 system using a block diagram. There are two main components, the *attentional* and *orienting* subsystems. The *attentional* subsystem contains, among other components, two layers of neurons, F_1 and F_2. The *comparison layer* F_1 has N input neurons, and the *recognition layer* F_2 has R output neurons. N is the input size, i.e. the number of input patterns (in our case, digital documents). R is the number of clusters produced by the algorithm, and is calculated dynamically. Neurons are fully connected with both feed-forward and feedback weighted links. The *orienting* subsystem contains the reset layer for controlling the overall dynamics of the *attentional* subsystem.

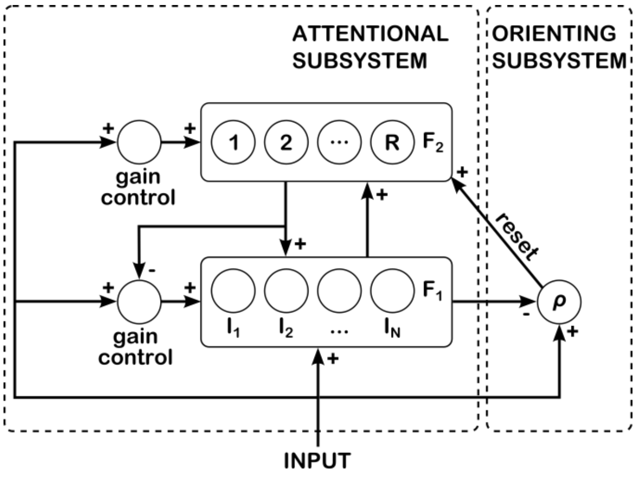

Figure 2. ART1 architecture [26]

Binary input patterns are presented to the F_1 layer, and the feed-forward links present the inputs to the F_2 layer. The feedback links from F_2 to F_1 store the prototype vectors, i.e. the vectors that represent the clusters embodied by the output neurons. The output neurons (clusters) in the F_2 layer compete for activation. The neuron with maximum activation takes the value 1 and inhibits the others, which take the value 0. If the input presented in F_1 and the winning neuron in F_2 "match", i.e. share enough dimensions in common, *resonance* occurs; *reset* occurs otherwise and the search is repeated. If no matching prototype vector is found, a new cluster is created based on the input vector. An input vector and a prototype vector are said to match if the *vigilance test* is passed. Vigilance (ρ) is a dimensionless parameter that dictates how close an input must be to a prototype vector for resonance to occur. ART1 neural networks are sensitive to the order of presentation of the input patterns, and may generate different clustering solutions when the same input patterns are presented in different order [24].

2.2.2. ART1 algorithm

The original work on ART1 describes its architecture and operation by means of differential equations, and does not supply a reference implementation [26]. Algorithm 1 (Fig. 3) presents the *Cluster-ART-I* algorithm described in [28] and cited in [27], with additional remarks for clarification. This algorithm is used in this paper as the basis of the software package developed, that was validated in the experiments. The Cluster-ART-I algorithm will be referred to in this paper as ART1 algorithm.

The ART1 algorithm groups binary input vectors into categories or clusters. Each cluster has a prototype vector, and each input vector is classified into the cluster that has the nearest prototype vector. The algorithm will create as many clusters as required by the input patterns. The number of clusters and their size depend on two parameters, β and ρ:

- β is the *choice parameter*, a small positive number that is added to the denominator in step 2 of the algorithm in order to avoid division by zero should it happen that $\|T_i\|_1=0$. The limit $\beta \to 0$ is called the *conservative limit* because small values of β tend to minimize recoding, i.e. updating of the prototype vectors, during the learning process [27]. β is used in the *category choice function* described in step 2 and also in the distance test in step 3 of the algorithm.
- ρ is the *vigilance* parameter, $0<\rho\leq 1$, that tests the similarity between the input and prototype vectors in step 3' of the algorithm and directly influences the number of clusters the algorithm creates, as well as their size. Low vigilance leads to coarse-grained generalization and abstract prototype vectors that represent many input vectors, the result being fewer clusters containing each a larger number of input vectors. High vigilance leads to fine-grained generalization and prototypes that represent fewer input vectors, the result being more clusters containing each a smaller number of input vectors. Though ART1 operation is unsupervised, the user can exert critical influence by means of tuning the vigilance parameter. Such tuning is useful when the number and size of the clusters is considered too small or too large.

It is stated in [27] that both Fuzzy ART (a member of the ART family of neural network architectures) and ART1 take three parameters: a choice parameter α, a learning rate β, $0<\beta\leq 1$, and vigilance ρ; although [28] mentions only two, choice parameter β and vigilance ρ. Despite the apparent difference [28] uses the same set of parameters as [27]. The ART1 algorithm described in [28] refers to the choice parameter as β instead of α and implements the *fast learning mode* described in [26], where the learning rate is fixed at 1. The parameters of the algorithm in this paper follow the description in [28].

```
0.  Start with zero cluster prototype vectors: the set $P$ of
    prototype vectors is $\{\}$.
1.  Let $I$ = next input vector. Let $P'$, the set of candidate
    prototype vectors, be equal to $P$.
2.  Find the closest prototype vector (if any). Call this
    prototype vector $T$. To find $T$ is to find $i \in P$ to
    maximize
        $\frac{T_i \cdot I}{\beta + \|T_i\|_1}$
    for some $\beta \ll 1$.
    $\|T_i\|_1$ counts the number of 1s in $T_i$ and acts as a tie-
    breaker, favoring smaller magnitude prototype vectors
    over larger prototype vectors which are subsets of them
    (have 1s in the same places) when input vector $I$ matches
    them equally well ($T_i \cdot I = T_{i'} \cdot I$).
3.  If $P' = \{\}$, or if $T_i$ is too far from $I$:
        $\frac{T_i \cdot I}{\beta + \|T_i\|_1} < \frac{I \cdot I}{\beta + n}$
    , then create a new cluster, $j$, and set $T_j = I$.
    Set $P = P \cup \{j\}$. Output $j$. Then go to step 1.
    $n$ is the number of dimensions (dimensionality) of $I$.
3'. If $T_i$ does not match $I$, in other words,
        $\frac{T_i \cdot I}{I \cdot I} < \rho$
    , $0 < \rho \leq 1$, then set $P' = P' - \{i\}$ and go to step 2.
4.  Otherwise ($T_i$ passes the tests in steps 3 and 3' and thus
    is close enough to $I$ in both senses), then update
    $T_i$: $T_i \leftarrow T_i \cap I$. Output $i$. Go to step 1.
```

Figure 3. ART1 algorithm [28]

2.3. Related Work

Document clustering of forensic text corpora has been done by researchers using different techniques and models, such as Kohonen's Self-Organizing Maps (SOM) [6] and the k-means algorithm [19]. Beebe and Dietrich in [5] proposed a new process model for text string searches that advocated the use of machine learning techniques, clustering being one of them.

Beebe and Clark in [6] used the Scalable Self-Organizing Map (SSOM) algorithm, which takes advantage of the sparseness of input vectors, to cluster search hits returned from queries to computer forensic text corpora. The results were reported to be promising. Beebe researched the subject further in a PhD thesis [20], where some issues and limitations that warrant further discussion are described:

1. The author performed searches at a physical level, completely skipping the file system. This approach is likely to miss documents that are not stored as plain text (Microsoft Office 2007/2010 user files and PDF documents, among others), or stored in noncontiguous areas of the media being examined.
2. The author also argues against full-text indexing, and states that *"Simply put, the startup cost for index creation is prohibitive – the return on investment is too low to make full text indexing worthwhile."* Both FTK and EnCase, two widely used computer forensic software packages [38,39], enable full-text indexing by default during pre-processing. This step is known to be resource-intensive, but by no means prohibitive when adequate hardware and software are available to the examiner.
3. The user was required to input the size of the 2-dimensional map that would display the clusters. The work does not mention how to determine or even suggest this value. The author states that "… *the primary purpose of this research was to ascertain the feasibility of clustering digital forensic text string search results and draw general conclusions regarding possible improvement in Information Retrieval effectiveness, studying cluster granularity optimization was out of scope"*, and follows stating that *"Future research should experimentally examine optimal map and document vector sizes given various data set and query characteristics."* The present work builds on the ideas discussed on Beebe's research [20].

The approach based on the k-means algorithm required the user to input the number of clusters, k , that the algorithm would create; Decherchi et al. [19] chose $k=10$, stating that *"… this choice was guided by the practical demand of obtaining a limited number of informative groups,"* and clustered the whole forensic text corpus they chose for their experiment. After the clustering was done, the words considered to be the most descriptive among the twenty most frequent words present in the documents belonging to each cluster were assessed and the results were reported to range from "interesting" to "extremely interesting". The authors did not discuss the technical details of the implementation.

As far as we are concerned, no other work related to document clustering of forensic text corpora was found; although there is a very large number of published works about document clustering. L. Massey published several research papers [21,22,23] and a PhD thesis [24] on the subject of document clustering using the ART1 algorithm, focusing on clustering entire collections. Our work rather focuses on clustering search results returned from computer forensic text collections.

3. PROPOSED APPROACH

The proposed approach consists of the conceptual model definition and the implementation of a software package with a modified ART1 algorithm, implemented to improve the running time of the algorithm. The validation method used experiments with a real world case with a two-fold approach, considering a quantitative and a qualitative method of analysis.

3.1. Conceptual Model

The conceptual model describes a pair of clustering solutions and a measure to compare them. The descriptions that follow are based on [30].

Definition 1. Let D={d_1,d_2,...,d_N} be the set of *N* documents matching the query expression, and U={u_1,u_2,...,u_R} and V={v_1,v_2,...,v_C} two partitions of the documents in *D* such that $\bigcup_{i=1}^{R} u_i = D = \bigcup_{j=1}^{C} v_j$ and $u_i \cap u_{i'} = \emptyset = v_j \cap v_{j'}$ for $1 \le i \ne i' \le R$ and $1 \le j \ne j' \le C$. Partition U has R subsets (clusters) and represents a clustering solution, and partition V has C subsets (classes) and represents the gold standard.

Definition 2. Let d_1 and d_2 be a pair of documents chosen from D. The total number of possible combinations of pairs in D is $\binom{n}{2}$, and the pairs can be of four different types:

- *a* – objects in a pair are placed in the same cluster in U and in the same class in V
- *b* – objects in a pair are placed in the same cluster in U and in different classes in V
- *c* – objects in a pair are placed in different clusters in U and in the same class in V
- *d* – objects in a pair are placed in different clusters in U and in different classes in V

A contingency table can be computed to indicate overlap between U and V as presented in Table II.

TABLE II. Contingency Table for Comparing Partitions U & V

$U \backslash V$	V_1	V_2	$\cdots$	V_C	Sums
U_1	n_{11}	n_{12}	$\cdots$	n_{1C}	a_1
U_2	n_{21}	n_{22}	$\cdots$	n_{2C}	a_2
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
U_R	n_{R1}	n_{R2}	$\cdots$	n_{RC}	a_R
Sums	b_1	b_2	$\cdots$	b_C	N

In Table II, n_{ij} represents the number of documents that were clustered in the *i*th subset (cluster) of partition U and in the *j*th subset (class) of partition V. The values of *a*, *b*, *c* and *d* can be computed from the n_{ij} values present in Table II.

Intuitively, *a* and *d* can be interpreted as indicators of agreement between U and V, while *b* and *c* can be interpreted as indicators of disagreement. Several indices based on these four types of pairs have been proposed in the literature. This work uses an information theoretic based measure, the Normalized Mutual Information (NMI). According to Strehl et al. [31], the NMI is a measure with a strong background rooted in Information Theory that gives a sound indication of the information shared between a pair of clusterings.

Definition 3. Let $I(U,V)$ be the Mutual Information (MI) between the discrete random variables described by clustering U and V, and $H(U)$ be the entropy[1] of the discrete random variable described by the cluster labeling U. The Mutual Information, or MI, defined by $I(U,V)$ is a metric or distance. As it has no upper bound, a normalized version that ranges between 0 and 1 is desirable [31]. The NMI is presented in Equation 1, which needs to be estimated from the sampled quantities provided by the clusterings. To meet this need, the normalized mutual information estimate $\phi^{(NMI)}(U,V)$ presented in Equation 2 is used [31].

$$NMI(U,V) = \frac{I(U,V)}{\sqrt{H(U) \times H(V)}} \tag{1}$$

$$\phi^{(NMI)}(U,V) = \frac{\sum_{i=1}^{R}\sum_{j=1}^{C} n_{ij} \times \log\left(\frac{n_{ij} \times N}{a_i \times b_j}\right)}{\sqrt{\left(\sum_{i=1}^{R} a_i \times \log\left(\frac{a_i}{N}\right)\right) \times \left(\sum_{j=1}^{C} b_j \times \log\left(\frac{b_j}{N}\right)\right)}} \tag{2}$$

The NMI has a fixed lower bound of 0, and a fixed upper bound of 1. It takes the value of 1 when two clusterings are identical and takes the value of 0 when they are independent, i.e. they do not share any information about each other.

3.2. Software package

In order to implement the conceptual model defined, two command line programs were produced. The first program, the *indexer*, was developed with the task of traversing a file system exposed from a mounted image file looking for: (i) all Microsoft Office user files (.doc, .docx, .xls, .xlsx, .ppt, .pptx, .pps, .ppsx); (ii) text files (.txt and .rtf) files; (iii) PDF files (.pdf); (iv) HTML and XML files (.htm, .html and .xml). The program also runs the following pre-processing steps:

[1] A clear and concise review of the fundamental concepts involved can be found in [32].

- Compute and store the MD5 hash of each document, and discard duplicates;
- Extract text from each document;
- Look for text in Portuguese (documents without text in Portuguese were discarded);
- Remove stopwords and numbers from the extracted text [37];
- Tokenize and stem the extracted text [37];
- Build an index with the metadata, e.g. *Author* and *Company*, from each file and its respective extracted text for later query and retrieval.

The second program, the *searcher*, was developed with the task of querying the index and presenting the results, i.e., the documents that matched the query. Upon execution, the program would return up to 10,000 documents in the index that matched the query expression, cluster them together with the ART1 algorithm, and then write a number of HTML files presenting all clusters and their respective documents along with the terms present in the prototype vector and the most frequent 20 terms present in the documents belonging to each cluster, which allow the examiner to obtain a general picture of the documents in each cluster. The choice for returning a maximum of 10,000 documents per query was made because the software package is research-oriented; a commercial tool would eschew such a limit. The choice for displaying the most frequent 20 terms present in the documents belonging to each cluster is based in [19]. Clusters are numbered sequentially and have no labels other than their numbers. The searcher program was written in Java and was based on the source code available in [29]. The program took five parameters:

a) *Case number* (mandatory);
b) *Query expression* (mandatory);
c) *Document type filter* (mandatory). Available document types were: structured text (MS Word, PDF and RTF), plain text, spreadsheet, and markup language (HTML and XML). An additional option was made available to not filter and return all matching documents regardless of their type.
d) *Vigilance value* (optional);
e) *Boolean flag to filter results from Windows system folders* (optional). Turning on this flag filters out results returned from the folders "Program Files" and "Windows".

After the searcher program is executed, it retrieves the documents that match the query expression and clusters them. The clustering solution must be compared to the gold standard with the classes defined by the domain experts using the validation method presented in Section 3.4.

3.3. ART1 Algorithm: Improvement

The ART software package available at [29] was not designed to handle input patterns that possess high dimensionality. The author of [29] states in [33] that ART algorithms are not ideal candidates for text clustering because they have problems with high-dimensional data. Our work shows that a modified ART1 algorithm can actually be used for text clustering with high-dimensional data in reasonable running time.

The data structure chosen to represent vectors (input patterns and prototypes) in [29] is the *vector* class template. For the purpose of document clustering with the ART1 algorithm, such an implementation is inadequate because it is very time consuming, and spends a large portion of time doing no useful work. Steps 2, 3, 3' and 4 of Algorithm 1 iterate over all dimensions of the input vector, of the prototype vector, or both. It is not necessary to iterate over all dimensions, because the calculations only need the dimensions whose value is 1, and those are relatively few in typical text collections, as explained in Section 2. Since in computer forensic examinations, it is commonplace to find document collections with thousands of documents that contain from tens to hundreds of thousands of unique terms (dimensions), it would be beneficial to represent documents with a data structure that would allow fast iteration of nonzero entries.

Since the C++ *vector* class template does not allow fast iteration of nonzero entries in vectors where most entries have a value of 0, our approach was to replace *vector*, which functions like

TABLE III. Comparison of C++ and Java ART1 algorithm implementations

N	*200*	*400*	*600*	*800*	*1000*	*1200*	*1400*	*1600*	*1800*	*2000*
Running time in seconds of art_1.exe (C++)	2	9	22	49	87	398	1.777	3.641	7.214	10.372
Running time in seconds of ART1.jar (Java)	1	3	5	8	13	39	82	157	257	268
Dimensions	1.313	1.969	2.403	2.760	2.975	4.578	7.269	8.902	10.961	13.384
Cells in BOW (C++)	262.600	787.600	1.441.800	2.208.000	2.975.000	5.493.600	10.176.600	14.243.200	19.729.800	26.768.000
Nonzero cells (Java)	13.258	26.400	41.481	55.993	71.326	89.963	112.418	140.458	166.880	183.812

a dynamic array. In other words, we propose to use a data structure that is more suitable for the purpose of document clustering: a *sparse vector*. This data structure should support fast iteration of nonzero entries, skipping entries whose value is 0. In doing so, it would allow faster traversal of the documents represented in the binary incidence matrix.

Our approach is intuitively inspired in the work presented in [6] that uses the Scalable Self-Organizing Map (SSOM) algorithm [34]. The SSOM algorithm is an implementation of Kohonen's Self-Organizing Map (SOM) [35] modified to take advantage of the sparseness of input vectors in order to reduce running time.

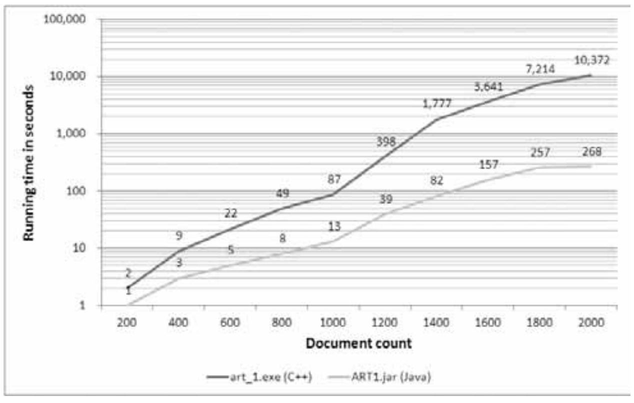

Figure 4. The C++ and Java implementations (running time in seconds with logarithmic scale)

In terms of implementation, the ART1 algorithm available in the ART software package [29] was rewritten in Java. The C++ *vector* class template was replaced with the *DefaultSparseBooleanMatrix* data structure from the UJMP library [36]. *DefaultSparseBooleanMatrix* was chosen because it is equipped to iterate quickly over nonzero entries. The new data structure was used to represent input vectors and cluster prototypes.

Tests were executed with subcollections obtained from computer forensic text collections, ranging from 200 to 2,000 documents in increments of 200 documents. The experimental results are presented in Table III and Fig. 4, and display a substantial reduction in running time in comparison to the C++ implementation of [29].

As noted in Fig. 4, the relationship between increase in document quantity and running time is not linear. A possible explanation is that running time is proportional to the number of cells in the binary incidence matrix that are iterated by each implementation. The C++ implementation iterates over all cells. The Java program iterates only over cells whose value is 1. This scenario is displayed in Fig. 5.

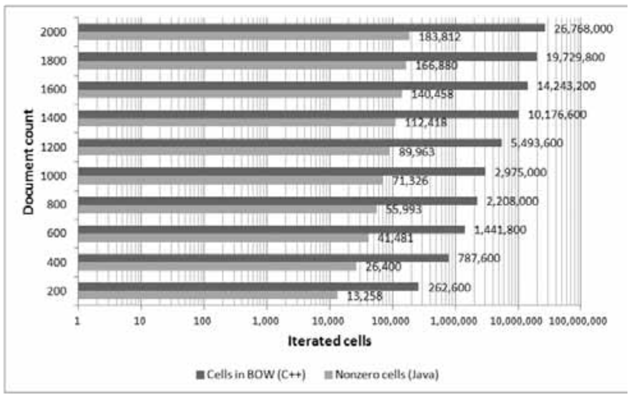

Figure 5. The C++ and Java implementations (number of iterated cells with logarithmic scale)

3.3.1. Computational complexity

Massey states in [24] that the ART1 algorithm has computational complexity of order $O(MN2^M)$

in the worst case, where N represents the number of documents and M the dimensionality of the input. At this point it is necessary to define R as the number of clusters created by the algorithm.

Factor MN represents the stabilization loop, and N is $O(M)$. Factor 2^M comes from the maximal number of clusters that can be created with M-bit inputs which may have to be searched to find the cluster that most closely resembles the current input vector. There can be at most 2^M unique inputs of M bits each in the worst case, which implies that the algorithm can create at most 2^M clusters containing a single unique document each. In such a scenario, $R = N = 2^M$.

The algorithm seems to be prohibitively expensive since M usually runs into the thousands in typical text collections and an exponential number of clusters is out of question in a real application. Realistic scenarios are far from this, however. Massey states in [24] that R is a constant determined by the application and should be at least several orders of magnitude smaller than N since the purpose of clustering is to compress the information space. Since we can only have as many clusters as we have documents, we have that $R \ll N \ll M \ll 2^M$ in typical text collections, which is very distant from the worst case. Massey states that R is expected to take values from tens to hundreds depending on the application and wraps up his explanation stating that ART1 is in fact $O(sRN)$, s being the number of iterations needed for stabilization. Our experimental results support this claim.

The complexity of our modified version is of the same order $O(sRN)$ as described by Massey [24], but had a much shorter running time than the ART software package available at [29] in our experiments because it skips dimensions whose value is 0.

3.4. Validation method

As cited before, the validation method is based on a two-fold approach, considering a quantitative and a qualitative method of analysis:

- First the search results for each query expression are classified by a human expert, i.e. a domain specialist, to form the classes that represent the gold standard.
- Then the clusters produced by the software package are compared to the classes, and the $\phi^{(NMI)}(U,V)$ estimate is computed (Equation 2).

Since $\phi^{(NMI)}$ is a quantitative measure, a subjective evaluation will also be performed by another domain specialist, who will assess the produced clusters to define whether they are "good" and "useful". In other words, judge whether the algorithm managed to group similar documents and give a good general picture of their contents.

4. EXPERIMENTS

To validate the conceptual model presented in Section 3.1 and the software package implemented (Section 3.2) we have conducted experiments with real world forensic cases. An image file acquired from the hard drive of a computer seized by a law enforcement agency during the course of an actual investigation was processed. The computer contained a single hard drive which was seized while executing a search warrant for collecting evidence of fraud in public procurement contracts. The hard drive contained a single NTFS file system. Table IV presents the number of documents used in our experiments (3,455), categorized by document type, after pre-processing. A number of documents were discarded because they were either duplicates or did not contain any text in Portuguese.

TABLE IV. Total Number of Documents

File type	*# of documents*
Microsoft Office user files (.doc, .docx, .xls, .xlsx, .ppt, .pptx, .pps, .ppsx)	578
Text files (.txt,.rtf)	118
PDF files (.pdf)	196
HTML and XML (.htm, .html,.xml)	2,563
TOTAL	3,455

4.1. ART1 parameters

As presented in Section 2.2.2, the *choice parameter* β was set to the small value of 0.001 in step of Algorithm 1 (Fig. 3). This small value was chosen to avoid division by zero, while attempting to not disturb the value of the denominator in steps 2 and 3 of the Algorithm 1 (Fig. 3).

The default value for the *vigilance* parameter was set to $\rho = 1/\ln(N \times M)$. This value was set as a starting point as discussed in [24], where it is suggested that the value for the *minimal useful vigilance* (ρ_{min}) should be $\rho_{min} = 1/M$ for a document collection with a maximum dimensionality of 2,600, and in practice much less (500 to 800). Such a value is not adequate for the purpose of this study because no dimensionality reduction techniques other than stopword removal and stemming are performed, which causes search results returned by queries to possess high dimensionality (>3,000). Although the default vigilance was calculated at runtime, it was only used as a suggestion and the user could quickly review the produced clusters and run the software again as many times as desired, supplying a value for vigilance, in order to find clusters that the user considers more adequate. That said, after being informed of the possibility of supplying a custom value for the vigilance parameter, the domain experts chose not to do so.

Table V lists the results of the experiments using 3,455 digital documents related to a real case of fraud in public procurement contracts. Company and individual names were masked to protect their rights. Tables VI, VII and VIII are the contingency tables for each query expression (Person#1, Company#1 and Company#2). Cells whose value is 0 were left blank to improve legibility.

Some preliminary results of our approach to cluster digital forensic documents were presented in [40]. The software package used in that publication did not discard duplicates and used a less accurate language detection software library, so the results were different.

TABLE V. Results

Query expression	***# of results***	***# of classes (V)***	***# of clusters (U)***	$\phi^{(NMI)}$	ρ
Person#1	70	16	19	0.76	0.075089
Company#1	185	23	27	0.719	0.07549
Company #2	63	10	13	0.5	0.077111

TABLE VI. Contingency Table for Query Expression "Person#1"

Continued

U/V	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**Sums**
1																1	1
2														1			1
3						4						2					6
4													8				8
5						1				1							2
6			1								2						3
7	1			1											1		3
8									9								9
9									4								4
10				3	1							1					5
11				2													2
12												4					4
13												6					6
14				2								1					3
15									4								4
16												4					4

TABLE VI. Contingency Table for Query Expression "Person#1"

Conclusion

U/V	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	Sums
17									2								2
18							1										1
19		1						1									2
Sums	1	1	1	8	1	5	1	1	19	1	2	18	8	1	1	1	70

TABLE VII. Contingency Table for Query Expression "Company#1"

U/V	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	Sums
1																					2			2
2						1											1		1					3
3								13																13
4								4	1			1												6
5	1														1									2
6								2																2
7										1	2													3
8													1											1
9						3											4							7
10		2						9											6			1		18
11																				22				22
12																1	2							3
13			34					5																39
14																			7					7
15								5									1		8					14
16				1																				1
17																							1	1
18														2										2
19								1								2								3
20								11										8						19
21																1								1
22							1	1											1					3
23																			2					2
24																			3					3
25														4										4
26					1			1																2
27																	2							2
Sums	1	2	34	1	1	4	1	52	1	1	2	1	1	6	1	4	10	8	28	22	2	1	1	185

TABLE VIII. Contingency Table for Query Expression "Company#2"

U/V	1	2	3	4	5	6	7	8	9	10	Sums
1				1					1	1	3
2					1						1
3					17						17
4					7						7
5					4						4
6					8						8
7					6						6
8		1				1	1				3
9								1			1
10					5						5
11	1										1
12					5						5
13			2								2
Sums	1	1	2	1	53	1	1	1	1	1	63

4.2. Discussion

The results of the experiments raise a number of topics for discussion and present opportunities for future work. The presented contingency Tables VI, VII and VIII will be commented according to their respective query expressions.

1) *Table VI - query expression "Person#1".* There were 16 classes (V) and 19 clusters (U).

   a) Class #9 accounted for 19 of the 70 results and was the biggest for this query expression. 17 out of its 19 results were concentrated in 3 of the 19 clusters generated by the algorithm (U = 8, 9, 15), and the remaining 2 results were assigned to a single cluster (U = 17). Upon closer inspection, the prototype vectors of the 4 clusters (U = 8, 9, 15, 17) were found to be very different from each other, as displayed in Table IX. The prototype vectors shared a number of common terms, as displayed in Table X.

TABLE IX. Cluster and prototype vector term counts for query expression "Person#1"

Cluster	*Term count (cluster)*	*Term count (prototype vector)*
1	4	4
2	6	6
3	380	19
4	17	2
5	139	8
6	222	13
7	75	3
8	**185**	**10**
9	**247**	**16**
10	125	8
11	60	51
12	777	35
13	641	22
14	244	15
15	**486**	**18**
16	671	46
17	**658**	**73**
18	5016	5016
19	4168	661

TABLE X. Term count intersection in the prototype vectors of class #9

Prototype vector for cluster	*#8*	*#9*	*#15*	*#17*
#8	10	10	10	10
#9	10	16	10	11
#15	10	10	18	14
#17	10	11	14	73

   b) Class #12 was the second biggest with 18 documents scattered in 6 different clusters, and 14 documents were concentrated in 3 clusters (U = 12, 13, 16). Visual inspection revealed that the prototype vectors of the clusters were very different, sharing few common terms.

   c) Class #13 contains 8 documents, and they were all assigned to cluster #4. Upon closer inspection, it was found that the documents contained scanned images but did not contain any text, and its documents vectors contained only textual metadata, which was the same for all documents.

   d) Each one of the 3 documents in cluster #7 comes from a different class. The documents differed in content but shared the 3 terms in the prototype vector, as displayed in Table IX. This was enough for the algorithm to assign them to the same cluster.

   e) The algorithm created cluster #18 for the single document in class #7, which is also the sole document in the cluster. The document contained a long text and many terms that could not be found in any other document in the collection.

The value of the $\phi^{(NMI)}$ estimate was 0.76. Considering that the upper bound for NMI is 1, this can be considered a good result – there is high agreement between the clustering solution and the gold standard. A subjective evaluation was made by three domain specialists, who made the following remarks:

- It is preferable to review the hits in clustered form than in an exhaustive list;
- Numbered cluster labels by themselves did not immediately convey what kind of content was to be found in the documents present in the cluster, but cluster labeling is a research topic by itself and was not addressed in this work;
- One of the specialists stated that, even in the case where it was required to review all documents from all clusters, the clustering approach had the merit of allowing documents with related content to be reviewed in sequence, whereas an exhaustive non-clustered list had no means to organize the documents in thematic sets;
- One specialist, after realizing resemblance between two clusters whose prototype vectors shared common terms, inquired whether the software package could merge clusters (this functionality was not developed).

The combined results of the $\phi^{(NMI)}$ estimate (0.76 of 1) and the subjective evaluation suggested that the proposed approach implemented by means of the software package is useful.

2) *Table VII - query expression "Company#1".* There were 23 classes (V) and 27 clusters (U). The 5 biggest classes account for 146 of the 185 documents, and the 5 biggest clusters contain 112 documents, with the $\phi^{(NMI)}$=0.719.

a) Class #8 is the biggest and also the most fragmented. Its 52 documents are scattered over 10 clusters. The class contained business statements of several disparate contents and purposes, and the clusters managed to capture related statements. This suggests that the class could be further refined into a number of classes.

b) Class #3 contains 34 documents, and they were assigned to cluster #13. The class contained certificates, and its documents were almost identical, except for a sequence number which was not indexed during preprocessing. Thus, from the point of view of the indexer and the searcher, the documents were identical. The perfect correspondence between the class and cluster caused the value of the $\phi^{(NMI)}$ estimate to be higher.

c) Class #19 contains 28 documents which were assigned to 7 clusters. The 3 biggest clusters contained 21 of the 28 documents. The prototype vectors for clusters #14, #15, #23, and #24 shared one term in common. The prototype vectors for clusters #14 and #15 contained 10 and 15 terms, respectively; the prototype vectors for clusters #23 and #24 contained respectively 69 and 47 terms, as displayed in Table XI. This suggest that the documents contained in clusters #14 and #15 were more generic, and the documents in clusters #23 and #24 were more specific; upon closer inspection, this was verified to be true.

d) Class #20 contains 22 documents, and they were all assigned to cluster #11. Upon closer inspection, it was found that the documents contained scanned images but did not contain any text, and its documents vectors contained only textual metadata, which was the same for all documents.

e) Class #17 contains 10 documents that were scattered in 5 different clusters, 4 of them in cluster #9. The same cluster #9 contains 3 documents that belong in class #6. The classes contained documents pertaining to quotations and bills of materials, all pertaining to public construction contracts. The prototype vector for cluster #9 contained 13 terms, and 4 of them could be found in documents from both classes #17 and #6.

TABLE XI. Cluster and prototype vector term counts for query expression "Company#1"

Cluster	***Term count (cluster)***	***Term count (prototype vector)***
1	8	2
2	190	11
3	159	6
4	108	4
5	50	12
6	87	6
7	80	34
8	18	18
9	269	13
10	327	9
11	31	2
12	522	28
13	225	10
14	**367**	**10**
15	**466**	**13**
16	63	63
17	116	116
18	36	10
19	418	20
20	373	10
21	162	162
22	219	9
23	**280**	**69**
24	**673**	**47**
25	668	37
26	271	33
27	574	154

3) *Table VIII - query expression "Company#2".* There were 10 classes and 13 clusters.

   a) Class #5, the largest, accounts for 53 of the 63 documents returned by the query. The company name is a common word used in interpersonal communications. Many of the hits were instant messaging chat logs that were not related to the company, and were considered to be false positives. The documents were scattered in 8 different clusters.

The other classes were well represented by the other clusters. Still, the dispersion displayed in class #5 induced the lowest $\phi^{(NMI)} = 0.5$ of all experiments.

4.2.1. General remarks

The ART1 algorithm managed to cluster together documents that shared common terms in all experiments. In order for the generated clusters to approximate the gold standard classes, their prototype vectors must contain the terms found in the documents belonging to each class, but the prototype vectors are not user-defined, rather being calculated on-the-fly by the algorithm.

In all tests the running time of the algorithm was very low, taking less than 5 seconds for each subset returned by the software package from the total 3,455 documents (min. 63 and max. 185 documents, see Table V). This is important because it allows the examiner to quickly peruse the results returned by a query expression and evaluate whether the clusters are too generic or too refined, and run the query again with a modified vigilance value. If the response time were slow, the algorithm could be considered unacceptable for interactive use and its appeal would be reduced.

5. CONCLUSIONS AND FUTURE WORK

This work presented research that is justified in the basis that it can be confusing and time-consuming to wade through hundreds to thousands of search results returned from queries to computer forensic text corpora. Document clustering in computer forensic examinations using ART1 neural networks was proved to allow forensic examiners to quickly and effectively obtain a general picture of document contents, leveraging van Rijsbergen's cluster hypothesis [15], as cited in Section 2.1.

Unfortunately, we found little previous research in document clustering of forensic text corpora to compare results, but considering the related work (Section 2.3) we may cite:

- For sure it not enough to cluster only results obtained at the physical level as done in [20], since we would miss structured documents, which accounted for many of the hits as we show in our experiments (many of the returned documents were structured: .doc, .docx, rather than plain text: .txt, .html).
- In opposition to the SOM [6] and k-means [19] algorithm with a fixed value for the number of clusters, it is possible to generate good clusters without the need to specify parameters beforehand (e.g., vigilance was calculated at runtime and could be tuned if desired).

Thus, our approach presents a clear advantage over other algorithms since it does not require a pre-defined number of clusters, resulting in a more flexible approach to clustering digital forensic documents. The vigilance parameter (ρ) is very important and we have empirically found a default value for it, but ideally a strong theoretical foundation is necessary, although the results were considered adequate by the group of specialists.

There is still much more investigation to be done in this area. The use of the ART1 algorithm was found to be feasible and useful for clustering of computer forensic documents. That said, our experiments were conducted with a relatively small corpus, since it is very laborious to prepare the gold standard classes to use to validate the model. Experiments with larger corpora are necessary and recommended for future work.

As future work we may cite also the use of soft and hierarchical clustering, generation of high quality cluster labels, development of methods for determining the optimal order of presentation of input patterns, and comparison with other clustering algorithms such as SOM, k-means and Expectation-Maximization (EM).

ACKNOWLEDGMENTS

This work was sponsored by the Ministry of Justice of Brazil under the PRONASCI Program. Authors thank Prof. Maria das Graças Volpe Nunes for kindly sharing the stopword list used in several research projects at the Núcleo Interinstitucional de Linguística Computacional (NILC) at the University of São Paulo. Thanks are also due to Luís Filipe da Cruz Nassif for suggesting the use of the NMI measure.

REFERENCES

[1] N. Beebe and J. Clark, "Dealing with terabyte data sets in digital investigations," Advances in Digital Forensics 194, pp. 3-16 (2005).
[2] N. Beebe and J. Clark, "A hierarchical, objectives-based framework for the digital investigations process," Digital Investigation, vol. 2, issue 2, June 2005, pp. 147-167 (2005).
[3] D. Forte, "The importance of text searches in digital forensics," Network Security, 2004, pp. 13 – 15 (2004).
[4] H. Jee, J. Lee, and D. Hong, "High Speed Bitwise Search for Digital Forensic System," World Academy of Science, Engineering and Technology, 32, pp. 104-107 (2007).
[5] N. Beebe and G. Dietrich, "A new process model for text string searching," in Research advances in digital forensics III, Shenoi S. and Craiger P. (Eds.). Norwell: Springer; 2007. pp. 73–85.
[6] N. Beebe and J. Clark, "Digital forensic text string searching: Improving information retrieval effectiveness by thematically clustering search results," in Digital Investigation, September 2007, vol. 4 (suppl. 1) (2007).
[7] J. Lee, "Proposal for Efficient Searching and Presentation in Digital Forensics," Third International Conference on Availability, Reliability and Security, IEEE Computer Society, 0, pp. 1377-1381 (2008).
[8] M. Schwartz and L. M. Liebrock, "A Term Distribution Visualization Approach to Digital Forensic String Search," in Proceedings of the 5th international workshop on Visualization for Computer Security (VizSec '08), John R. Goodall, Gregory Conti, and Kwan-Liu Ma (Eds.). Springer-Verlag, Berlin, Heidelberg, pp. 36-43 (2008).
[9] M. Schwartz, C. Hash, and L. M. Liebrock, "Term distribution visualizations with Focus+Context," in Proceedings of the 2009 ACM

symposium on Applied Computing (SAC '09). ACM, New York, NY, USA, pp. 1792-1799. 2009.
[10] F. Sebastiani, "Machine learning in automated text categorization," ACM Computing Surveys, 34(1), pp. 1–47 (2002).
[11] C. D. Manning, P. Raghavan, and H. Schütze, Introduction to Information Retrieval. Cambridge University Press, 2008.
[12] G. Salton. Automatic Text Processing. Addison-Wesley, 1989.
[13] H.-P. Kriegel, P. Kröger and A. Zimek, "Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering," ACM Transactions on Knowledge Discovery from Data, 3:1:1–1:58 (2009).
[14] J. Han and M. Kamber, Data Mining: Concepts and Techniques. Morgan Kaufmann Publishers Inc., 2006.
[15] C. J. van Rijsbergen, Information Retrieval. Butterworths, London, 2nd edition. 1979.
[16] A. Leuski, "Evaluating document clustering for interactive information retrieval," in Tenth international conference on information and knowledge management. Atlanta, Georgia: ACM Press; 2001.
[17] A. Leuski and J. Allan, "Improving interactive retrieval by combining ranked lists and clustering in RIAO," College de France; 2000.
[18] H. Zeng, Q. He, Z. Chen, W. Ma, and J. Ma, "Learning to cluster web search results," in Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR '04). 2004. ACM, New York, NY, USA, pp. 210-217.
[19] S. Decherchi, S. Tacconi, J. Redi, A. Leoncini, F. Sangiacomo, and R. Zunino, "Text Clustering for Digital Forensics Analysis," in Journal of Information Assurance and Security, No. 5, pp. 384-391, 2010.
[20] N. L. Beebe, "Improving information retrieval effectiveness in digital forensic text string searches: clustering search results using self-organizing neural networks," PhD thesis, Department of Information Systems and Technology Management, College of Business, The University of Texas at San Antonio, 2007.
[21] L. Massey, "Determination of Clustering Tendency With ART Neural Networks," in Proceedings of 4th Intl. Conf. on Recent Advances in Soft Computing, Nottingham, U.K., 12 & 13 December 2002.
[22] L. Massey, "On the quality of ART1 text clustering," Neural Networks(16)5-6, pp.771-778, 2003.
[23] L. Massey, "Real-World Text Clustering with Adaptive Resonance Theory Neural Networks," in Proceedings of 2005 International Joint Conference on Neural Networks, Montreal, Canada, July 31- August 4, 2005.
[24] L. Massey, "Le groupage de texte avec les réseaux de neurones ART1," PhD thesis, Faculté d'ingénierie du Collège militaire royal du Canada, 2005.
[25] S. Grossberg, "Adaptive pattern classification and universal recoding: I. Parallel development and coding of neural feature detectors," Biological Cybernetics, 23, pp. 121–134, 1976.
[26] G. A. Carpenter and S. Grossberg, "A massively parallel architecture for a selforganization neural pattern recognition machine," Computer Vision, Graphics, and Image Processing, vol. 37, pp. 54-115, 1987.
[27] G. A. Carpenter, S. Grossberg, and D. B. Rosen, "Fuzzy ART: Fast Stable Learning and Categorization of Analog Patterns by an Adaptive Resonance System," Neural Networks, 1991. Elsevier Science Ltd. Oxford, UK, UK, 232. 4, issue 6, pp.759-771.
[28] B. Moore, "ART 1 and Pattern Clustering," in Proceedings of the 1988 Connectionist Models Summer School, pp. 174-183, 1988.
[29] T. Hudík. Tool: ART software package. http://users.visualserver.org/xhudik/art/, accessed 07/23/2011.
[30] J. Santos and S. Ramos, "Using a clustering similarity measure for feature selection in high dimensional data sets," Intelligent Systems Design and Applications (ISDA), 2010 10th International Conference on, 2010, pp. 900-905
[31] A. Strehl, J. Ghosh, and C. Cardie (2002), "Cluster ensembles - a knowledge reuse framework for combining multiple partitions," Journal of Machine Learning Research, 3: pp. 583–617.
[32] N. Vinh, J. Epps, and J. Bailey, "Information theoretic measures for clusterings comparison: is a correction for chance necessary?," in Proceedings of the 26th Annual International Conference on Machine Learning (ICML '09)
[33] T. Hudík, "Learning algorithms in processing of various difficult medical and environmental data," PhD thesis, Faculty of Informatics, Masaryk University, 2009.
[34] Roussinov, D. G. & Chen, H. (1998), "A Scalable Self-organizing Map Algorithm for Textual Classification: A Neural Network Approach to Thesaurus Generation," Communication Cognition and Artificial Intelligence, Spring, 1998, 15: pp. 81–112
[35] Kohonen, T. (1981), "Automatic Formation of Topological Maps of Patterns in a Self-Organizing System," in Proceedings of the Second Scandinavian Conference on Image Analysis, Oja, E. & Simula, O. (Eds.) 1981, pp. 214–220.
[36] H. Arndt. Tool: UJMP library. http://sourceforge.net/projects/ujmp/, accessed 01/30/2012.
[37] R. A. Baeza-Yates and B. Ribeiro-Neto, Modern Information Retrieval. Addison-Wesley Longman Publishing Co., Inc., 1999.
[38] AccessData Group, LLC. Tool: Forensic Toolkit® 3.4.1. http://access-data.com/products/computer-forensics/ftk, accessed 10/29/2011.
[39] Guidance Software, Inc. Tool: EnCase® Forensic 7.01.02. http://www.guidancesoftware.com/forensic.htm, accessed 10/29/2011.
[40] G. R. F. de Araújo and C. G. Ralha, "Computer forensic document clustering with ART1 neural networks," in Proceedings of The Sixth International Conference on Forensic Computer Science (ICoFCS). 2011. Florianópolis, Brazil, pp. 106-114. DOI: http://dx.doi.org/10.5769/C2011011

Georger Rommel Ferreira de Araújo is a Federal Forensics Expert at the Federal Police Department (DPF) in Brazil since 2007. He received his M.Sc. in Electrical Engineering from the University of Brasília (UnB) in 2011, and his B.Sc. degree in Computer Science from the Federal University of Ceará (UFC) in 2002.

Célia Ghedini Ralha is an Associate Professor at the Department of Computer Science, University of Brasília, Brazil. She received her Ph.D. degree in Computer Science from Leeds University, England, UK in 1996. Her M.Sc. in 1990 was received at the Aeronautics Institute of Technology (ITA), São José dos Campos, São Paulo, Brazil. She has a BASc in Administration from the Catholic University of Brasília, Brazil. She has experience in Computer Science, focusing on Intelligent Information Systems, with research interest on the treatment of information and knowledge using Multi-Agent Systems approach, focusing the intelligent agent interaction protocol and the association to other technologies, such as SOA, Web Intelligence, Web Semantic, Business Process Management and Knowledge Management.