Fast and Effective Clustering of Spam Emails Based on Structural Similarity

Mina Sheikhalishahi[1], Andrea Saracino[2(✉)], Mohamed Mejri[1], Nadia Tawbi[1], and Fabio Martinelli[2]

[1] Department of Computer Science, Université Laval, Quebec City, Canada
mina.sheikh-alishahi.1@ulaval.ca,
{mohamed.mejri,nadia.tawbi}@ift.ulaval.ca
[2] Istituto di Informatica e Telematica, Consiglio Nazionale delle Ricerche, Pisa, Italy
{andrea.saracino,fabio.martinelli}@iit.cnr.it

Abstract. Spam emails yearly impose extremely heavy costs in terms of time, storage space and money to both private users and companies. Finding and persecuting spammers and eventual spam emails stakeholders should allow to directly tackle the root of the problem. To facilitate such a difficult analysis, which should be performed on large amounts of unclassified raw emails, in this paper we propose a framework to fast and effectively divide large amount of spam emails into homogeneous campaigns through structural similarity. The framework exploits a set of 21 features representative of the email structure and a novel categorical clustering algorithm named Categorical Clustering Tree (CCTree). The methodology is evaluated and validated through standard tests performed on three dataset accounting to more than 200k real recent spam emails.

1 Introduction

Spam emails constitute a notorious and consistent problem still far from being solved. In the last year, out of the daily 191.4 billions of emails sent worldwide in average [20], more than 70 % are spam emails. Spam emails cause several problems, spanning from direct financial losses, to misuses of Internet traffic, storage space and computational power [22]. Moreover, spam emails are becoming a tool to perpetrate different cybercrimes, such as phishing, malware distribution, or social engineering-based frauds.

Given the relevance of the problem, several approaches have already been proposed to tackle the spam email issue. Currently, the most used approach for fighting spam emails consists in identifying and blocking them on the recipient machine through filters, which generally are based on machine learning techniques or content features, such as keywords, or non ascii characters [5,8,25].

This research has been partially supported by EU Seventh Framework Programme (FP7/2007–2013) under grant no 610853 (COCO Cloud), MIUR-PRIN Security Horizons and Natural Sciences and Engineering Research Council of Canada (NSERC).

J. Garcia-Alfaro et al. (Eds.): FPS 2015, LNCS 9482, pp. 195–211, 2016.
DOI: 10.1007/978-3-319-30303-1_12

Unfortunately, these countermeasures just slightly mitigate the problem which still impose non negligible cost to users and companies [22].

To effectively fight the problem of spam emails, it is mandatory to find and persecute the spammers, generally hiding behind complex networks of infected devices which send spam emails against their user will, i.e. botnets. Thus, information useful in finding the spammer should be inferred analyzing text, attachments and other elements of the emails, such as links. Therefore, the early analysis of correlated spam emails is vital [2,7]. However, such an analysis, constitutes an extremely challenging task, due to the huge amount of spam emails, which vastly increases hourly (8 billions per hour) [20] and for the high variance that related emails may show, due to the use of obfuscation techniques [19]. To simplify this analysis, huge amount of spam emails, generally collected through honey-pots, should be divided into spam campaigns [29]. A *spam campaign* is the set of messages spread by a spammer with a specific purpose [6], like advertising a product, spreading ideas, or for criminal intents.

This paper proposes a methodology to fast and effectively group large amount of spam emails by structural similarity. A set of 21 discriminative structural features are considered to obtain homogeneous email groups, which identify different spam campaigns. Grouping spam emails on the base of their similarities is a known approach. However, previous works mainly focus on the analysis of few specific parameters [2,21,29,30], showing results whose accuracy is still somehow limited. The proposed approach in this work is based on a *categorical hierarchical clustering* algorithm named *Categorical Clustering Tree (CCTree)*, introduced in [27], which builds a tree whose leaves represent the various spam campaigns. The algorithm clusters (groups) emails through structural similarity, verifying at each step the homogeneity of the obtained clusters and dividing the groups not enough homogeneous (pure) on the base of the attribute which yields the greatest variance (entropy). The effectiveness of the proposed approach has been tested against 10k spam emails extracted from a real recent dataset [1], and compared with other well-known categorical clustering algorithm, reporting the best results in terms of clustering quality (i.e. purity and accuracy) and time performance.

The contributions of this paper can be summarized as follows:

- We present a framework to effectively and efficiently analyze and cluster large amounts of raw spam emails into spam campaigns, based on a Categorical Clustering Tree (CCTree) algorithm.
- We introduce a set of 21 categorical features representative of email structure, briefly discussing the discretization procedure for numerical features.
- The performance of CCTree has been thoroughly evaluated through *internal evaluation*, to estimate the ability in obtaining homogeneous clusters and *external evaluation*, for the ability to effectively classify similar elements (emails), when classes are known beforehand. Internal and external evaluation have been performed respectively on a dataset of 10k unclassified spam emails and 276 emails manually divided in classes.

– We propose and validate through analysis on 200k spam emails, a methodology to choose the optimal CCTree configuration parameters based on detection of max curvature point (knee) on an homogeneity-number of clusters graph.
– We compare the proposed methodology with two general categorical clustering algorithms, and other methodologies specific for clustering spam emails.

The rest of the paper is structured as follows. Section 2, reports the formal description and the theoretical background of the Categorical Clustering Tree algorithm. Section 3 describes the proposed framework, detailing the extracted features and reporting implementation details. Section 4 reports the experiments to evaluate the ability of CCTree in clustering spam emails, comparing the results with the ones of two well known categorical clustering algorithms. Also the methodology to set the CCTree parameters is reported and validated. Section 5 discuss limitations and advantages of the proposed approach reporting result comparison with some related work. Other related work on clustering spam emails is presented in Sect. 6. Finally Sect. 7 briefly concludes proposing future research directions.

2 Categorical Clustering Tree

In this section we recall notions on the Categorical Clustering Tree (CCTree) algorithm, presented in [27], recalling terminology and construction methodology.

2.1 CCTree Construction

The CCTree is constructed iteratively through a decision tree-like structure, where the leaves of the tree are the desired clusters. An example of CCTree is reported in Fig. 1. The root of the CCTree contains all the elements to be clustered. Each element is described through a set of *categorical* attributes, such as the *Language* of a message. Being categorical each attribute may assume a finite set of discrete values, constituting its domain. For example the attribute

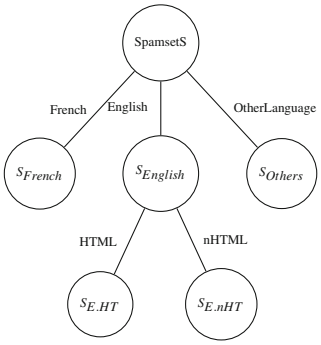


Fig. 1. A small CCTree.

Language may have as domain: {*English*, *French*, *Spanish*}. At each step, a new level of the tree is generated by splitting the nodes of the previous levels, when they are not homogeneous enough. *Shannon Entropy* [26] is used both to define a homogeneity measure called *node purity*, and to select the attribute used to split a node. In particular non-leaf nodes are divided on the base of the attribute yielding the maximum value for Shannon entropy. The separation is represented through a branch for each possible outcome of the specific attribute. Each branch or edge extracted from parent node is labeled with the selected feature which directs data to the child node. The process of CCTree construction can be formalized as follows.

Input: Let D be a set of data points, containing N tuples on a set A of d attributes, and let S be a set of stop conditions.

Attributes: An ordered set of d attributes $A = \{A_1, A_2, \ldots, A_d\}$ is given, where each attribute is an ordered set of mutually exclusive values. Thus, the j'th attribute could be written as $A_j = \{v_{1j}, v_{2j}, \ldots, v_{(rj)j}\}$, where r_j is the number of features of attribute A_j.

Data Points: A set D of N data points is given, where each data point is a vector whose elements are the features of attributes.

Stop Conditions: A set of stop conditions $S = (\{\mu, \varepsilon\})$ is given. μ is the "minimum number of elements in a node", i.e. when the number of elements in a node is less than μ, then the node is not divided even if not pure enough. ε represents the "minimum desired purity" for each cluster, i.e. when the purity of a node is better or equal to ε, it will be considered as a leaf. To calculate the node purity, a function based on Shannon entropy is defined as follows:

Let N_{kj_i} represents the number of elements having the k'th value of j'th attribute, in node i. Let N_i be the number of elements in node i. Thus, considering $p(v_{kj_i}) = \frac{N_{kj_i}}{N_i}$, the purity of node i, denoted by $\rho(i)$, is defined as: $\rho(i) = -\sum_{j=1}^{d}\sum_{k=1}^{r_j} p(v_{kj_i}) log(p(v_{kj_i}))$ where d is the number of attributes, and r_j is the number of features of j'th attribute.

Output: The final output of the CCTree algorithm is a set of clusters, constituted by the leaves of the CCTree. For additional information on the CCTree algorithm we refer the reader to [27].

3 Framework

The presented framework acts in two steps. At first raw emails are analyzed by a parser to extract vectors of structural features. Afterward the collected vectors (elements) are clustered through the introduced CCTree algorithm. This section reports details on the proposed framework for analysis and clustering spam emails and extracted features.

3.1 Feature Extraction and Definition

To describe spam emails, we have selected a set of 21 categorical attributes, which are representative of the structural properties of emails. The reason is that the general appearance of messages belonging to the same spam campaign mainly remain unchanged, although spammers usually insert random text or links [6]. The selected attributes extends the set of structural features proposed in [18] to label emails as spam or ham. The attributes and a brief description are presented in Table 1.

Table 1. Features extracted from each email.

Attribute	Description
`RecipientNumber`	Number of recipients addresses.
`NumberOfLinks`	Total links in email text.
`NumberOfIPBasedLinks`	Links shown as an IP address.
`NumberOfMismatchingLinks`	Links with a text different from the real link.
`NumberOfDomainsInLinks`	Number of domains in links.
`AvgDotsPerLink`	Average number of dots in link in text.
`NumberOfLinksWithAt`	Number of links containing "@".
`NumberOfLinksWithHex`	Number of links containing hex chars.
`SubjectLanguage`	Language of the subject.
`NumberOfNonAsciiLinks`	Number of links with non-ASCII chars.
`IsHtml`	True if the mail contains html tags.
`EmailSize`	The email size, including attachments.
`Language`	Email language.
`AttachmentNumber`	Number of attachments.
`AttachmentSize`	Total size of email attachments.
`AttachmentType`	File type of the biggest attachment.
`WordsInSubject`	Number of words in subject.
`CharsInSubject`	Number of chars in subject.
`ReOrFwdInSubject`	True if subject contains "Re" or "Fwd".
`NonAsciiCharsInSubject`	Number of non ASCII chars in subject.
`ImagesNumber`	Number of images in the email text.

Since the clustering algorithm is *categorical*, all selected features are categorical as well. It is worth noting that some features are meant to represent *numerical* values, e.g. `AttachmentSize`, instead that categorical ones. However, it is always possible to turn these features from numerical into categorical, defining intervals and assigning a feature value to each interval defined in such a way. We chose these intervals on the base of the *ChiMerge* discretization method [15], which returns outstanding results for discretization in decision tree-like problems [10].

3.2 Implementation Details

On the implementation side, an email parser has been developed in Java to automatically analyze raw mails text and extract the features in form of vectors. The software exploits the `JSoup` [14] for HTML parsing, and of the `LID`[1] Python tools for language recognition. The LID software exploits the technique of n-grams to recognize the language of a text. For each language that LID has to recognize, a database of words must be provided to the software, in order to extract n-grams. The language on which LID has been trained are the following: English, Italian, French, German, Spanish, Portuguese, Chinese, Japanese, Persian, Arabic, Croatian.

We have implemented the CCTree algorithm using the MATLAB[2] software, which takes as input the matrix of emails features extracted by the parser.

It is worth noting that the complete framework, i.e. feature extraction and clustering module, are totally portable on different operative system. In fact, both the feature extraction module and the clustering module (i.e. MATLAB) are Java-based and executable on the vast majority of general purpose operative system (Java, UNIX, iOS, etc.). Also the Python module for language analysis it is portable. Moreover, LID has been made as a disposable component, i.e. if the Python interpreter is missing, the analysis is not stopped. For the emails where the language is not inferable, the `UKNOWN_LANGUAGE` value for the attribute is used instead.

4 Evaluation and Results

This section reports the experimental results to evaluate the quality of the CCTree algorithm on the problem of clustering spam emails. A first set of experiments has been performed on a dataset of 10k recent spam emails (first week of February 2015 [1]), to estimate the capability of the CCTree algorithm in obtaining homogeneous clusters. This evaluation is known as *Internal Evaluation* and estimates the quality of the clustering algorithm, measuring how much each element of the resulting cluster is similar to the elements of the same cluster and dissimilar from the elements of other clusters. A second set of experiments aims at assessing the capability of CCTree to correctly classify data using a small dataset with benchmark classes known beforehand. This evaluation is named *External Evaluation* and measures the similarity between the resulting clusters of a specific algorithm and the desired clusters (classes) of the pre-classified dataset. For external evaluation, CCTree has been tested against a dataset of 276 emails, manually labeled in 29 classes[3]. The emails have been manually divided, looking both at the structure and the semantic of the message. Thus, emails belonging to one class can be considered as part of a single spam campaign. The results of CCTree are compared with those of two categorical clustering

[1] http://www.cavar.me/damir/LID/.
[2] http://mathworks.com.
[3] Available at: http://security.iit.cnr.it/images/Mails/cctreesamples.zip.

algorithms, namely COBWEB and CLOPE. COBWEB and CLOPE have been selected among the others, since they are well known to be respectively very accurate and extremely fast. The comparison has been done both for internal and external evaluation on the same aforementioned datasets. A time performance analysis is also reported. It is worth noting that the three algorithms are all implemented on Java-based tools, hence the validity of time comparison.

COBWEB [9] is a categorical clustering algorithm, which builds a dendrogram where each node is associated with a conditional probability which summarizes the attribute-value distributions of objects belonging to a specific node. Differently from the CCTree algorithm, also includes a merging operation to join two separate nodes in a single one. COBWEB is computationally demanding and time consuming, since it re-analyzes at each step every single data point. However, the COBWEB algorithm is used in several fields for its good accuracy, in a way that its similarity distance measure, named Category Utility, is used to evaluate categorical clustering accuracy [3]. The WEKA [12] implementation of COBWEB has been used for the experiments.

CLOPE [32] is a fast categorical clustering algorithm which maximizes the number of elements with the same value for a subset of attributes, attempting to increase the homogeneity of each obtained cluster. Also for CLOPE we have used the WEKA implementation for the performed experiments.

4.1 Internal Evaluation

Internal evaluation measures the ability of a clustering algorithm in obtaining homogeneous clusters. A high score on internal evaluation is given to clustering algorithms which maximize the *intra-cluster similarity*, i.e. elements within the same cluster are similar, and minimize the *inter-cluster similarity*, i.e. elements from different clusters are dissimilar. The cluster dissimilarity is measured by computing the distances between elements (data points) in various clusters. The used distance function changes for the specific problem. In particular, for elements described by categorical attributes, the common geometric distances, e.g. Euclidean distance, cannot be used. Hence, in this work the *Hamming* and *Jaccard* distance measures [13] are applied. Internal evaluation can be performed directly on the dataset on which the clustering algorithm operates, i.e. the knowledge of the classes (desired clusters) is not a prerequisite. The internal evaluation indexes are the Dunn Index [4] and the Silhouette [23], defined as follows:

Dunn Index. Let Δ_i be the diameter of cluster C_i, that can be defined as the maximum distance of elements of C_i: $\Delta_i = \max_{x,y \in C_i\ ,\ x \neq y}\{d(x,y)\}$, where $d(x,y)$ measures the distance of pair x and y, and $|C|$ shows the number of elements belonging to cluster C. Also, let $\delta(C_i, C_j)$ be the inter-cluster distance between clusters C_i and C_j, which is calculated as the pairwise distance between elements of two clusters. Then, on a set of k clusters, the *Dunn index* [11], is defined as: $DI_k = \min_{1 \leq i \leq k}\{\min_{1 \leq j \leq k}\{\frac{\delta(C_i,C_j)}{\max_{1 \leq t \leq k} \Delta_t}\}\}$.

A higher Dunn index value means a better cluster quality. It is worth noting that the value of Dunn index is negatively affected by the greatest diameter

between the elements of all generated clusters ($\max_{1 \leq t \leq k} \Delta_t$). Hence, even a single resulting cluster with poor quality (non homogeneous), will cause a low value of the Dunn index.

Silhouette. Let $d(x_i)$ be the average *dissimilarity* of data point x_i with other data points within the same cluster. Also, let $d'(x_i)$ be the lowest average dissimilarity of x_i to any other cluster, except the cluster that x_i belongs to. Then, the *silhouette* $s(i)$ for x_i is defined as:

$$s(i) = \frac{d'(i) - d(i)}{\max\{d(i), d'(i)\}} = \begin{cases} 1 - \frac{d(i)}{d'(i)} & d(i) < d'(i) \\ 0 & d(i) = d'(i) \\ \frac{d'(i)}{d(i)} - 1 & d(i) > d'(i) \end{cases}$$

where the definition result in $s(i) \in [-1, 1]$. As much as $s(i)$ is closer to 1, the more the data point x_i is appropriately clustered. The average value of $s(i)$ over all data of a cluster, shows how tightly related are data within a cluster. Hence, the more the average value of $s(i)$ is close to 1, the better is the clustering result. For easy interpretation, the silhouette of all clustered points is also represented through a *silhouette plot*.

Performance Comparison. As discussed in Sect. 2, CCTree algorithm requires two stop conditions as input, i.e. the minimum number of elements in a node to be split (μ), and minimum purity in a cluster (ε). Henceforth, the notation `CCTree`(ε, μ) will be used to refer the specific implementation of the CCTree algorithm.

Figure 2 graphs the internal evaluation measurements of CCTree with five different values of ε, when the minimum number of elements μ has been set to 1. It is worth noting that if $\mu = 1$, the only stop condition affecting the result is the node purity. Thus, we first fix $\mu = 1$ to find the best amount of required node purity for our dataset.

As shown in Fig. 2, the purity value reach the maximum and stabilize when $\varepsilon = 0.001$. More strict purity requirements (i.e., $\varepsilon < 0.001$) do not further increase the precision. This value of ε will be fixed for the following evaluations.

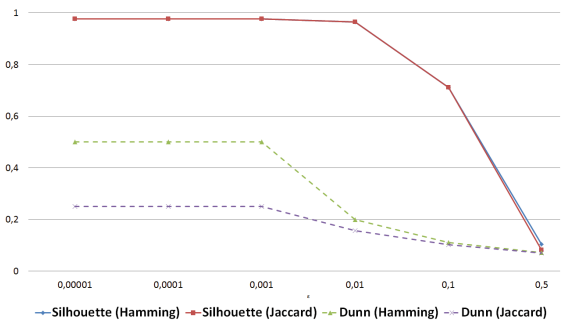


Fig. 2. Internal evaluation at the variation of the ε parameter.

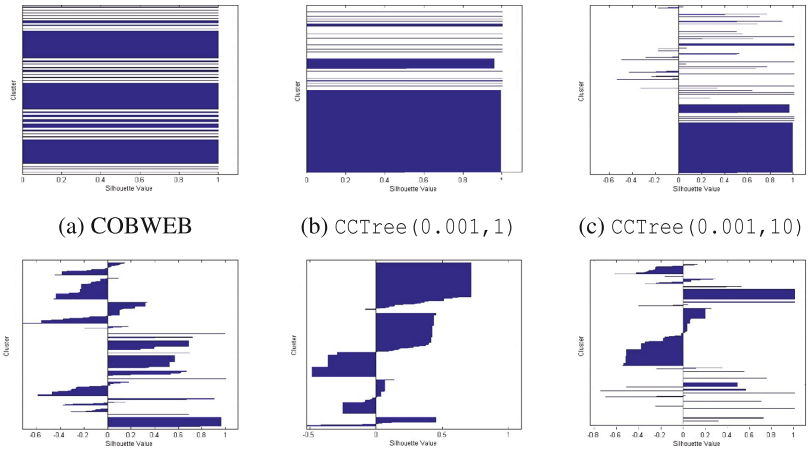
Fixing the node purity $\varepsilon = 0.001$, we look for the better value for the μ parameter to be able to compare CCTree performance with accurate COBWEB and fast CLOPE. To this end, we provide four different values of minimum number of elements in a cluster. Table 2 presents the Silhouette and Dunn index results for proposed values of μ, namely 1, 10, 100, and 1000. In addition, the last two rows of Table 2 reports the resulting number of clusters and the *time* required to generate the clusters. Table 2 also reports the comparison with the two categorical clustering algorithms COBWEB and CLOPE, previously described. The first two columns from left, show comparable results in term of clustering precision for the silhouette index. In fact, COBWEB and CCTree have both a good precision, when the CCTree purity is set to $\varepsilon = 0.001$ and the minimum number of elements is set to $\mu = 1$ (`CCTree(0.001,1)`). COBWEB performs slightly better on the silhouette index, for both distance measures. However, the difference (less than 2 percent) is negligible by considering that COBWEB creates almost twice more the number of clusters rather than `CCTree(0.001,1)`. It can be inferred that a higher number of small clusters improves the internal homogeneity (e.g., a cluster with one element is totally homogeneous). However, as it will be detailed in the following, a number of clusters, strongly greater than the expected number of groups, is not desirable. Moreover, `CCTree(0.001,1)` returns better result for the Dunn index, with respect to COBWEB. We recall that the value of Dunn index is strongly affected by the cluster homogeneity of the worst resulting cluster. The value returned for `CCTree(0.001,1)` shows that all the returned clusters globally have a good homogeneity, compared to COBWEB, i.e. the worst cluster for `CCTree(0.001,1)` is much more homogeneous than the worst cluster for COBWEB. The rightest column of Table 2 reports the results for the CLOPE clustering algorithm. CLOPE is a categorical clustering algorithm, known to be fast in creating as much as possible pure clusters. The accuracy of CLOPE is quite limited for Silhouette, and zero for the Dunn index.

Table 2. Internal evaluation results of CCTree, COBWEB and CLOPE.

Algorithm	COBWEB	CCTree - $\varepsilon = 0.001$				CLOPE
		$\mu = 1$	$\mu = 10$	$\mu = 100$	$\mu = 1000$	
Silhouette (Hamming)	0.9922	0.9772	0.9264	0.7934	0.5931	0.2801
Silhouette (Jaccard)	0.9922	0.9777	0.9290	0.8021	0.6074	0.2791
Dunn (Hamming)	0.1429	0.5	0.1	0.0769	0.0769	0
Dunn (Jaccard)	0.1327	0.25	0.1	0.0879	0.0857	0
Clusters	1118	619	392	154	59	55
Time (s)	17.81	0.6027	0.3887	0.1760	0.08610	3.02

A graphical description of the accuracy difference between the clustering of Table 2 can be inferred from the Hamming Silhouette plots of Fig. 3. The plots are horizontal histograms in which every bar represents the silhouette result, $s(i) \in [-1, 1]$, for each data point x_i, as from the aforementioned definition. Both

COBWEB and `CCTree(0.001,1)` show no negative values, with the majority of data points scoring $s(i) = 1$. In fact, for `CCTree(0.001, 1000)` the worst data points do not score less than -0.5, whilst for CLOPE some data points have a silhouette of -0.8, which will cause a strong non-homogeneity in their clusters. Also, the number of data point with positive values are much more for `CCTree(0.001,1000)`, than for CLOPE. Finally, Table 2 also reports the time elapsed for the clustering performed by the algorithms. It can be observed that COBWEB pay its accuracy with an elapsed time of 17 s on the dataset of 10k emails, against the 3 s of the much more inaccurate CLOPE. The CCTree algorithm outperforms both COBWEB and CLOPE, requiring only 0.6 s in the most accurate configuration (`CCTree(0.001,1)`). From internal evaluation we can thus conclude that the CCTree algorithm obtains clusters whose quality is comparable with the ones of COBWEB, requiring even less computational time than the fast but inaccurate algorithm CLOPE.



(a) COBWEB (b) `CCTree(0.001,1)` (c) `CCTree(0.001,10)`

(d) `CCTree(0.001,100)` (e) `CCTree(0.001,1000)` (f) CLOPE

Fig. 3. Silhouette plot.

4.2 CCTree Parameters Selection

Through internal evaluation and the results reported in Tables 2 and 3, we showed the dependence of the internal evaluation indexes and number of clusters to the values of μ and ε parameters. We will briefly discuss here some guidelines to correctly choose the CCTree parameters to maximize the clustering effectiveness.

Concerning the ε parameter, we showed in Sect. 4 that it is possible to find the optimal value of ε by setting $\mu = 1$ and varying the ε to find the fixed point in terms of accuracy, i.e. the optimal ε was considered the one for which the lesser amount of ε is not improving the accuracy.

Fixed the parameter ε, the parameter μ must be selected to balance the accuracy with the number of generated clusters. As the number of cluster is

affected by the μ parameter, it is possible to choose the optimal value of μ knowing the optimal number of clusters. The problem of estimating the optimal number of clusters for hierarchical clustering algorithms, has been solved in [24], by determining the point of maximum curvature (*knee*) on a graph showing the inter-cluster distance in function of the number of clusters. Recalling that silhouette index is inversely related to inter-cluster distance, it is sound to find the knee on the graph of Fig. 4 computed with the silhouette (Hamming) on the dataset used for internal evaluation, with seven different values of μ. The graph reports the values computed on the same dataset used for internal evaluation. For the sake of representation, we do not show in this graph plots for μ greater than 100. Applying the L-method described in [24], it is possible to find that the knee is located at $\mu = 10$. A further insight can be taken from the results of Table 3 and Fig. 5, reporting the analysis on three additional datasets of spam emails coming from three different weeks of March 2015 from the spam honeypot in [1]. The sets have comparable size with the one of the dataset used for internal evaluation (first week of February 2015), with respectively 10k, 10k and 9k spam emails. From both the table and the graph it is possible to infer how the same trend for both silhouette value and number of clusters holds for all the tested datasets. Hence we verify (i) the validity of the knee methodology, (ii) the possibility of using the same CCTree parameters for datasets with the same data type and comparable size.

To give statistical validity to the performed analysis on parameter determinacy, we have analyzed a dataset of more than 200k emails collected from October 2014 to May 2015 from the honey pot in [1]. The emails have been divided in 20 datasets, containing 10k spam emails each. Each set represents one week

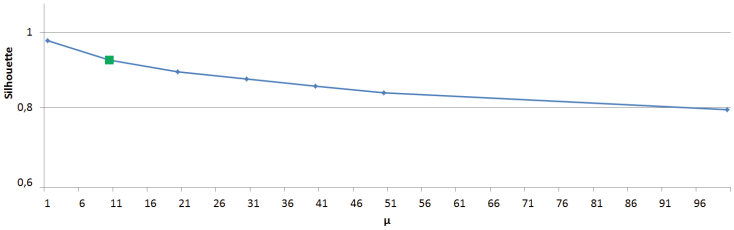
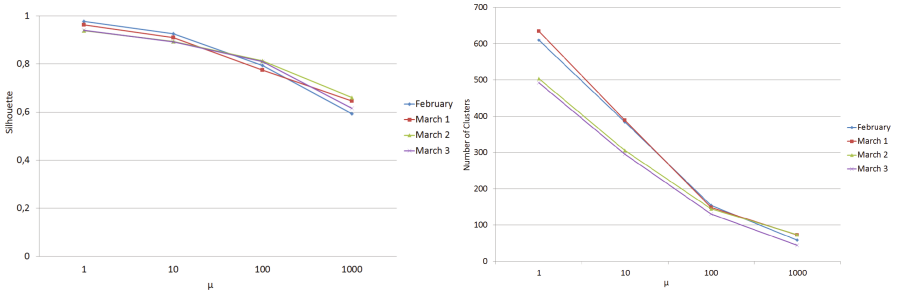


Fig. 4. Silhouette in function of the number of clusters for different values of μ.

Table 3. Silhouette values and number of clusters in function of μ for four email datasets.

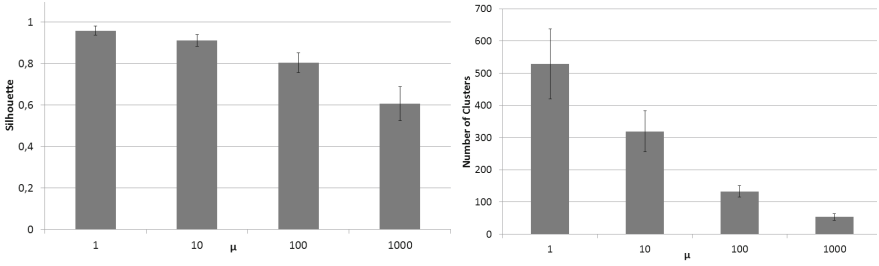
Data	$\mu = 1$		$\mu = 10$		$\mu = 100$		$\mu = 1000$	
	Silhouette	Clusters	Silhouette	Clusters	Silhouette	Clusters	Silhouette	Clusters
February	0.9772	610	0.9264	385	0.7934	154	0.5931	59
March I	0.9629	635	0.9095	389	0.7752	149	0.6454	73
March II	0.9385	504	0.8941	306	0.8127	145	0.6608	74
March III	0.9397	493	0.8926	296	0.8102	131	0.6156	44

of spam emails. Figure 5 shows the average values for number of clusters and silhouette computed on the 20 dataset varying the value of the μ parameter with the values of the former experiments (i.e. 1, 10, 100, 1k). The standard deviation is also reported as error bars. It is worth noting that, the standard deviation for the values of $\mu = \{1, 10\}$ on 20 datasets is slightly higher than 2 %, while it reaches 4 % for $\mu = 100$ and 8 % for $\mu = 1000$, which is in line with the results of Table 3. Comparable results are also obtained for the number of clusters where the highest value for standard deviation is, as expected, for $\mu = 1$, amounting to 108, which again is in line with the results of Table 3. Thus, for all the 20 analyzed datasets, spanning eight months of spam emails, we can always locate the knee for silhouette and number of clusters when $\mu = 10$ (Fig. 6).



(a) Sihouette (Hamming). (b) Generated Clusters.

Fig. 5. Trends of silhouette value and cluster number for four spam campaigns.



(a) Sihouette (Hamming). (b) Generated Clusters.

Fig. 6. Average and standard deviation for silhouette value and amount of generated clusters in function of the μ parameter for 20 datasets.

4.3 External Evaluation

The external evaluation is a standard technique to measure the capability of a clustering algorithm to correctly classify data. To this end, external evaluation is performed on a small dataset, whose classes, i.e. the desired clusters, are known

beforehand. A common index used for external evaluation is the F-measure [17], which coalesces in a single index the performance about correctly classified elements and misclassified ones.

Formally, let the sets $\{C_1, C_2, \ldots, C_k\}$ be the desired clusters (classes) for the dataset D, and let $\{C'_1, C'_2, \ldots, C'_l\}$ be the set of clusters returned by applying a clustering algorithm on D. Then, the F-measure $F(i)$ for each cluster C_i and the global F-measure F_c on the dataset are defined as follows:

$$F(i) = \max_{i,j} \frac{2|C_i \cap C'_j|}{|C_i| + |C'_j|} \qquad F_c = \sum_{i=1}^{k} F(i) \frac{|C_i|}{|\cup_{j=1}^{k} C_j|}$$

The F-measure result is returned in the range [0,1], where 1 represents the ideal situation in which the cluster C_i is exactly equal to one of the resulted clusters.

Experimental Results. For the sake of external evaluation, 276 spam emails collected from different spam folders of different mailboxes has been manually analyzed and classified. Emails have been divided in 29 groups (classes) according to the structural similarity of raw email message. The external evaluation set has no intersection with the one used for internal evaluation.

Table 4. External evaluation results of CCTree, COBWEB and CLOPE.

Algorithm	COBWEB	CCTree - $\varepsilon = 0.001$				CLOPE
		$\mu = 1$	$\mu = 5$	$\mu = 10$	$\mu = 50$	
F-Measure	0.3582	0.62	0.6331	0.6330	0.6	0.0076
Clusters	194	102	73	50	26	15

The results of external evaluation are reported in Table 4. Building on the results of internal evaluation, the value of node purity has been set to $\varepsilon = 0.001$ to obtain homogeneous clusters. The values of μ have been chosen according to the following rationale. $\mu = 1$ represents a CCTree instantiation in which the μ parameter does not affect the result. On the other hand $\mu = 50$ returns a number of clusters comparable with the 29 clusters manually collected. Higher values of μ do not modify the result for a dataset of this size. The best results are returned for $\mu = \{5, 10\}$. The F-measure for these two values is higher than 0.63, with a negligible difference, even if the number of generated clusters is higher than the expected one.

Table 4 also reports the comparison with the COBWEB and CLOPE algorithms. As showns CCTree algorithm outperforms COBWEB and CLOPE for the F-measure index, showing thus a higher capability in correctly classifying spam emails. We recall that for internal evaluation, COBWEB returned slightly better results than CCTree. The reason of this difference is resulted from the number of resulting clusters. COBWEB, in fact, always returns a high number of clusters (Table 4). This generally yields a high cluster homogeneity (several

small and homogeneous clusters). However, it not necessarily implies a good classification capability. In fact, as shown in Table 4, COBWEB returns almost 200 clusters on a dataset of 276 emails, which is six times the expected number of clusters (i.e., 29 clusters). This motivates the lower F-measure score for the COBWEB algorithm. It is worth noting that the CCTree outperform COBWEB even if the minimum number of elements per node is not considered (i.e., $\mu = 1$). On the other hand, CLOPE also performs poorly on F-measure for the 276 emails dataset. The CLOPE algorithm, in fact, only produced 15 clusters, i.e. less than half of the expected ones, with an F-measure score of 0.0076.

5 Discussion and Comparisons

Combining the analysis of 21 features, the proposed methodology, becomes suitable to analyze almost any kind of spam emails. This is one of the main advantages with respect to other approaches, which mainly exploit one or two features to cluster spam emails into campaigns. These features are links [2,16], keywords [5,8,25], or images [33] alternatively. The analysis of these methodologies remains limited to those spam emails that effectively contain the attributed features. However, emails without links and/or images are a consistent percentage of spam emails. In fact, from the analysis of the dataset used for internal evaluation, 4561 emails out of 10165 do not contain any link. Furthermore, only 810 emails are containing images. To verify the clustering capability of these methodologies we have implemented three programs to cluster the emails of the internal evaluation dataset on the base of the contained URLs, reported domains and links of remote images. The emails without links and pictures have not been considered. Table 5 reports the generated number of clusters for each methodology. It is worth noting that on large dataset these cluster methodologies are highly inaccurate, generating a number of campaigns close to the number of analyzed elements, hence, almost every cluster has a single element. For comparison purpose we reported the results of the most accurate implementation of CCTree and of COBWEB, which we recall being able to produce extremely homogeneous cluster, reporting a Silhouette value of 99 %. We point out that, comparing Silhouette is meaningless, due to the different sets of used features. Comparisons with other methodologies such as the FPTree-based approaches

Table 5. Campaigns on the February 2015 dataset from five clustering methodologies.

Cluster methodology	Analyzed emails	Generated campaigns
Link based clustering	4561	4449
Domain based clustering	4561	4547
Image based clustering	810	807
COBWEB (21 features)	10165	1118
CCTree (0.001,10)	10165	392

[6,7], which require the extraction and analysis of a different set of features, are left as a future work.

6 Related Work

Another clustering approach exploiting a pairwise comparisons of email subjects is presented in Wei et al. [31]. The proposed methodology introduces a set of eleven features to cluster spam emails through two clustering algorithms. At first an agglomerative hierarchical algorithm is used to cluster the whole data set based on subject pairwise comparison. Afterward, the *connected component graph* algorithm is used to improve performance. The authors of [29] applied a methodology based on k-means algorithm, named O-means clustering, which exploits twelve features extracted from each email. The O-mean algorithm works on the hypothesis that the number of clusters is known beforehand, which is not always a working hypothesis. Furthermore, the authors use Euclidean distance, which for several features that they apply, it does not bring meaningful information. Differently from this approach the CCTree exploits the more general distance measure, i.e. Shannon entropy. Moreover the CCTree does not require the desired number of clusters as input parameter. An application of the presented approach in a goal-based spam classification problem, is discussed in [28].

Frequent Pattern Tree (FP-Tree), is another technique applied to detect spam campaigns in large datasets. The authors of [6,7] extract set of features from each spam message. The FP-Tree is built based on the frequency of features. The sensitive representation of both message layout and URL features, causes that two spam emails with small difference be assigned to different campaigns. For this reason, FP-Tree approach becomes prone to text obfuscation techniques [19], used to deceive anti-spam filters and to emails with dynamically generated links. Our methodology, based on categorical features which do not consider text and link semantics is more robust against these deceptions.

7 Conclusion and Future Directions

In this paper we proposed a methodology, based on a categorical clustering algorithm named CCTree, to cluster large amount of spam emails in campaigns, grouping them by structural similarity. The set of features representing message structure, have been precisely chosen and the intervals for each feature has been found through discretization approach. The CCTree algorithm has been extensively tested on two dataset of spam emails, to measure both the capability in generating homogeneous clusters and the specificity in recognizing predefined groups of spam emails. The guideline for selecting CCTree parameters is provided, whilst the determinacy of selected parameter for the similar data set with the same size has been proven statistically. Considering the proven accuracy and efficiency, the proposed methodology may stand as a valuable tool to help authorities in rapidly analyzing large amount of spam emails, with the purpose of finding and persecuting spammers.

As a future work we plan to extend the proposed framework, to target specific attacks brought through spam emails. In particular we argue that it is possible to extract from the CCTree structure, rules to recognize emails related to well known cyber-crimes such as phishing or scam. The framework will be used to rapidly detect and group spam campaigns related to attributed cyber-crimes, among large sets of real spam emails.

References

1. Spam archive. http://untroubled.org/spam/
2. Anderson, D., Fleizach, C., Savage, S., Voelker, G.: Spamscatter: Characterizing internet scam hosting infrastructure. In: Proceedings of 16th USENIX Security Symposium (2007)
3. Andritsos, P., Tsaparas, P., Miller, R.J., Sevcik, K.C.: LIMBO: scalable clustering of categorical data. In: Bertino, E., Christodoulakis, S., Plexousakis, D., Christophides, V., Koubarakis, M., Böhm, K. (eds.) EDBT 2004. LNCS, vol. 2992, pp. 123–146. Springer, Heidelberg (2004)
4. Bezdek, J., Pal, N.: Cluster validation with generalized dunn's indices. In: Proceedings of Second New Zealand International Two-Stream Conference on Artificial Neural Networks and Expert Systems, pp. 190–193 (1995)
5. Blanzieri, E., Bryl, A.: A survey of learning-based techniques of email spam filtering. Artif. Intell. Rev. **29**(1), 63–92 (2008)
6. Calais, P., Pires, D., Guedes, D., Meira, W., Hoepers, C., Steding-Jessen, K.: A campaign-based characterization of spamming strategies. In: CEAS (2008)
7. Dinh, S., Azeb, T., Fortin, F., Mouheb, D., Debbabi, M.: Spam campaign detection, analysis, and investigation. Digit. Invest. **12**(1(0)), S12–S21 (2015). DFRWS 2015 Europe Proceedings of the Second Annual DFRWS Europe
8. Drucker, H., Wu, D., Vapnik, V.: Support vector machines for spam categorization. IEEE Trans. Neural Netw. **10**(5), 1048–1054 (1999)
9. Fisher, D.: Knowledge acquisition via incremental conceptual clustering. Mach. Learn. **2**(2), 139–172 (1987)
10. Garcia, S., Luengo, J., Saez, J.A., Lopez, V., Herrera, F.: A survey of discretization techniques: taxonomy and empirical analysis in supervised learning. IEEE Trans. Knowl. Data Eng. **25**(4), 734–750 (2013)
11. Halkidi, M., Vazirgiannis, M.: Clustering validity assessment: finding the optimal partitioning of a data set. In: Proceedings of IEEE International Conference on Data Mining, ICDM 2001, pp. 187–194 (2001)
12. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The weka data mining software: an update. SIGKDD Explor. Newslett. **11**(1), 10–18 (2009)
13. Han, J., Kamber, M., Pei, J.: Data Mining: Concepts and Techniques, 3rd edn. Morgan Kaufmann Publishers Inc., San Francisco (2011)
14. Hedley, J.: Jsoup cookbook (2009). http://jsoup.org/cookbook
15. Kerber, R.: Chimerge: discretization of numeric attributes. In: Proceedings of the Tenth National Conference on Artificial Intelligence, AAAI 1992, pp. 123–128. AAAI Press (1992)
16. Li, F., Hsieh, M.: An empirical study of clustering behavior of spammers and groupbased anti-spam strategies. In: CEAS 2006 Third Conference on Email and AntiSpam, pp. 27–28 (2006)

17. Manning, C.D., Prabhakar, R., Schütze, H.: Introduction to Information Retrieval. Cambridge University Press, New York (2008)
18. Martin, S., Nelson, B., Sewani, A., Chen, K., Joseph, A.D.: Analyzing behavioral features for email classification. In: CEAS (2005)
19. Pu, C., Webb, S.: Observed trends in spam construction techniques: a case study of spam evolution. In: CEAS, pp. 104–112 (2006)
20. Radicati, S.: Email statistics report 2013–2017 (2013). http://goo.gl/ggLntn
21. Ramachandran, A., Feamster, N.: Understanding the network-level behavior of spammers. ACM SIGCOMM Comput. Commun. Rev. **36**(4), 291–302 (2006)
22. Rao, J., Reiley, D.: On the spam campaign trail, the economics of spam. J. Econ. Perspect. **26**(3), 87–110 (2012)
23. Rousseeuw, P.J.: Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. J. Comput. Appl. Math. **20**, 53–65 (1987)
24. Salvador, S., Chan, P.: Determining the number of clusters/segments in hierarchical clustering/segmentation algorithms. In: Proceedings of the 16th IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2004, pp. 576–584. IEEE Computer Society, Washington, DC (2004)
25. Seewald, A.: An evaluation of naive bayes variants in content-based learning for spam filtering. Intell. Data Anal. **11**(5), 497–524 (2007)
26. Shannon, C.E.: A mathematical theory of communication. SIGMOBILE Mob. Comput. Commun. Rev. **5**(1), 3–55 (2001)
27. Sheikhalishahi, M., Mejri, M., Tawbi, N.: Clustering spam emails into campaigns. In: Library, S.D. (ed.) 1st International Conference on Information Systems Security and Privacy (2015)
28. Sheikhalishahi, M., Saracino, A., Mejri, M., Tawbi, N., Martinelli, F.: Digital waste sorting: a goal-based, self-learning approach to label spam email campaigns. In: Foresti, S. (ed.) STM 2015. LNCS, vol. 9331, pp. 3–19. Springer, Heidelberg (2015)
29. Song, J., Inque, D., Eto, M., Kim, H., Nakao, K.: O-means: an optimized clustering method for analyzing spam based attacks. IEICE Trans. Fundam. Electron. Commun. Comput. Sci. **94**, 245–254 (2011)
30. Tretyakov, K.: Machine learning techniques in spam filtering. In: Data Mining Problem-Oriented Seminar, MTAT, vol. 3, pp. 60–79. Citeseer (2004)
31. Wei, C., Sprague, A., Warner, G., Skjellum, A.: Mining spam email to identify common origins for forensic application. In: Proceedings of the 2008 ACM Symposium on Applied Computing, SAC 2008, pp. 1433–1437 (2008)
32. Yang, Y., Guan, X., You, J.: Clope: A fast and effective clustering algorithm for transactional data. In: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2002, pp. 682–687. ACM, New York, USA (2002)
33. Zhang, C., Chen, W., Chen, X., Warner, G.: Revealing common sources of image spam by unsupervised clustering with visual features. In: Proceedings of the 2009 ACM Symposium on Applied Computing, SAC 2009, pp. 891–892. ACM, New York, USA (2009)