Information Sciences 262 (2014) 137–158

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

CoBAn: A context based model for data leakage prevention

Gilad Katz *, Yuval Elovici, Bracha Shapira

Telekom Innovation Labs at Ben Gurion University and Department of Information Systems Engineering, Ben-Gurion University of the Negev, Beer Sheva, Israel

ARTICLE INFO

Article history:
Received 31 July 2011
Received in revised form 21 June 2013
Accepted 4 October 2013
Available online 18 October 2013

Keywords:
Information leakage
Security
Context

ABSTRACT

A new context-based model (CoBAn) for accidental and intentional data leakage prevention (DLP) is proposed. Existing methods attempt to prevent data leakage by either looking for specific keywords and phrases or by using various statistical methods. Keyword-based methods are not sufficiently accurate since they ignore the context of the keyword, while statistical methods ignore the content of the analyzed text. The context-based approach we propose leverages the advantages of both these approaches. The new model consists of two phases: training and detection. During the training phase, clusters of documents are generated and a graph representation of the confidential content of each cluster is created. This representation consists of key terms and the context in which they need to appear in order to be considered confidential. During the detection phase, each tested document is assigned to several clusters and its contents are then matched to each cluster's respective graph in an attempt to determine the confidentiality of the document. Extensive experiments have shown that the model is superior to other methods in detecting leakage attempts, where the confidential information is rephrased or is different from the original examples provided in the learning set.

1. Introduction

Information leakage is defined by [64] as the "accidental or unintentional distribution of private or sensitive data to an unauthorized entity". The definition of "sensitive data" is wide and can include (among other things) financial data, intellectual property, and customer details. In addition, once information has leaked it is nearly impossible to stop it from spreading.

One of the greatest threats to information security is the leakage of data by the organization's own employees. One common threat arises from the use of email. Confidential information can be sent to individuals outside the organization very easily and cause serious damage. There are many examples of such incidents:

1. In May 2000, a Walt Disney CEO accidentally sent out an email containing the company's quarterly earnings to a reporter, just prior to a public announcement about them.
2. In June 2001, an Eli Lilly employee sent an email to a group of e-mail service subscribers informing them that the service was about to be terminated. Due to a human error, the "To:" line contained all of the subscribers' names and email addresses, exposing the whole list.
3. In June 2002, a confidential e-mail containing information regarding Prince Charles' visit to Poland was accidentally sent to an incorrect address.

* Corresponding author. Tel.: +972 8 6479347; fax: +972 8 6479346.
E-mail address: katz.gilad@gmail.com (G. Katz).

0020-0255/$ - see front matter © 2013 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.ins.2013.10.005

Various laws and regulations have been passed, requiring organizations to protect their clients' private information. These include the Sarbanes–Oxley Act (SOX)[1]; HIPPA,[2] and the Gramm-Leach Bliley act.[3] These laws focus on specific types of client information. The HIPAA, for example, mainly deals with medical records, while the Sarbanes–Oxley Act focuses on financial information. Infractions of these statues carry severe punishments with violators facing hundreds of thousands of dollars in fines and up to 20 years in jail.

To address the many problems that arise from data leakage and content protection, many software and hardware solutions have been developed. The hardware solutions include the use of encrypted communication, fingerprint readers, and the disabling or removal of USB ports on computers. The software solutions are even more diverse and are discussed in Section 2 below.

In this study, we present a new context based model (CoBAn), which can deal with accidental and intentional data leakage via an organization's monitored channels of communication. Our model generates a representation of the confidential content of the documents with the context in which it is used. We propose an algorithm that can generate this representation and train a system with confidential content; another algorithm detects confidential documents. Our solution can accurately detect a document containing confidential information even when most of the document consists of non-confidential content, including cases in which the confidential content *has been rephrased*. This property is very important, as mixed-content documents (confidential and non-confidential) are very common. For example, an employee may send a non-confidential email message to her friend (unintentionally or intentionally), which also mentions a confidential project within it.

The remainder of this paper is organized as follows. In Section 2, we present previous work on content protection and data leakage prevention. In Section 3, we present the proposed model and the detection and decision algorithms. The experiments conducted to evaluate the model are discussed in Section 4. Section 5 concludes the paper with a summary and discussion of future research directions (see Tables 1–3).

2. Related work

In this section we review three topics relevant to this study: information leakage detection and prevention (ILD&P) methods; graph representation of textual documents; and the use of graph representations in the field of information security.

2.1. Information Leakage Detection & Prevention (ILD&P) methods

Considering the enormity of the threat of information leakage, there has been relatively little published research on the matter. Existing research can be divided into two main areas: content and behavior-based methods.

The content-based approach includes the *rule* and *classifier-based* approaches. In the rule-based approach, various rules are defined with regard to words and terms that may appear in a scanned text. When used to protect information, these rules determine the "confidentiality level" of the scanned text based on the number of appearances of certain words and/or phrases. This technique is discussed in various academic studies [32,11,55,67] and is implemented in several commercial products[4] as well.

The well explored classifier-based approach consists of various classification and other machine learning techniques, such as support vector machines (SVM) [12,18] and naïve Bayes [57,5,36]. In this approach, documents are represented as vectors, while the terms of the documents and their frequencies are the features of the vectors [58]. These vectors form the learning set for a probabilistic model that classifies documents as confidential or not. These techniques are often used in spam detection, a field related to ILD&P [15,16].

Recent studies attempt to not only identify confidential content but to determine the level of threat its leakage presents to the organization; works such as [26,31] proposes a "score" for determining how detrimental a leak of the analyzed content would be and a framework for applying it. Others focus on the identification of entities (people, places, companies etc.') [28] as a mean of improving their detection abilities.

Another approach, which does not exactly fit in either category is *fingerprinting*, used mostly in commercial products. In fingerprinting, documents are represented as a set of strings generated by using a hash function on a sliding window that spans X characters/words of a document. Each tested document is analyzed in the same manner and its hash values are compared against those in the database. If a sufficient number of matches are found, the document is considered confidential. Fingerprinting is used in commercial products offered by companies engaged in computer security, such as Symantec[5] and Websense,[6] but is studied in academic literature in relation to plagiarism detection [35], finding near-duplicate files [47], authorship detection [63] and even website summarization [2] rather than in relation to leakage prevention.

The behavior-based approach to ILD&P focuses on identifying anomalies in behavior. These anomalies can be tracked in the communication, in and out of the organization as a whole [66], or the analysis of past and current email communication

[1] http://www.soxlaw.com/.
[2] http://www.hhs.gov/ocr/privacy/.
[3] http://www.ftc.gov/privacy/privacyinitiatives/glbact.html.
[4] http://www.symantec.com/business/theme.jsp?themeid=vontu.
[5] http://www.symantec.com/business/products/family.jsp?familyid=data-loss-prevention.
[6] http://www.websense.com/assets/white-papers/PA_Information_Identification_Fingerprinting.pdf.

for a single person [39]. Other studies propose the use of decision trees to access illegitimate access to customers' personal data [44] and the identification of similar behavior when accessing databases [49].

The above methods do not effectively handle the following type of scenarios: an employee of a certain organization attempts to leak confidential details about a new, well-publicized project, on which his firm is working (via email) to a journalist. The majority of the document is not confidential but it includes a paragraph containing sensitive technical data. The methods mentioned above would probably fail to detect the confidentiality of the document due to the following reasons:

(a) Rule-based systems are probably too rigid to deal with this sort of ILD&P, since they tend to suffer from a high rate of false-positives, according to [1]. Organizations that use rule-based systems for ILD&P generally set a high threshold for detecting confidential content in documents to minimize false-positive detection. Thus, a document that is mostly non-confidential (except one paragraph, for example) would probably go undetected by such systems. It should be noted that because of their high false-positive rate, rule-based methods are not very common nowadays.
(b) Classifiers, such as SVM and naïve Bayes are also likely to fail in detection of such documents, as most of the document is not confidential and statistic-based classifiers look at the most significant features when classifying documents (our experiments, presented in Section 4, reinforce this statement).
(c) The fingerprint detection method usually maintains low false-positives as it looks for *exact matches* to sequences that only appear in confidential documents. It will also perform well in scenarios where a confidential sequence is inserted into a non-confidential document using "copy-paste" (because a bulk of hash sequences could be detected in those cases, almost ensuring detection). However, we hypothesize that the fingerprint performance will deteriorate considerably when the confidential text differs from the training set examples (because of the rephrasing of the text, for example). Since the amount of confidential text is relatively small, the method will have difficulties in detecting a sufficient number of matches in the modified text.

The behavior-based methods – those that analyze communication patterns, for example – are useless for this type of data leakage scenario since the email was sent to its intended recipient and, as noted previously, the vast majority of email's content is not confidential.

In contrast to the above methods, CoBAn is capable of tackling the problems posed by this scenario. The proposed method takes the best of both the keyword and classifier-based approaches. First, we use clustering to group together documents of similar content. Then, we extract the confidential content of each cluster and represent it in a graph. The graph contains both the terms that can identify a document as confidential and the context in which they appear. When attempting to determine whether a document is confidential, the proposed model looks for the key terms *in a confidential* context, thus reducing false-positives.

2.2. Graph representation of textual documents

Graphs offer an alternative method to the vector-space model for representing textual information. In the literature, a large number of methods for creating graphs from documents have been suggested, [33,21,40] being just a few examples. Aside from replacing existing models for text representation, graphs have also been used in many text-related tasks such as summarization [70], entity linking for the improvement of information retrieval [30] and the augmentation of matrix factorization techniques [8]. Another interesting study [37] integrates graph (and sub-graphs) mining together with the extraction of textual features for the purposes of text classification, and [15] utilizes graphs for the problem of topic detection. In this review, we focus on methods for the representation of text using graphs.

Schenker [62] presented six major groups of graph-related algorithms: standard, simple, n-distance, n-simple distance, absolute frequency, and relative frequency. In general, these methods present a graph, in which words appear as nodes connected by edges to words that appear in their vicinity. The differences between the various algorithms are related to variations in term-based techniques, such as whether to represent the order of appearance in the document; presenting the distance between each word (up to a certain predefined distance) and whether the frequencies of the terms' appearances together should be calculated.

The main advantage of a graph-based model for a document rather than a common vector representation is that the former is capable of capturing the structure of the document as well as its content. Graph-based methods represent the proximity of words in a document as well as capture its structure.

When using the vector space model for text-related tasks, the distance between vectors (calculated by methods such as Euclidean distance or Cosine measure) is used to determine the similarity among documents. This method is quite efficient, since a single vector needs to be read only once for it to be compared to another. When using graphs, the problem of detecting/retrieving similar objects is referred to as *graph matching*. As mentioned in [68,25,56], the standard methods for graph matching are *graph isomorphism*, *subgraph isomorphism* [50], and *maximum common subgraphs* [43,22]. All these methods will find an optimal solution, but they are NP in their complexity. Various methods for overcoming these problems have been proposed, including neural networks [14], genetic algorithms [9] and probabilistic relaxation schemes. Recently, hybrid approaches, which combine graph and vector representations have been proposed by [48]. These methods appear to achieve better results than earlier approaches in terms of both accuracy and running time.

Our method uses graphs to represent textual content as well as to capture the context of the content. The graphs generated by our model refer to **confidential content** only and not to the whole document. The graphs we utilize include two types of nodes; confidential and *contextual*. The creation process of the graphs is described in Section 3.

2.3. The use of graph representations in the field of information security

To the best of our knowledge, graphs have not been used in the field of ILD&P for the purposes of text representation. However, they have been used in the fields of access control (AC) [52,41]; network vulnerability analysis [53,3,20]; malware detection [10]; and network forensics [69]. They have also been utilized in conjuncture with online social networks for the purpose of spam detection [27] and organization mining [23] (inferring the structure of an organization based on the social network of its employees). In all these studies, graphs are used to represent states and entities. The reasons graphs are used in these areas are that they are easy to modify and expand and because they assist users in understanding (using visualization), the connections between states, rules, and entities.

3. The proposed model

CoBAn consists of two phases, *learning* and *detection*. During the learning phase, a *context-based confidential terms graph* is generated for each type of the organization's confidential documents, using a training set. During the detection phase, documents are analyzed and matched to one or several graphs to calculate their *confidentiality score*. If this score exceeds a predefined threshold, the document is considered confidential. In the following subsections, we describe each of these phases in detail.

3.1. The learning phase

The detailed learning phase algorithm is presented in Algorithm 1, while Fig. 1 is its graphic representation. The objective of this phase is to generate a representation of the confidential content of each of the organization's fields of activity. This representation should include not only key words and terms, but also the context in which they appear and the strength of the connection between the two. This idea of an "augmented" representation has been previously proposed in related fields, such as information retrieval [38] and concept analysis [13].

The learning phase requires two sets of documents as input:

C – a set of **confidential** documents reflecting the complete set of the organization's confidential fields of activity (e.g., financial reports, client information, R&D, etc.).

N – a set of **non-confidential** documents. Some of the non-confidential documents may be of the same type or subject as the confidential ones, but not necessarily; this could be the case as there are types of documents, which are entirely confidential. More formally put, for each of the organization's areas of interest v:

- $v \subset N$ if the area of interest contains only non-confidential documents.
- $v \subset C$ if the area of interest contains only confidential documents.
- $v \cap C \neq \varnothing$, $v \cap N \neq \varnothing$ if the area of interest contains both confidential and non-confidential documents.

The learning phase begins by identifying the various subjects represented by *all* documents (both confidential and non-confidential). This is done by first applying stemming [45] and stop-words removal to the documents, transforming them into vectors of weighted terms [59] and applying *unsupervised clustering* [60,46,16,24]. For clustering, we used the k-means algorithm with the cosine measure as the distance function. The resulting clusters represent an approximation of the various subjects found in the dataset, many of which may contain both confidential and non-confidential documents.

The next step is aimed at representing the confidential content for all clusters containing confidential documents. This is accomplished by applying the following procedures:

(1) Detect the *confidential key terms*, which are the key terms that provide the initial indication of the existence of confidential content.
(2) For each of the detected confidential terms, analyze the *context in which it appears* in the learning set.
(3) Create a graph representation of the confidential content and context on the cluster level.

We now review each of these procedures in detail.

(1) **Detecting the confidential "key terms"**
The purpose of this step is to identify the terms,[7] which indicate with a high probability that a document in a cluster is confidential. We refer to these terms as *confidential key terms*. Our first intuition was to choose terms with a high prob-

[7] Throughout this paper, we define "terms" as a sequence of words whose length varies between one and three.

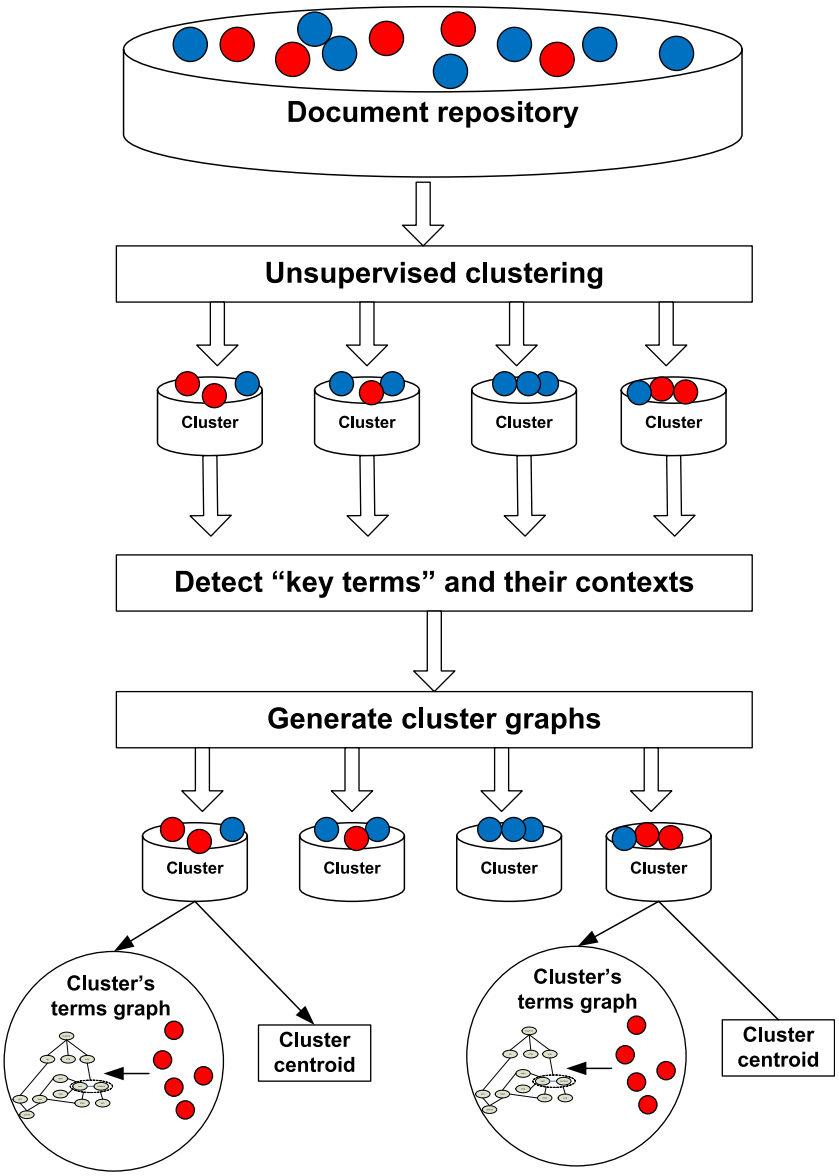

Fig. 1. The learning phase. (Red circles = confidential documents; blue circles = non-confidential documents.) (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

ability of appearing in confidential documents and a low probability of appearing in non-confidential documents. The former probability could be divided by the latter in order to generate a "confidentiality score" for each term. This type of computation is referred to as language modeling [54,65,42] and is usually used in the field of information retrieval. Language modeling utilizes the probability of randomly sampling a word (or a sequence of words), from a corpus of documents $P(w_1, w_2, w_3 \ldots \backslash C)$ for tasks such as choosing terms for a query or ranking documents [42].

Implementing this approach, however, presents some difficulty in the selection of the non-confidential documents used in this calculation. On the one hand, using only the non-confidential documents of the currently analyzed cluster may pose the following problems:

1. **The cluster may have only a few non-confidential documents or possibly none at all.** This may result in an inadequate representation of the non-confidential content.
2. **Adjacent clusters may contain similar documents to those in the analyzed cluster**. Not considering these may result in an incorrect representation of the confidential content. This is the case because the adjacent clusters may have non-confidential terms with content similar to those of the analyzed cluster's confidential documents. Such a scenario may result in choosing non-indicative confidential terms.

On the other hand, selecting the non-confidential documents from a broader scope (for example, the adjacent fifty clusters), may also have a negative effect on the produced probabilities due to the inclusion of unrelated documents. If this is the case,

many terms could be considered as indications of confidential content simply because they are *related* to the general subject of the documents in the examined cluster.

We solved this problem by using what we call *hierarchical language modeling*. We first create a separate language model for the confidential and non-confidential documents of the cluster itself. Then, we iteratively expand the non-confidential language model to include a larger number of clusters. The effect of each term on the language model diminishes as the number of clusters grows. We use the similarity between the original cluster and the candidate cluster to choose the clusters that are added to the language model (thus, similar clusters are added first and we lower the required similarity threshold every iteration). The formula for calculating the score of each term is presented in Eq. (1), where $P_{confidential\_LM}(term)$ and $P_{non\_confidential\_LM}(term)$ denote the probability of randomly sampling the analyzed term from the confidential and non-confidential language models, respectively, and iteration is a simple counter of the iterations already performed.

$$\forall term \in confidential\_Lm, \quad score_{term} += \frac{1}{iteration}\left(\frac{P_{confidential\_LM}(term)}{P_{non\_confidential\_LM}(term)}\right) \tag{1}$$

We believe that this method incorporates the best of both worlds. By giving larger weight to the examined cluster and its adjacent clusters, we are able to estimate accurately which terms are most likely to indicate the confidentiality of the document. At the same time, we are able to take into account the attributes of a larger portion of the documents in the dataset and to adjust the terms' weights accordingly.

Upon completion of this phase, a list of confidential terms is obtained for each cluster containing confidential documents. From this list, only the terms whose assigned score is greater than 1 are used in order to determine the confidentiality of documents during the detection phase (that is, we only use terms, which are more likely to appear in confidential documents than in non-confidential ones).

(2) **For every cluster, analyze the context of the confidential key terms**

The incorporation of the context in the representation of the confidential information is important because it enables a better understanding of the confidentiality of each term. Intuitively, the probability of a term being a part of confidential information is higher if it appears in similar contexts in other confidential documents. If a confidential term appears in an unrecognized context (or in a context only found in non-confidential documents), it is much less likely to be related to confidential content.

In the proposed model, the context of a term is defined using a parameter referred to as the *context span*, which is used to determine what the scope of terms surrounding the confidential term to be used as context should be. For example, if the context span is defined as twenty words, the context of a term is defined as the ten words that precede it and the ten that follow it. After experimenting with various values of this parameter, we found that the optimal value for our experiments is 20.[8] We chose the context terms of each confidential term from the context span.

Intuitively, the terms that ought to be considered as context terms are those that appear near the confidential terms *in confidential documents*. For this reason, when attempting to determine the context of a confidential term, only the documents in which that term appears are considered.

We calculated the probability of each context term to appear near the confidential term both in the confidential and non-confidential documents. This probability is defined as the number of documents in which the context term appeared near the confidential term divided by the total number of documents in which the confidential term appeared. This probability is calculated separately for the confidential and non-confidential documents.

The score of each context term is computed by subtracting the probability of its appearance in a non-confidential context from the probability of its appearance in a confidential one. This method is initially applied only to clusters containing the confidential documents, but then it is iteratively expanded to include other clusters, in the same manner as described in the previous section. The formula for calculating the score of each context term is presented in Eq. (2), where $P_{conf}(context\_term/conf\_term)$ and $P_{non\_conf}(context\_term/conf\_term)$ denote the probability of a context term to appear near a "key term" when it appears in confidential and non-confidential documents, respectively. As previously, *iteration* is a simple counter of the iterations in which we include more clusters but reduce their influence on the score of each context term.

$$score_{context\_term} += \frac{1}{iteration}(P_{conf}(context\_term/conf\_term) - P_{non\_conf}(context\_term/conf\_term)) \tag{2}$$

Here, unlike in Eq. (1), we subtract rather than divide the non-confidential probability. The reason behind this is that division can cause large fluctuations in the values of the context terms. Because of the small number of documents involved – only the documents in which the confidential terms appear are taken into account – even a single document can drastically change the probabilities.

[8] A lower value decreased the performance of our model, while a higher value did not improve it.

Algorithm 1. The learning phase algorithm

Generate_Clusters_Confidential_Terms_Graphs
Input:
C – set of confidential documents
N – set of non-confidential documents
TR_{min} – a minimal cluster similarity threshold
Output:
CR – a set of clusters, each with a centroid and a confidential terms graph
1: $T \leftarrow C \cup N$(combine both datasets)
2: $CR \leftarrow$ Perform Unsupervised Clustering on T
3: For each cluster $cr \in CR$
4: Calculate similarity to all other clusters
5: **Detect_Confidential_Key_Terms**
6: **Detect_Confidential_Context_Terms**
8: End For
9: **return** *CR*
Detect_Confidential_Key_Terms
Input:
CR – a set of all the clusters
C – the cluster currently being analyzed
Output:
CT[$\mathbf{C}_i$] – the set of confidential terms for the cluster
1: Create language model for cluster C_i, and calculate confidentiality score for each term
2: $TR \leftarrow$ set initial cluster similarity threshold
3: While $TR > TR_{min}$
4: Find all cluster whose cosine similarity to C_i is above *TR*
5: Create language model from all relevant clusters' documents
6: $CT \leftarrow$ Use new language model to add and update confidential terms' scores
7: Reduce the value of *TR*
8: End While
9: **return** *CT*
Detect_Confidential_Context_Terms
Input:
CR – a set of all the clusters
$\mathbf{C}_i$ – the cluster currently being analyzed
TR_{min} – a minimal cluster similarity threshold
Output:
CXT – sets of context terms for each confidential term
1: For each confidential term *ct* in C_i
2: Detect all occurrences of *ct* – both in confidential and non-confidential documents
3: $CXT \leftarrow$ For each context term in scope, calculate it's probability of appearing with the term in a confidential document vs. a non-confidential document
4: $TR \leftarrow$ set initial cluster similarity threshold
5: While $TR > TR_{min}$
6: Find all clusters whose similarity is above *TR* and detect all occurrences of term in clusters
8: $CXT \leftarrow$ Update each context term's probability of appearing next to the term in confidential context
9: Reduce *TR*
10: End While
11: End For
12: **return** *CXT*

By using subtraction, we ensure the diminishing effect of the iterations on the score of the term, since the value of each iteration can vary between [-1,1].
For each confidential term, all context terms whose "score" is positive – those that are more likely to appear as the context of that confidential term in confidential documents than in the rest of the dataset – are considered as its context.

(3) **Creating a graph representation of the confidential content of a cluster**
At the end of the process described in the previous section, confidential terms and the context in which they appear

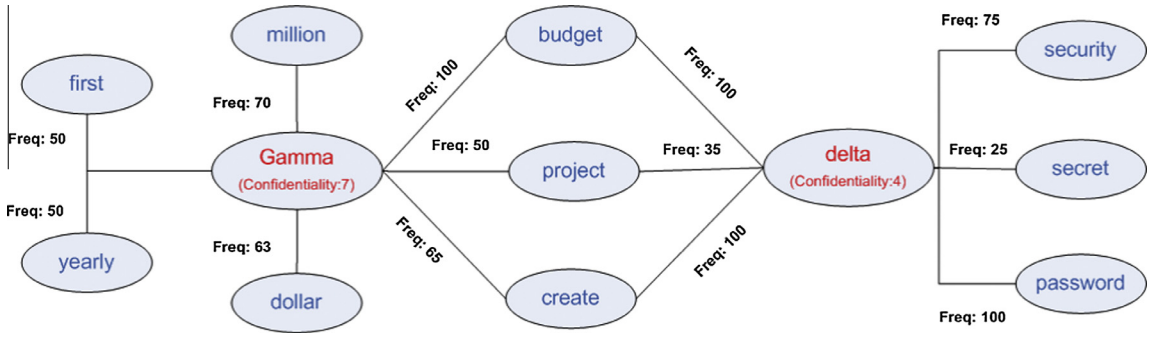

Fig. 2. An example of a confidential terms graph of a cluster (red nodes – confidential "key terms", blue nodes – context terms). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

are defined for each cluster. This information can easily be represented as a graph consisting of two types of nodes: confidential and context nodes. Confidential nodes can only be connected through their context nodes. That is, if two confidential terms have at least one common context term then they are connected in the graph. An example of such a graph is displayed in Fig. 2.

The graphs created in this procedure may be used by domain experts as a visual tool for understanding, verifying, and modifying (if necessary) the representation of the confidential content by the model. This will not only increase confidence in the proposed method but will also make it possible to detect mistakes and weaknesses in the model. Moreover, the graph representation enables the use of various graph and network analysis algorithms (as we show in Section 6, where we discuss future work). An example of such a graph is presented in Fig. 2.

3.2. The detection phase

In the detection phase (see Fig. 3), we attempt to deal with three types of challenges: (1) detection of whole confidential documents; (2) containment [7] – detecting small portions of confidential text embedded in a larger, non-confidential text; and (3) containment of modified context – the detection of small portions of modified confidential text embedded in a larger, non-confidential text (by "modified" we mean that some of the words have been replaced by synonyms; we elaborate further on this subject in Section 4).

The algorithm of the detection phase is presented in Algorithm 2. It consists of the following steps:

(1) Assign the inspected document to relevant clusters.
(2) For each of the assigned clusters, identify all the confidential and context terms that appear both in the document and in the cluster's confidential terms graph.
(3) For each of the assigned clusters, calculate the document's confidentiality score, based on the detected terms.
(4) Determine whether the document is confidential.

We now review each step in detail:

(1) **Assign the inspected document to relevant clusters**
The purpose of this step is to determine which of the confidential terms graphs (meaning, the clusters from which they were generated) will be used to determine whether the inspected document is confidential.
To deal with the scenarios described at the beginning of this section, we propose the following assignment algorithm:
(a) **Assign the document to the X clusters whose centroids are most similar to the document**
Theinspected document is transformed into a vector (after stemming [45] and stop-words removal) and its similarity to each of the cluster centroids is calculated. This is done using the cosine distance measure [61]. All the clusters whose similarity is above a predefined threshold are selected.
(b) **For each of the selected clusters, look for "irregular" terms for possible assignment to more clusters**
The goal of the previous step was to discover the clusters to which the document most likely belongs. This method is expected to work well for the detection of whole confidential documents but it is insufficient for detecting confidential sections embedded in non-confidential documents. Since most of the document is non-confidential, clustering – being a statistical process – will result in assigning the document to clusters containing non-confidential content (based on the majority of the document's text).

To deal with this problem, we look for "irregular" terms in the text i.e., terms that are very unlikely to appear in the document considering the clusters to which it was assigned. In order to identify these terms, we first create a language model for both the tested document and the clusters it was assigned to. Then, for each term in the tested document, we divide its probability of appearing in the document by its probability of appearing in the cluster. The formula used for this calculation is:

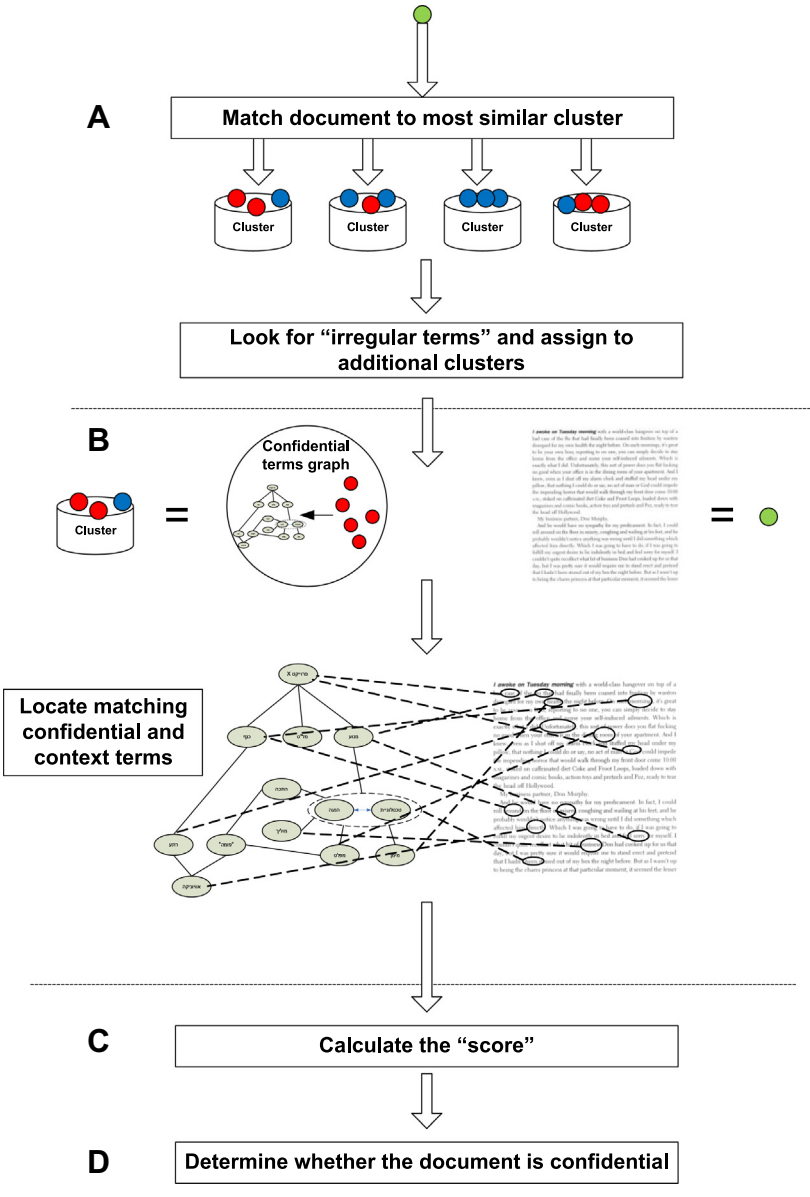

Fig. 3. The detection phase.

$$\forall d \in D, \quad d_{score} = \frac{P(d/D)}{P(d/C_i)} \tag{3}$$

where D is the tested document; d is a term in D; and C_i is one of the selected clusters. The higher the score of a term, the less likely it is to belong to the cluster.

The threshold that should be set for a term to be considered "irregular" is quite high, as to avoid "false alarms" and the slowing down of the model (due to the need to further analyze the document). The threshold we used in our experiments was 20[9]; that is, a term has to be twenty times as likely to appear in the tested document than in the cluster it was assigned to in order to be considered "irregular".

Once the "irregular" terms are found they are matched against the confidential terms graph of every cluster (provided the clusters contain confidential documents). In case of a match between a term and the confidential terms graph of a cluster, the corresponding cluster is added to the list of "candidate" clusters for which the "confidentiality score" of the document is calculated.

[9] The value was empirically set.

(2) **For each of the assigned clusters, identify all the confidential and context terms that appear both in the document and in the cluster's confidential terms graph**

In order to determine the confidentiality of a document, we perform for each of the graphs the following two steps: (a) scan the text in search of "confidential terms" and their context; and (b) determine which of the detected terms will be included in calculating the document's confidentiality.

(a) **Scan the text in search of all confidential terms and their context:**

The text of the document undergoes stemming and stop-words removal and is then matched against the confidential terms represented in the cluster's confidential terms graph. In order not to be hindered by a change in the order of the words (malicious or accidental), we look for the presence of a confidential term using a "sliding window" of twenty words.

Another issue is that of the confidential terms that are contained in other confidential terms (a typical example would be "world trade" and "world trade organization"). We select the term with the highest "score" in the cluster's graph (see Section 3.2), as it is the strongest indication of a confidential content.

Once a confidential term is detected, its vicinity is scanned in search of the context terms assigned to it in the graph. The scan is conducted using the same "sliding window" method described above. In the case of multiple occurrences of a term in one document, we record the term, including its context terms, from all its occurrences.

(b) **Determine the detected terms to be included in calculating the document's confidentiality:**

At this point, we have detected the confidential terms in the document and the contexts in which they appear. The following process aims at selecting the detected terms that should be considered when determining the document's confidentiality. Consider the following two confidential terms, each with two detected context terms. One has a "score" of three, which means it is 3 times more likely to appear in a confidential document than in a non-confidential one. The other has a score of 50. The presence of the first term is – obviously – a weaker indication of the document's confidentiality. In order to avoid the inclusion of many "weak" terms with low confidentiality scores (which could be "false alarms"), we use the following criteria. The inclusion of a confidential term depends *both* on its own score *and* on the number and score of its *context terms*. The higher the score of the confidential term itself, the less we rely on its context terms in determining whether to include it. The following table presents the criteria we used in our experiments (the values were set empirically):

(3) **Calculate the document's "confidentiality score":**

The confidentiality score of the document for each cluster is calculated by summing up the scores of all the confidential terms that met the above criteria.

In addition to summing them up, the scores of the terms were modified in the following way: if two confidential terms appear near each other *and* share at least one context term, then their scores are summed up and multiplied by 1.5. The rationale for this action is that the appearance of several terms in (most probably) the same context is a strong indication that the document is indeed confidential (a similar approach was used in [17], which applied the idea of related key terms to text summarization).

We experimented with different thresholds for the distance between two terms and found that the model performs best with the distance set to at most 50 words. This is the distance we used for the experiments described in Section.

(4) **Determine whether the document is confidential**

At this point we have a confidentiality score for the document, more than one if the document was assigned to several clusters with confidential terms graphs. If the document's confidentiality score for a cluster is higher than the threshold defined for that cluster then the document is considered confidential and is blocked.

Every cluster may have a different threshold confidentiality score above which a document is considered confidential. In a real-world scenario, domain experts will probably make this decision. In our experiments, we used one threshold for all confidential clusters. It is quite possible that a different threshold can be automatically set for every confidential cluster (based on the confidentiality scores of its training set documents), but we leave this subject for future work.

Algorithm 2. The detections phase algorithm

Calculate_Document_Confidentiality_Score
Input:
D – the document whose confidentiality we wish to determine
CL – a set of clusters, with their centroids and confidential terms graphs
Output:
conf_score – represents the analyzed document's level of confidentiality
1: *conf_score* ← 0
2: *C* ← get set of most similar clusters to document
2: *Irr* ← **Get_Clusters_That_Match_Irregular_Terms**
3: *T* ← *C* ∪ *Irr*

4: For each cluster $t \in T$
5: Find all matches to cluster confidential terms in the document
6: For each confidential term match, find all relevant context terms
7: Calculate_Document_Confidentiality_Score
8: End for
9: return
Get_Clusters_That_Match_Irregular_Terms
Input:
D – the analyzed document
CL – a set of cluster
C – the clusters **D** is currently assigned to
TR_{min} – the threshold above which a term is considered "irregular"
Output:
Irr – a set of additional clusters the document needs to be matched against
1: $Irr \leftarrow$ initialize the set of clusters to return
2: For each $c \in C$
3: $LM_{doc} \leftarrow$ create a language model from document D
4: $LM_c \leftarrow$ create a language model from cluster cl
5: $irr\_terms \leftarrow$ find every term t in D where $\frac{P_D(t)}{P_c(t)} \geqslant TR_{min}$
6: For each $cl \in CL$
7: If there is any $term \in irr\_terms$ also appears in the confidential terms of cl
8: $Irr \leftarrow Irr \cup cl$
9: End for
10: End for
11: **retrun** *Irr*
Calculate_Document_Confidentiality_Score
Input:
d – the document whose score we wish to compute
C – the cluster whose confidential terms graph will be used to calculate the score of D
Output:
Score – the score of the document
1: $score \leftarrow 0$
2: For each confidential term ct found both in C and d
3: If ct has a sufficient number of context terms (with sufficiently high values) then $score$ += ct_{score}
5: End id
6: End for
7: If several confidential terms are close *and* share context, multiply their joint score by a factor
8: **return** *score*

3.3. Complexity of the test phase

As explained above, CoBAn consists of two phases – learning and test. The learning phase is executed offline, so the evaluation of its performance is not very relevant when attempting to understand its applicability. The test phase, on the other hand, is applied in real-time, so assessing its complexity is crucial for the feasibility of the model. The complexity of the test phase depends on the complexity of the three following tasks: (a) assigning the tested document to some of the clusters (by computing the similarity of the document vector to the cluster centroids); (b) the search for "irregular" terms; and (c) calculating its confidentiality score once it has been assigned.

Before analyzing the complexity of each action, we would like to elaborate on our assumptions, as they greatly affect the complexity of the proposed solution:

(1) We assume that the *centroids*, *language models*, and *confidential terms graphs* of all the clusters have already been calculated (as was done during the learning phase).
(2) The confidential terms graphs were implemented using hashtables – an initial hashtable was used to store the "key" confidential terms, with each entry pointing to another hashtable that contained the relevant context terms. The cost of searching a hashtable with a low collusions rate is $O(1 + \frac{n}{k})$, with n being the number of terms in the table and k being the number of slots. Since the confidential terms graphs remain constant throughout the evaluation (no new items are added), the cost of searching it remains constant. Therefore, from this point on we refer to the cost of looking up a value on a hashtable as $O(1)$ for the sake of convenience.

Since the centroids of the clusters are calculated during the learning phase, the operations that need to be considered are:

(1) **The generation of the vectors for the tested documents and the calculation of their cosine similarities to each of the centroids**
The parameters that need to be taken into account are: (a) $\boldsymbol{T}$ – the number of terms in a document and (b) $\boldsymbol{C}$ – the number of centroids (or clusters). In order to generate a *tf* vector from the document, it needs to be "read" once – an action with a linear complexity of $O(T)$. In addition, we need to calculate the similarity of the document vector to each of the cluster centroids. This calculation requires comparing each value in the document vector to its corresponding value in the centroid. Therefore, the complexity of this step is $O(T \cdot C)$. Thus, the overall complexity of this step is $O(T + (T \cdot C))$, which can be reduced to $O(T \cdot C)$.

(2) **Identifying the "irregular" terms and their comparison to the cluster confidential terms graphs**
The parameters that need to be taken into account are: (a) $\boldsymbol{T}$ – the number of terms in the document and (b) $\boldsymbol{C}$ – the number of clusters.
To generate a language model for the examined document, we need to go over all its terms; an act, which has a complexity of $O(T)$. As the language models of the clusters were calculated during the learning phase, they are not part of the calculation. In the "worst case scenario", where all terms in the document are found to be "irregular", every term in the document will need to be matched to the confidential terms graphs of all the clusters. Since we assume that the cost of looking up a term in a hashtable is constant (see assumption 2), this action has a complexity of $O(T \cdot C)$. Therefore, the overall complexity of this step is $O(T + (T \cdot C))$, which can be reduced to $O(T \cdot C)$.

(3) **Calculating the confidentiality score for each of the clusters to which the document was assigned**
The complexity of finding the confidentiality score depends on two parameters: (a) $\boldsymbol{T}$ – the number of terms in an inspected document and (b) $\boldsymbol{C}$ – the number of clusters. In this step, we check for each relevant cluster, which terms in the document also exist in that cluster's confidential terms graph.
As we explained above, the cost of checking whether a term $t \in T$ is a confidential term has a complexity of $O(1)$. If a term is found in the hashtable, then the text in its vicinity is searched for relevant context. This too requires looking up every term in the dictionary, but it is important to remember that the scope of the search is *constant*. For example, if we define the scope of the search to be 20 words, then we have 21 hashtable lookups overall (the key term and the context). Since this is merely a constant, the complexity of performing this action is equal to that of a single hashtable lookup – $O(1)$.
The actions described above need to be applied for every term $t \in T$ and every cluster, so the complexity of this step is also $O(T \cdot C)$.
The complexity of each of the three steps described above is $O(T \cdot C)$ and therefore, the overall complexity of the test phase is $O(3T \cdot 3C) = O(T \cdot C)$. It is easy to see that the complexity is linear to the number of terms in the inspected document and to the number of clusters. Therefore, we conclude that CoBAn is capable of being applied in real time.

4. Evaluation

In this section, we evaluate CoBAn's performance on three datasets: Reuters news articles, the Pan-PC-11 Plagiarism dataset, and the Enron emails dataset. We first use Reuters' news articles in order to evaluate three scenarios: (a) the detection of whole confidential documents; (b) the detection of confidential excerpts (exact copies) embedded in non-related, non-confidential documents; and (c) the detection of *rephrased confidential excerpts* embedded in non-related, non-confidential documents. The third scenario is the one that is of most interest to us, as the two others are addressed via existing solutions, namely, classifier based methods for whole documents classification and fingerprinting based methods for the detection of unchanged confidential excerpts.

Initially, we simulated the third scenario on the Reuters dataset using only simple obfuscation, generated by applying Microsoft Word's thesaurus function (as detailed below in Section 4.3). We then used two additional datasets to evaluate scenarios in which a *more sophisticated rephrasing of the text* was applied to make the detection more difficult. The two datasets are the Pan-PC-11 plagiarism dataset (in which the text is rephrased specifically to avoid detection) and the Enron email dataset, which is used for data leakage prevention in emails. The challenge in this dataset is the short length and amount of "noise" in many emails.

We begin this section with a short review of the measures used to evaluate the performance of the various leakage detection methods followed by a description of our baseline methods. We then explain the preprocessing steps applied on the datasets, the experiments, and their results.

4.1. Evaluation measures

We use two criteria to evaluate the performance of the various methods: *true positive rate* (TPR) and *false positive rate* (FPR). The true positive rate is the percentage of confidential documents *correctly classified* during the evaluation phase. The false positive rate is the percentage of non-confidential documents *mistakenly classified* as confidential ("false alarms").

Table 1
The required number and score of context terms for each score of the confidential terms.

Confidential term score	Required num of context terms	Min score of context terms
$1 < X < 3$	11	85
$3 \leqslant X < 7$	9	75
$7 \leqslant X < 10$	7	60
$10 \leqslant X < 15$	6	50
$15 \leqslant X < 30$	5	40
$30 \leqslant X < 45$	3	30
$45 \leqslant X$	2	0

Naturally, our goal is to maximize the TPR while minimizing the FPR. To illustrate the tradeoff between the TPR and the FTR for a range of thresholds, we plot *receiver operating characteristics* (ROC) curves [6] that enable us to view the TPR and FPR tradeoffs on a single chart and to compare the performances of different methods.

ROC curves are usually compared by calculating their *area under the curve* (AUC) [6]. The comparison is conducted using statistical tests that attempt to determine whether the performance of one model (the AUC) is significantly different from another. In our experiments, we used paired T-tests in order to verify the significance of our results.

4.2. The baseline methods

We compared the performance of our model with the following two detection methods (baseline methods): (1) the well-known SVM classifier [12,13]. We used a polynomial kernel, which has been shown by [29] to perform well for text categorization tasks. We believe that the results of the comparison to the SVM could be generalized to most classifiers, and (2) an implementation of a fingerprint detection algorithm. The fingerprint algorithm was chosen due to its leading leakage detection methods in commercial products (for example, WebSense[10]). Since evaluation of commercial products is impossible (due to legal issues, the fact that usually more than one method is employed, and because every product is a "black box"), a basic fingerprint version was implemented for our evaluation. A sliding window size of 40 characters was used to generate the hash values. The size parameter was set by empirical calibration.

Furthermore, in the preliminary experiments on the Reuters dataset we compared the performance of the proposed method with and without the utilization of the context terms to demonstrate their effect on the results.

4.3. Reuters news articles

4.3.1. Creating the dataset

Since no public dataset of confidential and non-confidential documents was available, we had to generate one. The dataset was compiled from news articles collected from the Reuters news feed for two months. The documents of the dataset relate to 17 different news categories, e.g., art, culture, economics, and sports. Each category consists of several sub categories, and altogether there are 1300 categories and sub-categories. The documents were tagged in accordance with the IPTC News code taxonomy that Reuters uses for classifying news feeds.[11] A news item (document) was usually tagged with more than one tag.

The dataset contained 6102 documents, of which 1697 related to the "economics" category (i.e., at least one tag of the document is "economics") and the remainder of the documents were randomly selected from the other categories. We chose to define the sub-category "international (foreign) trade" as confidential. Out of the economic related documents, 310 were tagged with the "international (foreign) trade" tag, while the remainder of the "economics" document was distributed evenly among the other sub categories of economics. This setting was designed to simulate a scenario where documents of the same general subject (economics), can be either confidential or not.

4.3.2. Experiments

Three sets of experiments, which evaluate the three scenarios described above, were conducted on the dataset:

(1) For the detection of whole confidential documents – the basic scenario. We expected all methods to perform well, as this is a basic classification problem.
(2) Detection of small confidential sections embedded in non-confidential text – in this experiment, we simulated a situation where a person accidentally or intentionally inserts some confidential information in a document of an entirely dissimilar subject. This simulates the real-world scenario where confidential content is inserted using a "copy-paste" function.

[10] http://www.websense.com.
[11] http://www.iptc.org.

We hypothesized that classifier-based methods (such as SVM), would fail to identify these embedded sections due to the global statistics they apply on documents; the "bag of words" representation used by these methods does not enable focusing on specific sections. Therefore, classification is dominated by the non-confidential text, which consists of the majority of the document.

Fingerprint-based methods, on the other hand, are expected to perform best in this scenario since they were designed to detect sections of text, which are *identical* to those seen in the training phase.

(3) Detection of small rephrased/modified confidential sections embedded in non-confidential texts where the meaning of the confidential content is preserved. This experiment simulates scenarios of a person writing the confidential content "in one's own words", possibly to avoid detection.

Classifier based methods are expected to fail in this scenario as well due to the same reasons that caused their failure in the previous scenario. We also expect the fingerprint-based method to perform worse than in the previous scenario, since the rephrased text reduces the number of *exact matches* with the training set. We believe that CoBAn is capable of handling this scenario because of its focus on key terms and their contexts, rather than a specific sequence of texts.

4.3.3. Data preprocessing

Embedding the confidential section in a non-confidential text

To simulate the scenarios in which a confidential section is embedded in a non-confidential document, we manually extracted confidential sections from the 310 documents defined as confidential and embedded them into non-confidential documents according to the process described below. It is important to note that for this scenario *the confidential content was only extracted but not modified in any way*.

The steps of embedding the confidential section in a non-confidential document are as follows:

For each confidential document.

(1) Pick a random starting point (word) in a confidential section of the paper.
(2) Select a sequence of 50 words from the position of the chosen word (if the end of the section was reached, continue to the beginning of the confidential section).
(3) Pick a random, non-confidential document whose length is at least 10 times larger than the confidential section (to simulate a scenario in which the confidential section is hidden in a much larger, non-confidential document).
(4) Embed the confidential section at a random point in the non-confidential document.

The result of these steps is a set of new confidential documents, each containing at least 90% non-confidential content.

Rephrasing the confidential section

To generate a rephrased confidential text for the embedded modified confidential text scenario on the Reuters dataset, we simply replaced several words of the extracted confidential content with their synonyms. This way, the text is modified, yet retains its *coherence* and *original meaning*.

This solution was implemented using the Microsoft Word's thesaurus. For each word in the confidential section, we defined a 33% probability that a synonym would be sought for it. If a single synonym existed, it replaced the word. If several existed, one was randomly chosen. If none existed, no modification was made (so in effect, the ratio of replaced words was lower than 33%).

The advantage of this implementation was that the chances of modifying "unique" terms (initials or field specific terms), were very small, thus retaining the original meaning of the text. In Fig. 4, we present an example from one of our experiments. The first section is the original text and the second is the modified text. The modifications in the second section appear in bold. Although some of the modifications do not constitute "proper English", the idea of the text is clearly preserved. It should be noted that the results could not be considered as a sophisticated attempt to obfuscate the confidential text.

4.3.4. Reuters dataset experiments results

The experiments were conducted using a 3-fold cross-validation: two-thirds of the dataset was used as a training (learning) set and the remaining third was used as the test set.

Detecting whole confidential documents

The results are presented in Fig. 5. It is clear that all four methods perform well. There are no significant differences between the performances of the methods.

Detecting small confidential sections embedded in non-confidential text

The results for this scenario are presented in Fig. 6. As expected, the SVM model is inferior to the other two models due to the statistical nature of the algorithm. It can be seen that the proposed method, which utilizes the context of terms, outperforms the version that does not. The fingerprints algorithm fares best and is significantly superior to the proposed model ($p = 0.001$). This result is expected since the fingerprint algorithm is targeted towards this scenario. However, *the fingerprint maximal detection rate is 90%,* since some of the leaked pieces are too small and fragmented to be detected. In this respect, the other methods are superior.

Original text

"... Africa, hoping to put in place mutually beneficial trade terms and cooperation over immigration and peacekeeping. But the thorny trade issue, which was especially pertinent because of the end-of-year deadline, upset the summit's efforts. Merkel said EU leaders would discuss trade with Africa at an EU summit on Friday. To..."

modified text

*"...Africa, hoping to **set** in place **equally** beneficial trade **provisions** and cooperation **larger than** immigration and **mediation**. But the **pointed** trade issue, which was especially **relevant** because of the end-of-year deadline, **disconcerts** the summit's **labors**. Merkel said EU leaders would **argue** trade with Africa at an EU summit on Friday. To..."*

Fig. 4. An example of an original confidential section and its modified version after the use of synonyms.

Detecting small confidential sections embedded in non-confidential text when some of the words are replaced with synonyms

The results for this scenario are presented in Fig. 7. As observed from the graphs, the performance of the SVM algorithm remains low. Considering the fact that the SVM algorithm analyzes the document as a whole, this was as expected. The performance of the fingerprint algorithm has deteriorated considerably; its maximum detection rate has dropped to about 65% (although it maintains a low false-positive rate). CoBAn, when compared to fingerprinting, has a slightly higher false-positives rate but its performance is not upper-bounded like the fingerprinting algorithm.

It should be noted again that in this scenario we tested a rather simple obfuscation attempt in which long excerpts of text were unchanged. Moreover, the modified text had none of the characteristics that could make detection more difficult:

(a) An intent to avoid detection (as in plagiarism, for example).
(b) "Noisy text" – the use of abbreviations, emoticons, and misspelling of words (as is the case for emails).

For these reasons, we chose to conduct two additional sets of experiments designed to test these scenarios. The first dataset contained plagiarized documents (in other words, the authors' goal was to avoid detection), while the second contains email messages, short and "noisy" texts. The results of these experiments are described in the following sections.

4.4. Plagiarism dataset

In the previous section, we demonstrated that even a naive rephrasing of confidential text substantially reduces the performance of the fingerprinting algorithm (from a maximal detection rate of 90% to 65%). We wanted to analyze this issue further and therefore, conducted additional experiments on a dataset from the field of plagiarism. This field is very suitable for the evaluation of CoBAn's ability to detect rephrased texts, as plagiarizers very often go to great lengths to change and rephrase the copied text in order to avoid detection.

In our experiments, we used the PAN-PC-11 dataset.[12] This dataset contains 22,186 documents, each having a section (or sections), which have been plagiarized. The plagiarism in the dataset takes place in three forms: by machine, by humans, and through inter-language translation as well as three levels of obfuscation: low, medium, and high. For our experiments, we used the text, which was plagiarized by humans through Amazon's *mechanical Turk*[13] with a high level of obfuscation.

Most of the documents that form the training set in this dataset were very long (sometimes hundreds of pages), and did not adhere to the organizational information leakage scenario that usually occurs with shorter text documents. Such short texts are very commonly leaked when sending emails with confidential content in the message body or with short documents attached, but certainly not with documents of this length. Therefore, we had to adjust the length of documents to fit our scenario. In addition, each text did not necessarily consist of a single issue (some texts were entire books or manuals), but included references to many subjects and issues, which again, is atypical to our scenario.

Our goal was to generate a training set, which had the two following characteristics: (1) short and focused documents; (2) a small percentage of their content was modified (plagiarized), in the test set. A second goal was to generate documents for the test set that adhere to the scenario of non-confidential documents that include hidden and embedded confidential sections.

The following preprocessing steps were performed to generate short, focused documents in the training set and a test set that includes confidential modified "hidden" sections in non-confidential documents (the process is graphically illustrated in Fig. 8):

[12] http://www.webis.de/research/corpora/corpus-pan-pc-10.
[13] https://www.mturk.com/mturk/welcome.

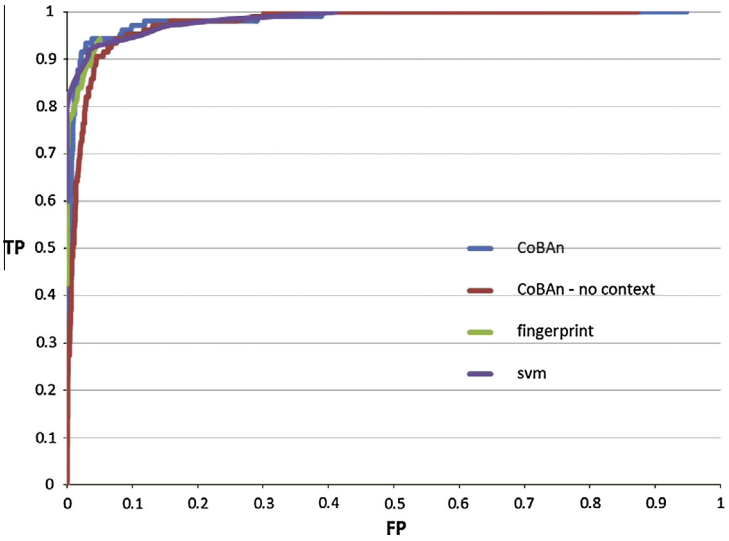

Fig. 5. Performance of the methods when detecting whole confidential documents.

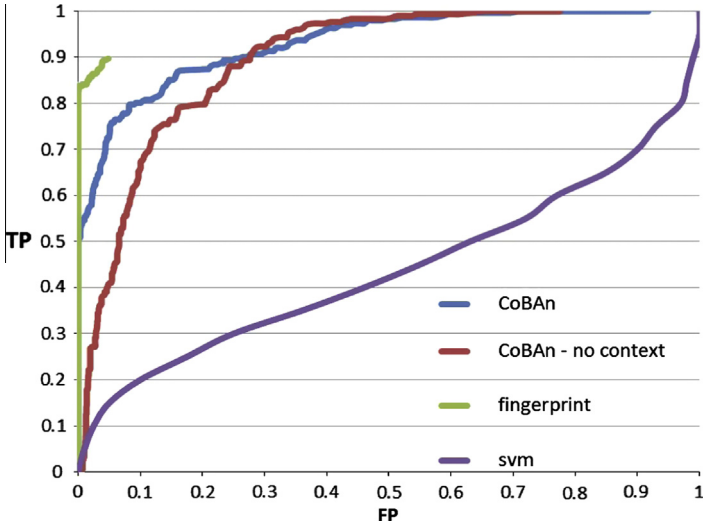

Fig. 6. Performance of the methods when detecting confidential sections in non-confidential documents.

Training set generation:

(1) For each document in the training set, we extracted the *text* that was plagiarized in the test set along with some surrounding text. The length of the surrounding text was defined as *5 times* that of the original text (step 1 in Fig. 8). By applying this step, we were able to generate a sufficiently short document of which a small section was used for the "leak attempt" during the evaluation phase.
(2) All the items that were generated were clustered, and for each cluster, the confidential and context terms were generated in the manner described in Section 3 of this paper (Step 2 in Fig. 8).

The result of these two steps is a set of clusters that form the training set and represent the various subjects of the documents.

Test set generation:

We wanted to "conceal" each plagiarized section in a larger randomly chosen text. The following describes the process for choosing the random text and the embedding of the plagiarized section (steps 3–5 in Fig. 8):

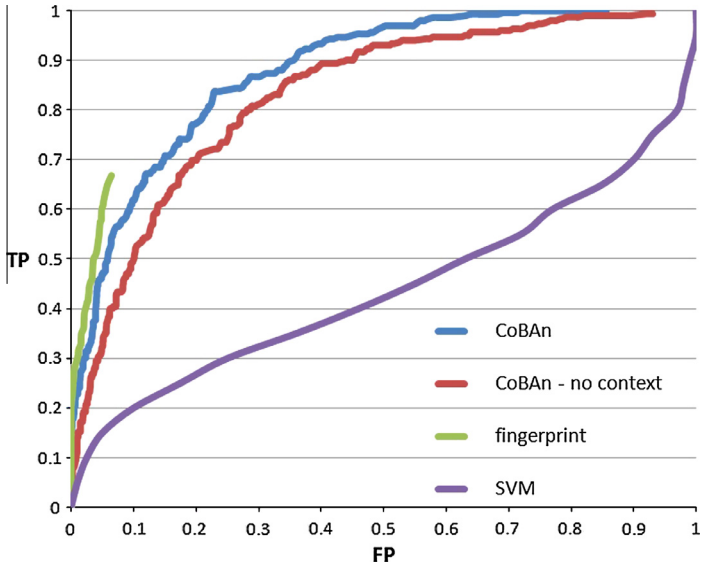

Fig. 7. Performance of the methods when detecting modified confidential sections in non-confidential documents.

(1) Extract the plagiarized section from the test set document (step 3 in Fig. 8).
(2) Select a random section of appropriate length (5 times the length of the plagiarized section) from one of the test set documents. (Step 4 in Fig. 8).
To assure that the plagiarized text is sufficiently different from the randomly extracted text, the following heuristic was applied:
(a) **If** the chosen cluster is one of the 10 *furthest* from the cluster to which the original text (the one that the plagiarized section was copied from), is assigned.
(b) **Then** place the plagiarized section in a random position in the chosen text (Step 5 in Fig. 8).
(c) **Otherwise** go back to *a*.

We conducted three sets of experiments with the plagiarism dataset with confidential sections of different lengths: less than 300 characters, 300–750 characters, and 751–1500. For each section length, three experiments were conducted – one for each of the three largest clusters produced during the clustering of the learning set. The same clusters were used for each of the three algorithms. The results are presented in Fig. 9.

As we can see, CoBan outperforms fingerprint and SVM in all experiments. As expected, both CoBAn and fingerprinting improve with the increase of the plagiarized section length. We believe SVM's poor performance can once again be explained by the fact that the majority of the text refers to an unrelated subject. The poor performance of the fingerprinting algorithm is explained by the heavy obfuscation of the plagiarized text, namely, the rephrasing and reordering of the text left very little identical text to that of the learning set (see Fig. 10).

This set of experiments demonstrates that CoBAn's focus on key terms and their context renders it far less "vulnerable" to text obfuscation and concealment. Key terms are less likely to be modified because their removal is more likely to lead to the loss of the text's original meaning.

4.5. Email leakage detection

The proposed method was also integrated into an email leakage detection solution, described in [71]. This paper describes a method designed to tackle the challenge of email leakage detection by clustering email messages via their content and then assigning senders and recipients to the clusters. The method then analyzes the types of content a user is allowed to access according to the cluster assignments of their emails and detects leakage when a user attempts to send an email to an unauthorized recipient.

The proposed method has two important advantages, which are not fully addressed in similar existing solutions that analyze the communication between senders and recipients: (a) users can share information on a confidential subject even if they have not interacted before at all (assuming they are both authorized); and (b) even if users communicate frequently, an attempt to send an email containing information that the recipient is not authorized to access will be blocked.

One of the key elements required for the method described in [71] to perform well is the ability to compute similarity between an email message and the various clusters. This task is challenging due to three problematic characteristics of the average email; short length, spelling mistakes, and abbreviations. CoBan is expected to perform well for this task, since

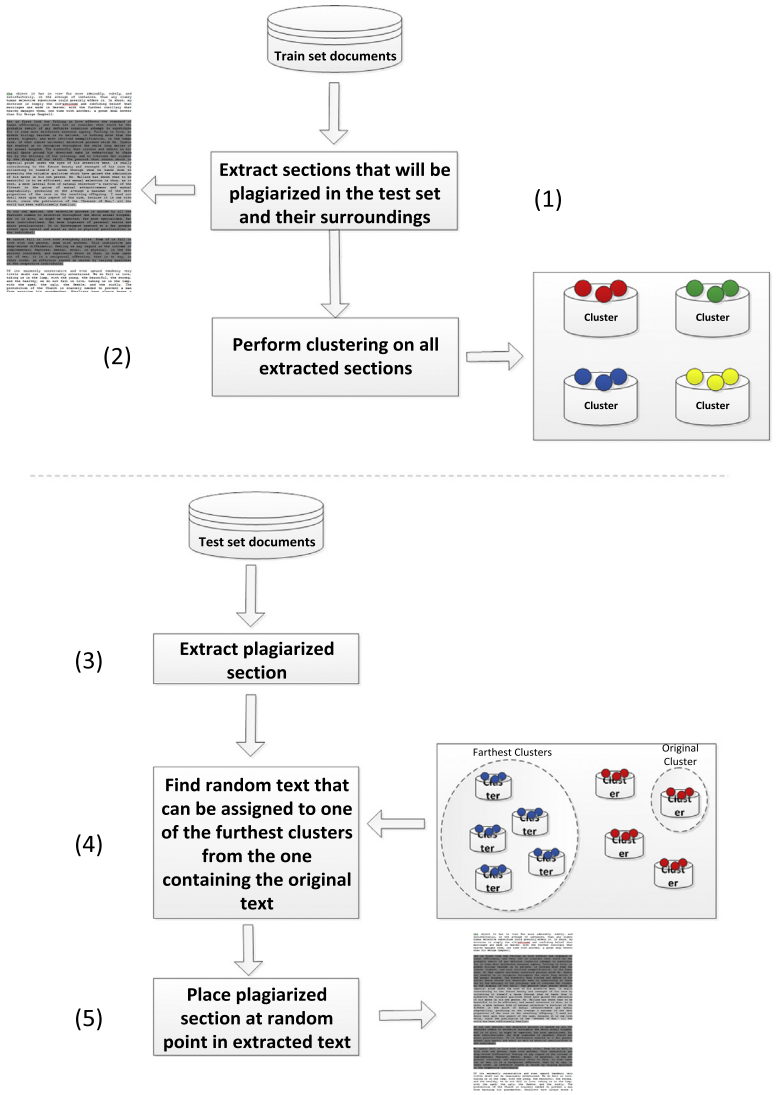

Fig. 8. The preprocessing of the training and test set documents of the plagiarism dataset.

it is able to focus on key terms and their context, i.e., ignores most of the noise in the text. In [71] we compare the performance of the proposed model when it employs CoBAn and the cosine similarity for this task.

We extended the experiments conducted in [71] to include an application of the fingerprint method for email-cluster similarity computation in addition to CoBAn and the cosine measure. The same clusters were generated and used by all methods to ensure a fair evaluation. The results are presented in Fig. 9 and show that CoBAn outperforms all other methods ($p = 0.01$). The fingerprinting algorithm fares worst, likely due to the "noisy" nature of the analyzed text.

Unfortunately, it was not possible to use the SVM method on this dataset since the SVM is a binary classifier, and although there are many works that demonstrate how it can be used for multiclass problems [4,19], we were not able to find an effective way to measure the *relative certainty* of the classification. The solution presented in [71] requires an estimation of the relative "strength" of the assignment of a message to several clusters at a time, and the SVM classifier is simply not suitable for this task.

These results are very encouraging as they show that the proposed method performs well in two very different domains, reaffirming the robustness of the model and its consistent contribution.

4.6. Running time comparisons

In this section we present a summary of the running time comparisons of both the training and test phases for each of the methods and each of the experiments described in the previous sections. For the first two datasets (Reuters and Plagiarism),

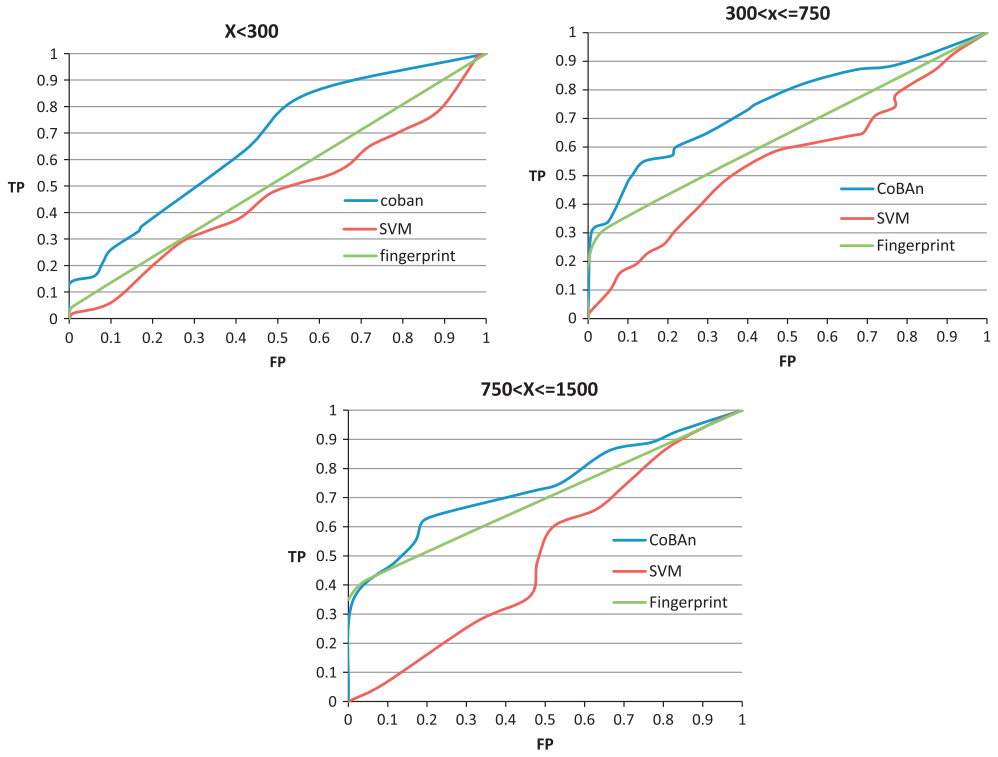

Fig. 9. The performance of the various evaluated methods on the plagiarism dataset with different lengths of plagiarized text. The *TP* axis denotes the detection rate of real plagiarism cases, while *FP* denotes "false alarms".

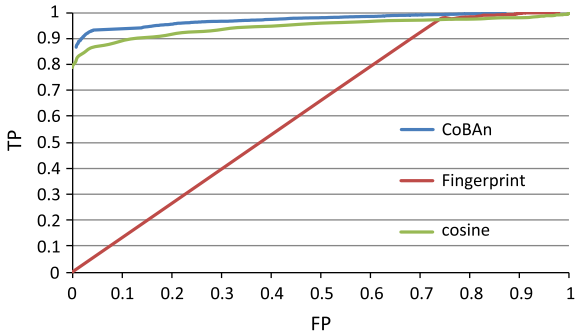

Fig. 10. Email leakage detection system.

Table 2
Comparison of the average training running times for the different classification methods on the experiment datasets (the results are in an hh:mm format).

	Number of instances	CoBAn	SVM	Cosine measure	Fingerprinting
Reuters	3051	00:25	00:07		00:10
Plagiarism	2115 (average)	00:52	0:35		00:28
Enron	50,112	05:00		03:00	01:05

Table 3
Comparison of the average evaluation running times for the different classification methods on the experiment datasets (the results are in an hh:mm format).

	Number of instances	CoBAn	SVM	Cosine measure	Fingerprinting
Reuters	2051	00:18	00:04		00:07
Plagiarism	2115 (average)	00:54	0:34		00:22
Enron	12,528	04:15		03:25	01:56

we present the running times of CoBAn, SVM, and Fingerprinting. For the third dataset, Enron emails, we present the results for the cosine similarity instead. All experiments were run on an Intel i7 quad core laptop with 8 GB of memory.

The results clearly show that CoBAn requires longer running times both for the learning and the detection phases. Although some of the difference between CoBAn and SVM could be attributed to the fact that the CoBAn was implemented as a research prototype, while we used a professional commercial product for the SVM (RapidMiner [51]), it is clear that the additional operations the proposed model requires do result in longer running times. Nonetheless, the differences are not in order of magnitude, as can be observed from the complexity analysis in Section 3.3 and we believe they could be substantially reduced with some optimization to our research prototype. The running time analysis proved the feasibility of the method.

5. Discussion

The proposed model has the following advantages:

(1) **The ability to detect confidential information "hidden" in large non-confidential documents.** The proposed model has the ability to detect small sections of confidential content, even if they differ from the training set examples. This is due to the model's ability to focus on content and context – the appearance of certain terms in a certain context. Consequently, it is much less vulnerable to rephrasing or alterations of the text. This sets the proposed model apart from classifiers such as SVM, (which fail in these scenarios), and fingerprinting algorithms, (which have high levels of success in detecting the confidential text, but only when it is very similar to an example from the training set).
(2) **Explainable model.** The graphs that are generated are visually very comprehensible and can be easily understood by domain experts. For each cluster containing confidential content, the expert is able to see the confidential terms that form the base of the detection rate and their relevant context (as shown in Fig. 2).Furthermore, the expert is able to easily adjust the values of nodes in the graph and even add or delete nodes. We believe this might help domain experts to better define their security policies, since they are able to understand and define their policies at the cluster level.
(3) **Easily configurable model.** Domain experts can easily add, delete or modify nodes in the graphs. This flexibility will enable domain experts to refine the representation of the confidential information in each cluster and consequently enhance the performance of the model.

We believe the proposed model has the following theoretical contributions:

(1) **A novel approach regarding the context of key terms for classification purposes.** The vector space model does not consider context at all, while existing graph-based methods do so indiscriminately. Fingerprinting algorithms only look for predefined sequences of characters. In this paper, we demonstrate how the *context of key terms* improves classification. We believe that the use of context has the potential to enhance many existing models that involve classification. The work of Hess and Holt is one such example [34].
(2) **A new approach for the graph representation of text.** Although graph-based textual representation is not new and was used before for classification, there are differences in its usage as presented in this paper and in earlier works ([23–28]). The proposed graph model consists of two node types, each with a different role. Moreover, instead of using isomorphism as a measure of similarity, we propose a completely different scheme based on matching key nodes in the graph.

Some may argue that the false positive rates achieved by the proposed model are too high to be used in a real system. In this work, we address new scenarios that have not been addressed before, particularly the rephrasing of small sections of confidential text that in many cases refer to intentional confidential information leakage. We consider our results as preliminary and encouraging and believe that they can be improved with additional research. In addition, we think that the proposed method should be used in conjunction with existing information leakage solutions (for example, fingerprinting), as is done in all commercial products to leverage the advantages of each method.

6. Conclusion and future work

In this paper, we present a new method for information leakage detection and prevention. The model is superior in the following aspects:

a. It detects small sections of confidential information embedded in non-confidential documents. As proven by the experiments in Section 4, existing methods provide only a partial solution to this problem and are unable to successfully deal with the problem of rephrased texts.
b. It generates a well-understood model that can be reviewed and even modified by its users.

We plan to examine the following directions:

(1) **The use of external data sources to enhance the representation of confidential content**. We believe that one of the main advantages of our method is its expandability. We believe that the graph representation used by CoBAn can be enriched with information from sources, such as ontologies, metadata, and the World Wide Web. The data found in these sources of information can be used to discover additional confidential terms and to enrich the context of existing ones.
(2) **The use of network analysis methods**. Currently, the confidential terms graph of a cluster is treated as a single unit. Possibly the use of graph clustering and other network analysis algorithms could enable better analysis of documents.
(3) **The use of natural language processing (NLP) and entity extraction**. In the current model, the role of the term in the sentence and its type (person, place, etc.), are not taken into account. It is possible that by considering this information we will be able to modify the values of nodes in the graph or change the ranking scheme entirely.

Acknowledgement

The research was supported by ISF Grant 1116/12 and Grant 150378-0-0553 of the Israeli Ministry of Science & Technology.

References

[1] "Information Week Global Security Survey". Information Week, 2004.
[2] D. Alassi, R. Alhajj, Effectiveness of template detection on noise reduction and websites summarization, Information Sciences 219 (0) (2013) 41–72.
[3] P. Ammann, D. Wijesekera, et al, Graph-based network vulnerability analysis, in: Proceedings of the 9th ACM Conference on Computer and Communications Security, ACM, Washington, DC, USA, 2002, pp. 217–224.
[4] J. An, Z. Wang, et al, A new SVM multiclass classification method, Information and Control – Shenyang 33 (2004) 262–267.
[5] I. Androutsopoulos, J. Koutsias, et al, An experimental comparison of naive Bayesian and keyword-based anti-spam filtering with personal e-mail messages, in: Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, Athens, Greece, 2000, pp. 160–167.
[6] A.P. Bradley, The use of the area under the ROC curve in the evaluation of machine learning algorithms, Pattern Recognition 30 (7) (1997) 1145–1159.
[7] A.Z. Broder, On the resemblance and containment of documents, Compression and Complexity of Sequences, 1997. Proceedings 21–29.
[8] D. Cai, X. He, et al, Graph regularized nonnegative matrix factorization for data representation, IEEE Transactions on Pattern Analysis and Machine Intelligence 33 (8) (2011) 1548–1560.
[9] W.J. Christmas, J. Kittler, Structural matching in computer vision using probabilistic relaxation, IEEE Transactions on Pattern Analysis and Machine Intelligence 17 (8) (1995) 749–764.
[10] M. Christodorescu, S. Jha, et al, Mining specifications of malicious behavior, in: Proceedings of the 6th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on The foundations of Software Engineering, ACM, Dubrovnik, Croatia, 2007, pp. 5–14.
[11] W.W. Cohen, Learning rules that classify e-mail, in: Proceedings of the AAAI Spring Symposium on Machine Learning in Information Access, 1996, pp. 18–25.
[12] W.W. Cohen, Y. Singer, Context-sensitive learning methods for text categorization, ACM Transactions on Information Systems 17 (2) (1999) 141–173.
[13] T.P. Cronan, P.R. Embry, et al, Identifying factors that influence performance of non-computing majors in the business computer information systems course, Journal of Research on Computing in Education 21 (4) (1989) 431–446. %@ 0888-6504.
[14] A. Cross, A. Wilson, et al, Genetic search for structural matching, Lecture Notes in Computer Science 1121 (1996) 150–159.
[15] J. D'hondt, P.-A. Verhaegen, et al, Topic identification based on document coherence and spectral analysis, Information Sciences 181 (18) (2011) 3783–3797.
[16] W. De Mulder, Optimal clustering in the context of overlapping cluster analysis, Information Sciences 223 (0) (2013) 56–74.
[17] R.F. Deckro, H.W. Woundenberg, M.B.A. admission criteria and academic success, Decision Sciences 8 (4) (1977) 765–769.
[18] H. Drucker, D. Wu, et al, Support vector machines for spam categorization, IEEE Transactions on Neural Networks 10 (5) (1999).
[19] K.B. Duan, S. Keerthi, Which is the best multiclass SVM method? An empirical study, Multiple Classifier Systems (2005) 732–760.
[20] S.T. Eckmann, V. Giovanni, et al, STATL: an attack language for state-based intrusion detection, Journal of Computer Security 10 (1/2) (2002) 71–104.
[21] G. Erkan, D.R. Radev, LexRank: graph-based lexical centrality as salience in text summarization, Journal of Artificial Intelligence Research (2004).
[22] J. Feng, M. Laumy, et al, Inexact matching using neural networks, Machine Intelligence and Pattern Recognition (1994).
[23] M. Fire, R. Puzis, et al., Organization Mining Using Online Social Networks, arXiv preprint, 2013. arXiv:1303.3741.
[24] R. Forsati, M. Mahdavi, et al, Efficient stochastic algorithms for document clustering, Information Sciences 220 (0) (2013) 269–291.
[25] S. Fortin, The graph isomorphism problem, Technical Report 96-20, University of Alberta, Edomonton, Alberta, Canada, 1996.
[26] M.a. Gafny, A. Shabtai, et al., Detecting data misuse by applying context-based data linkage, in: Proceedings of the 2010 ACM workshop on Insider threats, ACM, 2010.
[27] H. Gao, J. Hu, et al, Detecting and characterizing social spam campaigns, in: Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement, ACM, 2010.
[28] J.M. Gómez-Hidalgo, J.M. Martin-Abreu, et al, Data Leak Prevention Through Named Entity Recognition, 2010 IEEE Second International Conference on Social Computing (SocialCom), IEEE, 2010.
[29] A. Hagberg, P. Swart, et al., Exploring network structure, dynamics, and function using networkx, 2008.
[30] X. Han, L. Sun, et al, Collective entity linking in web text: a graph-based method, in: Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, 2011.
[31] A. Harel, A. Shabtai, et al, M-score: estimating the potential damage of data leakage incident by assigning misuseability weight, in: Proceedings of the 2010 ACM Workshop on Insider Threats, Springer, 2010.
[32] J.I. Helfman, C.L. Isbell, Ishmail: Immediate identification of important information, AT&T Labs Texhnical Report, 1995.
[33] J.W. Henry, M.J. Martinko, et al, Attributional style as a predictor of success in a first computer science course, Computers in Human Behavior 9 (4) (1993) 341–352.
[34] A. Hess, J. Holt, et al, Content-triggered trust negotiation, ACM Transactions on Information and System Security 7 (3) (2004) 428–456.
[35] T.C. Hoad, J. Zobel, Methods for identifying versioned and plagiarized documents, Journal of the American Society for Information Science and Technology 54 (3) (2003) 203–215.
[36] J. Hovold, Naive Bayes spam filtering using word-position-based attributes, in: Proceedings of the 2nd Conference on Email and Anti-Spam, 2005.

[37] C. Jiang, F. Coenen, et al, Text classification using graph mining-based feature extraction, Knowledge-Based Systems 23 (4) (2010) 302–308.
[38] A.C. Joseph, M. Suzanne, et al, Tutoring for retention, in: Proceedings of the 42nd ACM Technical Symposium on Computer Science Education, ACM, Dallas, TX, USA, 2011, pp. 213–218.
[39] C. Kalyan, K. Chandrasekaran, Information leak detection in financial e-mails using mail pattern analysis under partial information, in: Proceedings of the 7th Conference on 7th WSEAS International Conference on Applied Informatics and Communi-cations, 2007, pp. 104–109.
[40] K.J. Keen, L. Etzkorn, Predicting students' grades in computer science courses based on complexity measures of teacher's lecture notes, Journal of Computing Sciences in Colleges. 24 (5) (2009) 44–48.
[41] M. Koch, L.V. Mancini, et al, A graph-based formalism for RBAC, ACM Transactions on Information and System Security 5 (3) (2002) 332–365.
[42] V. Lavrenko, W.B. Croft, Relevance based language models, in: Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, New Orleans, Louisiana, United States, 2001, pp. 120–127.
[43] G. Levi, A note on the derivation of maximal common subgraphs of two directed or undirected graphs, Calcolo 9 (1972) 341–354.
[44] C.-C. Lien, C.-C. Ho, et al, Applying fuzzy decision tree to infer abnormal accessing of insurance customer data, in: 2011 Eighth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD), IEEE, 2011.
[45] J.B. Lovins, Development of a stemming algorithm, Mechanical Translation and Computational Linguistics 11 (1) (1968) 22–31.
[46] M.I. Malinen, P. Fränti, Clustering by analytic functions, Information Sciences 217 (0) (2012) 31–38.
[47] U. Manber, Finding similar files in a large file system, in: Proc. USENIX Winter 1994, 1994, pp. 1–10.
[48] A. Markov, M. Last, et al, The hybrid representation model for web document classification, International Journal of Intelligent Systems 23 (6) (2008) 654–679.
[49] S. Mathew, M. Petropoulos, et al, A Data-centric Approach to Insider Attack Detection in Database Systems, Recent Advances in Intrusion Detection, Springer, 2010.
[50] J.J. McGregor, Backtrack search algorithms and the maximal common subgraph problem, Software: Practice and Experience 12 (1) (2006) 23–34.
[51] I. Mierswa, M. Wurst, et al, Yale: rapid prototyping for complex data mining tasks, ACM, 2006.
[52] M. Nyanchama, S. Osborn, The role graph model and conflict of interest, ACM Transactions on Information and System Security 2 (1) (1999) 3–33.
[53] C. Phillips, L.P. Swiler, A graph-based system for network-vulnerability analysis, in: Proceedings of the 1998 Workshop on New Security Paradigms, ACM, Charlottesville, Virginia, United States, 1998, pp. 71–79.
[54] J.M. Ponte, W.B. Croft, A language modeling approach to information retrieval, in: Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, Melbourne, Australia, 1998, pp. 275–281.
[55] J.D.M. Rennie, ifile: An application of machine learning to e-mail filtering, in: Proceeding of the KDD Workshop on Text Mining, 2000.
[56] K. Riesen, H. Bunke, Approximate graph edit distance computation by means of bipartite graph matching, Image and Vision Computing 27 (7) (2009) 950–959.
[57] M. Sahami, S. Dumais, et al., A Bayesian Approach to Filtering Junk Email, AAAI-98 Workshop on Learning for Text Categorization, 1998.
[58] G. Salton, Introduction to Modern Information Retrieval, McGraw Hill, New York, 1983. p. 448.
[59] G. Salton, Another look at automatic text-retrieval systems, Communications of the ACM 29 (7) (1986) 648–656.
[60] G. Salton, C. Buckley, Term-weighting approaches in automatic text retrieval, Information Processing and Management 24 (5) (1988) 513–523.
[61] G. Salton, H. Wu, A term weighting model based on utility theory, in: Proceedings of the 3rd Annual ACM Conference on Research and Development in Information Retrieval, Cambridge, England, Butterworth \ & Co., 1981, pp. 9–22.
[62] A. Schenker, Graph-Theoretic Techniques for Web Content Mining, University of South Florida, PhD thesis, 2003.
[63] S. Schleimer, D.S. Wilkerson, et al, Winnowing: local algorithms for document fingerprinting, in: Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data, ACM, San Diego, California, 2003, pp. 76–85.
[64] A. Shabtai, Y. Elovici, et al, A Survey of Data Leakage Detection and Prevention Solutions, Springer, 2012.
[65] F. Song, W.B. Croft, A general language model for information retrieval, in: Proceedings of the Eighth International Conference on Information and Knowledge Management, ACM, Kansas City, Missouri, United States, 1999, pp. 316–321.
[66] J. Song, H. Takakura, et al, Toward a more practical unsupervised anomaly detection system, Information Sciences 231 (0) (2013) 4–14.
[67] J. Staddon, P. Golle, et al, A content-driven access control system, in: Proceedings of the 7th Symposium on Identity and Trust on the Internet, ACM, Gaithersburg, Marylands, 2008, pp. 26–35.
[68] J.R. Ullmann, An algorithm for subgraph isomorphism, Journal of the ACM 23 (1) (1976) 31–42.
[69] W. Wang, T.E. Daniels, A graph based approach toward network forensics analysis, ACM Transactions on Information and System Security 12 (1) (2008) 1–33.
[70] F. Wei, W. Li, et al, A document-sensitive graph model for multi-document summarization, Knowledge and Information Systems 22 (2) (2010) 245–259.
[71] P. Zilberman, S. Dolev, et al, Analyzing Group Communication for Preventing Data Leakage via email, IEEE, 2011.