Data Loss Prevention Based on Text Classification in Controlled Environments

Kyrre Wahl Kongsgård[1,2,4(✉)], Nils Agne Nordbotten[1,2,4], Federico Mancini[1], and Paal E. Engelstad[1,3]

[1] Norwegian Defence Research Establishment (FFI), P.O. Box 25, 2027 Kjeller, Norway
{kyrre-wahl.kongsgard,nils.nordbotten, federico.mancini,paal.engelstad}@ffi.no

[2] Department of Informatics, University of Oslo, Blindern, 0316 Oslo, Norway

[3] Oslo and Akershus University College of Applied Sciences (HiOA), 0130 Oslo, Norway

[4] University Graduate Center Kjeller, UNIK, Kjeller, Norway

Abstract. Loss of sensitive data is a common problem with potentially severe consequences. By categorizing documents according to their sensitivity, security controls can be performed based on this classification. However, errors in the classification process may effectively result in information leakage. While automated classification techniques can be used to mitigate this risk, little work has been done to evaluate the effectiveness of such techniques when sensitive content has been transformed (e.g., a document can be summarized, rewritten, or have paragraphs copy-pasted into a new one). To better handle these more difficult data leaks, this paper proposes the use of controlled environments to detect misclassification. By monitoring the incoming information flow, the documents imported into a controlled environment can be used to better determine the sensitivity of the document(s) created within the same environment. Our evaluation results show that this approach, using techniques from machine learning and information retrieval, provides improved detection of incorrectly classified documents that have been subject to more complex data transformations.

1 Introduction

Organizations and companies are handling increasing amounts of digital sensitive information, such as personal (e.g., health) data, military classified documents, trade secrets, and so forth. It is critical that such sensitive data is not leaked to unauthorized parties.

Many vendors offer data loss prevention (DLP) solutions that automatically monitor the storage, network, and users to detect and prevent such leakages [19,25]. Among the mechanisms employed by these solutions, data classification, where data objects are labelled according to their sensitivity, is recognized as an important enabler to improve their effectiveness [27]. Furthermore, given correct

© Springer International Publishing AG 2016
I. Ray et al. (Eds.): ICISS 2016, LNCS 10063, pp. 131–150, 2016.
DOI: 10.1007/978-3-319-49806-5_7

classification, data can be protected using appropriate access control and other security controls.

For instance, in a military or governmental context, a correct security label forms the basis for conducting information flow control using a so called guard (e.g., [12]), that ensures that only data allowed by policy (e.g., non-sensitive data) is released to external parties. If a Confidential domain and an Unclassified domain are connected, the guard will typically be configured to only allow data marked with a security label of Unclassified to pass from the Confidential domain to the Unclassified domain. While there may be additional security classifications (e.g., Restricted and Secret), the main concern to prevent data loss in such a scenario is to be able to detect when a document is incorrectly claimed to be releasable (i.e., Unclassified in this case).

Since security decisions are based on the classification specified by a data object's security label, it is crucial that data objects are classified correctly. However, human users or applications may mislabel data by mistake. Furthermore, an insider, or malware, could intentionally mislabel data to bypass security controls. For this reason, it is important to be able to validate the classification specified by users/applications, in order to detect mistakes and exfiltration attempts.

In this paper we explore the use of automated methods, based on machine learning (ML) and information retrieval (IR), to detect misclassification that could result in data loss. Text classification for DLP purposes has usually been based on techniques like fingerprinting of documents, keyword matching, and regular expressions, but more recently methods like ML and IR have been recognized as important for automated classification of unstructured documents [27]. However, little research is to be found on the subject [2,7,13,14,18]. A common trait of these previous works is that the classifier or index is based on a set of documents with well-known classification, either intended to include all the sensitive documents to be protected or a subset of documents used as a training set. It may be noted that it is necessary to assume a set of correctly classified documents which can be used as the base to estimate the correctness of new classifications. However, the noise in such a large generic set of index documents may result in misclassification, especially if new documents have been generated using content from sensitive sources, but where this content has been re-phrased, summarized or modified in other ways. Apart from recent work based on sequence alignment techniques [26], intended solely for detecting inadvertent data leaks, there is little previous research on detection of modified/transformed data leaks, the most relevant being the use of synonym substitution (spinning) [2].

In order to improve the classification accuracy also in these situations, we propose a controlled environment where all imported documents are dynamically monitored. The collected information constitutes the basis for classification of new documents created within the environment itself. The intuition is that in order to generate a new document, various sources are usually accessed and consulted, and the resulting work will share some common traits with such references. Also, if one wants to leak out a sensitive document, it would have to be imported first, and therefore be among the indexed ones. Compared to the usual approach where one classifier based on all sensitive documents available in a company (or a generic subset thereof) is used to classify any outgoing document, our results show that a classifier built dynamically for each controlled environment provides improved accuracy especially for the more difficult content transformations. While this is a surprising result given that more data generally provides higher accuracy, it illustrates that determining sensitive content is much more context dependent than for instance topic categorization. Also, we assume that any sensitive content can be traced back to the imported documents. Furthermore we study how different algorithms from both ML and IR performs in different controlled environments by varying the amount of imported documents, i.e., the noise in the training set/index, and by generating modified outputs that share a variable amount of information with the imported documents, both in quality and quantity. While we find that the classification accuracy of ML and IR algorithms is only moderately affected by document spinning, we find that there are other types of modifications (e.g., summarisation and mixing of content) that have a more severe effect on classification accuracy. To our knowledge the effect of these more difficult modifications has not been previously studied in the context of ML and IR for DLP. Our results show that the use of controlled environments improve detection of these less obvious data leaks.

For our tests we use documents from the U.S. Digital National Security Archive (DNSA) [1] and reports from our own institution for validation. While previous work has performed experiments using *tens* or *hundreds* of leaked classified documents (WikiLeaks, DynCorp and TM[1]) and public documents from sources such as Wikipedia, the Enron email dataset and various online PC magazines [2,14], and a smaller subset of the DNSA documents [7], we believe the work presented here constitutes the first study of transformed data leak detection based on such a large number of actual classified documents (i.e., *thousands*).

The rest of this paper is organized as follows. Section 2 describes the proposed solution, including the controlled environment, classification approaches, and deployment options. Section 3 describes the methodology used to evaluate the proposed solution, including: the corpora used as datasets and the methods used to simulate the generation of new documents. Section 4 presents the evaluation results; Sect. 5 provides a presentation of related work; and Sect. 6 provides some final conclusions and summarises the main contributions of the paper.

2 Proposed Solution

We introduce a *controlled environment* as an environment where we have control on all imported documents and their classification. The set of imported known documents is defined as *input*, and any new generated document(s) is defined as *output*. We assume that the classification (i.e., sensitivity) of the input documents are known, e.g., through existing cryptographically signed security labels.

A controlled environment could for instance be a single computer, a virtual machine, a network/security domain, or an isolated process/application. Our

[1] Transcendental meditation.

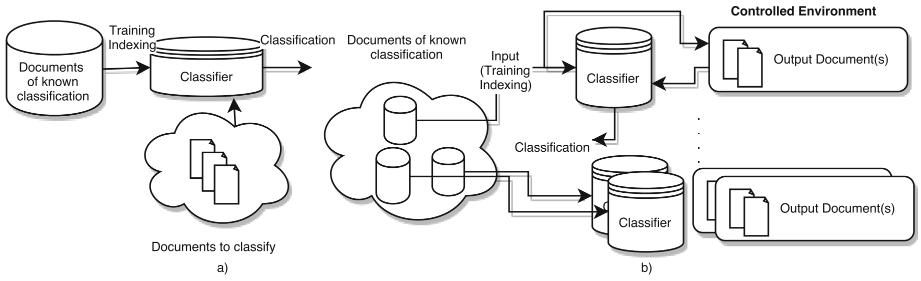

Fig. 1. (**a**) Usually, a classifier used in DLP is trained on all available documents (**b**) With a controlled environment, only the documents of known classification accessed from the environment are used to train the classifier, which in turn is used to classify documents generated within the environment. Multiple controlled environments can exist simultaneously, each characterized by its own input and output.

proposed solution inspects the information flow to the controlled environment as shown in Fig. 1b, and estimates the classification of output documents based on the information about the input documents. This is contrary to the traditional approach where a larger generic index/training set is used to classify any output document, as illustrated in Fig. 1.

Our guiding hypothesis is that by limiting the set of input documents to those relevant to an output document, thereby reducing the noise in the classification process, we can more accurately estimate the classification of output documents.

While there are different ways to implement controlled environments, our hypotheses suggests that the more tightly enclosed environments such as a virtual machine or isolated process have potential for better accuracy than the more loosely enclosed environments such as a network domain. As will be seen in Sect. 3, this is supported by our evaluation results. To fully take advantage of this, a controlled environment may be *instantiated* for a specific task (e.g., the writing of a single output document), thereby clearing all state (i.e., *input*)

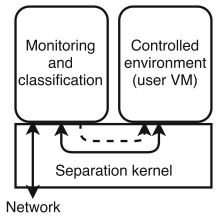
Monitoring and classification
Controlled environment (user VM)
Separation kernel
Network

Fig. 2. Illustration of how the proposed solution can be deployed on a separation kernel. The solid arrows represent the allowed communication channels while the dotted arrow indicates a control channel to restart the partition of the controlled environment (i.e., removing all input).

between tasks/instantiations. We refer to this as an instantiated controlled environment.

In the following two subsections we present first how the classification of output documents is performed and then discuss the different deployment alternatives.

2.1 Classification Methods

To construct the classifier we use two approaches: one based on machine learning algorithms and one relying on information retrieval techniques (see Sect. 3.3 for additional details). In the case of machine learning, the input documents are used as the training set for the classifier, which is then used to determine the classification of the output document(s) directly. In the information retrieval approach, we build an index of all the input documents and their associated classification, so that we can query it to obtain a list of the input documents matching the output one(s), ranked by a similarity score. This list is then used to determine the more likely classification of the output.

2.2 Deployment

A controlled environment may in principle be any computing environment where the incoming information flow can be monitored. A monitoring mechanism similar to a guard, inspecting data at the application level, can be applied to control the information flow to a network/security domain, a virtual machine, a computer, a process, or some isolated container such as a sandbox or Docker instance. Cloud computing, with its extensive use of virtual machines, may also facilitate controlled environments.

Figure 2 shows a possible deployment of the proposed solution using a separation kernel. A separation kernel is a minimalistic type 1 hypervisor, running directly on the computer hardware, providing high assurance in the isolation between partitions (virtual machines) and controlled communication channels. As can be seen, monitoring of input and classification of output is performed within a separate partition, while the controlled user/application environment resides within another partition. The separation kernel ensures that all communication (e.g., documents from a file server) to/from the controlled environment has to pass through the monitoring and classification solution. The separation kernel can also be configured to allow the monitoring and classification partition to restart the controlled environment partition in order to clear its input state.

2.3 Applicable Scope

Unlike the work in [7,13], we do not aim to construct a classifier that is able to universally distinguish between what the government or some other organization considers to be classified and unclassified information; instead, we want to discover if sensitive information from a particular set of documents has been included (possibly in modified form) into a new document. Thus, the proposed method is not intended to be able to detect misclassification of output documents where the user has included sensitive content accessed through other channels (e.g., a second computer system or a paper document). Although this could potentially be addressed by combining our approach with those discussed in [7,13], this is not pursued here.

Evasion and Poisoning. There are multiple ways to influence the machine learning classifier. By carefully choosing what to import, the training set can potentially be shaped in such a way that the algorithm later misclassifies a document containing sensitive content that the user wants to exfiltrate. This is an example of what is referred to in the literature as a *Causative* attack [4] in which the training set is intentionally poisoned. The Reject On Negative Impact (RONI) defense technique addresses this by modifying the learning process to dynamically discount those data points in the training set that have a significant negative impact on the performance [4]. Usage of procedures from the field of Robust Statistics has also shown to partially remedy the poisoning issue [4]. Also, performing a causative attack to exfiltrate larger amounts of data would likely result in detectable anomalies in the import patterns.

The IR approaches are not as susceptible to the *Causative* class of attacks because: (1) there is no training phase involved and (2) the presence of a classified document with a similarity score greater than the threshold value (see Sect. 3.3) will result in assigning the strictest label, e.g., "Classified", to the document. Both the ML and IR methods remain vulnerable to the fact that if a highly skilled attacker has knowledge of the training set and hypothesis space, then he can shape the contents of the output document such as to stealthily avoid detection. This is known as an *Exploratory Attack*, in which the attacker does not influence the training phase [4]. This is a generic challenge in data loss detection, where this work advance compared to most previous work only considering unmodified data leaks.

The possibility of a skilled attacker evading the detection mechanism underlines why detection of malicious insider data leakage is considered a challenging research problem (e.g., [26]), and why said detection methods need to be deployed in combination with other countermeasures. As a controlled environment maintains control of the documents imported into the environment, we plan as further work to investigate how this information can be used to determine the risk an environment poses with regard to information leakage and to identify potential insiders. Also, it should be kept in mind that much of the data leakage by internal actors is due to accident (i.e., half of the serious incidents according to a recent study [15]).

3 Evaluation Methodology

In order to evaluate the effectiveness of the proposed solution, we needed to perform experiments using documents of known classification. As information from a sensitive document may leak by being included in a new document, we wanted to evaluate how modifications and introduction of external noise in output documents affect performance. We therefore introduced several methods to create new output documents based on manipulation of input documents and inclusion of external noise. This is further described in Sect. 3.1.

As the aforementioned approach does not accurately reflect how a document would be generated by a real user, we also use a second approach to validate our results. Using this approach we did not generate new output documents based on input documents (and noise), as in the first approach. Instead we used another set of documents with known classification as output documents, and used each of these document's reference list as an approximation of the input documents accessed during its creation. This latter approach is further described in Sect. 3.2.

Section 3.3 briefly describes the machine learning and information retrieval algorithms used as classifiers in the evaluation.

3.1 Evaluation Approach 1

Here we present the first dataset and describe each of the document transformations that are used to generate the output documents.

DNSA Dataset. The U.S. Digital National Security Archive contains the most comprehensive collection of declassified U.S. government documents available to the public [1]. From this repository we extracted three subcollections:

1. *(Af)ghanistan: The Making of U.S. Policy, 1973–1990*;
2. *(Ch)ina and the United States: From Hostility to Engagement 1960–1998* and
3. *The (Ph)ilippines: U.S. Policy during the Marcos Years, 1965–1986.*

These were chosen because they contained a mix of both classified and unclassified documents from unrelated domains and from partially overlapping time periods.

Out of the 5853 retrieved documents, 1612 were dismissed because they had gone through a declassification process. The "PublicUse" (3 docs) and "Restricted" (1 doc) documents were removed due to their limited numbers. All documents marked either as "Unknown" (620) or "LimitedOfficial" (685) were removed. Documents that consisted of less than 30 words were also excluded. Post-filtering, the final data set consisted of 2884 documents. These documents were then partitoned into two categories depending on their original label. The 1117 documents marked "Unclassified" or "Non-Classified" were assigned the label *Unclassified.* While the remaining 1767 documents marked either "Confidential" or "Secret" were assigned the label *Classified.* In the final dataset, 525 out of the 883 documents belonging to the AF subset, 661 out of the 959 CH documents, and 610 out of the 1042 PH documents were labeled as *Classified.*

The final data set was divided into the *Input* (used as input documents) and the *Noise* datasets. The *Noise* dataset was used to introduce external noise into output documents.

For each PDF file we utilized the 3rd party OCR service Abbyy[2] to extract the textual contents. Any non-English words, i.e., artifacts of the OCR process, were then filtered out as part of a pre-processing phase as were any variations of tokens of the type *Secret*, *Unclassified* etc., as their inclusion would potentially result in the classifiers yielding artificially good results, that effectively would only determine the security classification based on the absence or presence of such words.

Output Generation. We create new output documents the following ways:

1. **Existing Documents:** In this case an existing document from the DNSA dataset is used unmodified.
2. **Mixed Documents:** In this case we randomly sample sentences from two documents of different classification, with each document contributing 50 % of the output document. We take one document from the *Input* set and the other from the *Noise* set. The resulting document is then assigned the highest security classification of the two documents used in the mixing, the rationale being that the introduction of a single classified sentence or keyword into a unclassified document would imply that the correct security classification of the resulting document would be classified.
3. **Rewritten Documents:** A document can be rewritten while still being semantically identical to the original, and should thus retain it's security label. We implement this by fitting a n-gram language model [17] for each document, and then use these probability distributions to generate new documents that contain semantically similar content. The correct security classification of the output document is assumed to be the same as that of the original document.
4. **Abstract:** For each document in the corpus there is an accompanying short abstract (not included as input) which we take as a condensed version of the document. The correct security label of the condensed document is assumed to be the same as that of the full document.
5. **Spinner:** An article spinner is a tool used in search engine optimization to generate documents for content farms and to circumvent plagiarism detection algorithms in general. It works by rewriting an article by replacing words, phrases and paragraphs with synonyms. We used the online commercial tool "Spin Rewriter 6.0"[3]. As this is proprietary software the exact algorithm used to "spin" documents is not known.
6. **Translation:** Using the online service Google Translate[4] we introduce noise into the document by performing a sequence of translation steps. In our experiments we utilize the translations steps: English $\rightarrow$ Simplified Chinese $\rightarrow$ English.

The "Exisiting", "Translation" and "Synonym" ("Spinner") transformation methods have also been used in the CLEF 2014 plagarism dection challenge [24].

[2] http://www.abbyy.com/.
[3] https://www.spinrewriter.com/.
[4] https://cloud.google.com/translate/docs.

It should be noted that “Abstract” is the most realistic transformation as it uses real abstracts created by human editors.

3.2 Evaluation Approach 2

Here we present the second dataset used and then briefly describe how input documents were obtained based on the reference list of each output document in the dataset. As mentioned earlier, the intent behind introducing this second set of data and accompanying experiments is to further ensure that the results of the first approach is not attributed to the artificial way we generate output documents.

Private Dataset. This dataset was based on an internal repository of technical reports at the Norwegian Defence Research Establishment (FFI). We collected a document set of 10 classified and 10 unclassified documents, to be used as output documents. The reference lists of these documents consisted of a mixture of both classified and unclassified reports, notes, and conference papers. The number of references per document ranged from 5 to 17, with a total set of 166 references (used as input documents).

For each document and its references we extracted the textual contents and then pre-processed the resulting text using the same procedure as for the first dataset, but with additional word filters customized to remove location and domain specific tokens that identifies the security label. For each of the documents we also removed the list of references from the text.

Input Generation. As mentioned, this approach approximates the set of input documents by using the documents in the reference list of the output document instead. In this case the correct security classification of the output document is assumed to be the one stated on the original (output) document. Likewise, the security classification of an input document is also the one originally indicated on the document itself.

3.3 Algorithms

In this section we introduce the information retrieval and machine learning algorithms utilized in the experiments. For each algorithm, we performed 5-fold cross-validation and a grid-search to find the optimal configuration of tunable parameters.

Information Retrieval. In the field of information retrieval, a collection of indexed documents can be searched by computing the similarity between the documents and a query string. In the context of controlled environments, the similarity for each output/input pair can be used to detect if parts of the output document originates from the input document.

Each output document comes attached with the user’s assigned label, and it is only interesting to run the detection routine for instances where this label may result in the document being released/leaked, i.e., *Unclassified* in our dataset. The following three-step algorithm computes the predicted security label:

1. Compute the similarity between the output document and each of the input documents.
2. Take the label of the input documents whose score is the highest as a *tentative* label.
3. If the tentative label is *Unclassified*; we test whether the best *Classified* match has a higher score than some threshold θ. If this is the case: the final label is taken to be *Classified*. Otherwise we proceed with the *tentative* label.

The implementations provided by open-source search engine ElasticSearch [11] and the Python package gensim [22] were used for the experiments. As there are a number of different retrieval and scoring methods, we perform initial experiments with algorithms belonging to each of the four families:

1. **TF-IDF/VSM:** In the tf-idf vector-space model (VSM), the cosine similarity between the term frequency inverse-document (tf-idf) vector representations of a document and the query term is computed (refer to Sect. 3.3). This model is typically used as a baseline ranking method [3].
2. **BM25:** Okapi BM25 is a probabilistic algorithm built on top of the TF-IDF method and has been empirically shown to outperform the regular TF-IDF method in many experiments [23].
3. **LMJelinekMercerSimilarity:** A method that uses a statistical language model, e.g., a multinomial distribution over a sequence of one or more words, to represent each document in the collection [21]. Ranking is then performed by computing the likelihood of a document generating a query. This particular implementation uses Bayesian smoothing with Dirichlet priors [28].
4. **IB:** A family of information based retrieval models that builds on the core idea that a words' statistical behaviour differs on the document and collection level [9].

Machine Learning. Machine learning aided text classification builds on the ability of the underlying algorithm to correctly learn the essential features that characterize a certain category of documents from a set of given examples, i.e., the training set, so that it can automatically classify new documents in the right category among those it learned.

While the goal of the traditional machine learning classification task is to train a classifier that is able to handle out-of-sample data points, e.g., tasks such as the real-time detection of pedestrians, malicious binaries or OCR, there is a strong overlap between the in-sample and out-of-sample data points in our setting, as we assume that information in the input documents (used for training) are used to generate the output documents. Under these conditions one can argue that a certain degree of overfitting is not only unharmful but in fact beneficial.

All fitting was performed using the open-source Python machine-learning library scikit-learn [20].

Features. For features we compute the tf-idf weights, and represent each document by a high-dimensional sparse vector $\mathbf{x}$, whose x_i entry is the tf-idf weight of the word (token) associated with dimension i. For example, a document denoted by d containing the words 'man', 'missile' and 'aide' would be transformed into the vector:

$$\mathbf{x}_d = \left[\overset{\text{man}}{\overbrace{x_{d,1}}} \; \overset{\text{missile}}{\overbrace{x_{d,2}}} \; \ldots \; \overset{\text{aide}}{\overbrace{x_{d,N}}} \right] \tag{1}$$

where N is the vocabulary size, with the word *aide* being mapped to the last dimension. The entries $\mathbf{x}_{d,t}$ are the tf-idf weights defined as[5]

$$\mathbf{x}_{d,t} = \underbrace{\sqrt{f_{t,d}}}_{\text{tf}} \times \underbrace{\left(1 + \log \frac{|D|}{1 + |\{d \in D : t \in d\}|}\right)}_{\text{idf}} \tag{2}$$

where $f_{d,t}$ denotes the frequency of term t in document d and D the set of all documents.

We performed experiments using the RandomForest [6], Adaboost [28], SVM [5] and logistic regression [5] packages implemented in sklearn [20]. However, we only discuss the linear SVM multiclass algorithm in further detail as it yielded the best performance. It works by searching for linear functions (one for each class) whose score for the correct class (e.g., *Secret*) is at least 1 higher than the other classes (e.g.,. *Unclassified*, *Top Secret*, *Classified*). This idea is mathematically expressed by the non-convex minimization problem:

$$\min_{\mathbf{W}} L(\mathbf{W}) = \underbrace{\frac{1}{M} \sum_i \sum_{j \neq y_i k} \max(0, \mathbf{w}_j^T \mathbf{x}_i - \mathbf{w}_{y_i}^T \mathbf{x}_i + 1)}_{\text{data loss}} \quad + \quad \underbrace{\frac{\lambda}{2} ||\mathbf{W}||^2}_{l_2 \text{ regularization loss}} \tag{3}$$

Because there is an inbalance between the classes in the dataset, which is a situation one must expect to handle in a real-life deployment, we experimented with the use of a pre-training reweighting mechanism where:

$$y_i^* = \frac{M}{|C| \times |\{y \in Y : y = y_i\}|} \tag{4}$$

with $|C|$ and M denoting the number of classes and samples respectively. This has the effect of automatically adjusting the weights proportional to the inverse class frequencies.

4 Evaluation Results

We here first provide evaluation results from using the DNSA dataset (evaluation approach 1), and then additional validation results using the private dataset (evaluation approach 2).

[5] It should be noted that there exists a great number of heuristic variations of the tf and idf formulas.

4.1 Evaluation Approach 1

The training and indexing is performed on the DNSA dataset, which is also used to generate the sets of output documents described in Sect. 3.1. In order to verify whether a controlled environment improves the detection of content from a classified input document being embedded in some transformed way in an output document, we set up the following experimental set-up with three types of classifiers:

1. **Global:** First we create a global classifier trained on all available documents from the DNSA dataset. This constitutes the baseline accuracy achieved by traditional ML and IR methods, where more information is supposed to give better results.
2. **Domain specific classifiers:** Then we create three more context-sensitive classifiers, where only documents pertaining one of the three subcollections AF, CH and PH are used for training/indexing. This is done as a first verification that a classifier/index set built on more context-specific documents can indeed give better accuracy than one built on possibly more, but less specific data, even when creating the output documents from a relatively diverse and large input set. That is, better results are not only an effect of creating output documents from a small and not very diverse set of input documents.
3. **Dynamic per-user:** Finally we create controlled environments where we gradually increase the amount of imported documents used for training/indexing. This to test our hypotheses that using additional documents for training besides those strictly relevant to the output documents does not necessarily increase accuracy, but rather constitutes noise.

For all three types of classifiers we test how they perform on the different transformations of documents from each separate subcollection as defined in Sect. 3.1. We also test how they behave when used both as a binary classifier (i.e. Unclassified and Classified labels) and a ternary one (i.e. Unclassified, Confidential and Secret labels). For IR algorithms we use a threshold value $\theta = 0.15$ and we measure also how often they are able to identify the input document used to generate the output document, i.e., the number of exact matches.

If our hypothesis is correct, then dynamic per-user classifiers should provide the best accuracy when used on smaller but relevant input sets, and their accuracy converge toward domain specific and global classifiers as more and more noise is introduced in the input set. Domain specific classifiers should also provide some better accuracy than global ones. The results we obtained are summarized in Table 1 and partially illustrated in Fig. 3, and confirm exactly this kind of behaviour.

Global Classifiers. We train a classifier using the complete set of documents as the training set, and then measure the predictive performance in terms of accuracy on each of three subcollections AF, CH and PH separately. For ML we only include results for SVM as it consistently outperformed the other ML methods included in our experiments (see Sect. 3.3) on all output classes.

Table 1. Mean accuracy for the SVM and IR methods when deployed using a **global** (Global), **domain specific** (DS) and **dynamic per-user** (User) classifier. The performance is reported when operating with: two security classes (**Binary** - Unclassified/Classified), three classes (**Tenary** - Unclassified, Confidential and Secret) and exact match, i.e., counting only instances where the best match is the input document used to generate the output for the IR approach (**Exact**). The reported controlled environment accuracy is when using a classifier trained with 25 Unclassified and 25 Classified input documents.

		Transformation	Binary			Tenary			Exact		
			Global	DS	User	Global	DS	User	Global	DS	User
IR	China	`Abstract`	.79	.81	.93	.66	.68	.86	.33	.36	.75
		`Spinner`	.94	.94	.99	.99	.99	.99	.99	.99	.99
		`Rewritten`	.99	.99	.99	.99	.99	.99	.99	.98	.99
		`Translation`	.99	.99	.99	.98	.98	.99	.97	.97	.99
		`Mixture`	.72	.78	.89	-	-	-	-	-	-
		`Unmodified`	.99	.99	.99	.99	.99	.99	.99	.99	.99
	Afghanistan	`Abstract`	.58	.57	.85	.46	.47	.77	.22	.27	.60
		`Spinner`	.96	.96	.99	.93	.95	.99	.92	.93	.99
		`Rewritten`	.99	.99	.99	.99	.99	.99	.99	.99	.99
		`Translation`	.99	.99	.99	.99	.99	.99	.99	.98	.99
		`Mixture`	.81	.85	.92	-	-	-	-	-	-
		`Unmodified`	.99	.99	.99	.99	.99	.99	.99	.99	.99
	Philippines	`Abstract`	.69	.70	.85	.50	.56	.78	.30	.30	.66
		`Spinner`	.99	.99	.99	.99	.99	.99	.95	.95	.99
		`Rewritten`	.99	.99	.99	.99	.99	.99	.99	.99	.99
		`Translation`	.99	.99	.99	.99	.99	.99	.99	.96	.99
		`Mixture`	.67	.68	.75	-	-	-	-	-	-
		`Unmodified`	.99	.99	.99	.99	.99	.99	.99	.99	.99
ML	China	`Abstract`	.63	.81	.84	.54	.59	.64	-	-	-
		`Spinner`	.98	.98	.99	.95	.95	.99	-	-	-
		`Rewritten`	.99	.98	.99	.96	.97	.98	-	-	-
		`Translation`	.96	.98	.99	.90	.97	.97	-	-	-
		`Mixture`	.76	.79	.92	-	-	-	-	-	-
		`Unmodified`	.99	.99	.99	.99	.99	.99	-	-	-
	Afghanistan	`Abstract`	.50	.61	.76	.53	.54	.62	-	-	-
		`Spinner`	.96	.96	.99	.94	.93	.99	-	-	-
		`Rewritten`	.99	.99	.99	.97	.96	.99	-	-	-
		`Translation`	.95	.97	.99	.89	.95	.97	-	-	-
		`Mixture`	.81	.82	.96	-	-	-	-	-	-
		`Unmodified`	.99	.99	.99	.99	.99	.99	-	-	-
	Philippines	`Abstract`	.62	.68	.80	.45	.53	.62	-	-	-
		`Spinner`	.97	.99	1.0	.96	.98	.99	-	-	-
		`Rewritten`	.97	.99	1.0	.98	.97	.99	-	-	-
		`Translation`	.96	.97	.99	.89	.96	.99	-	-	-
		`Mixture`	.59	.61	.70	-	-	-	-	-	-
		`Unmodified`	.99	.99	.99	.99	.99	.99	-	-	-

Domain Specific Classifiers. For the domain specific classifiers we train three classifiers using each of the subcollections AF, CH, and PH separately as training sets. We then evaluate the classifiers on the corresponding subcollections that were used in the training phase. Each classifier provides slightly better accuracy than the global classifier when detecting the classification of output documents generated from the corresponding subcollection.

Dynamic Per-User Classifier. We predict that a smaller input set will give better results because of less noise from potentially unrelated sources. In order to test this, we perform experiments in which we initially start with a sample of 25 classified and 25 unclassified documents. We then iteratively increase the set of input documents (namely the noise) by sampling from the remaining dataset while testing the classifier on outputs generated only from the initial 50 documents. This sequence of steps is then repeated 200 times and the mean accuracy is computed. Figure 3 (top) shows how the accuracy, for the "Abstract" output documents, decays as the size of the set of input documents increases. This shows how the classification task becomes increasingly difficult as more noise is introduced into the environment, approaching the performance of the domain specific classifier as the size of the input set increases. For the "Mixture" class we have taken one classified document from the *Input* and one unclassified document from the *Noise* datasets and combined a sample section of (50%) from each. This is meant to illustrate the effect of introducing noise from an external source, i.e., content which is not present in the training set/index. When evaluating the performance for the "Mixture" documents we measure the accuracy only for Classified output documents.

4.2 Discussion

Table 1 displays the accuracy for the ML and IR approaches when evaluated using a Global, Domain Specific and Dynamic Per-User classifier. It is clear that all approaches are robust with respect to the "Rewritten", "Spinner" and "Translation" transformations and further experiments show that only a minor benefit is achieved when deploying a controlled environment for these classes. The high accuracy can be traced back to how these output documents are generated from the input documents and how the SVM and TF-IDF methods works. When using a bag-of-word representation in which the order of words are discarded, the relative frequencies of the input documents remain invariant with respect to the transformations, resulting in what (for the algorithms) are very similar documents. For the "Translation" transformation the intermediate steps distorts the semantics while retaining the TF-IDF weights of the orignal document.

The "Abstract" and "Mixture" categories appear to be more challenging transformations. If we look at the accuracy plots displayed in Fig. 3, we see how the introduction of the more context aware Domain Specific classifier offers a moderate increase in performance compared to the Global classifier. Proceeding one step further by deploying the Dynamic Per-User classifier yields a significant

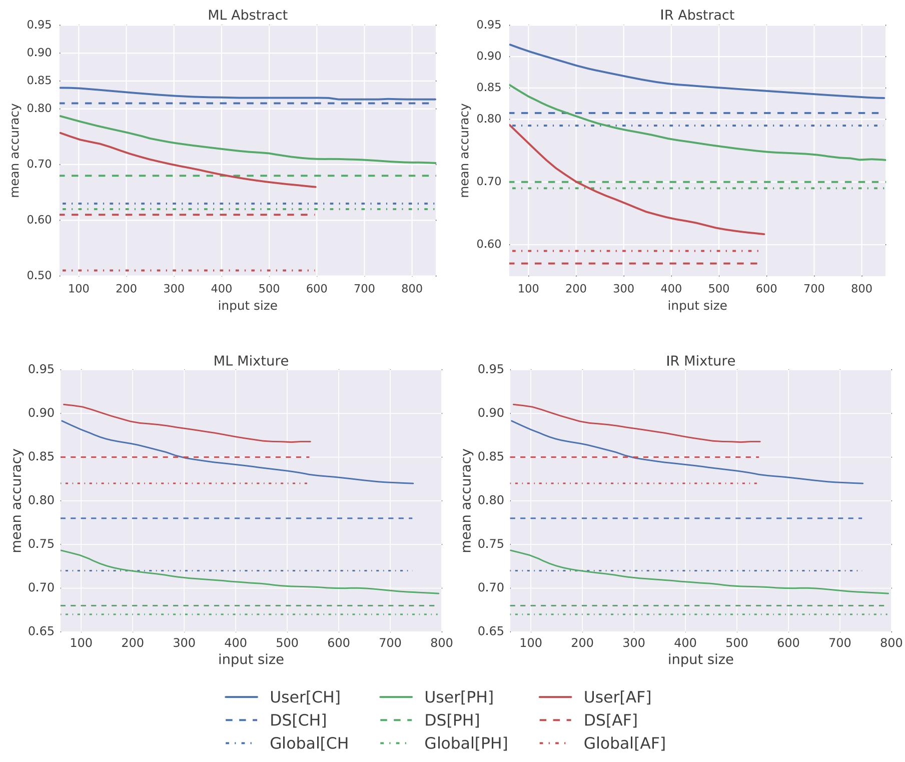

Fig. 3. Accuracy as a function of the size of the controlled environment for the "Abstract" and "Mixture" output class. Left: Machine learning (l_2-regularized SVM) based classifier. Right: Information retrieval (TF-IDF VSM/Cosine) based classifier. **Legends**: *Global[XY]* a global classifier trained an the complete training set (AF, PH and CH) and evaluated on the XY dataset, where XY can be AF (Afghanistan), PH (Philippines) and CH (China). *DS[XY]* denotes a domain specific classifier trained and evaluated on the XY dataset. *User[XY]* a dynamic per-user classifier trained on the input for a controlled dataset from the XY dataset.

increase in performance, and the plot illustrates how the performance decays as the size of the input increases until it finally converges to the performance offered by the Domain Specific classifier.

From the graph it is clear that the introduction of a second source of noise further reduces the effectiveness of the algorithms, but that the use of a controlled environment still significantly improves accuracy compared to a global or domain specific classifier. There is a notable drop in accuracy as we go from two to three security classes and when we go from three classes to only measuring exact matches. However the table shows that we still achieve a bump in performance by using a more tightly controlled environment.

From Table 1 and Fig. 3 we can conclude that the IR and ML methods yield comparable performance when deployed using a controlled environment. However, the IR approach offers a few advantages. Firstly, the threshold value θ (Sect. 3.3) limits the viability of *Causatitive* attacks. Secondly, since they work by comparing the output document with a set of input documents, it is straightforward to explain a classification outcome to the end user. Likewise, it is easy to perform an error analysis by studying the nature of the documents it gets wrong. For example, in our study we randomly sampled a set of classified documents that were misclassified. We discovered that there were several instances in which the best match was an official invitation to an event or meeting, while the classified document, the one used as a query, was an internal memo detailing what the attending officials political position, talking, and view points should be.

4.3 Evaluation Approach 2

The primary motivation behind a second evaluation approach is to investigate whether or not the previous results can be attributed to the artificial way we generate output documents. E.g., if the generated output documents are much less diverse than those created by users of a real-life system, it could be that a larger but less specific training set might outperform the more context-aware given more diverse output documents.

For data we use the internal dataset described in Sect. 3.2, and we evaluate using the IR approach. We did not evaluate the ML classifiers on this dataset as many of the documents did not have enough references to perform the fitting. For each document we index all of its references (and their classification), and use the text of the document as the query string. From the set of results we take the security label of the document with the highest score to be the correct one. All three scoring algorithms have an accuracy of approximately 78%. This might seem like a lackluster result given that the small size of the input set should provide a more meaningful context for the classifier. However, an analysis of the set of misclassifications reveals that most of the incorrectly classified documents are unclassified reports whose best match is the larger and more extensive classified version, which naturally has a large amount of overlap. If we remove these documents, accuracy improves to 89%. As shown in Fig. 4, there is significant benefit in using a controlled environment compared to a global classifier for this second set of documents, providing further evidence in support of our hypothesis.

5 Related Work

Despite the long history of applying machine learning and information retrieval to the text categorization task [16]; not much research has focused on using these methods to automate security classification in the context of DLP. This despite being recognized as a relevant area by commercial vendors [27].

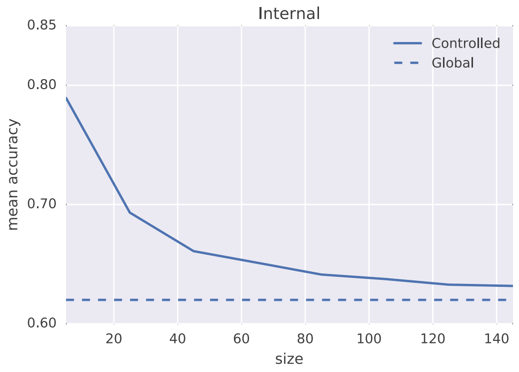

Fig. 4. Accuracy as a function of the size of the controlled environment for the internal dataset compared to when using a global classifier.

The first work we are aware of is that of Brown et al. [7]. This study used a smaller part of the DNSA corpus [1] to investigate whether a ML classifier trained on documents regarding a particular topic or time-period would generalize to other topics/time-periods. Their goal was to develop a tool to aid manual classification and declassification of documents, rather than detect leakages. These results were later validated and expanded upon in [13] and the classifiers then fine-tuned in [10]. In all these cases only unmodified out-of-sample documents were used as a test-set.

The concept of an automatic security classification framework based on machine learning has also been proposed by Kassidy [8], but without any implementation or validation. Lewellen et al. [18] proposed to use plagiarism techniques to detect data leaks. This is mostly a technical document on how to set up a system that uses standard indexing techniques to detect parts of sensitive text copied and pasted, for instance, in e-mails. Another related work uses N-grams statistical analysis to detect sensitive documents even after text spinning [2]. While our work is similar in that we investigate how the classifier precision changes when documents are manipulated or degraded, we use additional manipulation techniques and also utilize a large and more realistic dataset. Besides, their classification is based on topics rather than sensitivity, so it is not possible to directly relate our results. We argue that classifying documents per topic as they do, is easier as a topic can be well-characterized by specific words and expressions, while sensitivity can be based, for instance, also on political or strategic motivations.

Hart et al. [14] first evaluate various ML algorithms as we do, but then settle on a SVM-based classifier and then develop a new supervised training algorithm that can automatically distinguish between secret, public, and non-enterprise documents. They use various supplement training and adjustment techniques in order to compensate for the imbalance and false positives due to the non-enterprise documents. While we operate with a data set whose security labels have been assigned by government personnel and has later gone through a declassification process, the data used in their study comes from several smaller private sources, and is a mixture of internal handbooks for religious organizations (LDS[6], TM[7]), corporate emails and private blogposts regarding software-engineering. Furthermore, they do not seem to account for possible manipulation of the documents to be classified either, but use only documents known to the classifier without any modification. Shu et al. [26] introduces a scalable sequence alignment algorithm for detecting data leaks in transformed documents. However, their focus is restricted to a special class of transformations, e.g., replacing white space characters with the "+" character.

6 Conclusion

This paper explored the idea of using controlled environments and dynamic classifiers to better determine the security classification of transformed documents, by providing a more relevant context. By deploying a controlled environment that monitors and registers all imported documents, i.e., the documents accessed by a user during a session, we can subsequently check if any documents that are later exported contain information whose origin is a classified source. In our work we proposed two ways to automate this process using methods from the fields of machine learning and information retrieval. A controlled environment can be instantiated per single user, virtual machine, network, or process, potentially for each work session or for longer time periods, and can be re-initialized when required.

In contrast to the traditional approach where a global classifier is trained using all the available data of an enterprise or organization, a classifier in a controlled environment has the advantage of a more targeted context, as it knows which data has been accessed during a session. Extensive evaluation presented in Sect. 4, using a dynamic per-user environment (trained on the documents accessed by the user), a domain specific (trained using documents from the same subcollection) as well as a global classifier (trained using all available data), supports the hypothesis that the introduction of a controlled environment is beneficial to DLP. Especially for difficult detection tasks there is substantial benefit in using a controlled environment as witnessed in Fig. 3, where we see how the accuracy decays as more irrelevant data is imported into the controlled environment. For the easier detection tasks (results presented in Table 1) there was no significant gain to be achieved by deploying a controlled environment.

In order to validate that our results also hold water in a operational setting, i.e., that the observed boost in performance can not be attributed to the artificial way we generate output documents, we conducted experiments (Sect. 4.3) using a separate data set consisting of a private repository of classified and unclassified reports from our research institution. For each of these reports we assumed that the references were the documents imported into the controlled environment while the report itself was the output at the end of a session.

[6] The Church of Jesus Christ of Latter-day Saints.
[7] Transcendental meditation.

As part of future work we intend to perform a more fine-grained analysis in which we operate on the sentence or paragraph level, e.g., we want to investigate whether we can detect if a transformed chunk of text in an output document can be traced back to a classified document in the input set.

We also plan to use the predicted label probabilites or scores of the classifiers to build an insider threat meta score, which would help facilitate the detection of anomalous behaviour on the user level, thus mitigating the threat of *Causative* attacks. Furthermore, an insider score would mitigate false positives by analyzing user labeling trends over a longer time horizon instead of operating on a per-document level.

References

1. Digitial National Security Archive. http://nsarchive.chadwyck.com/home.do. Accessed 26 Mar 2015
2. Alneyadi, S., Sithirasenan, E., Muthukkumarasamy, V.: A semantics-aware classification approach for data leakage prevention. In: Susilo, W., Mu, Y. (eds.) ACISP 2014. LNCS, vol. 8544, pp. 413–421. Springer, Heidelberg (2014). doi:10.1007/978-3-319-08344-5_27
3. Baeza-Yates, R., Ribeiro-Neto, B., et al.: Modern Information Retrieval, vol. 463. ACM Press, New York (1999)
4. Barreno, M., Nelson, B.A., Joseph, A.D., Tygar, D.: The security of machine learning. Technical report UCB/EECS-2008-43, EECS Department, University of California, Berkeley, April 2008. http://www.eecs.berkeley.edu/Pubs/TechRpts/2008/EECS-2008-43.html
5. Bishop, C.M.: Pattern Recognition and Machine Learning. Springer, New York (2006)
6. Breiman, L.: Random forests. Mach. Learn. **45**(1), 5–32 (2001)
7. Brown, J.D., Charlebois, D.: Security classification using automated learning (scale): optimizing statistical natural language processing techniques to assign security labels to unstructured text. Technical report, DTIC Document (2010)
8. Clark, K.P.: Automated security classification. Master thesis Vrije Universiteit (2008)
9. Clinchant, S., Gaussier, E.: Information-based models for ad hoc IR. In: Proceeding ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 234–241 (2010)
10. Engelstad, P.E., Hammer, H., Kongsgård, K.W., Yazidi, A., Nordotten, N.A., Bai, A.: Automatic security classification with lasso. In: Proceeding International Workshop on Information Security Applications (2015)
11. Gormley, C., Tong, Z.: Elasticsearch: The Definitive Guide. O'Reilly Media Inc., Sebastopol (2015)
12. Haakseth, R., Nordbotten, N.A., Jonsson, Ø., Kristiansen, B.: A high assurance guard for use in service-oriented architectures. In: Proc. International Conference on Military Communications and Information Systems (2015)
13. Hammer, H., Kongsgård, K.W., Bai, A., Yazidi, A., Nordbotten, N.A., Engelstad, P.E.: Automatic security classification by machine learning for cross-domain information exchange. In: Proceeding IEEE Military Communications Conference, vol. 31 (2015)

14. Hart, M., Manadhata, P., Johnson, R.: Text classification for data loss prevention. In: Fischer-Hübner, S., Hopper, N. (eds.) PETS 2011. LNCS, vol. 6794, pp. 18–37. Springer, Heidelberg (2011). doi:10.1007/978-3-642-22263-4_2
15. Security, I.: Grand theft data - data exfiltration study: actors, tactics, and detection (2015). http://www.mcafee.com/us/resources/reports/rp-data-exfiltration.pdf
16. Joachims, T.: Text categorization with support vector machines: learning with many relevant features. In: Nédellec, C., Rouveirol, C. (eds.) ECML 1998. LNCS, vol. 1398, pp. 137–142. Springer, Heidelberg (1998). doi:10.1007/BFb0026683
17. Jurafsky, D., Martin, J.: Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition. Prentice Hall series in Artificial Intelligence, Pearson Prentice Hall (2009). https://books.google.no/books?id=fZmj5UNK8AQC
18. Lewellen, T., Silowash, G.J., Costa, D.L.: Insider threat control: Using plagiarism detection algorithms to prevent data exfiltration in near real time. Technical report CMU/SEI-2013-TN-008, Carnegie Mellon University (2013)
19. Ouellet, E.: Magic quadrant for content-aware data loss prevention. Gartner Inc. (2013)
20. Pedregosa, F., et al.: Scikit-learn: machine learning in Python. J. Mach. Learn. Res. **12**, 2825–2830 (2011)
21. Ponte, J.M., Croft, W.B.: A language modeling approach to information retrieval. In: Proc. ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 275–281 (1998)
22. Řehůřek, R., Sojka, P.: Software Framework for Topic Modelling with Large Corpora, pp. 45–50, May 2012. http://is.muni.cz/publication/884893/en
23. Robertson, S., Zaragoza, H.: The probabilistic relevance framework: BM25 and beyond. Foundations and trends in information retrieval (2009)
24. Rosso, P., Stein, B.: Overview of the 6th international competition on plagiarism detection
25. Shabtai, A., Elovici, Y., Rokach, L.: A Survey of Data Leakage Detection and Prevention Solutions. Springer, New York (2012)
26. Shu, X., Zhang, J., Yao, D., Feng, W.C.: Fast detection of transformed data leaks. IEEE Transactions on Information Forensics and Security (2016)
27. Symantec: Machine learning sets new standard for data loss prevention: describe, fingerprint, learn (2010). http://eval.symantec.com/mktginfo/enterprise/white_papers/b-dlp_machine_learning.WP_en-us.pdf
28. Zhai, C., Lafferty, J.: A study of smoothing methods for language models applied to ad hoc information retrieval. In: Proceeding ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 334–342 (2001)