Knowl Inf Syst (2012) 32:167–189
DOI 10.1007/s10115-011-0438-9

REGULAR PAPER

Similarity assessment for removal of noisy end user license agreements

Niklas Lavesson · Stefan Axelsson

Received: 2 May 2010 / Revised: 29 March 2011 / Accepted: 18 July 2011 /
Published online: 28 July 2011

Abstract In previous work, we have shown the possibility to automatically discriminate between legitimate software and spyware-associated software by performing supervised learning of end user license agreements (EULAs). However, the amount of false positives (spyware classified as legitimate software) was too large for practical use. In this study, the false positives problem is addressed by removing noisy EULAs, which are identified by performing similarity analysis of the previously studied EULAs. Two candidate similarity analysis methods for this purpose are experimentally compared: cosine similarity assessment in conjunction with latent semantic analysis (LSA) and normalized compression distance (NCD). The results show that the number of false positives can be reduced significantly by removing noise identified by either method. However, the experimental results also indicate subtle performance differences between LSA and NCD. To improve the performance even further and to decrease the large number of attributes, the categorical proportional difference (CPD) feature selection algorithm was applied. CPD managed to greatly reduce the number of attributes while at the same time increase classification performance on the original data set, as well as on the LSA- and NCD-based data sets.

Keywords End user license agreement · Latent semantic analysis · Normalized compression distance · Spyware

1 Introduction

The amount of spyware has increased dramatically due to the high value for marketing companies of the information that is collected. Spyware is designed to collect user information for marketing campaigns without the informed consent of the user. This type of software is commonly distributed by bundling it with popular applications available for free download. A spyware application is typically difficult to remove once it has been installed, and it

N. Lavesson (✉) · S. Axelsson
School of Computing, Blekinge Institute of Technology, 371 79 Karlskrona, Sweden
e-mail: Niklas.Lavesson@bth.se

can seriously degrade system performance and compromise the privacy of the user [5]. The motivation for conducting this study is that spyware is privacy intrusive and affects system performance. Moreover, spyware is difficult to detect and most users do not have the time or the experience needed to identify possible spyware inclusions from reading the end user license agreements that accompany most software installers.

Previous work has investigated the relationship between the contents of end user license agreements (EULAs) and the legitimacy of the associated software applications [26,27]. In the most recent study, a data set was generated from 996 EULA instances of legitimate (good) and spyware-associated (bad) software. The experimental results showed that a majority of the evaluated supervised learning algorithms significantly outperformed a baseline classifier, indicating the feasibility of the approach. Furthermore, it was determined that bag-of-words was a suitable data model for the learning task.

However, the results obtained when conducting the previous experiments also indicated that the amount of false positives (spyware classified as legitimate software) was too great for practical use. Since most software is not associated with spyware, the detector has to operate in a skewed environment, with many more good than bad instances. Due to the base rate fallacy, the false positives suppression capability of the classifier dominates its effectiveness, much as in the computer security classification task of intrusion detection [2]. The false positives problem is a general problem in data mining, but there is no universal or general solution to the problem. However, there are some approaches that are commonly applied; for example, it is possible to address the class imbalance problem by increasing the number of training instances in order to reduce the number of false positives in some situations [44]. This is not a realistic option in our case, since it is difficult to obtain more spyware instances in order to reduce the imbalance. Instead, we focus on removing noisy examples contained in the data.

The innovation presented in this paper is that we address the problem of false positives by providing means for automatically removing instances of EULAs that even human experts cannot distinguish as belonging to either spyware or legitimate software. Thus providing the learning algorithm with less "confusing" data.

1.1 Aim and scope

In the present study, the EULA classification task is further examined by investigating two approaches for decreasing the number of false positives. The aim is to experimentally compare latent semantic analysis (LSA) in conjunction with cosine similarity (CS), with normalized compression distance (NCD) applied to the problem of noisy EULA detection. Additionally, we investigate whether the categorical proportional difference (CPD) feature selection method can increase the classification performance further, while at the same time decreasing the number of attributes. The informal hypothesis is that the noisy cases identified by either method will, with some variation, match the borderline cases identified by the human examiner.

If the informal hypothesis is accepted, it could be argued that the removal of the noisy EULAs is justified. To expand on this notion, if the identification performed by a human domain expert is consistent with the identification of an automatic approach, the automatic approach may be regarded as an appropriate way of decreasing the number of false positives. The formal hypothesis is then that, once these noisy EULAs are removed, the false positive rate is significantly lower than the false positive rate on the original data set.

The informal hypothesis is difficult to test and would require extensive manual examinations of EULAs, not to mention an human expertise in information security in general

and EULA analysis in particular. Hence, in the present study, we will describe a small case study on manual examination but we will focus on the formal hypothesis and work under the assumption that the informal hypothesis is true.

1.2 Related work

Similarity detection and search is a heavily researched area since a larger number of applications, e.g., document comparison, object recognition, and so on, depend on this functionality. However, the objectives and requirements can be quite different between these applications. Typically, most similarity-based search methods depend on an indexing method and an index structure. For example, Zhang and Alhajj [43] compare different graph-based index structures for similarity search in high-dimensional metric spaces. However, in relation to the problems studied by Zhang and Alhajj, the EULA classification problem may be regarded as quite small in terms of dimensionality.

Noise reduction has been researched extensively, especially with regard to finding remedies for specific classes of problems. Recently, a thorough review of noise reduction approaches for instance-based learning algorithms has been carried out [10]. This review identifies problems associated specifically with instance-based learners and presents an approach, based on case-based reasoning, to detect instances that are either contributing negatively or positively to a particular case or problem. However, a majority of these recent approaches only focus on remedies for instance-based learning algorithms, which means that they would apply only to a small subset of the algorithms we study. Moreover, our recent work on EULA analysis has shown that instance-based algorithms do not seem to be particularly useful for the task of EULA classification.

LSA was introduced by Deerwater et al. [9] as a method for automatic indexing and retrieval, based on singular-value decomposition. It has been studied extensively during the last two decades. Berry et al. described how LSA can be used for intelligent information retrieval and demonstrated its applicability over then-standard lexical methods [4].

Hofmann highlighted some possible issues of LSA for certain applications and presented a new method, based on LSA, called probabilistic latent semantic indexing (PLSA) [20]. PLSA was claimed to have a solid statistical foundation and to define a proper generative data model. Moreover, Hofmann's experiments indicated performance gains over LSA when using PLSA. Nevertheless, LSA remains the most popular of the two and is a widely applied indexing and retrieval method with successful use in, for example: web applications [25] and spam filtering [17].

NCD was introduced by Cilibrasi [8], and its applicability to various problems such as clustering and classification was demonstrated. NCD has been experimentally evaluated on a number of problems, including: classification of biological sequences and structures [15], novelty detection in patient histories [13], mining of sequential data [23], and even static analysis of source code [3]. It has also been evaluated in terms of the impact of information distortion on the compression [19].

It is our firm belief that there has been no attempts to develop methods for detecting noisy or duplicate end user license agreement text documents before, simply because the research on the basic problem of using EULAs to distinguish between spyware and legitimate software has started rather recently. However, we argue that LSA and NCD are two good candidates to compare as a starting point of this direction of research since they have been shown to be very successful at similar tasks. For example, NCD has been shown to work well for the problems of similarity analysis and document clustering [8,19,37]. LSA-based reflection of human knowledge has been established as adequate in a several ways. For example, it

has been shown to mimic human word sorting and category judgments [24], which is quite relevant for the studied problem. Furthermore, LSA has been used to cluster other types of natural language-based text with good performance [12].

The contribution of this study is thus that we identify the problem of noisy EULAs in the context of EULA-based spyware detection and that we provide a comparison of two good candidates for conducting the task of noisy EULA removal. It is not the aim of the presented study to investigate how to best find duplicate EULAs. However, for reasons that will be presented later, the task of identifying duplicates in data sets is related to the detection of noise. There are many reliable studies on duplicate document detection. For example, refer to the study by Ye et al. on large-scale duplicate document detection [42], which is focused on shingling-based algorithms as opposed to the presented study, which focuses on term-based algorithms.

1.3 Outline

The remainder of this paper is organized as follows: in Sect. 2, a more extensive background to the studied problem is provided. Section 4 gives a detailed description of the approach to address the problem. The subsequent sections introduce the suggested similarity assessment methods. In Sect. 5, the experimental procedure is presented. The results of the experiments are then analyzed in Sect. 6. The last section includes conclusions and pointers to future work.

2 Background

From now on, the terms *bad* and *good* are used to signify spyware-associated and legitimate applications, respectively. The distributors of bad software usually try to disguise it as good in an attempt to reach as many users as possible. However, to avoid legal repercussions, they are required to mention in the end user license agreement (EULA) that spyware will indeed be installed. For obvious reasons, this information is given in a way most users find difficult to understand. (Even EULAs for legitimate software can be hard to comprehend due to their length and their extensive use of legal terminology [18].) If the distributors do *not* mention in the EULA that unwanted applications or behavior is installed then detecting it is outside the scope of this study as we study the EULA specifically for signs of spyware.

Consequently, we recognize the need for an efficient method for helping users to distinguish between good and bad software during the installation process. If spyware is detected through such a method, users can be warned about the potentially hazardous behavior of the application and avoid installing it.

2.1 Anti-virus techniques

Anti-virus techniques are used for removing malicious software (malware) such as computer viruses and worms. Malware is illegal in many countries but the same is not necessarily true for spyware. The reason for this is that spyware resides in a gray zone between what is considered legal or illegal. Thus, the actions of a piece of spyware can be interpreted as either legal or illegal depending on who you ask; what one individual regards as spyware could be considered a legitimate business application by another individual. Because of this, it is difficult to define spyware and many anti-virus companies do not categorize questionable

applications as spyware, since that exposes them to possible lawsuits from the application vendors.

2.2 Previous studies on EULA classification

In a pilot study [26], we investigated whether it was possible to take advantage of the fact that the installation of bad software has to be mentioned in the EULA. We addressed this problem by applying supervised learning algorithms to classify EULAs of both good and bad applications, in order to detect if the associated software hosts spyware. The results indicated that the approach was feasible. However, the amount of experimental data was limited (the data set featured 100 EULAs in total).

In a later study, we collected 996 EULAs (900 good, 96 bad) and the experimental results were quite promising: EULA-based spyware detection seems to be a feasible approach to alert users before the installation of downloaded applications [27].

2.3 The EULA classification task

EULA classification, as a problem, is quite analogous to that of spam classification, i.e., to distinguish between unsolicited commercial email (spam) and legitimate email. Suppose that we have a collection, I, of EULAs, each labeled either *good* or *bad*. The set of possible classes is thus $C = \{\text{good}, \text{bad}\}$.

We would like to approximate the unknown target function, $f : I \times C = \{1, 0\}$. The value of $f(i, c)$ is equal to 1 if the EULA, i, belongs to the class c or zero otherwise. It is now possible to define a classifier as an approximation function, $a : I \times C = \{1, 0\}$. The objective of the learning task is to generate a classifier that produces results as close to that of F as possible.

Theoretically, the goal is to generate a classifier, a, where $\forall i(f(i) = a(i))$, that is, a classifier that achieves a globally optimal classification accuracy. This goal is unreachable in practice, however. It is at least impossible to know whether the goal has been met since one does not have access to all possible instances. In practice, one therefore has to assume that, if a certain classification performance is estimated reliably on I, and I is regarded as a good representation of the complete set of definable instances, then that level of performance may be expected for the complete set (or at least within reasonable proximity).

2.4 Data representation

2.4.1 Pre-similarity analysis

The raw data set consists of 996 ASCII text documents.[1] The size of these documents range from 1 to 35 KB (good EULAs) and from 2 to 56 KB (bad EULAs), respectively.

In general, the bad EULA documents are definitely of greater size than the good EULA documents: 21 out of the 96 bad EULAs have a file size that is greater than 20 KB. By comparison, only 4 out of 900 good EULAs are larger than 20 KB. The raw EULA documents have not been modified in any way. The text featured in each document has been copied and pasted from an actual software product installation wizard, as described in previous work on EULA classification [27].

[1] Original data available at: http://www.bth.se/com/nla.nsf/sidor/resources.

2.4.2 Post-similarity analysis

The bag-of-words model is a common model to use in order to represent documents as feature vectors [14]. In fact, it has been found in several experiments that more sophisticated representations do not yield any significant increase in effectiveness [34], although there are some recent approaches that have shown promise. For example, Wang automatically constructed a thesaurus of concepts from *Wikipedia* and introduced a unified framework to expand the bag-of-words representation with semantic relations [39]. More research is needed in order to establish whether this type of expansion really increases performance significantly over the traditional model.

In the bag-of-words model, every word in a document represents a feature. Thus, a text document is transformed into a vector with one element for each word that occurs in the document. A weight is associated with each word, see below. A collection of documents is then represented by a set of word vectors and the dimension of the feature space is equal to the number of different words in the whole document collection. Some studies have investigated whether the use of phrases is more suitable than words when performing text classification. However, the experimental results achieved in these studies have not been as encouraging as those in studies that use words as terms [34].

There are basically two methods for assigning weights to features. The simplest is the binary method, which either assigns a value of one if the word is present in the document or a value of zero otherwise. The binary method can be used if the chosen learning algorithm can handle only nominal attributes. However, the more common method is to take the frequency of the word into account. We adopt this method and calculate frequencies using the standard *term frequency—inverse document frequency* (TF IDF) scheme [14]. The TF IDF function embodies the intuitions that the more often a term occurs in a document, the more it is representative of its content, and the more documents a term occurs in, the less discriminating it is [34].

Several variations of TF IDF exist, and they differ from each other in terms of logarithms, normalization, or other factors. The following variant of TF IDF is used in this study: given a word, w, in a EULA, d, the TF IDF weight is calculated according to Eq. 1, where N is the total number of EULAs, DocFreq is the number of EULAs containing the word and TermFreq represents the frequency of the word in the particular EULA.

$$\text{weight}(w, d) = \text{TermFreq}(w, d) \cdot \log \frac{N}{\text{DocFreq}(w)}. \tag{1}$$

3 Approach

Real-world data sets often contain classes that are not linearly separable. This is to be expected for some problems because of the very nature of those problems. However, data sets may also contain noisy examples, e.g., instances that are either incorrectly labeled or represent something that would not exist in the real world, e.g., an individual with a height of 6 ft and an age of 4 years. It is obvious that such instances may confuse a supervised learning algorithm or a generated classifier.

The main approach presented here for the detection of noisy EULAs additionally assumes that there exists a subset of examples from the good and bad class that are more related to each other than to instances of their respective class per se. In other words, the assumption is that this subset represent an undefined, and hence hidden, neutral class.

3.1 Similarity-based identification of noise

If I represents a set of EULA documents with a distribution of k bad document instances and $|I| - k$ good document instances, the objective is to find which bad and good instances are closer to instances of the other document class, as in the instance space or some transformed version of the instance space.

Definition 3.1 A neutral instance is an instance whose original class label is unequal to the class label of its nearest neighbor.

Using Definition 3.1 and a suitable similarity measure, we may obtain a set, $E \subset I$, of all neutral instances in I. It is now possible to generate two new data sets by regarding E either as noise or by introducing a third class (neutral) to more appropriately label the instances of E.

3.2 Removal of noisy instances

An important question to ask is: why the detection and removal of *noisy* document examples should be performed to begin with. It is obvious that in any non-trivial supervised classification learning task, some classes are not easily separable. Thus, removing examples to increase the separation of classes in the instance space may be seen as over-simplifying the studied problem. To answer the stated question, and address the related issues, we would like to draw the reader's attention to the discussion in Sect. 2 about the collection and labeling of end user license agreements; since there is no consensus about the definition of spyware, it is impossible to correctly label a collection of EULAs, in the sense that everyone will agree upon the correctness of the associated class labels. Thus, the objective is not to generate a *perfectly* labeled collection of EULAs. Rather, we are interested in developing a tool that can be used to alert users of the possible inclusion of spyware during the installation of downloaded applications. The tool needs to be robust in its classification of the clear cases of legitimate software and spyware-associated software. Hence, we consider it better if the borderline or noisy cases are labeled as belonging to a third class; *neutral*, or not labeled at all, as opposed to being incorrectly labeled as good or bad (were the former case is the most problematic).

3.3 Removal of noisy attributes

The EULA classification task does not suffer only from the possible inclusion of noisy instances in the data set: it is also quite common that the dimensionality of the data with regard to the number of attributes is large in comparison to the number of EULA documents, or instances. A common approach to address this issue for the general problem of text categorization is to apply a feature selection method. The potential benefits of feature selection are that it may:

1. reduce computational cost and storage requirements,
2. deal with the degradation of classification efficiency due to the finite size of training sample sets,
3. reduce training and prediction time, and
4. facilitate data understanding and visualization [1], as cited by Lin et al. [30].

3.3.1 Statistical and algorithmic feature selection

Consequently, it may be worth investigating whether feature selection can be used to decrease attribute dimensionality and increase classification performance for the EULA classification task. There is a large number of feature selection algorithms available. Generally, one could make a distinction between statistical and algorithmic feature selection approaches. Whereas the former type makes use of statistics or measures calculated directly on the data, the latter type may involve the application of a learning algorithm to generate a classifier, from which information about the quality of different features can be obtained. In general, the algorithmic approach is much more computationally expensive than the statistical approach. (There are also hybrid approaches available, for example, based on determination of an optimal feature set by means of running an appropriate optimization technique [30]).

3.3.2 Categorical proportional difference

A recent statistical feature selection technique, categorical proportional difference (CPD), has been shown to outperform commonly applied feature selection methods, such as χ^2, information gain, document frequency, mutual information, odds ratio, and simplified χ^2, on several text categorization corpora in terms of F-measure performance improvement [36]. Moreover, CPD has also managed to significantly reduce the number of attributes in the aforementioned cases of performance improvement. Thus, we argue that CPD is a suitable choice of feature selection technique for the EULA classification problem.

Intuitively, CPD is a measure of the degree to which a word contributes to differentiating a particular class from other classes. The possible values that CPD can take on are limited to the interval of -1 and 1, where CPD values close to -1 indicate that a word occurs in approximately an equal number of EULAs in all classes and a CPD value of 1 indicates that a word occurs in the EULAs of only one class. Let A be the number of times word w and class c occur together, and let B be the number of times word w occurs without class c. According to Simeona and Hilderman, we may then define CPD for a particular word, w, and class, c, as follows:

$$\mathrm{CPD}(w, c) = \frac{A - B}{A + B}. \tag{2}$$

This is the ratio of the difference between the number of EULAs of a class in which a word occurs and the number of EULAs of other classes in which the word also occurs, divided by the total number of EULAs in which the word occurs. The CPD for a word is the ratio associated with the category c_i for which the ratio is the highest, i.e.:

$$\mathrm{CPD}(w) = \max_i \{\mathrm{CPD}(w, c_i)\}. \tag{3}$$

4 Similarity analysis methods

We will now present two different similarity-based analysis approaches for reducing the number of false positives for the EULA classification problem by removing noisy instances.

4.1 Latent semantic analysis

A document-term matrix, M, is constructed from a given text document collection of n documents containing m terms. A feature extraction component is used to extract the values

v_{ij} for each term, m_i, from each document, j. The m-dimensional column vector, V_j, with the components, $v_{ij}; i = 1, 2, \ldots, m$, then represents document j in the so-called Vector Space Model. The $m \times n$ document-term matrix, M, is composed using the vectors V_j for all documents as columns [17].

This matrix, M, is then decomposed via *singular-value decomposition* (SVD) into: the term vector matrix, T, the document vector matrix, D, and the diagonal matrix, S, according to Eq. 4.

$$M = TSD^T. \tag{4}$$

These matrices can then be reduced to a given number of dimensions, k, resulting in the truncated matrices T_k, S_k and D_k: the Latent Semantic Space.[2] If these matrices are multiplied, the product is a matrix, M_k, which is the least-squares best fit approximation of M with k singular values (see Eq. 5).

$$M_k = \sum_{i=1}^{k} t_i \cdot s_i \cdot d_i^T. \tag{5}$$

If a similarity measure is used, it is now possible to determine the similarity between different documents represented in M_k or between an external document and the documents of M_k. One measure, that is commonly used together with Latent Semantic Analysis, is cosine similarity (CS) [28]. Given a query vector q of length f, the distances to all documents represented in M_k can be measured in terms of the cosines of the angles between q and the columns of M_k. The closest column represents the closest match between the document collection and the query vector [17]. Given the query vector, q, and an arbitrary document vector in the collection, p, the cosine similarity is defined using the dot product as shown in Eq. 6.

$$\cos(\theta) = \frac{q \cdot p}{\|q\| \, \|p\|}. \tag{6}$$

For our particular task, it is useful to construct a cosine similarity matrix (CSM), as shown in Eq. 7. The CSM includes the similarity scores, θ, or distances between all EULA documents, $N = \{1 \ldots n\}$ (good and bad). Naturally, the cosine similarity between a certain document and itself is 1.

$$\text{CSM} = \begin{pmatrix} \theta_{11} & \theta_{12} & \ldots & \theta_{1n} \\ \theta_{21} & \theta_{22} & \ldots & \theta_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \theta_{n1} & \theta_{m2} & \ldots & \theta_{nn} \end{pmatrix}. \tag{7}$$

4.2 Normalized compression distance

An alternative approach to that of latent semantic analysis and cosine similarity is a fairly recent algorithm that computes distances between arbitrary data vectors: normalized compression distance (NCD) [8] as it is generally applicable [15], parameter free [22], noise resistant [6] and demonstrated theoretically optimal [38]. The NCD is an approximation to the uncomputable normalized information distance, which is based on the notion of Kolmogorov complexity.

[2] Documentation of the LSA package for the R-project, http://cran.r-project.org/web/packages/lsa/lsa.pdf.

Since the NCD is a distance function, it must satisfy (verbatim from [38]):

- $D(x, y) = 0$ if and only if $x = y$, (identity)
- $D(x, y) = D(y, x)$ (symmetry)
- $D(x, y) \leq D(x, z) + D(z, y)$ (triangle inequality)

In order to work well, there are also requirements on distance densities for the population, i.e., only a certain number of distances of pairs of input may be the same, not all of them. This is to avoid undesirable distances functions such as $D(x, y) = 1$ when $x \neq y$, and 0 when $x = y$. Although that function satisfies the requirements above, it is clearly useless [38].

NCD is based on the idea that by using a compression algorithm on data vectors (in whatever shape or form these may come) both individually and concatenated, we will receive a measure of how distant they are. The better the combination of the two vectors compress, compared to how the individual vectors compress on their own (normalized to remove differences in length between the set of all vectors), the more similar the vectors are. More formally, NCD is a metric:

$$\mathrm{NCD}(x, y) = \frac{C(x, y) - \min(C(x), C(y))}{\max(C(x), C(y))}$$

where $C(x)$ is the compressed length of x and $C(x, y)$ is the compressed length of x concatenated with y. The range of the NCD function is ideally within $0 \leq NCD \leq 1$. This always holds true for the lower bound, but due to failure of the chosen compression algorithm, $C(x)$, to accurately approximate the incomputable Kolmogorov complexity function the upper bound is sometimes not met, and the NCD will be greater than one. This does not materially affect operation as long as the distance does not greatly exceed one, as that signifies major problems with the compression algorithm for use as part of the NCD. At least for the particular data set.

The reason for choosing $\max(C(x), C(y))$ for the denominator is not obvious. In fact, most of the alternatives that come to mind will not work as the NCD then does not satisfy the distance function requirements. The interested reader is referred to Vitanyi et al. [38] for further details. That the resulting function is indeed a distance metric is far from self evident, is demonstrated by Li et al. [29].

In addition to the advantages listed above, the NCD has several additional advantages, a significant of which is the lack of parameters that need to be set. Parameter-free methods are advantageous in that setting the correct parameters requires detailed knowledge of the internal operation of the algorithm. Setting the wrong parameters can have a sizable impact on the performance of the algorithm, both when it comes to false positives and false negatives. It is of course also problematic to compare the same algorithm on differing problem sets if the parameters are also different. How to set the correct parameters in a real-world usage scenario, when labels are absent from the data set is also fraught with peril [38]. Many of the same problems also abound when it comes to feature selection, selecting the right features can involve intimate knowledge of the problem domain, and in the instances when it does not, it can still be a time consuming and error-prone process [38].

The only parameter one has to set when performing NCD is to choose a compression algorithm. The best performing algorithm overall is most often the statistical compressors in the *PPM*-family [41]. However, in this experiment, we have chosen to use the *bzip2* [35] algorithm. This algorithm was selected since it has previously been demonstrated to work well on the type of text classification problem that we are faced with here [7] and the runtime is much shorter than for the *PPMD*. With a window length of as much as 900 KB, we are comfortably within the length of the concatenation of our two largest EULAs (105 KB),

which is in the range where bzip2 works well. (It is, however, too large for the *gzip* algorithm as it can only comfortably handle concatenated document sizes of approximately 60 KB).

4.3 Detection of noisy instances

The LSA and NCD methods can both be used to generate similarity scores for all pairs of EULA documents. In order to actually detect noisy instances, we need to choose an appropriate detector. The 1-nearest neighbor algorithm is one of the most commonly used methods for this purpose although it would be quite possible to make use of more sophisticated methods as well. The K-nearest neighbor algorithm could, for example, be used instead (refer to the discussion on using NCD as a learner in the previous section). However, for the purpose of the present study, no alternative methods to 1-nearest neighbor will be evaluated.

4.3.1 LSA detection

Based on the cosine similarity matrix (CSM), as defined in Eq. 7, we obtain some interesting results. Primarily, we are interested in finding out which documents from a certain class are most likely to be confused with documents from the other class. That is, we want to find the set of documents that are closest (in the CSM-sense), to a document of the opposite class. We therefore order the column of similarity scores from highest to lowest for each document and compare the class of each document with the class of its closest neighbor.

Note that since the text documents are tokenized into word elements prior to LSA analysis, and since LSA results in a truncated vector space model, the nearest neighbor to a certain document is only approximately identical to that document, even if the cosine similarity is 1. We say that if the nearest neighbor to a document belongs to another class than that of the document itself, the document is classified as incorrectly labeled by the LSA-based analysis. Consequently, we hypothesize that the document may contribute to a higher false positive rate were it included during classifier generation. For an arbitrary document, d, and its nearest neighbor, d^*, we define $c(d)$ and $c(d^*)$ as their corresponding classes. We may then define label correctness with Eq. 8.

$$\text{label}(d) = \begin{cases} \text{incorrect} : c(d) \neq c(d^*) \\ \text{correct} : \text{otherwise} \end{cases} \tag{8}$$

Secondarily, we may use the information provided in the CSM to find duplicate documents. Again, we will only find approximately duplicate documents with our approach, but the information about these documents may prove to be valuable in further research on EULA classification. We define the set of duplicates of a document, n, as the documents for which the cosine similarity scores to n are 1.

4.3.2 NCD detection

Analogously to LSA-based detection, the NCD-based detection is carried out by finding the nearest neighbor. For each EULA document, d, a vector of all available EULA documents is generated, ordered by the NCD-based similarity to the document in question. The EULA with the highest similarity score from this vector, d^*, is then the nearest neighbor. Consequently, Eq. 8 can once again be used to determine the correctness of the labeling of d according to the NCD. Similarly, the set of EULAs for which the NCD-based similarity is above a certain threshold could be regarded as duplicates according to the NCD.

Table 1 LSA-based analysis of the original data

Class	#Instances	#Duplicates	#Correct	#Incorrect
Good	900	158	869	31
Bad	96	8	52	44
Total	996	166	921	75

Table 2 NCD-based analysis of the original data

Class	#Instances	#Duplicates	#Correct	#Incorrect
Good	900	109	873	27
Bad	96	0	64	32
Total	996	109	937	59

5 Experiments

5.1 Detection of noisy instances

Table 1 includes the results from the LSA-based analysis of the original 996 raw EULA text documents. As can be seen in the fifth column, there is a rather large amount of misclassified instances, especially from the bad class. Recall that, duplicates and incorrectly classified instances actually denote instances that are determined by the cosine similarity measure to be either equal to another instance or most similar to an instance of another class. Table 2 features the results from the NCD-based analysis of the original EULAs. Similarly to the results of the LSA-based analysis, the fifth column shows a quite substantial amount of misclassified instances from the bad class although the amount of incorrectly labeled instances according to NCD is smaller than what was detected by LSA. The information about duplicate instances is not used in the presented study. That is, the duplicate EULAs are not removed from the collection. Recall that, duplicates and incorrectly classified instances actually denote instances that are determined by either the cosine similarity measure or the normalized compression distance measure to be either equal to another instance, or more similar to an instance of the other class, than to any instances of their own class.

5.2 Comparison of noise detection approaches

In the primary experiment, the aim is to determine whether the removal of instances, labeled as incorrectly classified cases by any of the similarity-based instance removal approaches, yields an increase in classification performance. Instances from either class, determined by LSA and NCD to be closer to an instance of the other class, are removed from the corresponding data set before transforming it to a bag-of-words model. Thus, to generate the LSA-based data set, 75 instances are removed from the original data set (31 good, 44 bad), resulting in a new data set that features 921 instances. Similarly, to generate the NCD-based data set, 59 instances are removed from the original data set (27 good, 32 bad). The NCD-based data set features 937 instances in total.

5.2.1 Bag-of-words generation

The total number of words, contained in all of the EULAs, is close to 10,000 but experimental evidence suggests that this number can be reduced by circa 90% without reducing classifier

performance [14]. In order to decrease the number of attributes, we convert all characters to lowercase and consider only alphabetic tokens. Furthermore, we remove stop words and words that occur only once (*hapax legomena*) and store a maximum of 1,000 words per class. Finally, we apply the Iterated Lovins Stemmer [31] to be able to store only the stems of the remaining words. The result is an original data set with 1,269 numeric attributes, an LSA-based data set with 1,295 attributes, and an NCD-based data set with 1,302 attributes. All data sets include a nominal target attribute, which indicates the class of the EULA (good or bad).

5.2.2 Feature selection

Prior to the experimental comparison of noise detection approaches, the three data sets under investigation are subjected to feature selection using the categorical proportional difference (CPD) technique. As described by Simeon and Hilderman [36], the search for a suitable cutoff point for CPD is computationally expensive due to the possible non-linearity of the function of the number of kept words and the resulting performance. We opted to define a keep ratio interval and select a reasonable step size. In the presented study, we use an interval of 1.0 to 0.6 together with a step size of 0.01, which yields 40 iterations for each data set. These settings let us investigate, in increments of one percent, the possible performance gain, from using CPD-based feature selection, for each data set when keeping from 60% and up to 100% of the attributes.

Since Naive Bayes Multinomial [32] achieved the top AUC score in the previous study [27], it is argued that this algorithm could serve as a reasonable estimator of general performance for the purpose of feature selection. Thus, the Multinomial Naive Bayes algorithm is evaluated according to area under the ROC curve (AUC) performance by running 10 ten-fold cross-validation tests for each iteration and data set. For each of the three data sets, which attributes to keep is determined by selecting the subset of attributes from the iteration that yields the highest AUC performance by the multinomial Bayesian algorithm.

5.2.3 Evaluation and hypothesis testing

In the primary experiment, all algorithms investigated during the previous study [27] except K-star are compared in terms of the area under the ROC Curve (AUC) performance on the feature selected versions of the original data set, the LSA-based data set, and the NCD-based data set. A decision was taken to remove K-star since it achieved the poorest AUC performance in the previous study and, moreover, it was by far the most computationally expensive algorithm.

In order to obtain reliable results, the comparison is performed by subjecting each of the included algorithms to ten runs of re-sampled and stratified tenfold cross-validations. The cross-validation folds are generated by using an initial seed of 1 for each algorithm. Thus, given a particular data set, each algorithm is accessing identical folds. In essence, this setup also increases the reproducibility of the results. For the selected set of algorithms and data set, we aim to test the following null hypothesis, h_0: there is no AUC performance difference between the original data set, the LSA-based data set, and the NCD-based data set. In order to investigate the validity of this hypothesis, the non-parametric Friedman's test [16] and the corresponding Nemenyi post hoc test [33] are used since the hypothesis testing involves comparisons of multiple algorithms and data sets [11].

Friedman's test is based on calculating the average ranks of each data set rather than using the actual AUC performance score. For each algorithm, the data set for which the algorithm

achieves the highest performance is awarded the rank of 1, the second best the rank of 2, and so on. In the case of a tie, the average rank of the tied data sets is assigned. We can now state the null hypothesis more formally. Given that R_j represents the average rank of the j-th out of k data sets, we wish to test $H_0 : R_1 = \cdots = R_k$. The regular statistic used for Friedman's test, χ_F^2, is defined, with $k - 1$ degrees of freedom, as follows:

$$\chi_F^2 = \frac{12}{Nk(k+1)} - \sum_j (NR_j)^2 3N(k+1). \tag{9}$$

Since it has been shown that Friedman's statistic is too conservative [21], we employ the recommended alternative statistic based on the F-distribution, as defined in Eq. 10 where N is the number of included algorithms and k is the number of data sets to compare. Moreover, we perform hypothesis testing at $p < 0.05$ and with $k - 1$ and $(k - 1)(N - 1)$ degrees of freedom.

$$F_F = \frac{(N-1)\,\chi_F^2}{N\,(k-1) - \chi_F^2}. \tag{10}$$

If the null hypothesis is rejected, the Nemenyi *post hoc* test can be applied to determine whether the performance achieved on two particular data sets is significantly different. In order for this difference to be significant, the corresponding average ranks of the two data sets must differ by at least the critical difference (CD), which is defined in Eq. 11, where q_α is the critical value for the two-tailed Nemenyi *post hoc* test at $p < \alpha$. For $k = 3$ and $p < 0.05$, the critical value, q_α, is 2.343. Thus, the critical difference for $p < 0.05$ is $2.343\sqrt{1.125} \approx 0.828$, given that $k = 3$ and $N = 16$.

$$\mathrm{CD} = q_\alpha \sqrt{\frac{k\,(k+1)}{6N}}. \tag{11}$$

As in previous work, the Weka [40] default configurations are applied for a majority of the algorithms since it is not the main objective of the study to find the optimal algorithm. The exceptions are IBk and Stacking. For convenience, a comprehensive description of the configurations of the included algorithms can be viewed in Table 3.

5.3 Introduction of the neutral class

In the secondary experiment, a third class is introduced: the instances determined to be incorrectly labeled on the basis of either the LSA or the NCD analysis are now categorized as belonging to the *neutral* class instead of being removed from the data set. The procedure to transform the EULA document collection to a bag-of-words model is identical to the procedure followed in the primary experiment. Thus, the data sets generated for the secondary experiment both feature 996 instances. The LSA-based three-class data set features 869 good, 75 neutral, and 52 bad instances and the NCD-based three-class data set features 873 good, 59 neutral, and 64 bad instances.

Since this data set includes three classes instead of two, as in the original data set, it makes little sense to formally compare the performance of learning algorithms across the different types of data sets. Instead, our aims are to investigate whether the performance in classifying bad and good instances deteriorate when introducing the neutral class and to examine the classification confusion between the good, bad, and neutral classes. The experiment is conducted by subjecting the top performing algorithm from the primary experiment to a stratified tenfold cross-validation test.

Table 3 Learning algorithm configurations

Algorithm	Configuration
NaiveBayes	Kernel estimator: false, supervised discretization: false
NaiveBayesMultinomial	N/A
RBFNetwork[a]	Ridge: 1.0E-8
SMO[b]	Kernel: polynomial, complexity: 1.0
VotedPerceptron	Exponent: 1.0, max kernel alterations: 10,000
IBk[c]	Number of neighbors: 10, distance weighting: false
AdaBoostM1	Classifier: DecisionStump
Bagging	Classifier: REPTree (Pruning: true)
Stacking	Meta: SMO, committee: SMO, VotedPerceptron, NaiveBayesMultinomial
HyperPipes	N/A
JRip[d]	Pruning: true, number of optimizations: 2
PART	Binary splits: false, pruning: true (confidence factor: 0.25)
Ridor	N/A
DecisionStump	N/A
J48[e]	Pruning: subtree raising, pruning confidence factor: 0.25
RandomForest	Number of trees: 10

This table is from the previous EULA study [27]
[a] Radial basis function network
[b] Support vector machines
[c] K-nearest neighbor
[d] Ripper
[e] C4.5

6 Results

We now review and discuss the results of the experiments. First, the results from the CPD-based feature selection are revealed. The results from the main experiment are then described in detail. As earlier described, in the main experiment, the performances on the original data set, the LSA-based data set, and the NCD-based data set are compared. Finally, we present results from and discuss the possibility of adding a third, neutral, class to the EULA classification task by including the instances previously identified as noise. Prior to the present study, a domain expert manually examined a subset of the bad EULAs (all instances with a smaller size than 10 KB). It was concluded that for 18 of the 36 examined EULAs, it would be impossible for a human to infer that spyware would be installed by the software product. Interestingly, the LSA-based approach managed to remove 13 of these 18 bad instances (72%) and the NCD-based approach removed over 80% of these bad instances, stemming from the fact that they were more similar to good EULAs. Thus, the initially stated informal hypothesis seems to be true, which arguably increases the meaningfulness of using an automated approach to remove noisy instances.

6.1 Feature selection results

The complete results of the feature selection performed on the two-class data sets are presented in Table 4. As can be seen in the Table 5, the AUC performance gains for the original data set, the LSA-based data set, and the NCD-based data set are: 1.2, 0.9, and 0.6%,

Table 4 Post-feature selection AUC results for Naive Bayes Multinomial on 40 levels of keep ratio

Keep ratio	Original mean (SD)	LSA-based mean (SD)	NCD-based mean (SD)
0.61	**0.934 (0.054)**	**0.998 (0.004)**	**0.994 (0.006)**
0.62	0.933 (0.054)	0.998 (0.004)	0.994 (0.006)
0.63	0.933 (0.054)	0.998 (0.004)	0.994 (0.006)
0.64	0.933 (0.054)	0.998 (0.004)	0.994 (0.006)
0.65	0.932 (0.054)	0.998 (0.004)	0.994 (0.006)
0.66	0.931 (0.054)	0.998 (0.004)	0.994 (0.006)
0.67	0.931 (0.054)	0.998 (0.004)	0.993 (0.006)
0.68	0.931 (0.054)	0.998 (0.004)	0.993 (0.006)
0.69	0.931 (0.054)	0.998 (0.004)	0.993 (0.006)
0.70	0.931 (0.054)	0.998 (0.004)	0.993 (0.006)
0.71	0.931 (0.054)	0.998 (0.004)	0.993 (0.006)
0.72	0.931 (0.054)	0.997 (0.004)	0.993 (0.006)
0.73	0.929 (0.054)	0.997 (0.004)	0.993 (0.005)
0.74	0.929 (0.054)	0.997 (0.004)	0.993 (0.006)
0.75	0.929 (0.054)	0.997 (0.004)	0.993 (0.005)
0.76	0.929 (0.054)	0.997 (0.004)	0.993 (0.005)
0.77	0.929 (0.054)	0.996 (0.005)	0.993 (0.005)
0.78	0.928 (0.055)	0.996 (0.005)	0.993 (0.005)
0.79	0.928 (0.055)	0.996 (0.005)	0.993 (0.006)
0.80	0.929 (0.055)	0.996 (0.005)	0.993 (0.005)
0.81	0.929 (0.055)	0.995 (0.005)	0.993 (0.006)
0.82	0.929 (0.055)	0.995 (0.006)	0.993 (0.006)
0.83	0.928 (0.055)	0.995 (0.006)	0.992 (0.006)
0.84	0.928 (0.056)	0.995 (0.006)	0.992 (0.006)
0.85	0.928 (0.056)	0.995 (0.006)	0.992 (0.006)
0.86	0.928 (0.055)	0.994 (0.006)	0.992 (0.006)
0.87	0.928 (0.056)	0.994 (0.006)	0.992 (0.006)
0.88	0.927 (0.056)	0.993 (0.006)	0.992 (0.007)
0.89	0.927 (0.056)	0.993 (0.007)	0.992 (0.007)
0.90	0.928 (0.055)	0.992 (0.007)	0.991 (0.007)
0.91	0.927 (0.056)	0.992 (0.007)	0.991 (0.007)
0.92	0.927 (0.056)	0.992 (0.007)	0.990 (0.007)
0.93	0.926 (0.056)	0.991 (0.007)	0.989 (0.008)
0.94	0.926 (0.057)	0.991 (0.007)	0.989 (0.008)
0.95	0.925 (0.057)	0.991 (0.007)	0.988 (0.008)
0.96	0.925 (0.057)	0.991 (0.007)	0.988 (0.008)
0.97	0.923 (0.057)	0.990 (0.007)	0.988 (0.008)
0.98	0.923 (0.057)	0.990 (0.008)	0.988 (0.009)
0.99	0.923 (0.058)	0.990 (0.008)	0.988 (0.008)
1.00	0.922 (0.058)	0.989 (0.008)	0.988 (0.008)

Optimal AUC performance is indicated with bold

Table 5 Data set statistics post- and pre-feature selection

Data set	#Instances	Pre	Post	
		#Attributes	#Attributes	AUC gain
Original	996	1,269	775	0.012
LSA-based	922	1,295	791	0.009
NCD-based	937	1,302	795	0.006

respectively. Table 5 also shows that the number of attributes has decreased to 775, 791, and 795 for the three data sets.

We conclude that CPD is successful in substantially decreasing the number of attributes for each data set while at the same time managing to increase the AUC performance reasonably well. Intuitively, the largest performance gain is attributed to the feature selection on the original data set, since the potential for improvement was the highest on this data set, due to the comparably low initial performance.

For the three-class problem, feature selection was again performed using CPD and 40 iterations with Multinomial Naive Bayes as a general performance estimator. This feature selection process resulted in a substantial decrease of attributes for both the LSA- and NCD-based three-class data sets. The best AUC performance was found with a keep ratio of 0.61 and 0.65 for the NCD and LSA data sets, respectively. In other words, the original three-class data sets included 1440 attributes, and after feature selection, this number was reduced to 879 for the NCD data set and to 937 for the LSA data set since 61% of the original 1440 attributes were kept when generating the NCD data set and 65% of the original attributes were kept when generating the LSA set.

6.2 Data set Comparison Results

The results from the main experiment, in which we compare the original EULA data set with the LSA-based data set and the NCD-based data set, are summarized in Table 6. Clearly, the AUC performances of all included algorithms are substantially higher when their classifiers are generated from the LSA-based data set or the NCD-based data set, in which noisy examples have been removed. With 3 data sets and 16 algorithms, F_F is distributed according to the F-distribution with $3-1=2$ and $(3-1)(16-1)=30$ degrees of freedom. The critical value of $F(2, 30)$ for $\alpha = 0.05$ is 3.32, so we may reject the null hypothesis (h_0). In fact, it is even possible to reject the null hypothesis with the χ^2 statistic and $3-1=2$ degrees of freedom since the critical value of $\chi^2(2)$ for $\alpha = 0.05$ is 5.991 and $\chi^2 = 26$.

Since a significant average rank difference has been detected, we proceed with the Nemenyi *post hoc* test to determine individual significant differences between all possible pairs of data sets. As described in Sect. 5.2.3, the critical difference for $p < 0.05$, 3 data sets, and 16 algorithms is 0.828. The difference in average rank between two data sets must therefore be larger than 0.828 in order for the difference to be significant according to the Nemenyi test. Since $3 - 1.75 = 1.25$ and $3 - 1.25 = 1.75$, we conclude that both the LSA-based and NCD-based data sets are significantly better than the original data set. However, since $1.75 - 1.25 = 0.5$, the conclusion is that there is no significant difference, in terms of average AUC performance of the evaluated algorithms, between the LSA-based and the NCD-based approach.

Similarly to the previous study, the Multinomial Naive Bayes algorithm is superior in terms of AUC performance. In fact, this algorithm achieves the highest AUC performance

Table 6 Comparison of AUC between the original data set and the two similarity-based data sets

Algorithm	Type	Original mean (SD)	LSA-based mean (SD)	NCD-based mean (SD)
NaiveBayes	Bayes	0.898 (0.055)	0.990 (0.019)	0.977 (0.037)
NaiveBayesMultinomial		0.934 (0.054)	0.998 (0.004)	0.994 (0.006)
RBFNetwork	Function	0.821 (0.096)	0.982 (0.038)	0.980 (0.025)
VotedPerceptron		0.829 (0.080)	0.893 (0.091)	0.904 (0.084)
SMO		0.814 (0.080)	0.926 (0.086)	0.948 (0.061)
IBk	Lazy	0.826 (0.086)	0.966 (0.074)	0.990 (0.018)
AdaBoostM1	Meta	0.864 (0.079)	0.966 (0.046)	0.969 (0.032)
Bagging		0.889 (0.067)	0.914 (0.075)	0.954 (0.071)
Stacking		0.847 (0.084)	0.921 (0.088)	0.948 (0.061)
HyperPipes	Misc	0.811 (0.090)	0.978 (0.018)	0.983 (0.016)
JRip	Rules	0.805 (0.083)	0.845 (0.104)	0.886 (0.089)
PART		0.796 (0.097)	0.872 (0.145)	0.852 (0.136)
Ridor		0.739 (0.089)	0.832 (0.105)	0.860 (0.103)
DecisionStump	Trees	0.814 (0.069)	0.901 (0.076)	0.902 (0.085)
J48		0.737 (0.115)	0.853 (0.138)	0.859 (0.125)
RandomForest		0.884 (0.069)	0.962 (0.062)	0.982 (0.030)
Average rank		3.00	1.75	1.25

Table 7 Comparison of LSA and NCD on the three-class problem

Confusion matrices generated from one tenfold cross-validation test of Multinomial Naive Bayes

Dataset	Classifications			Actual
	Bad	Good	Neutral	class
LSA	44	0	8	Bad
	10	706	154	Good
	9	24	41	Neutral
NCD	56	0	8	Bad
	12	714	147	Good
	13	23	23	Neutral

on all three data sets. The overall algorithm performance results indicate that both the LSA-based and the NCD-based approach represent feasible ways to reduce the false positive rate for the EULA classification task.

6.3 Introduction of the Neutral class

The results of the secondary experiment can be viewed in Table 7 and consist of two confusion matrices generated for the Multinomial Naive Bayes algorithm on the three-class versions of the LSA-based and NCD-based data sets. Interestingly, no bad instances are classified as good, which would be the worst type of classification error in a real-world setting. The neutral instances are classified slightly more correctly in the LSA-based data set. A large amount of the good instances are classified as neutral in both the data sets.

LSA and NCD perform comparably in generating suitable three-class data sets. However, as can be interpreted from the AUC performance, the introduction of a neutral class affects

Table 8 The ten wordstems with the highest conditional probability for each class and data set

Data set	Bad	Neutral	Good
NCD	search	toolbar	evalu
	toolbar	search	register
	arbitr	act	program
	whenu	effect	el
	view	jurisdict	ver
	cook	commerc	tr
	url	el	sharewar
	anonym	sol	fil
	18	direct	pack
	avail	respon	author
LSA	search	search	evalu
	toolbar	serv	register
	whenu	toolbar	el
	view	act	program
	cook	download	ver
	download	oper	tr
	anonym	effect	sharewar
	serv	direct	pack
	acc	commerc	fil
	upd	control	author

Word stem probabilities generated by Multinomial Naive Bayes

the performance negatively. Arguably, the conclusion that can be drawn is that the neutral instances are confusingly similar to either good or bad instances. Further research is required to determine whether completely different categorizations of spyware may resolve the issue. For example, one might classify the spyware applications according to functionalities like: *browser hijacking*, *advertisement pop-ups*, *user behavior reporting*, and so on. However, it is quite obvious that this type of functionality-based labeling of instances would most certainly be time consuming and difficult to conduct manually.

6.4 Discussion

As in many other domains, false positives are particularly troublesome in the EULA classification task, since the principal objective of this classification is to provide decision support to computer users during software product installation. A false positive (a spyware-associated application classified as legitimate software) would then mislead the user to install the product, which could possibly lead to system performance degradation or worse; loss of system control and theft of information. A false negative (a legitimate application classified as spyware) is not as potentially costly. However, if the user knows about the legitimacy of the application, a false negative may result in a decrease of trust in the decision support tool. The decreased trust would, in the worst scenario, leads the user to not pay attention to the recommendation from the decision support tool, much like most users today do not pay attention to the displayed EULA when installing software downloaded from the Internet.

Table 8 includes the word stems with the highest conditional probabilities for each of the three classes from the secondary experiment. In analyzing the conditional probability-based prioritized lists of word stems, some evidence is uncovered as to why bad and neutral instances are indeed hard to separate, which confirms the conclusions drawn from the manual EULA

examination. Additionally, it is also obvious that, despite their demonstrated differences, the LSA-based and NCD-based approaches yield data sets in which the lists of the most representative words for each class are quite correlated.

We have not been able to find any additional public domain collection of EULAs, and the generation of this type of collection is a labor intensive task since each EULA has to be extracted during the actual installation of a software product. Since research on the definition of spyware is still on-going, it is often also difficult to correctly label the collected instances. This fact, together with the conclusions drawn on the basis of the aforementioned results of the manual examination, motivates the need for an efficient method for detecting noisy examples. It is intuitively not a good idea to remove instances from a data set based on the mere fact that they were misclassified by one or more algorithms. In contrast, both the manual examination and the experimental evidence of the present study suggest that the noisy examples should be removed since they are not at all semantically separable from instances from the other class.

7 Conclusions and future work

End user license agreement (EULA)-based spyware detection is essentially a text categorization task. Previous work by us has reported good classification performance results of supervised learning algorithms, such as Multinomial Naive Bayes and Random Forests, on a collection of 96 spyware-related EULAs and 900 legitimate EULAs. However, the false positive rate (the rate of spyware EULAs classified as legitimate EULAs) has been reported as too high for practical use.

This study examines the use of feature selection and compares two candidate approaches for removal of noisy instances to decrease the false positive rate. The first approach is based on using latent semantic analysis (LSA) in conjunction with cosine similarity to find noisy instances in the EULA collection under study. The second approach is based on normalized compression distance (NCD). After performing the feature selection, we experimentally evaluated the performance of 16 supervised learning algorithms on the original EULA collection studied in previous work and the LSA-based and NCD-based data sets. These were generated by removing noisy examples using the aforementioned methods, to investigate whether it is possible to reduce the false positive rate.

The area under the ROC curve (AUC) performances are significantly higher on the LSA-based and NCD-based data sets in comparison to the AUC scores on the original EULA collection. However, no significant difference in AUC performance could be revealed between the LSA-based and NCD-based approaches. Thus, we conclude that LSA and NCD both represent suitable approaches for finding noisy instances whose removal reduce the number of false positives during classification.

The categorical proportional difference (CPD) feature selection method was applied on the three studied data sets prior to experimental comparison. CPD managed to substantially decrease the number of instances while at the same time increasing AUC performance on all data sets. Moreover, the experimental evidence confirms the results of previous work in that the Multinomial Naive Bayes algorithm seems to be the most suitable algorithm for the EULA classification task.

Future work is focused on compiling a larger database of EULAs in collaboration with the anti-spyware industry. Moreover, we are in the process of developing a spyware prevention tool, based on Multinomial Naive Bayes, that will be able to automatically extract and analyze EULAs in order to provide users with a means to give an informed consent when installing

applications downloaded from the Internet. The availability of the prevention tool will also allow for more human-oriented studies on spyware awareness and informed consent. We aim to compare additional feature selection methods and learning algorithms in the subsequent experiments to be carried out on new data.

References

1. Abe N, Kudo M (2006) Non-parametric classifier-independent feature selection. Pattern Recogn 39: 737–746
2. Axelsson S (2000) The base-rate fallacy and the difficulty of intrusion detection. ACM Trans Inf Syst Sec 3(3):186–205
3. Axelsson S, Baca D, Feldt R, Sidlauskas D, Kacan D (2009) Detecting defects with an interactive code review tool based on visualisation and machine learning. In: 21st international conference on software engineering and knowledge engineering, Boston, USA
4. Berry MW, Dumais ST, O'Brien GW (1995) Using linear algebra for intelligent information retrieval. SIAM Rev 37(4):573–595
5. Boldt M, Carlsson B, Jacobsson A (2004) Exploring spyware effects. In: Eight nordic workshop on secure IT systems, pp 23–30
6. Cebrian M, Alfonseca M, Ortega A (2007) The normalized compression distance is resistant to noise. IEEE Trans Inf Theory 53(5):1895–1900
7. Cebrian M, Alfonseca M, Ortega A (2005) Common pitfalls using normalized compression distance: what to watch out for in a compressor. Commun Inf Syst 5(4):367–400
8. Cilibrasi R (2007) Statistical inference through data compression. PhD thesis, Institute for Logic, Language and Computation Universiteit van Amsterdam, Plantage Muidergracht 24, 1018 TV Amsterdam. http://www.illc.uva.nl/
9. Deerwester S, Dumais S, Furnas G, Landauer T, Harshman R (1990) Indexing by latent semantic analysis. J Am Soc Inf Sci 41(6):391–407
10. Delany SJ (2009) The Good, the bad and the incorrectly classified: profiling cases for case-base editing. In: 8th international conference on case-based reasoning, pp 135–149
11. Demsar J (2006) Statistical comparisons of classifiers over multiple data sets. J Mach Learn Res 7:1–30
12. Dong Z (2002) Towards web information clustering. PhD thesis, Southeast University, Nanjing, China
13. Edsberg O, Nytro O, Rost TB (2007) Novelty detection in patient histories: experiments with measures based on text compression. In: Berthold MR, Shawe-Taylor J, Lavrac N (eds) Advances in intelligent data analysis VII. Springer, New York, pp 367–378
14. Feldman R, Sanger J (2007) The text mining handbook. Cambridge University Press, Cambridge
15. Ferragina P, Giancarlo R, Greco V, Manzini G, Valiente G (2007) Compression-based classification of biological sequences and structures via the universal similarity metric: experimental assessment. BMC Bioinf 8(1)
16. Friedman M (1940) A comparison of alternative tests of significance for the problem of *m* rankings. Ann Math Stat 11:86–92
17. Gansterer WN, Janecek AGK, Neumayer R (2007) Spam filtering based on latent semantic indexing. In: Berry MW, Castellanos M (eds) Survey of Text Mining II. Springer, New York
18. Good N, Grossklags J, Thaw D, Perzanowski A, Mulligan DK, Konstan J (2006) User choices and regret: understanding users' decision process about consensually acquired spyware. I/S Law Policy Inf Soc 2(2):283–344
19. Granados A, Cebrian M, Camacho D, Rodriguez FB (2008) Evaluating the impact of information distortion on normalized compression distance. In: Barbero A (ed) Coding Theory and Applications. Springer, Berlin, pp 69–79
20. Hofmann T (1999) Probabilistic latent semantic indexing. In: 22nd annual international ACM SIGIR conference on research and development in information retrieval. ACM Press, pp 50–57
21. Iman RL, Davenport JM (1980) Approximations of the critical region of the friedman statistic. Commun Stat A9(6):571–595
22. Keogh E, Lonardi S, Ratanamahatana CA (2004) Towards parameter-free data mining. In: Tenth ACM SIGKDD international conference on Knowledge discovery and data mining. ACM Press, New York, NY, USA, pp 206–215
23. Keogh E, Lonardi S, Ratanamahatana CA, Wei L, Lee S-H, Handley J (2007) Compression-based data mining of sequential data. Data Min Knowl Discov 14(1):99–129

24. Landauer TK, Foltz PW, Laham D (1998) Introduction to Latent Semantic Analysis. Discourse Process 25:259–284
25. Langville AN, Meyer CD (2004) The use of linear algebra by web search engines. Bull Int Linear Algebra Soc 33:2–6
26. Lavesson N, Boldt M, Davidsson P, Jacobsson A (2008) Spyware prevention by classifying end user license agreements. In: Nguyen NT, Katarzyniak R (eds) New Challenges in Applied Intelligence Technologies, Studies in Computational Intelligence. Springer, Berlin
27. Lavesson N, Boldt M, Davidsson P, Jacobsson A (2011) Learning to detect spyware using end user license agreements. Knowl Inf Syst 26(2):285–307
28. Leydesdorff L (2005) Similarity measures, author cocitation analysis,and information theory. J Am Soc Inf Sci Technol 56(7):769–772
29. Li M, Chen X, Xin ML, Ma B, Vitanyi PMB (2004) The similarity metric. IEEE Trans Inf Theory 50(12):3250–3264
30. Lin S-W, Chen S-C, Wu W-J, Chen C-H (2009) Parameter determination and feature selection for back-propagation network by particle swarm optimization. Knowl Inf Syst 21(2):249–266
31. Lovins JB (1968) Development of a stemming algorithm. Mech Transl Comput Linguist 11:22–31
32. McCallum A, Nigam K (1998) A comparison of event models for naive bayes text classification. In: AAAI-98 workshop on learning for text categorization
33. Nemenyi PB (1963) Distribution-free multiple comparisons. Ph.D. thesis, Princeton university
34. Sebastiani F (2002) Machine learning in automated text categorization. ACM Comput Surv 34(1):1–47
35. Seward J (2001) Space-time tradeoffs in the inverse B-W transform. Data Compression Conference, Washington DC, USA
36. Simeon M, Hilderman R (2008) Categorical proportional difference: a feature selection method for text categorization. In: Roddick JF, Li J, Christen P, Kennedy PJ (eds) Seventh Australasian Data Mining Conference, volume 87 of CRPIT. ACS, Glenelg, South Australia, pp 201–208
37. Telles GP, Minghim R, Paulovich FV (2007) Normalized compression distance for visual analysis of document collections. Comput Graph 31:327–337
38. Vitanyi PMB, Balbach FJ, Cilibrasi RL, Li M (2008) Information theory and statistical learning, Chap. 3. Springer, New York
39. Wang P, Hu J, Zeng HJ, Chen Z (2009) Using wikipedia knowledge to improve text classification. Knowl Inf Syst 19:265–281
40. Witten IH, Frank E (2005) Data mining: practical machine learning tools and techniques, 2nd edn. Morgan Kaufmann, San Francisco
41. Cleary JG, Witten IH (1984) Data compression using adaptive coding and partial string matching. IEEE Trans Commun 32(4):396–402
42. Ye S, Wen J-R, Ma W-Y (2008) A systematic study on parameter correlations in large-scale duplicate document detection. Knowl Inf Syst 14(2):217–232
43. Zhang M, Alhajj R (2010) Effectiveness of NAQ-tree as index structure for similarity search in high-dimensional metric space. Knowl Inf Syst 22(1):1–26
44. Japkowicz N, Stephen S (2002) The class imbalance problem: a systematic study. Intell Data Anal 6(5):429–449

Author Biographies

Niklas Lavesson is associate professor of computer science at Blekinge Institute of Technology, Sweden. He received his M.Sc. in software engineering and Ph.D. in computer science, in 2003 and 2008, respectively, from Blekinge Institute of Technology. His main area of research is machine learning with a special focus on evaluation of supervised learning algorithms. Current research interests involve the application of supervised learning to data mining problems in the domains of healthcare and information security. He is the coordinator of the Ph.D. education at Blekinge Institute of Technology.

Stefan Axelsson is a senior lecturer at Blekinge Institute of Technology. He received his M.Sc. in computer science and engineering in 1993, and his Ph.D. in computer science in 2005, both from Chalmers University of Technology. His research interests revolve around computer security, especially the detection of anomalous behavior in computer networks, financial transactions and ship/cargo movements to name a few. And also how to combine the application of machine learning and information visualization to better aid the operator in understanding how the system classifies a certain behavior as anomalous. Stefan has 10 years of industry experience, most of it working with systems security issues at Ericsson.