A machine learning solution to assess privacy policy completeness

Conference Paper · October 2012

DOI: 10.1145/2381966.2381979

4 authors, including:

Elisa Costante
Eindhoven University of Technology
20 PUBLICATIONS **146** CITATIONS

SEE PROFILE

A Machine Learning Solution to Assess Privacy Policy Completeness

Elisa Costante
Eindhoven University of Technology
The Netherlands
e.costante@tue.nl

Yuanhao Sun
Eindhoven University of Technology
The Netherlands
yuanhaosun2009@gmail.com

Milan Petković
Eindhoven University of Technology
Philips Research Laboratories
The Netherlands
m.petkovic@tue.nl
milan.petkovic@philips.com

Jerry den Hartog
Eindhoven University of Technology
Netherlands
j.d.hartog@tue.nl

ABSTRACT

A privacy policy is a legal document, generally used by websites to communicate how the personal data they collect will be managed. By accepting it, the user agrees to release his data under the conditions stated by the policy. To enable the user to make informed decision, policies should be as complete as possible. Unfortunately, privacy policies fail in their goal of informing the user because of their complexity. Users tend not to read privacy policies finding them long, and difficult to understand, and thus users do not even know whether the policy covers the aspects that are important to them.

We present a solution to assist the user by automatically evaluating the completeness of natural language privacy policies and by providing a structured way to browse the policy content. We extract a set of privacy categories from privacy regulations and directives, and use text categorization and machine learning techniques to analyze a policy and assign an appropriate privacy category to each section. Our results show this approach is technically feasible; an automatic classifier with an accuracy approximating that obtainable by a human judge can be effectively created. Once a privacy policy structured into categories is available, it is possible to grade its completeness, i.e. its coverage of the categories important to the user. With this structure the user can also directly browse the text related to a category he finds interesting.

Categories and Subject Descriptors

H.5.2 [**User Interface**]: Miscellaneous; I.2.7 [**Natural Language Processing**]: Text analysis; K.4.1 [**Privacy**]: Privacy Policy

General Terms

Human Factors, Algorithms, Languages

Keywords

privacy, privacy policy, privacy principle, natural language, machine learning, end-user

1. INTRODUCTION

Protection of and control over ones personal data are rights recognized by the privacy regulations in many countries. The European Union is very strict when it comes to personal data protection. Its Charter of the Fundamental rights clearly states that "*Everyone has the right to the protection of personal data concerning him or her*" [1].

Though organizations that handle personal data are obliged to implement the privacy regulations, the few mechanisms available to offer users of online services control over their data are neither adequate nor easy to use. Often a user's control is practically limited to selecting what organizations to give personal data to. To make an informed decision about whether or not to share personal data, the user should be able to understand how the organization will treat his data.

As a key communication channel to the user, privacy policies should provide complete information on the way personal data is collected, used, stored or shared. However, this information is often 'hidden' in free text full of technical terms, that the user refuses to read [15, 24], preferring the easier way of unconditionally acceptance. This leads to situations where the user has no clue of what conditions he has agreed to, e.g. whether he accepted to share data with third parties, to provide his location information every time he uses a website, etc. or even whether the privacy policy has this information at all. The usability of privacy policies needs to be improved if we want to allow users to truly make informed decisions.

To assist the user in making such decisions, we develop a system that automatically evaluates the completeness of a privacy policy. A more complete privacy policy alleviates the (user's) concerns on data disclosure [3], and forms a better tool for communication. A privacy policy is complete when it addresses the most important privacy principles, as defined by OECD and privacy legislations.

To assess the level of completeness of a policy, we define a set of privacy categories that the policy should cover based on privacy directives, regulations and common practice. Examples of categories are *Collection*, *Share*, *Cookies*, and *Security*. We then use text categorization and machine learning techniques to check which paragraphs in the natural language privacy policy belong to which category, and grade the policy based on the categories covered. When we detect that the user is viewing a privacy policy, the grade is provided to him, to give immediate information over the policy quality. The user can then inspect the policy in a structured way, selecting to look at only those paragraphs belonging to the categories he is interested in.

The main contribution of this paper is thus a system consisting of: i) an automatic completeness analyzer to verify which privacy categories are covered by a policy ; ii) a grading system to assess the level of completeness according to the categories covered; iii) a mechanism allowing the user to browse only the text related to categories of interest, and iv) an automatic privacy policy detector to trigger the policy analysis when a user visits a privacy policy web page.

Note that our evaluation is only based on the contents of the privacy policy, therefore the completeness grade does not give any guarantees on whether the policy is enforced by the website. In addition, a high completeness grade only means the policy covers the most of the categories, but says nothing about their semantic value. For example, in the category *security*, a policy could state "*our transaction are protected by the use of the SSL protocol*", but it would achieve the same level of completeness by stating "*during our transaction we do not make use of the SSL protocol*". Possible solutions to this problem are discussed in Section 6. It is also worth noting that simplifying a privacy policy, to make it easier to understand, may lead to a lack of precision that can have legal implications [19]. For this reason, the privacy policy *as-is* has to remain the main source of information. That is why, although we summarize the quality of a privacy policy with a completeness grade, we also provide a structured way to browse the actual text of the privacy policy, to let the user have easily access to the complete information.

The remaining part of the paper is organized as follows: in Section 2 we analyze the related work, describing what has been already done in the field of privacy policies analysis, and the position of our contribution. In Section 3 we present the goals, the methodology and the implementation of the completeness analyzer, while in Section 4 we discuss the feasibility of the approach. In Section 5 we describe how we detect privacy policies amongst other web pages, while in Section 6 we discuss an extended framework, part of our future work, aiming at evaluating the general privacy protection level offered by a website. Finally, we address conclusions in Section 7.

2. RELATED WORK

The difficulty users have in understanding privacy policies, often resulting from complex and ambiguous language, has been observed in several studies [22, 4]. Approaches to improve privacy protection by means of privacy policies can be categorized by considering which of the two main stakeholders are involved: the end-user (client-side), to whom the policy is addressed, or the policy authors or enforcement authorities (server-side). Approach such as SPARCLE [11, 10] or PPMLP [55, 56] addresses the server-side, while solutions such as P3P [17] involve both sides. Final there are the approaches such as the PrimeLife Dashboard [47] that focus on the client-side.

SPARCLE (Server Privacy ARchitecture and CapabiLity Enablement) [11, 10] is a privacy management workbench aiming at facilitating privacy policy authoring, implementation, and compliance monitoring. The framework is intended to assist an organization in the writing, auditing and enforcement of privacy policies. Its main goal is to help organizations to create understandable privacy policies, compliant with privacy principles. The framework takes privacy policies written in constrained natural language, checks them for compliance with privacy principles, and translates them into a machine readable and enforceable format, e.g. EPAL [6] or XACML [31]. The use of specific patterns in the sentences and constrained natural language, i.e. a subset of a natural language with a restricted grammar and/or lexicon [39], makes it possible to parse such policies.

PPMLP (Privacy Policy Modeling Language Processor) [55, 56], as SPARCLE, aims to help organizations in generating privacy policies, making such policies compliant with the privacy principle extracted from the Australian National Privacy Principles. The privacy policy authors specify a meta-privacy policy, that is then translated in a template of rules for the enforcement. Such a meta-policy is analyzed by the system that suggests new rules, to allow the compliance with the privacy principles. Once the meta-policy is ready, it is translated both in EPAL, used for its enforcement, and in natural language (using static matching rules), for the presentation to the end-user. Within the system a PPC (Privacy Policy Checker) is used to check whether the policy is enforced: the PPC is plugged into the website, and has access to the data the application stores about an user. When the end-user performs a transaction, the PPC analyzes it, to verify that the policy is enforced, and assigns a compliance rate to the website. Such grade is based on whether the enforcement of the policy takes place, and on the weight the end-users gives to each of the different privacy principles. In this context the PPC is trusted by both the end-user and the website.

P3P (Platform for Privacy Preferences) is the W3C's attempt to manage privacy policies, allowing the website to express policies in a XML-based machine-readable format, and the user to automatically check those policies against his preferences, by the mean of P3P-enabled browsers [17]. As first step the website sends a machine-readable proposal of its privacy practices. The proposal can be automatically parsed by a user agent (such as web browsers, or plugins) and compared with the user's privacy preferences, thus, users do not need to read the privacy policies of every web site they visit [36]. Privacy Bird [16] and Privacy Finder [45] are examples of P3P user agents, able to compare P3P policies with users preferences. A limitation of the P3P, shared by SPARCLE and PPMLP, is that it needs server-side adoption, which is not easily obtained: according to [8] only the 20% of the websites amongst the E-Commerce Top 300 is

P3P enabled.

The PrimeLife Privacy Dashboard [47] is a recent browser extension aiming at evaluating the quality of a website. To this end, it collects information about the website the user is currently visiting, such as whether it has a P3P policy, whether it collects cookies, and whether it is certified by trust seals. The dashboard provides then a visual 'privacy quality' rating of a website: the presence of a P3P version of the privacy policy increases the quality rating, while the presence of external or flash cookies decreases it. In addition, the dashboards allows the user to set preferences and check what data has been exchanged with a website; also, if available, it translates the website P3P policy into plain English. The low adoption of P3P limits the effectiveness of this approach: a website may have a good privacy policy, but may be rated low because of the lack of a P3P version.

In contrast, our approach is a pure client side solution and only requires the existence of a plain text privacy policy. As such, it can easily be adopted by privacy-concerned end users as no special support from the server side is needed. The trade-off is that it does not cover enforcement of the privacy policy at the server side. To the best of our knowledge, our work represent the first attempt to automatically grade the coverage level of plain text privacy policies (in [2], for example, the work to grade a privacy policy, has to be done manually).

3. COMPLETENESS ANALYZER

In this section, we describe the goals of our policy completeness analyzer, and the methodology we use to assess the completeness level of a privacy policy.

3.1 Goals

The main goal of the completeness analyzer is to assign a completeness grade to a privacy policy. This is done by parsing the contents of a privacy policy, and checking which privacy categories are covered. As the privacy policy is meant to be a communication channel to the user, the level of completeness is an important quality aspect to evaluate. To denote the completeness level of a privacy policy p, we use $G_c(p)$ which we compute as follows:

$$G_c(p) \;=\; N \cdot \sum_{i=1}^{n} w_i \cdot c_i$$

where $N = 10/\sum_{i=1}^{n} w_i$ is a normalization factor[1] which scales results to $[0, 10]$, n is the number of privacy categories, w_i is the weight, in $[0, 1]$, assigned by the user to the category i, and c_i is the coverage level which is either 1 (covered) or 0 (not covered). In this paper we assume the user weights (w_i) are given. They could be obtained in different ways, such as directly specified in the user's preferences, or automatically generated by collaborative filtering.

Given a privacy policy, the completeness analyzer classifies each of its paragraphs and computes its grade. Figure 1 shows how the results of the completeness analyzer are visualized. The method to detect that a user is visiting a privacy policy page is described in Section 5. The overall completeness grade $G_c(p)$ is shown using the traffic light metaphor, together with the categories that are covered (green tick; $c_i = 1$, $w_i \neq 0$), not covered (red cross; $c_i = 0$, $w_i \neq 0$) or not considered relevant by the user (gray circle; $w_i = 0$). Additionally, the user can select one of the covered categories, and browse the paragraphs of the policy that belong to that category.

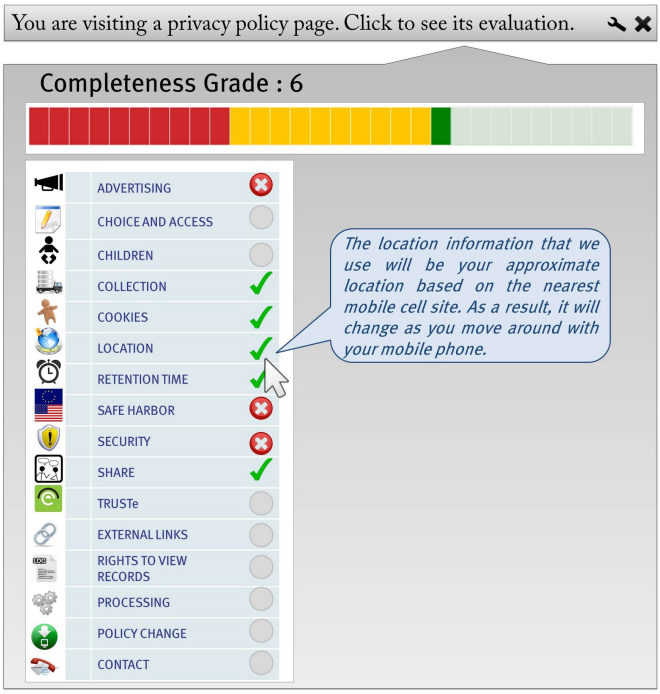


Figure 1: The completeness analyzer interface.

To assess the value of c_i, i.e. whether category i is covered by a privacy policy, we apply text classification and machine learning techniques. Text classification is the automatic activity of labeling natural language text with thematic categories from a predefined set, while machine learning is an inductive process to automatically build a text classifier by learning from a set of pre-classified documents [40].

In our context, the pre-classified documents are paragraphs from manually labeled privacy policies. This set of policy paragraphs (a.k.a. corpus) is used to train a classifier, to correctly assign a privacy category to the paragraphs of an unlabeled policy. For example, to the paragraph "*We collect other information when you submit it to us. It may include your gender, education, occupation, financial information (such as your income, investments, portfolio transactions and value, ...), interests, photos, connections, and comments*" should be automatically assigned the category *Collection*.

Figure 2 depicts the process we used to build and evaluate a text classifier. The privacy categories are extracted from privacy regulations and common practice (Section 3.2). The list of categories is then used to annotate the corpus (Section 3.3), by manually labeling each of the privacy policy paragraphs forming the corpus with one of the privacy categories.

The labeled corpus is then transformed into a suitable representation by applying standard text preparation operations such as tokenization, stemming and vectorization [49].

The corpus is also split into two different sets: a train-

[1]If all weights w_i are 0, i.e. the user does not care about any of the categories, the grade of any policy is defined to be 10.

ing and a testing set. The training set is used as input to the learning phase, while the testing set is kept apart from the training set and leaning process, so that a completely fresh set is available for the final evaluation of the created classifiers.

The learning phase starts with pre-processing the training set, optimizing it for effective learning (Section 3.4). In the learning, different classifiers are built according to different machine learning algorithms (Section 3.5). The performance (Section 4.1) of the classifiers is tested using the *k-fold cross-validation* technique [37] (Section 4.2). Different settings and configurations are tested, and several optimization techniques are applied to find the best performing classifier (Section 4.3). Finally, the resulting classifiers are validated by evaluating their performance on the testing set.

3.2 Privacy Category Definition

In defining the privacy categories we considered different privacy regulations and directives, such as the EU 95/46/EC and the EU 2006/24/EC Directive on the protection of individuals, the guidelines issued by the Organization for Economic Cooperation and Development (OECD) in 1980, the Fair Information Practice Principles published by the U.S. Federal Trade Commission (FTC), and the Safe Harbor Framework [18]. We also considered regulations concerning the privacy of specific groups of people, like the Children's Online Privacy Protection Act) from which we derived the category '*Children*'. A more comprehensive survey of current regulations on privacy protection can be found in [20].

Although they are not mandatory, the principles in the legislations mentioned above provide guidance for drafting privacy policies. Organizations such as TRUSTe[2], eTrust[3], and Webtrust[4] certify policy compliance with such directives, requiring implementation of fair information practice.

In defining the categories we also accounted for common practice, i.e. topics that are usually addressed by privacy policies such as *Advertising*, *Location* for location based services, etc. Below we give, and briefly describe, the resulting privacy categories.

- **Advertising** explains how the website manages advertisements (e.g. banners, sms, e-mail), and whether advertising is controlled by the website itself or by third parties.
- **Choice and Access (C&A)** provides information about the user's privacy choices, such as opt-in/opt-out options, and user's rights to access, modify and/or delete the information collected by the website.
- **Children** explains the company's policy regarding the collection and use of personal information of children.
- **Collection** explains how and what kind of personal information may be collected by the website.
- **Cookies** explains whether the website makes use of cookies. It may also state the purpose(s) of using cookies, and the information the cookies store.
- **Location** explains how the website manages user's location information.
- **Retention** explains the purpose(s), and duration of the retention of personal data.

[2] http://www.truste.com/
[3] http://www.etrust.org/
[4] http://www.webtrust.org/

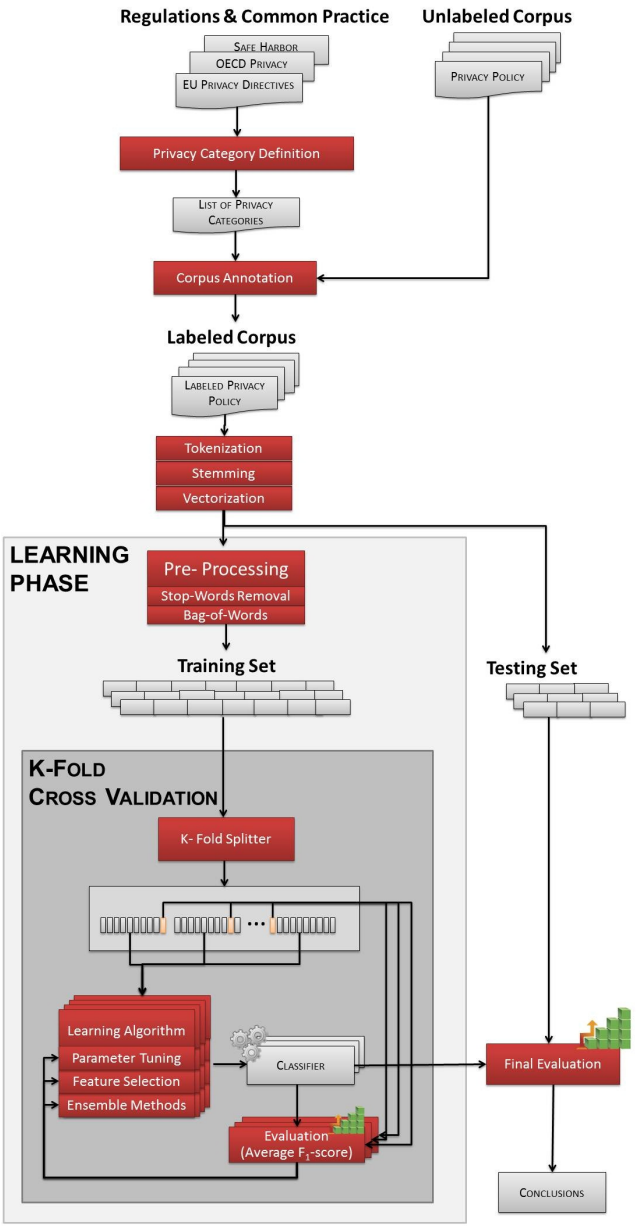


Figure 2: Classifier building and evaluation process.

- **Safe Harbor** explains the website participation in, and self-compliance with, the U.S.-EU/Swiss Safe Harbor Framework.
- **Security** explains the security technologies applied by the website, e.g. use of SSL on the website or access control policies regulating employees' practices.
- **Share** explains whether, and under which conditions, the website will share user's information.
- **TRUSTe** explains whether the website has been awarded the TRUSTe's Privacy Seal. This seal signifies that the website's privacy policy and practices are compliant with the TRUSTe program's requirements such as transparency,

accountability and user's choice regarding collection and use of personal data.

- **External Links** warns the users about the fact that the current privacy policy does not cover third party websites reachable with external links.
- **Rights to view records** refers to the users' right to request the access to the records of his personal information disclosed to third parties.
- **Processing** explains where the personal data is transferred to, stored and processed (the country where storing and processing take place impacts which regulations apply).
- **Policy Change** explains how updates to the privacy policy are managed, and whether and how the users will be informed of such changes.
- **Contact** provides company's contact information, such as the registered office, or the address users can use for further questions or complaints.

3.3 The Corpus

The corpus is an initial set of paragraphs extracted from privacy policies and labeled with the categories described before, and it is used during the learning phase to build the classifier. The corpus is usually divided in two subsets: the training set ($\sim \frac{7}{10}$), used to train and validate the classifier, and the testing set ($\sim \frac{3}{10}$), used as a final evaluation of the classifier. The two subsets are strictly separated, i.e. the testing set cannot be used during the learning phase.

Through machine learning one can only train the patterns present in the corpus. Thus the corpus must be large enough to contain all the patterns of interest, and to cover all the categories with a sufficient number of samples.

Our corpus is based on 64 privacy policies (including policies from e.g. Google, Amazon, FoxNews) of which each paragraph has been manually annotated. This resulted in 1049 annotated paragraphs, 772 of which are used as training set, and 277 as testing set. Table 1 specifies the number of annotated paragraphs belonging to each category, for the training and the testing set.

Table 1: Amount of labeled paragraphs in the training and in the testing set (divided by privacy categories)

Category	Training	Testing
Collection	121	34
Sharing	128	47
Choice & Access	107	41
Cookies	85	27
Children	33	18
TRUSTe	19	5
Safe Harbor	23	5
Link outs.	32	11
California	23	4
Retention	13	2
Processing	17	8
Security	50	28
Advertising	39	9
Change	35	19
Location	17	1
Contact	30	18
Total	772	277

Reliability of the annotated corpus is clearly a prerequisite for training a high quality classifier. To verify the reliability of the annotations, a basic agreement test was performed. An agreement test aims at measuring the level of agreement amongst different annotators: a high level of agreement shows objectivity of the annotation process, and indicates a high quality of the corpus.

The labeling of our corpus was performed by a single annotator. For the agreement test, a second annotator had independently labeled a selected subset composed by of 102 paragraphs, extracted from the training set. The subset was selected in a way to have a distribution of paragraphs over categories close to the one of training set.

The two annotators agreed in the 91% of the cases. Three of the nine items upon which they disagreed were marked as *difficult to label* by the second annotator, and the labels assigned by the first annotator were indicated as a second choice. Discarding these cases would increase the agreement level to 94%. This leads us to conclude the corpus annotation is sufficiently reliable, but also that in some cases human classifier can disagree on the classification of paragraphs, something that needs to be kept in mind when analyzing the results of the automated classifiers.

3.4 Preprocessing

In general data preprocessing consists of applying a set of techniques, such as cleaning, normalization, transformation, feature extraction and selection, to the ('raw') training set to eliminate noise that could decrease the quality of the learnt classifier [28]. Using the resulting ('polished') training set as input, usually significantly increases the effectiveness of the machine learning algorithms and/or the quality of the resulting classifiers.

Here we applied cleaning (stop-words removal) and transformation (bag-of-words representation) techniques. The stop-words removal has been done using a common English stop-words list, with the aim of reducing the noise produced by very common words. The bag-of-words transformation represents a document as a vector of words. The weight of each word is given by the number of its occurrences. Instead of the simple word frequency, we apply the tf-idf (term frequency - inverse document frequency) weighting [38], because it has the advantage of decreasing the weight of words appearing very often in the document while increasing the weight of rare words.

Feature selection is another common pre-processing technique that can have positive impact on the performance of the classifiers. It will be discussed in Section 4.3, as part of the optimization methods, rather then applying it during preprocessing.

3.5 Learning Algorithms

The problem of constructing a text classifier has been widely treated in literature, resulting in the existence of many machine learning algorithms. The goal of these algorithms is to build a function that, given a document, returns a value, usually called CSV_i, representing the evidence that the document should be assigned class C_i). The CSV_i score for classifying documents may represent e.g. a probability, or a measure of vector closeness [40] depending on the algorithm used.

Machine learning algorithms can be divided based on several characteristics [7] including supervised versus unsupervised learning, and linear versus non-linear combination of

features. In text classification a feature usually corresponds to a word. As we work with an annotated corpus we use the supervised learning. For the algorithms we selected we indicate whether they use linear or non-linear methods to combine features.

Since the algorithm selection problem, i.e. the problem of selecting the algorithm that performs best for a given task, is still unsolved [29], we test the performance of different algorithms to determine which algorithm achieve the best results, and whether such results are satisfactory. Below we describe the main characteristics of the algorithms we selected, and the reasons behind our choice.

We selected Naïve Bayes (NB), Linear Support Vector Machine (LSVM), and Ridge Regression (Ridge) from the 'linear' algorithms and the k-Nearest Neighbor (k-NN), the Decision Tree (DT), and the Support Vector Machine (SVM) with non-linear Kernel from the 'non-linear' algorithms. We also applied several ensemble learning methods that try to build more effective classifiers by combining (the outcome) of different algorithms.

Naïve Bayes represents a group of close related probabilistic methods, where CSV_i is the probability that a document belongs to a category C_i. We selected it because of its computational efficiency, important for real time applications [26]. Its most commonly used variants are the multivariate Bernoulli event model, and the multinomial event model. We selected the latter variant, because it generally outperforms the former [30].

The LSVM [46, 13] is one of the most popular machine learning algorithms, based on the representation of training items as points in space. The mapping is then done in such a way that items belonging to different categories are divided by a clear linear gap. The classifier obtained with this algorithm labels new items into a category based on which side of the gap they fall on. It has been selected because it suits well with text classification [25].

The ridge regression classifier [21] is a variant of the least squares regression model, selected because it is computationally efficient, and because it solves the problem of possible not-unique solutions [57]. Moreover, just like all the others linear classifiers, it is supposed to do well for text classification.

In the k-NN classifier, the k represents the number of closest neighbors (training items) considered. The k-NN has a minimal training stage and an intensive (and timing consuming) testing stage. The basic idea is that the k-NN classifier assigns to a new item the category owned by the majority of its k nearest neighbors. The k-NN classifier is known as one of the top-performing approaches in text categorization tasks [53], therefore, we want to check how it performs in our context.

Decision tree based approaches are particularly appealing for text learning, because their performance compares favorably with other learning techniques in this setting [5]. There are several classic decision tree algorithms, such as ID3 [34], C4.5 [35] and CART [9]. The latter is the one we selected.

Non-linear SVM algorithms are useful when the gap between different categories cannot be linearly modeled and, thus, a more complex function is needed. There are several alternatives to the linear kernel used in the linear SVM [43, 44]. In our experiments we use the RBF kernel, because, for text classification tasks, it outperforms several other variants [48].

Ensemble learning [33] is an optimization technique that generates a classifier as a combination of others. To create ensembles of different types of classifiers, (e.g. SVM, Ridge regression and Naïve Bayes) it is advised to use methods such as the *voting committee* [12], and the *stack generalization* [52]. The voting committee method combines the results of different classifiers, into a voting committee: the category that has been selected by the most of the classifiers is chosen to label the item. On the other hand, the main idea in the stack generalization method is to learn a meta-level (or level-1) classifier based on the output of base-level (or level-0) classifiers [42]. In our experiments we test the voting committee and two variants of the stacking method: the *probability variant* (Prob.), and the *prediction variant* (Pred.) [42].

4. EVALUATION

The evaluation phase of the machine learning approach to building text classifiers aims to validate that classifiers of adequate effectiveness have been created. Different iterations are typically needed to find effective classifiers and the evaluation results in each iteration represent a guide to further improvements and configuration choices for the next iteration. In this section we describe how we measure the effectiveness of a classifier and what method we use for the validation. We also discuss the results obtained applying optimization techniques to the classifiers and the results of the final evaluation over the testing set.

4.1 Metrics

The effectiveness of a classifier is usually computed in terms of the information retrieval notions of precision and recall [40, 54]. Intuitively, precision with respect to a class, is the rate of items classified as being in this class which are actually from the class, while recall is the rate of items from the class which are indeed labeled as belonging to this class. The F_β score of a classifier combines precision and recall into a single score which can be used to compare the quality of classifiers.

Given a corpus of documents D divided into $D_1, \ldots, D_N$ according to the category label, and a classifier $L : D \to \{1, \ldots N\}$ we define the True/False Positive rates (TP/FP) and Negative rates (TN/FN), the precision, recall and F_β by:

$$
\begin{aligned}
TP_i &= |\{d \in D | L(d) = i \wedge d \in D_i\}| \\
FP_i &= |\{d \in D | L(d) = i \wedge d \notin D_i\}| \\
TN_i &= |\{d \in D | L(d) \neq i \wedge d \notin D_i\}| \\
FN_i &= |\{d \in D | L(d) \neq i \wedge d \in D_i\}| \\
Precision &= \sum_{i=1}^{N} \frac{|D_i|}{|D|} \cdot \frac{TP_i}{TP_i + FP_i} \\
Recall &= \sum_{i=1}^{N} \frac{|D_i|}{|D|} \cdot \frac{TP_i}{TP_i + FN_i} \\
F_\beta &= (1 + \beta^2) \cdot \frac{Precision \cdot Recall}{\beta^2 \cdot Precision + Recall}
\end{aligned}
$$

Where $|S|$ denotes the size of set S. Precision and recall are a weighted average over the different categories. The β in F_β determines the relative importance of precision and recall

where smaller numbers, e.g. $1/2, 1/3, \ldots$ indicate a higher importance of precision, while large numbers, e.g. $2, 3, \ldots$, are used if recall is more important. In the following we use the term effectiveness or performance to refer to the F_1 score of a classifier, where precision and recall are given the same importance. Note that, the higher the F_1 score, the lower the false predictions($FP + FN$).

4.2 Cross Validation

In the experiments described below, we make use of the *k-fold cross validation* technique [37] to compute the F_1 score and to estimate how good a classifier is going to perform on future unknown data. Typically, the k-fold cross validation consist of partitioning the corpus into k equally (or nearly equally) sized folds. After the partitioning, k iterations of training and testing are performed, in such a way that in each iteration a different fold of the data is used for testing, while the remaining k - 1 folds are used for learning. We use the 10-fold cross validation because this value of k is particularity attractive [37].

In several experiments we apply $n \times 10$-fold cross validation [51], which corresponds with repeating the 10-fold cross validation n-times using different random foldings, i.e. the partitioning of documents from the training set over the folds is randomly re-selected each time. This thus results in $n \times 10$ different training and testing sets. The F_1 score is then computed as average of the scores obtained at each repetition.

4.3 Experiments

In this section we describe the learning phase experiments that we carried out to evaluate the performance of the classifiers, and the impact of different configuration settings and optimization techniques. We also discuss the results of the experiments and the implications of these results.

(a) **Parameter Tuning**: The machine learning algorithms we used in our process, except for ridge regression and naïve Bayes, depend on the value of some parameters that need to be tuned to work well in a specific setting. Tuning parameters is normally done empirically, by running the algorithm with different values of a parameter, and select the value that leads to the best F_1 score. When an algorithm depends on multiple parameters a grid search is used for the tuning [23]. Figure 3(a) shows the different parameters and the values considered for these parameters (parameter domain) during the experiment. For readability reasons, we actually show the $\log_2$ of the value for some of the parameters. The figure also shows which setting for the parameter gives the best results in training the algorithms on the training set. The parameters are fixed to these best performing values in the remaining experiments unless stated differently.

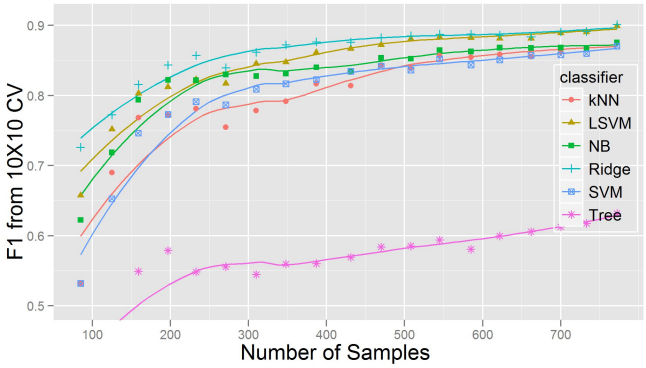
(b) **Impact of the Sample Size**: Choosing the training set size is a compromise between the effectiveness achievable by the classifiers and the effort involved in building the training set. If the training set is too small this may significantly degrade the effectiveness of a classifier. On the other hand, diminishing returns in improvements to the classifier make that increasing the training set is, at a certain point, no longer worth while. In this experiment, we test how increasing of the training set size influences the F_1 scores of the classifiers, and validate that the size of our training set is adequate, in that increasing its size should give only marginal benefits. We extracted 18 subsets from the training set, starting with 10% of the whole set and increasing size in steps of 5% of the training set until the final subset, consisting of the whole training set. The F_1 score of the six classifiers is computed for each subsets, using the 10×10-fold cross validation. Figure 3(b) shows the results we obtained by running this experiment. By looking at the trend of the classifiers performance, we can note that 300 samples would not be enough, since the performances of all the classifiers can still considerably improve. On the other hand, after 700 samples, an increase in the size leads to marginal improvements. For this reason, we claim that our training set size is adequate, especially given that our goal is to showing feasibility rather then obtaining maximal performance. The figure also provides some initial indications on the performance of the classifiers we are testing. Considering the F_1 score obtained over the complete training set, Ridge ($F_1 = 90.0\%$), LSVM ($F_1 = 89.9\%$), and Naïve Bayes ($F_1 = 87.5\%$) are the classifiers performing best, while k-NN ($F_1 = 87.4\%$) and SVM ($F_1 = 87.1\%$) are less effective. The decision tree, with its F_1 score equals to 63.1% is performing very poorly and, for this reason, is discarded for the further experiments.

(c) **Impact of Feature Selection**: Feature selection [28] is an optimization technique, enabling learning algorithms to operate faster and more effectively by removing irrelevant and redundant information. During the process, a correlation score between a word and each category is computed; only the words with the highest scores are selected. In this experiment, we evaluate the performance of the classifiers using the χ^2 feature selection method [41]. Also, because of the variation in the number of samples, we changed the parameters k (for k-NN) and γ (for the SVM) to the values $\gamma = 1/\#features$, and $k = \#features/2\#classes$. Figure 3(c) displays the impact feature selection has on the performance. The results indicates that Ridge and LSVM reach their best performance at 20% of feature selected, while Naïve Bayes reaches such peak with slightly more features (30%), and start to decay with more than approximately 70% of features selected. The k-NN classifier performs poorly until almost all features are selected. For the next experiments a 40% feature selection is applied, given the fact that for that value, all classifiers (except for k-NN) have already reached their best performances.

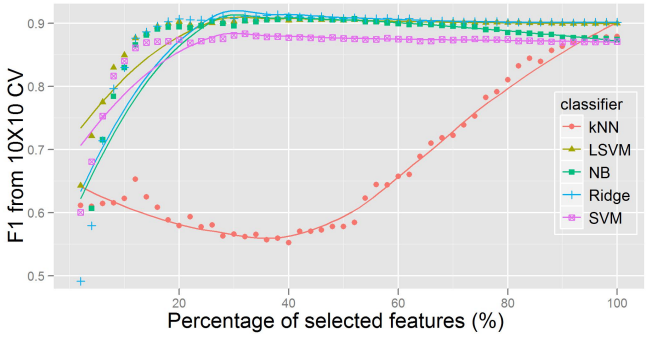
(d) **Impact of Ensemble methods**: In this experiment we compare the performance of single classifiers (LSVM, NB, Ridge) with the ones obtained by applying ensemble methods such as the voting committee, and the probability (Prob.) and prediction (Pred.) variants of the stacking. The F_1 score is obtained applying a 50×10-fold cross validation. The results are depicted in Figure 3(d). SVM and k-NN do not appear in the graph because of their low performances with 40% feature selection ($\sim 55\%$ for k-NN, and $\sim 87\%$ for SVM as shown in Figure 3(c)). The figure shows the mean and the standard deviation for the F_1. As we can see, the probability variant of the stacking ensemble outperforms the

Classifier	Parameter(s)	Parameter Domain	Best value
k-NN	k	$1, 3, 5, \ldots, 49$	19
SVM (loose search)	$log_2(\gamma)$	$-15, -11, -7, -5, -3, -1$	-5
	$log_2(C)$	$-3, -1, 1, \ldots, 15$	7
SVM (fine search)	$log_2(\gamma)$	$-8, -7, -6, -4, -2$	-4
	$log_2(C)$	$5, 6, 7, 8, 9, 10, 11$	5
LSVM	$log_2(C)$	$-3, -1, 1, \ldots, 15$	-1
	regularization method	$l1$, $l2$	$l2$
Decision Tree	max_depth	$6, 8, 10, 12, 14, 16$	16
	min_split	$2, 3, 4, 5, 6$	5

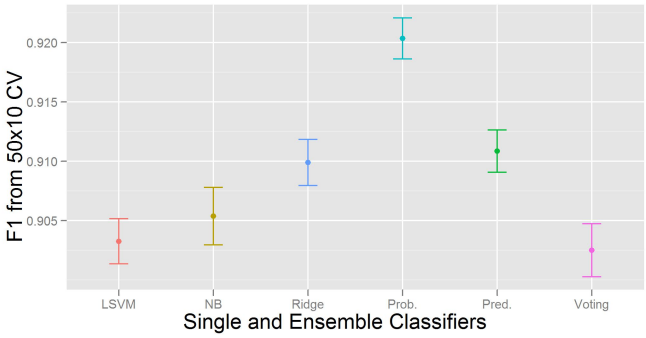
(a) Parameter Tuning



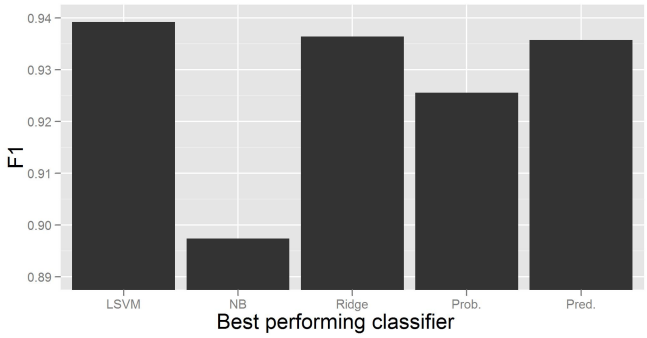
(b) Impact of the Sample Size



(c) Impact of Feature Selection



(d) Impact of Ensemble methods



(e) Final Evaluation over the testing set

Figure 3: Experimental Results

best single classifiers, achieving an F_1 score which is more than 1% higher. To confirm these results we also apply a paired t-test that showed that the probability variant of the stacking ensemble significantly improves the performance compared to the best single classifier (Ridge regression). The voting ensemble can be discarded from further experiments, because it is both less effective and more computationally expensive than the best single classifier. Even though we do not address the computational costs in our experiments, it is easy to see that the ensemble methods are more expensive than the single classifiers, simply because they use several of them, and not necessarily in parallel.

(e) **Final Evaluation on the testing set**: The final evaluation test aims at validating the best performing classifiers by running them on the testing-set. This checks that there is no over-fitting problem; where the classifier is too specialized toward the training set and cannot correctly classify other data. In case of over-fitting the results on the test set would be significantly less than on the training set. This test is done at the very end when all parameters have been set and the classifiers are fully specified. Because no information about the test set was used during the training of the classifiers, the results of this experiment should be indicative of the performance in real scenarios. Figure 3(d) presents the results of the final evaluation. All classifiers perform at least as good as on the training set suggesting there is no over-fitting problem. Ridge and Prob. Stacking still perform good, with an F_1 score between 92% and 94%, confirming the results obtained during the training-phase, while LSVM exceeds the expectations, showing a 94% F_1 score.

As general conclusions, we can say that the choice of the best classifier is a trade-off between performance and computational costs. The probability variant of the stacking ensemble performs very well, providing a significant improvement of performance compared to the best single classifier. There is, however, added computational costs associated with this performance (ensemble methods costs approximately 10 times more than single classifiers). Since single classifiers, as Ridge and LSVM, show good performances in training and test evaluation, and are less computationally expensive, they may be a good choice when it is important to keep computational costs down (e.g. in real-time applications). Also, Ridge and LSVM cost 10 or 20 times more than the Naïve Bayes, so the latter may represent a reasonable fall-back option when costs are very important (e.g. on mobile devices) and 90% effectiveness is considered sufficient. In our scenario, where a good effectiveness is crucial to show the feasibility of our approach, and costs are not really a problem, we can state that the stacking ensemble method represents the best choice.

As final remark, note that the maximum effectiveness we reach with an automatic classifier ($\sim$ 92%), can be considered adequate to our task, since it approximates the results obtainable with a human classifier. As seen in the agreement test, when different human judges are asked to classify a privacy policy, the objectivity (that can be seen as a measurement of effectiveness) is close to the 92% level. However, study to increase the overall effectiveness of the classifiers may be left as future works.

5. PRIVACY POLICY DETECTION

Before a privacy policy can be structured and evaluated, we need to detect when a user is visiting a privacy policy page. This is done by the means of a privacy policy detector, a component able to recognize privacy policies amongst other types of web pages. We construct a detector based on a set of regular expressions, assuming that a privacy policy may be detected according to the presence of certain keywords and patterns. If the number of regular expressions (from our set) that a web page matches is bigger than a certain threshold, then it is tagged as a privacy policy. This detection then triggers the processing and evaluation of the policy. The detector has been implemented as a Google Chrome extension[5].

The relatively simple detecting approach described above achieves good accuracy. We tested it by simulating a user browsing session: we accessed 40 different websites and we randomly visited their pages. For each website, we were made sure to follow the link to the privacy policy page to check the true positive/false negative rates and to access generic pages to check for false positives and true negatives. We visited around 20 pages per website giving a total of 873 web pages and obtaining an accuracy (seen as the ratio of correct predictions over the total number of samples) of 99.3%. The precision is high as well (97.9%), but the recall is relatively low (90%) due the fact that some policies are hosted on non-html files (.pdf or .txt), which are not detected by our implementation. The approach, however, is still valid; extending the implementation to cover different file formats would be possible.

To make a more stressing test, we simulated another scenario where the user enters the key-words 'privacy policy' in a search engine (we used Google Search for this experiment). We analyzed the first 875 web pages, retrieved as the search result, and we measured a detection accuracy of 92.0%. The deterioration of the accuracy in this scenario is mostly due to the fact that the pages returned by the engine, though very relevant to the privacy policy topic, are not always policies leading to an increased false positive rate.

Once the detector has recognized a web page containing a privacy policy, the policy text needs to be extracted before the text classification can take place. The main contents of the policy must be separated from other building blocks such as headers, footers, menus, advertisements, etc. Different methods are available in literature to accomplish this task ([27, 50, 14, 32]). We have implemented a solution using Boilerpipe[6], a Java library based on the method described in [27].

6. EXTENDED FRAMEWORK

The completeness analyzer presented in this paper, is intended to be part of a larger framework which automatically grades the privacy protection offered by a website. Assessing the level of privacy protection the user has while using a certain website is a more ambitious goal than just assessing the completeness of the privacy policy. While the completeness level gives a good objective measure of the quality of a privacy policy as a tool to inform the user, it may not match the user's subjective view on the level of privacy protection. For example, a web site may have a complete privacy policy that correctly states what data they gather. However, the user may find the amount that the web site collects excessive for the service provided (e.g. a credit card number is required during the registration process to a free service, an email or phone number need to be given even though there should be no need to contact the user, etc.). In this case the web site would be considered by the user to have a low privacy protection level.

The machine learning approach demonstrated shows good results for the completeness analysis but it only considers the content (in terms of keywords) of a privacy policy and does not consider its semantic meaning. Two policies completely different from a semantic perspective, but covering the same amount of categories, will achieve the same level of completeness. With a high completeness grade the user knows the categories he cares about are covered but he still needs to read the related paragraph(s) to know *how* the category is addressed (e.g. which data is collected, with whom it is shared, etc.).

If we want to assess not only completeness but also the protection level, we need to be able to extract some valid semantic information from a privacy policy. For example, we need to distinguish a privacy policy stating "*we WILL share your Personal Data with third parties* " from one saying that "*we WILL NOT share your Personal Data with third parties* ". Providing feedback on policies indicating not only what topics are covered but also whether what the policy states is 'safe' or a 'potentially risk' from a privacy perspective helps the user in making a more informed decision.

A solution to extract semantic information out of a privacy policy may come from the application of information extraction and natural language processing techniques. For

[5]available at `https://github.com/YuanhaoSun/ChromePP`

[6]`http://code.google.com/p/boilerpipe/`

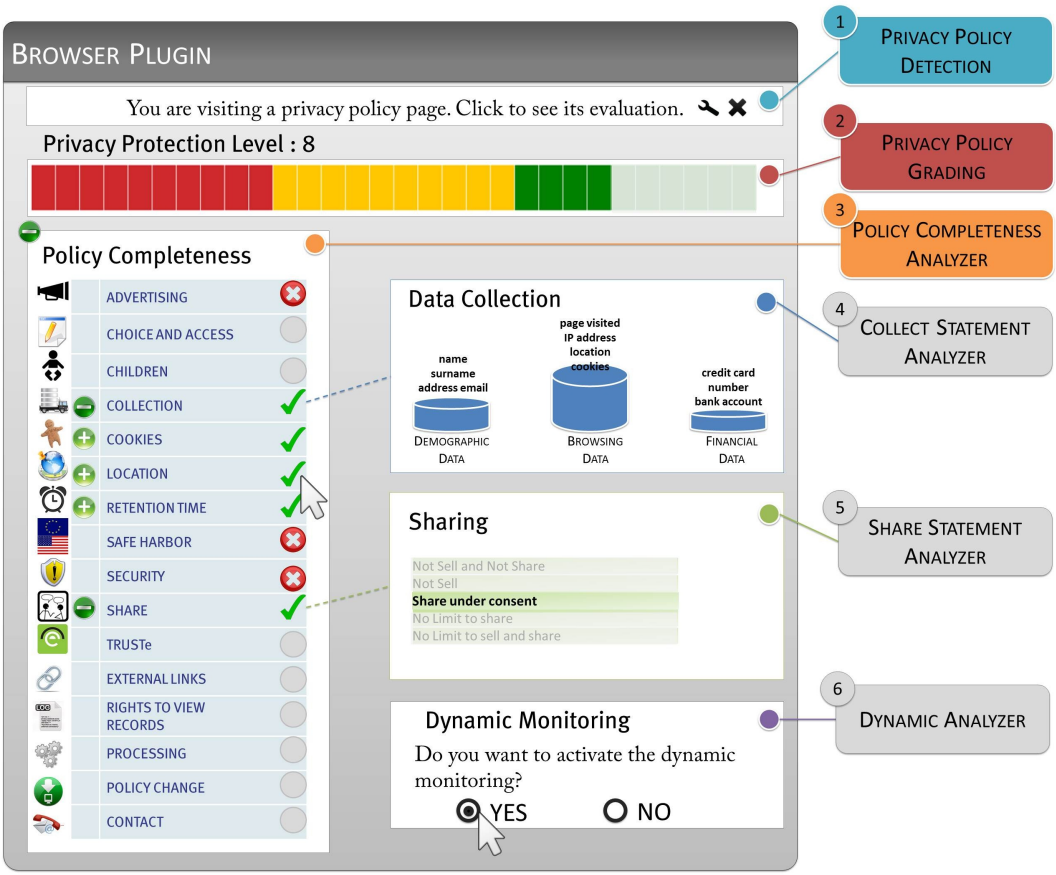


Figure 4: The Extended Framework Interface.

each privacy category, for example, a set of semantic patterns of interest may be defined (e.g. we share/do not share your personal information), and the privacy policy may be scanned looking for such patterns. The results may be then analyzed and presented to the user in a way like the one depicted in Figure 4.

The figure aims at representing several characteristics of the solution we have in mind. We want to provide the user visiting a privacy policy page, with an overall grade of the privacy protection level offered by that website. The user can then decide to base his privacy decision on that grade, or to obtain more information by inspecting the detailed results. The completeness analysis represents a starting point for further investigation: the semantic analysis of a category can only take place if the category is actually covered by the policy. Let us consider the category *Collection*. If the policy covers such category, we may show to the user the amount of personal data the website collects, divided by categories (e.g. demographic, browsing or financial data). In this way the user can have an idea of what data is at stake by using the website. In the figure, the results of the share statements analysis, are also presented. All the information presented to the user is retrieved from the privacy policy, by using apposite modules (the blocks on the right end of the figure).

However, adding the semantic analysis may still not be enough to assess privacy protection level. What websites state in their privacy policy may not reflect what they actually do (e.g. they may collect more personal data than stated in the policy). Enforcement of policies and verification within the website domain is not possible within a completely client-side oriented solution such as ours. However, it is possible to monitor the website behavior by analyzing the client-side data flow during the transactions with the website. This is what we call dynamic analysis in Figure 4.

The problems we discussed in this section are still under investigation. The study of their feasibility, the approach to follow, the development and the evaluation is part of our current and future work.

7. CONCLUSIONS

In this paper we present a solution to evaluate privacy policies completeness by applying machine learning and text classification techniques. We argue that given the importance and complexity of privacy policies increasing their usability would give significant benefits.

We test several automatic classifiers, obtained by applying machine learning over a corpus of pre-annotated privacy policies, to prove the feasibility approach and give an indication of the accuracy that can be achieved. The results of our experiments show that it is possible to create automatic classification with a high accuracy($\sim$ 92%). This accuracy level is similar to the one obtainable with a human classifiers.

Based on the results, we can provide some directions to guide the choice of the classifier to use for the completeness

analyzer: both, LSVM and Ridge regression preform well. The probability variant of the stacking ensemble method significantly improves the F_1 scores in the testing but does require a lot more computation. Thus it represents a good choice only for environments with no computational constraints. If computational costs matter, Ridge regression and the LSVM offer a good trade-off between effectiveness and costs. Naïve Bayes, does not perform as good as the other classifiers in the final evaluation, but computational costs are 10 to 20 times less than LSVM and Ridge, making it a possible choice for settings where computational costs are very important.

8. ACKNOWLEDGMENTS

We thank Dr. Mykola Pechenizkiy for his valuable help and comments. This work has been partially funded by the EU-IST-IP-216287 TAS3 project, and the Dutch National THeCS project in the COMMIT program.

9. REFERENCES

[1] Charter of the Fundamental rights of the Union., 2000.

[2] R. Agrawal, W. I. Grosky, and F. Fotouhi. Ranking Privacy Policy. *2007 IEEE 23rd International Conference on Data Engineering Workshop*, pages 192–197, Apr. 2007.

[3] E. Andrade, V. Kaltcheva, and B. Weitz. Self-disclosure on the web: The impact of privacy policy, reward, and company reputation. *Advances in Consumer Research*, 29(1):350–353, 2002.

[4] A. I. Anton, J. B. Earp, H. Qingfeng, W. Stufflebeam, D. Bolchini, and C. Jensen. Financial privacy policies and the need for standardization. *Security and Privacy,*, 2(2):36 – 45, 2004.

[5] C. Apté, F. Damerau, and S. Weiss. Automated learning of decision rules for text categorization. *ACM Transactions on Information Systems (TOIS)*, 12(3):233–251, 1994.

[6] P. Ashley, S. Hada, G. Karjoth, C. Powers, and M. Schunter. Enterprise privacy authorization language (epal). Technical report, IBM Research, 2003.

[7] T. Ayodele. Types of Machine Learning Algorithms. In Y. Zhang, editor, *New Advances in Machine Learning*, pages 19–48. InTech, 2010.

[8] P. Beatty, I. Reay, S. Dick, and J. Miller. P3P Adoption on E-Commerce Web sites: A Survey and Analysis. *IEEE Internet Computing*, 11(2):65–71, Mar. 2007.

[9] L. Breiman. *Classification and regression trees*. Chapman & Hall/CRC, 1984.

[10] C. Brodie, C. Karat, J. Karat, and J. Feng. Usable security and privacy: a case study of developing privacy management tools. In *Proceedings of the 2005 symposium on Usable privacy and security*, pages 35–43. ACM, 2005.

[11] C. a. Brodie, C.-M. Karat, and J. Karat. An empirical study of natural language parsing of privacy policy rules using the SPARCLE policy workbench. *Proceedings of the second symposium on Usable privacy and security - SOUPS '06*, page 8, 2006.

[12] G. Brown. Ensemble learning. *Encyclopedia of Machine Learning*, pages 1–24, 2010.

[13] C. Burges. A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery*, 2(2):121–167, 1998.

[14] D. Cai, S. Yu, J. Wen, and W. Ma. VIPS: a visionbased page segmentation algorithm. Technical report, Microsoft Technical Report, MSR-TR-2003-79, 2003.

[15] E. Costante, J. den Hartog, and M. Petkovic. On-line trust perception: What really matters. In *Socio-Technical Aspects in Security and Trust (STAST), 2011 1st Workshop on*, pages 52–59, Milan, Italy, 2011. IEEE.

[16] L. Cranor and M. Arjula. Use of a P3P user agent by early adopters. In *the 2002 ACM workshop on Privacy in the Electronic Society*, pages 1–10, 2002.

[17] L. Cranor, M. Langheinrich, M. Marchiori, M. Presler-Marshall, and J. Reagle. The platform for privacy preferences 1.0 (P3P1. 0) specification. *W3C*, 2002.

[18] H. Farrell. Constructing the International Foundations of E-Commerce - The EU-U.S. Safe Harbor Arrangement. *International Organization*, 57(02):277–306, 2003.

[19] P. Guarda and N. Zannone. Towards the development of privacy-aware systems. *Information and Software Technology*, 51(2):337–350, Feb. 2009.

[20] J. Hiller. The Regulatory Framework for Privacy and Security. *International Handbook of Internet Research*, pages 251–265, 2010.

[21] A. Hoerl and R. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

[22] L.-E. Holtz, K. Nocun, and M. Hansen. Towards Displaying Privacy Information with Icons. *Privacy and Identity Management for Life*, pages 338–348, 2011.

[23] C. Hsu, C. Chang, and C. Lin. A practical guide to support vector classification. *Bioinformatics*, 1(1):1–16, 2003.

[24] A. Jø sang, L. Fritsch, and T. Mahler. Privacy policy referencing. In *Trust, Privacy and Security in Digital Business*, pages 129–140. Springer, 2010.

[25] T. Joachims. Text categorization with support vector machines: Learning with many relevant features. *Machine Learning: ECML-98*, pages 137–142, 1998.

[26] A. Kibriya, E. Frank, and B. Pfahringer. Multinomial naive bayes for text categorization revisited. *Lecture notes in computer Science*, pages 488–499, 2004.

[27] C. Kohlschütter, P. Fankhauser, and W. Nejdl. Boilerplate detection using shallow text features. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 441–450. ACM, 2010.

[28] S. Kotsiantis and D. Kanellopoulos. Data preprocessing for supervised leaning. *International Journal of Computer Science*, 1(2):111–117, 2006.

[29] L. Kotthoff, I. Gent, and I. Miguel. A Preliminary Evaluation of Machine Learning in Algorithm Selection for Search Problems. In *The Fourth International Symposium on Combinatorial Search*, pages 84–91, 2011.

[30] A. McCallum and K. Nigam. A comparison of event models for naive bayes text classification. In *AAAI-98 workshop on learning for text categorization*, volume 752, pages 41–48, 1998.

[31] OASIS. extensible access control markup language (xacml) version 2.0. Technical report, OASIS, 2008.

[32] J. Pasternack and D. Roth. Extracting article text from the web with maximum subsequence segmentation. In *Proceedings of the 18th international conference on World wide web*, pages 971–980. ACM, 2009.

[33] R. Polikar. Ensemble based systems in decision making. *Circuits and Systems Magazine, IEEE*, 2006.

[34] J. Quinlan. Induction of decision trees. *Machine learning*, pages 81–106, 1986.

[35] J. Quinlan. *C4. 5: programs for machine learning*. Morgan kaufmann, 1993.

[36] J. Reagle and L. Cranor. The platform for privacy preferences. *Communications of the ACM*, 42(2):48–55, 1999.

[37] P. Refaeilzadeh, L. Tang, and H. Liu. Cross Validation. In M. T. Ozsu and L. Liu, editors, *Encyclopedia of Database Systems*, volume 25. Springer, Jan. 2009.

[38] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523, 1988.

[39] R. Schwitter. English as a formal specification language. *Proceedings. 13th International Workshop on Database and Expert Systems Applications*, pages 228–232, 2002.

[40] F. Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47, Mar. 2002.

[41] R. Setiono. Chi2: feature selection and discretization of numeric attributes. *Proceedings of 7th IEEE International Conference on Tools with Artificial Intelligence*, pages 388–391, 1995.

[42] G. Sigletos, G. Paliouras, and C. Spyropoulos. Combining information extraction systems using voting and stacked generalization. *The Journal of Machine Learning Research*, 6:1751–1782, 2005.

[43] G. Smits and E. Jordaan. Improved SVM regression using mixtures of kernels. In *Neural Networks, 2002. IJCNN'02. Proceedings of the 2002 International Joint Conference on*, volume 3, pages 2785–2790. IEEE, 2002.

[44] J. Suykens and J. Vandewalle. Least squares support vector machine classifiers. *Neural processing letters*, 9(3):293–300, 1999.

[45] J. Tsai, S. Egelman, L. Cranor, and A. Acquisti. The effect of online privacy information on purchasing behavior: An experimental study. *Information Systems Research*, 21(June), 2010.

[46] V. Vladimir. *The nature of statistical learning theory*. Springer-Verlag, 1995.

[47] W3C. Privacy Enhancing Browser Extensions. Technical report, W3C, 2011.

[48] Z. Wang, Y. He, and M. Jiang. A comparison among three neural networks for text classification. In *Signal Processing, 2006 8th International Conference on*, volume 3, pages 1–4. IEEE, 2006.

[49] S. Weiss. *Text mining: predictive methods for analyzing unstructured information*. Springer-Verlag New York Inc, 2005.

[50] T. Weninger and W. H. Hsu. Text Extraction from the Web via Text-to-Tag Ratio. *2008 19th International Conference on Database and Expert Systems Applications*, pages 23–28, Sept. 2008.

[51] D. S. Wilks. Statistical forecasting. In *International Geophysics*, chapter 7, pages 215–300. Academic Press, 2011.

[52] D. Wolpert. Stacked generalization. *Neural networks*, 5(2):241–259, 1992.

[53] Y. Yang. A re-examination of text categorization methods. *Proceedings of the 22nd annual international ACM*, 1999.

[54] Y. Yang. An evaluation of statistical approaches to text categorization. *Information retrieval*, 1(1):69–90, 1999.

[55] W. Yu, S. Doddapaneni, and S. Murthy. A Privacy Assessment Approach for Serviced Oriented Architecture Application. *2006 Second IEEE International Symposium on Service-Oriented System Engineering (SOSE'06)*, pages 67–75, Oct. 2006.

[56] W. D. Yu and S. Murthy. PPMLP: A Special Modeling Language Processor for Privacy Policies. *2007 IEEE Symposium on Computers and Communications*, pages 851–858, July 2007.

[57] J. Zhang and Y. Yang. Robustness of regularized linear classification methods in text categorization. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 190–197. ACM, 2003.