A Machine Learning Solution to Assess Privacy Policy Completeness

(Short Paper)

Elisa Costante
TU/e
Eindhoven, Netherlands
e.costante@tue.nl

Yuanhao Sun
TU/e
Eindhoven, Netherlands
yuanhaosun2009@gmail.com

Milan Petković
TU/e & Philips Research
Eindhoven, Netherlands
milan.petkovic@philips.com

Jerry den Hartog
TU/e
Eindhoven, Netherlands
j.d.hartog@tue.nl

ABSTRACT

A privacy policy is a legal document, used by websites to communicate how the personal data that they collect will be managed. By accepting it, the user agrees to release his data under the conditions stated by the policy. Privacy policies should provide enough information to enable users to make informed decisions. Privacy regulations support this by specifying what kind of information has to be provided. As privacy policies can be long and difficult to understand, users tend not to read them. Because of this, users generally agree with a policy without knowing what it states and whether aspects important to him are covered at all. In this paper we present a solution to assist the user by providing a structured way to browse the policy content and by automatically assessing the completeness of a policy, i.e. the degree of coverage of privacy categories important to the user. The privacy categories are extracted from privacy regulations, while text categorization and machine learning techniques are used to verify which categories are covered by a policy. The results show the feasibility of our approach; an automatic classifier, able to associate the right category to paragraphs of a policy with an accuracy approximating that obtainable by a human judge, can be effectively created.

Categories and Subject Descriptors

I.2.7 [**Natural Language Processing**]: Text analysis; K.4.1 [**Privacy**]: Privacy Policy

General Terms

Security, Human Factors, Languages

Keywords

privacy, privacy policy, natural language, machine learning

WPES'12, October 15, 2012, Raleigh, North Carolina, USA.

1. INTRODUCTION

Protection of and control over one's personal data are rights recognized by the privacy regulations in many countries. The EU Charter of the Fundamental Rights clearly states that "*Everyone has the right to the protection of personal data concerning him or her*". Though organizations that handle personal data are obliged to implement 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 consent to give his personal data or not using a service at all. To make informed decisions about whether or not to share personal data, users should fully understand how data will be treated.

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 [6], preferring the easier way of unconditional acceptance. To assist the user in making informed decisions, we develop a system that automatically evaluates the completeness of a privacy policy. By measuring the level of completeness, we give the user information on whether the privacy categories of his interest are covered or not by a policy. Privacy categories are defined following privacy regulations and guidelines. Example of privacy categories can be *Data Collection* or *Data Sharing*.

Once privacy categories are defined, we use text categorization and machine learning techniques to check which paragraphs in the natural language privacy policy belong to which category, and assess the completeness level (or grade) according to which of the categories important for the user are covered by the policy. The user can then inspect the policy in a structured way, looking at only those paragraphs belonging to the categories he is interested in.

The main contribution of this paper is thus a system consisting of an automatic completeness analyzer to verify which privacy categories are covered by a policy, and accordingly assess the level of completeness. With the work presented in this paper we want to help the user to easily discover *if* privacy categories of his interested are covered by a policy: at this point the user still has to read the related paragraphs to know *how* the category is addressed (e.g. whether personal information is shared or not with third parties). Providing initial information on whether or not important categories are covered by a policy can help the user in making initial assumption on how the website takes into account his (vision) of privacy. Moreover, if we are able to detect which paragraph is related to which category, more sophisticated natural language processing techniques can be applied to explore the semantic content

of the policy. A solution to extract semantic value from privacy policies is discussed in [7].

The remaining part of the paper is organized as follows: in Section 2 we analyze the related work, in Section 3 we discuss the goals and the methodology, while in Section 4 we present the results regarding the accuracy of our approach. Finally, in Section 5 we discuss the conclusions and the further work.

2. RELATED WORK

Solutions to improve privacy protection by means of privacy policies can be categorized according to which of the two main stakeholders, the end-user (client-side), to whom the policy is addressed, or the policy authors or enforcement authorities (server-side), are considered. Approaches such as SPARCLE [4] or PPMLP [22] are intended for the server-side, while solutions such as P3P [8] are intended for both. Finally, there are approaches such as the PrimeLife Dashboard [19] that primarily focus on the client-side.

SPARCLE [4] is a privacy management workbench intended to assist an organization in the writing and enforcement of privacy policies by helping authors to create understandable policies, compliant with privacy principles. The framework takes privacy policies written in constrained natural language, i.e. a *subset* of a natural language with a restricted grammar, checks them for compliance with privacy principles, and translates them into a machine readable and enforceable format, e.g. EPAL [1] or XACML [13].

PPMLP [22] also aims to help policy authors in generating privacy policies, making them compliant with the privacy regulations. In this case the authors specify a meta-privacy policy, that is then translated in a template of rules for the enforcement. Once the meta-policy is ready, it is translated both in EPAL, for its enforcement, and in natural language, for the presentation to the end-user.

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 [8]. A limitation of the P3P, shared by SPARCLE and PPMLP, is that it needs server-side adoption, which is not easily obtained: according to [2] only the 20% of the websites amongst the E-Commerce Top 300 is P3P enabled.

The PrimeLife Privacy Dashboard [19] is a 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 policy increases the rate, while the presence of cookies decreases it. 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.

3. COMPLETENESS ANALYZER

The main goal of the completeness analyzer is to assign a completeness grade to a privacy policy. To denote the completeness level of a privacy policy p, we use $G_c(p)$ which we compute as follows:

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

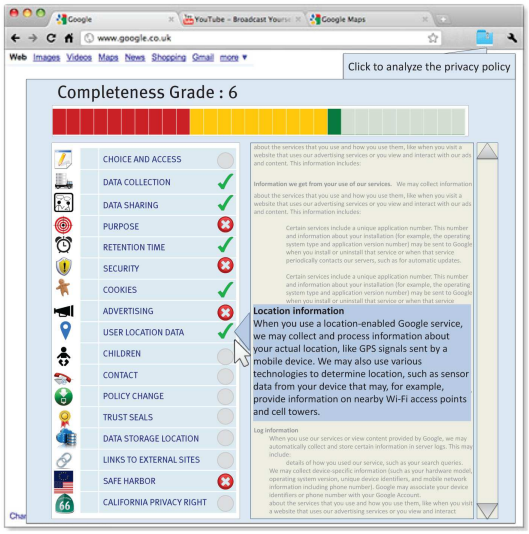


Figure 1: The completeness analyzer user interface.

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 parses it, and classifies each of its paragraphs, computing computes its grade. Figure 1 shows how the results of the completeness analyzer may be visualized. 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 relevant* for the user (gray circle; $w_i = 0$). Additionally, the user can select one of the covered categories, and browse the related text in the policy: in the figure the user is browsing the category *User Location Data*.

To assess the value of c_i 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 [17]. In our context, the pre-classified documents are paragraphs from manually labeled privacy policies (a.k.a. corpus). The corpus is used to train a classifier to correctly assign a privacy category to the paragraphs of an unlabeled policy.

To build and evaluate the text classifiers, we extracted the privacy categories from privacy regulations (Section 3.1), and we manually labeled each paragraph of the policies forming the corpus (Section 3.2) with one of such category. The labeled corpus is then transformed into a suitable representation by applying standard preprocessing techniques (Section 3.3). During the learning phase, different classifiers are built according to different machine learning algorithms (Section 3.4). The performance (Section 4.1) of the classifiers is validated using the *k-fold cross-validation* technique [15] (Section 4.2). Finally, different settings and optimization techniques are applied to find the best performing classifier(s) (Section 4.3).

[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.

3.1 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), and the Safe Harbor Framework [9]. 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*. Although they are not mandatory, the principles in the legislations mentioned above provide guidance for drafting privacy policies. In defining the categories we also accounted for common practice, i.e. topics that are usually addressed by privacy policies such as *Advertising*, *Policy Change*, etc. Below we briefly describe the privacy categories used in our settings. Note that categories can be divided in two types: *Core Categories*, directly adapted from the EU directives and the OECD guidelines, and *Additional Categories*, representing more specific information we believe are useful to the users, but that can overlap with other categories, e.g. *User Location Data* sometimes overlaps with the *Data Collection* category.

Core Categories.

- **Choice and Access** provides information about the user's privacy choices, such as opt-in/opt-out options, and user's rights to access or modify the information collected by the website.
- **Data Collection** explains how and what kind of personal information may be collected by the website.
- **Data Sharing** explains whether, and under which conditions, the website will share user's information.
- **Purpose** explains for which purposes the data will be collected.
- **Retention Time** explains for how long personal data is retained.
- **Security** explains whether security technologies, e.g. SSL or access control policies, are applied by the website.

Additional Categories.

- **Cookies** explains whether the website makes use of cookies, and the information the cookies store.
- **Advertising** explains how user's data is used for advertisements, and whether it is managed by third parties.
- **User Location Data** explains how the website manages user's location information.
- **Children** explains the company's policy regarding the collection and use of personal information of children.
- **Contact** provides company's contact information, such as address or phone number, useful for questions or complaints.
- **Policy Change** explains how updates to the privacy policy are managed, and whether and how users will be informed of that.
- **Trust Seals** explains whether the website has been awarded with trust seals (e.g. TRUSTe), signifying the website's policy is compliant with trusted third party's requirements.
- **Data Storage Location** explains where the personal data is transferred to, stored and processed (e.g. whether it goes from Europe to USA, where a different privacy legislation applies).
- **Links to External Sites** explains whether the website contains links to external sites, not covered by the current privacy policy.
- **Safe Harbor** explains the website participation in, and compliance with, the Safe Harbor Framework.
- **California Privacy Rights** is specific to California residents; it defines if (and how) users can request the records of the personal data disclosed to third parties.

Users can decide what categories are relevant to them. Those from outside California will likely not be interested in the *California Privacy Rights*. That category, as well as e.g. the category *Safe Harbor*, could be (automatically) discarded for websites that are not located in the USA. Discarded categories are not taken into account while computing the final completeness grade; therefore an European website lacking the *Safe Harbor* category, can still potentially reach the maximum grade available. Also, our approach allows the definition of new categories, e.g. as consequence of changes in legislation like the upcoming EU Data Protection Regulation.

3.2 The Corpus

The corpus is an initial set of paragraphs extracted from privacy policies, labeled with the categories described before, and used during the learning phase to build the classifier. The corpus is divided in two subsets: the training set ($\sim 70\%$), used to train and validate the classifier, and the testing set ($\sim 30\%$), used as a final evaluation of the classifier. The two subsets are strictly separated, i.e. the testing set is kept apart from the training set and learning process, so that a completely fresh set is available for the final evaluation.

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 has been extracted from 64 privacy policies 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.

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 independently labeled a selected subset consisting of 102 paragraphs extracted from the training set. The two annotators agreed in the 91% of the cases, leading us to conclude the corpus annotation is sufficiently reliable, but also that in some cases human classifiers can disagree on the classification of paragraphs, something that needs to be kept in mind when analyzing the results of the automated classifiers.

3.3 Preprocessing

Data preprocessing consists of applying a set of techniques, such as cleaning, normalization, and transformation to the ('raw') corpus to eliminate noise that could decrease the quality of the learnt classifier [10]. Using the resulting ('polished') corpus 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, while the bag-of-words transformation is done applying the tf-idf weighting [16]. We also apply feature selection, an optimization technique enabling learning algorithms to operate faster and more effectively by removing irrelevant and redundant information [10]. The impact this technique has in our settings is discussed in Section 4.3.

3.4 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 func-

tion 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 may represent e.g. a probability, or a measure of vector closeness depending on the algorithm used [17].

Since the algorithm selection problem, i.e. the problem of selecting the algorithm that performs best for a given task, is still unsolved [11], we test the performance of different algorithms to determine which one achieves the best results, and whether such results are satisfactory. The algorithms we selected are the Naïve Bayes (NB) with the multinomial event model, the Linear Support Vector Machine (LSVM), the Ridge Regression (Ridge), the k-Nearest Neighbor (k-NN), the CART version [3] of the Decision Tree (DT), and the Support Vector Machine (SVM) with non-linear RBF Kernel. We selected such algorithms based on some characteristics important in our settings, such as computational efficiency, or good accuracy in text classification tasks.

We also applied ensemble learning methods trying to build more effective classifiers by combining (the outcome) of different algorithms. Ensemble learning [14] is an optimization technique that generates a classifier as a combination of others. Several ensemble techniques can be used, here we test the *voting committee* [5], and two variants of the *stack generalization* [21]: the *probability variant* (Prob.), and the *prediction variant* (Pred.) [18].

4. EVALUATION

The evaluation task aims to validate that classifiers of adequate effectiveness have been created. Different iterations are needed to find effective classifiers and the results in each iteration are a guide to further improvements. In this section we describe how we measure the effectiveness of a classifier and the results we obtained in our experiments.

4.1 Metrics

The effectiveness of a classifier is usually computed in terms of the information retrieval notions of precision and recall [17]. 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. 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.

4.2 Cross Validation

In the experiments described below, we use the *k-fold cross validation* technique [15] to select the best performing classifiers and estimate their performance. Typically, the k-fold cross validation consist of partitioning the corpus into k 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. In our experiments we apply the $n \times 10$-fold cross validation [20], i.e. the 10-fold cross validation is repeated n-times, every time using a different random selection of folds from the training set. The F_1 score and the error are then estimated respectively as average of the values obtained at each repetition. Note that the cross-validation method does not produce any single classifier, but an estimation of the average performance of classifiers learned from sub-sets of the training set. The performance (F_1 score) obtained from cross-validation is used as an informal estimation of the performance of a final classifier[2].

4.3 Experiments

Following we describe the experiments we carried out to evaluate the performance of the classifiers, and the impact of different configuration settings and optimization techniques. Note that in the first experiments (a-e) the privacy category *Purpose* is left out, while it is introduced in the last experiments (f-g): this allows us to discuss the impact that adding a new category has on our model.

(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 selecting the value that leads to the best F_1 score. Figure 2(a) shows the parameters we tuned, the domain that was considered, and the value leading to the best performance. For readability reasons, some of the parameters are expressed in $\log_2$. In the following experiments, unless stated differently, the parameters are set to their *Best value*.

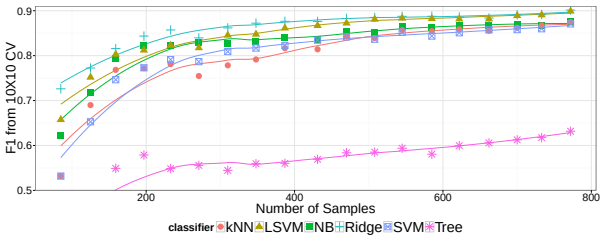
(b) **Impact of the Sample Size**: Choosing the training set size is a compromise between the effectiveness achievable by the classifiers and the effort involved in building the training set. If the training set is too small this may significantly degrade the effectiveness of a classifier. On the other 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 with an increasing step size of 5%. The final subset consists of the whole training set. The F_1 score of our classifiers is computed for each subsets, using the 10×10-fold cross validation. By looking at the trend of the classifiers performance, Figure 2(b), 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 for our settings. Figure 2(b) 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 ($F_1 = 63.1\%$) is performing very poorly and, for this reason, is discarded from further experiments.

(c) **Impact of Feature Selection**: In this experiment, we evaluate the performance of the classifiers when applying the χ^2 feature selection method [12], as shown in Figure 2(c). The results indicate 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 60% 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

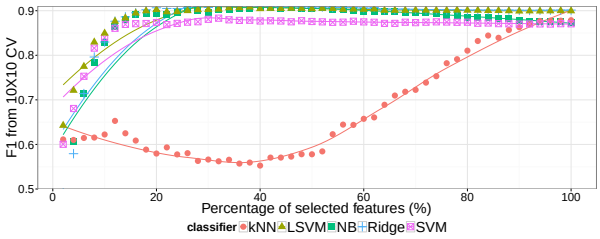
[2] `http://cseweb.ucsd.edu/~elkan/250B/classifiereval.pdf`

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

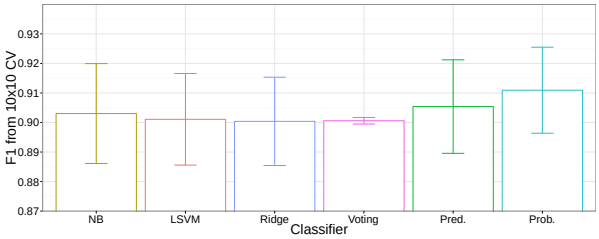
(a) Parameter Tuning



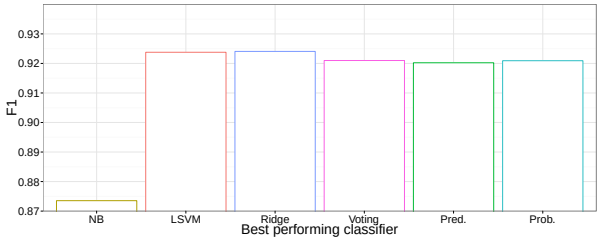
(b) Impact of the Sample Size



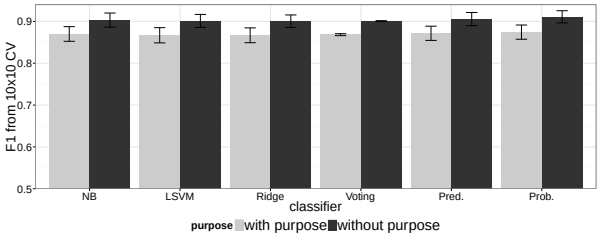
(c) Impact of Feature Selection



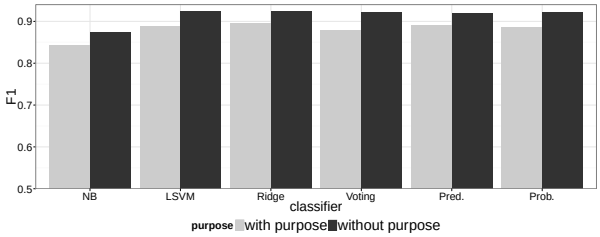
(d) Impact of Ensemble methods



(e) Evaluation over the testing set



(f) Impact of adding a new category (Purpose): Cross Validation



(g) Impact of adding a new category (Purpose): Evaluation over the testing set

Figure 2: Experimental Results

the fact that for that value, all classifiers (except for k-NN) have already reached their best performances, and NB performance is not yet decreasing.

(d) **Impact of Ensemble methods**: In this experiment we compare the performance of the best single classifiers (NB, LSVM and 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. Figure 2(d) shows the average mean and error obtained by applying a 10×10-fold cross validation and a 40% feature selection. SVM and k-NN do not appear in the figure because of their low performances with 40% feature selection ($\sim 55\%$ for k-NN, and $\sim 87\%$ for SVM). We can notice that the probability variant (Prob.) of the stacking ensemble outperforms the best single classifier (NB) with a F_1 score that is about 1% higher (91.1% versus 90.3%). However, all the classifiers have performances between 90-91%, that does not allow to prefer one over the others. Note that the voting ensemble is the one presenting less variance that is usually appreciated when choosing a classifier. On the other side, ensemble methods are intuitively more computationally expensive than single classifiers, simply because they use several of them, and not necessarily in parallel. Therefore, if the gain in performance is not relevant, single classifiers have to be preferred.

(e) **Evaluation on the testing set**: The evaluation over the testing set aims at providing an estimation of the classifier performance in real scenarios. This test has been done at the very end, when all the classifiers were fully specified. The test checks

there is no over-fitting problem, i.e. that the classifiers are not too specialized toward the training set and can therefore correctly classify other data. In case of over-fitting the results on the testing set would be significantly less than on the training set. Figure 2(e) presents the results of the test: all the classifiers (but the NB) show very similar performance ($\sim 92\%$), better than during the validation phase. This suggests they do not suffer the over-fitting problem, at the contrary of the NB classifier that should be therefore discarded.

(f) **Impact of adding a new category**: In this test we check the impact that adding a new category has on our model, in terms of costs and performance. To perform this test we added 50 new paragraphs to the corpus (35 to the training set and 15 to the testing set) regarding the category *Purpose*. We chose to add the category *Purpose* at the very end, mostly because of the difficulty we encountered during its manually labeling. Since paragraphs regarding *Purpose* are difficult to isolate (they are often interleaved with other categories such as *Data Collection* or *Retention*) the performances the system has when *Purpose* is included, are likely to be a lower bound for our system. Figure 2(f) shows the results of the cross validation test when *Purpose* is included and when it is left out. We can notice that the best classifiers continue to be the best even when the new category is added, suggesting that the choice of the classifier is independent from the categories. However, adding *Purpose* leads to a decrease in performance of about 3.6%, confirming that *Purpose* is indeed a category difficult to annotate. Figure 2(g) shows the results obtained over the testing set and compares them to the one obtained when *Purpose* is not included. The trend is again confirmed, with the NB classifier performing poorly and the others showing similar F_1 score, between 89-90%. These results show that, if required, new categories can be added to the systems. The costs in terms of time depend on the number of new samples added to the corpus, while the costs in terms of performance seems to be limited to a decrease of about 3%.

5. 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, being able of automatically evaluate their completeness would give significant benefits to the end users. We test several automatic classifiers, obtained by applying machine learning over a corpus of pre-annotated privacy policies, to prove the feasibility of our approach. The results show that LSVM, Ridge, Voting, Pred. and Prob. have comparable performance. There is, however, added computational costs associated to the use of ensemble methods (they costs approximately 10 times more than single classifiers). Since single classifiers are less computationally expensive, they represent a good option for our settings. 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 87% effectiveness is considered sufficient. However, NB suffers of the over-fitting problem, so it could lead to lower performance. 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. However, study to increase the overall effectiveness of the classifiers are left as future works, as well as study on the impact of applying techniques to deal with interleaving categories, such as the multi-labeling.

6. ACKNOWLEDGMENTS

We thank Dr. Mykola Pechenizkiy for his valuable help and comments. This work has been partially funded by the Dutch National THeCS project in the COMMIT program.

7. REFERENCES

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

[2] 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), 2007.

[3] L. Breiman. *Classification and regression trees*. Wadsworth International Group, 1984.

[4] C. Brodie, C. Karat, J. Karat, and J. Feng. Usable security and privacy: a case study of developing privacy management tools. In *Proc. of SOUPS*, 2005.

[5] G. Brown. Ensemble learning. *Encyclopedia of Machine Learning*, 2010.

[6] E. Costante, J. d. Hartog, and M. Petkovic. On-line Trust Perception: What Really Matters. In *Proc. of STAST*, 2011.

[7] E. Costante, J. d. Hartog, and M. Petkovic. What Websites Know About You. In *Proc. of DPM*, 2012.

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

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

[10] S. Kotsiantis and D. Kanellopoulos. Data preprocessing for supervised leaning. *IJCSI*, 1(2), 2006.

[11] L. Kotthoff, I. Gent, and I. Miguel. A Preliminary Evaluation of Machine Learning in Algorithm Selection for Search Problems. In *Proc. of SoCS*, 2011.

[12] H. Liu and R. Setiono. Chi2: feature selection and discretization of numeric attributes. In *Proc. of ICTAI*, 1995.

[13] OASIS. eXtensible Access Control Markup Language (XACML) version 2.0. Technical report, OASIS, 2008.

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

[15] P. Refaeilzadeh, L. Tang, and H. Liu. Cross-validation. In *Encyclopedia of Database Systems*. Springer, 2009.

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

[17] F. Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1), 2002.

[18] G. Sigletos, G. Paliouras, and C. D. Spyropoulos. Combining Information Extraction Systems Using Voting and Stacked Generalization. *JMLR*, 6, 2005.

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

[20] D. S. Wilks. Statistical forecasting. In *International Geophysics*, chapter 7. Academic Press, 2011.

[21] D. Wolpert. Stacked generalization. *Neural networks*, 5(2), 1992.

[22] W. Yu, S. Doddapaneni, and S. Murthy. A Privacy Assessment Approach for Serviced Oriented Architecture Application. In *Proc. of SOSE*, 2006.