What Websites Know About You*

Privacy Policy Analysis Using Information Extraction

Elisa Costante[1] and Jerry den Hartog[1] and Milan Petković[1,2]

[1] Eindhoven University of Technology, The Netherlands
{e.costante, j.d.hartog, m.petkovic}@tue.nl
[2] Philips Research Europe, High Tech Campus, The Netherlands
milan.petkovic@philips.com

Abstract. The need for privacy protection on the Internet is well recognized. Everyday users are asked to release personal information in order to use online services and applications. Service providers do not always need all the data they gather to be able to offer a service. Thus users should be aware of what data is collected by a provider to judge whether this is too much for the services offered. Providers are obliged to describe how they treat personal data in privacy policies. By reading the policy users could discover, amongst others, what personal data they agree to give away when choosing to use a service. Unfortunately, privacy policies are long legal documents that users notoriously refuse to read. In this paper we propose a solution which automatically analyzes privacy policy text and shows what personal information is collected. Our solution is based on the use of Information Extraction techniques and represents a step towards the more ambitious aim of automated grading of privacy policies.

1 Introduction

Protection of online privacy is lately attracting a lot of attention. Concerns about the ease at which excessive collection of personal data can take place online are reflected in research, legislation and public opinion. Within the new data protection directives, the European Commission has identified enhancing the control a user has over his personal data as a key objective [1]. Privacy breaches regularly appear in popular media along with suggestions for tools to protect users during their browsing [2]. On the other hand the data collected from thousands of customers represents a big economical asset for companies, one they would not easily give up [1].

Surveys show that also users are deeply concerned about privacy [3], and that they would be prone to pay some money to shop on websites with good privacy policies [3, 4]. However, there are some misconceptions about privacy policies, since users think that the sole presence of a privacy policy means the website will protect, and will not share, their personal data [5]. In reality, privacy policies often serve more as liability disclaimers for service providers than as assurances of privacy for end-users [6]. Moreover, since policies are often ignored by users, they fail in their goal of increasing privacy awareness.

* This work has been partially funded by the THeCS project in the Dutch National COMMIT program.

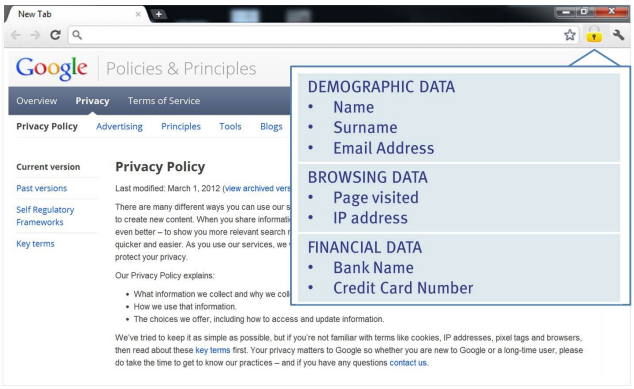

Fig. 1. An example of how the list of personal data collected can be presented to the user.

In this paper we describe a solution, based on Information Extraction (IE) techniques, to extract the list of data collected by a website, according to what it is stated in its privacy policy. We believe that presenting such information to the user, for example using a browser extension as depicted in Figure 1, will increase his privacy awareness, and will help him to take more informed decisions. Without having to read the complete privacy policy, the user can see what data is collected, and can judge whether or not the information required is justified by the service offered (e.g. the Social Security Number is required by an e-commerce service). Moreover, solving the problem of automatically extracting useful information from privacy policy is a step towards being able to rate a website, based on the contents of its privacy policy.

The work described in this paper is, indeed, part of a larger framework which aims at automatic grading the privacy protection offered by a website. This grading is intended to be done by considering several aspects of privacy management, starting from the privacy policy [7]. In this paper we focus on analyzing the contents of a policy, namely the part regarding data collection. The same approach, once has been shown effective, can be applied to extract other information, such as the sharing practices or the security mechanisms applied, useful to verify how good a website protects the privacy of its users.

The rest of the paper is organized as following: we first discuss related work in Section 2, and employed extraction methods and tools in Section 3, before presenting the process followed to build our system in Section 4. The results are described in Section 5, while conclusion and further work are addressed in Section 6.

2 Related Work

Understanding privacy policies, often complex and ambiguous, is not an easy task for end-users [8, 9]. Privacy policies represent an important resource for privacy protection since they describe how personal data is managed. Several approaches, aiming at improving privacy protection by the means of privacy policies, exist. Some of them, such as SPARCLE [10, 11] and PPMLP [12, 13] are intended for privacy policy authors,

while others, such as P3P [14] or UPP [15], are directed to both actors: privacy authors and end-users. Finally, approaches like the PrimeLife Dashboard [16] mainly focus on supporting the end-user.

SPARCLE (Server Privacy ARchitecture and CapabiLity Enablement) [10, 11] is a framework intended to assist an organization in the writing, auditing and enforcement of privacy policies. The framework takes privacy policies written in a specific constrained natural language, checks for their compliance with privacy principles, and translates them into a machine readable and enforceable format, e.g. EPAL [17] or XACML [18]. 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 [19], makes it possible to parse such policies.

PPMLP (Privacy Policy Modeling Language Processor) [12, 13], like SPARCLE, aims at helping organizations in generating privacy policies compliant with privacy principles: authors specify a meta-privacy policy that will automatically be enriched with rules allowing such compliance. The meta-policy is then translated in EPAL rules, ready for the enforcement. The meta-policy is also translated in natural language (using static matching rules), for the presentation to the end-user. Within the system, a PPC (Privacy Policy Checker) is also present, to allow the users to verify whether a website enforces its privacy policy.

P3P (Platform for Privacy Preferences) [14] is the W3C's attempt to manage privacy policies, allowing the website to express them in a XML-based machine-readable format. The user is able to automatically check those policies against his preferences, by the means of P3P-enabled browsers [14]. Privacy Bird [20] and Privacy Finder [4] are examples of P3P user agents, able to compare P3P policies with users preferences, thus users do not need to read the privacy policies of every web site they visit [21].

UPP (User Privacy Policy) [15] is an approach similar to P3P, but mainly focused on social network websites. The mechanism allows a user to define policies to protect his resources (e.g. pictures or videos). Other users (his *friends*) can access such resources only if they guarantee the enforcement of the policies.

A limitation of the P3P approach, shared by SPARCLE, PPMLP and UPP, is that it needs server-side adoption, which is not easily obtained: according to [22] only 20% of the websites amongst the E-Commerce Top 300 is P3P enabled.

The PrimeLife Privacy Dashboard [16] is a recent browser extension aiming at helping the user to track what information is collected by the websites he visits. 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 then provides a visual 'privacy quality' rating of a website: the presence of a P3P version of the privacy policy increases the rating, while the presence of external or flash 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. Also, the content of the privacy policy, if it does exist, is not taken into account.

The solution presented in this paper, like the PrimeLife Dashboard, aims at letting the user know what personal data a website collects. However, it only requires the exis-

tence of a plain text privacy policy and no P3P version is needed. As such, it can easily be adopted by privacy-concerned end users, without requiring server-side adoption.

3 Methodologies and Tools

We use *Information Extraction* (IE) to extract the list of data collected by a website, by analyzing what is stated in its privacy policy. IE is a technique for analyzing natural language texts, to extract relevant pieces of information [23]. The analysis takes raw text as input and produces fixed-format, unambiguous data as output [24].

An IE system applies IE techniques to a specific scenario or domain. To build an IE system, an in-depth understanding of the contents of texts is needed [25]. IE systems have the advantage of accounting for the semantic contents of the text, rather than only for the presence/absence of given key words as it happens e.g. in Information Retrieval. Taking semantics into account gives the potential for systems that are accurate enough to really help people, reducing the time they need to spend reading texts [24].

Privacy policies, due to their legal nature, show strong formality and fixed patterns. Because of this, applying IE techniques to privacy policy text may lead to high accuracy, as confirmed by the results presented in Section 5. The fact that IE only extracts information on/in a-priori selected subjects/format fits well with our idea of showing to the user only specific information, i.e. the list of data collected.

The general architecture of an IE system may be defined as"*a cascade of transducers or modules that, at each step, add structure to the documents and, sometimes, filter relevant information, by means of applying rules*" [25, 26]. Figure 2 describes the architecture of our IE system, showing the cascade of modules we used. The raw privacy policy is given as input to the *Tokenizer* that splits the text into very simple tokens, to which a type (i.e. number, punctuation or word) is associated. The *Sentence Splitter* divides the text into sentences, while the *POS Tagger* annotates each word with the related part-of-speech (POS) tag. A POS tag specifies whether a word is a verb, a noun or another lexical category[3]. The *Named Entity (NE)* recognition module seeks the text for concepts of the application domain, while the *Annotation Patterns Engine* provides the means to define extraction rules, needed to detect the information wanted, according to specific syntactic-semantic patterns.

Note that, as described in Figure 2, at each stage a new *Annotation Set* is added to the document. *Annotations* can be seen as meta-data attached to part of the text giving specific information. For example POS tagger annotations can state that the word '*and*' is a *conjunction*. Every annotation has a type (e.g. *conjunction*) and a value (e.g. *'and'*), and belongs to an *Annotation Set* (e.g. POS Annotation Set) that groups together similar annotations. Annotations Sets that are output of earlier modules, can be used as input of later stages.

The structure shown in Figure 2 is a basic cascade of modules. Such a cascade can be extended by adding other modules for language processing. For example, co-reference and anaphora resolution [27] could be added to verify whether two noun phrases refer to the same entity [28], e.g. '*email address*' and '*it*' in the sentence "*We store your **email***

[3] The complete list of POS tags we used is available at `http://gate.ac.uk/sale/tao/splitap7.html#x37-729000G`

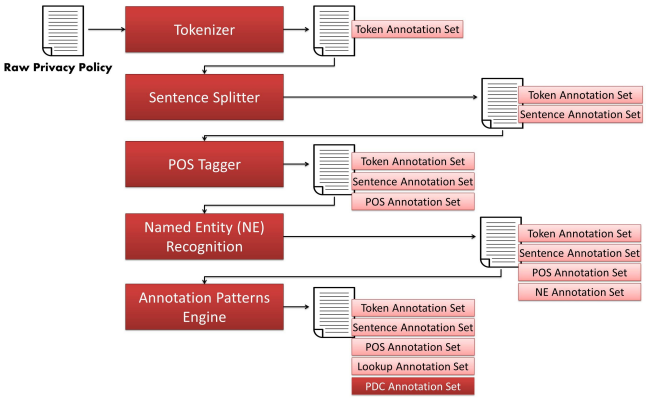

Fig. 2. The cascade of modules used in our IE system.

address, ***it*** *will be used to inform you that your order has been shipped*". The use of ontologies, which associate a concept with the terms expressing it [23], may enrich the domain knowledge. Finally, syntactic parsing could help to find the main constituents of a sentence such as noun, verbs, and prepositional phrases. The decision of whether or not to add one or more of these modules is a trade off between the increased accuracy they can grant and the added computational costs they will bring. Since we aim at building a system with a good response time, we only use the modules presented in Figure 2 in this paper. The benefits and costs of adding other modules are discussed in Section 6, as part of our further works.

To develop our system we used the GATE (General Architecture for Text Engineering) framework [29, 30], and the ANNIE (A Nearly-New Information Extraction System) [31] suite. These tools, amongst other functionalities, make available multiple implementations of the modules depicted in Figure 2.

4 The Process

The process we followed to build and evaluate our IE system is described in Figure 3. In this section we give details for activities of this process such as the definition of Named Entity (Section 4.1), the selection, annotation and splitting of the Corpus (Section 4.2), and the creation of extraction rules (Section 4.3). The evaluation and the results are discussed in Section 5.

4.1 Named Entities

Information extraction uses Named Entities (NEs) to represent important concepts within a certain domain[4]. For example, in the domain of novels, the *author*, the *title*, and the *characters* are important concepts that can be modeled as NEs.

[4] A discussion about the difficulties of defining *Named Entity* is available at `http://webknox.com/blog/2010/09/named-entity-definition/`

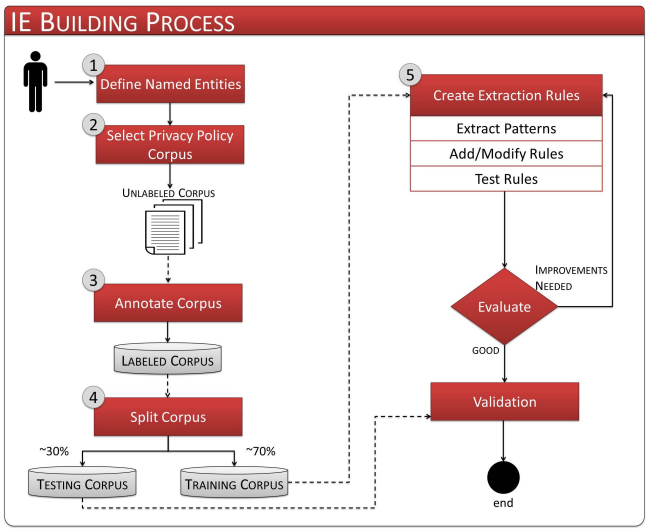

Fig. 3. IE building process.

Defining NEs requires a deep knowledge of both the application domain and the type of information to be extracted. Since we aim at extracting the list of personal data collected by a website, we need entities modeling concepts of this scenario. To select our named entities we analyze several privacy policies. In this way, we discovered that the description of data collection practices is usually done in one of the following ways:

1. The policy states the website may require user's personal data for some purposes, e.g. registration or shipping;
2. The policy states the user may provide his personal data to the website, e.g. by filling in profile information;
3. The policy states the website may use automatic tools to get users' personal data, e.g. cookies to track user's on-line behavior.

This knowledge leads to choose the NEs presented in Figure 4 as part of our system. The **Data Provider** represents the data owner, generally the user, that provides his personal information; the **Data Collector** refers to the website (or the company behind it) that collects the personal information; while the **Collection Tools** is used to describe web technologies, such as cookies or web beacons, used to track users' online activities. The NE mentioned so far can be seen as *subjects* of the *actions* leading to data collection. The actions related are respectively modeled as **Provider Action**, including verbs such as *provide* or *supply*; **Collector Action**, with verbs such as *request, collect, use, store*; and **Tools Action** including verbs describing tracking activities, such as *track* or *monitor*. The core entity of our domain is the **Personal Data Collected** (or **PDC**), actually divided into two sub-entities: **Generic Data** to refer to general concepts of personal information such as *contacts information*, *personal identifiable information*, *browsing information*; and **Specific Data** to refer to personal data units such as *name*, *surname*,

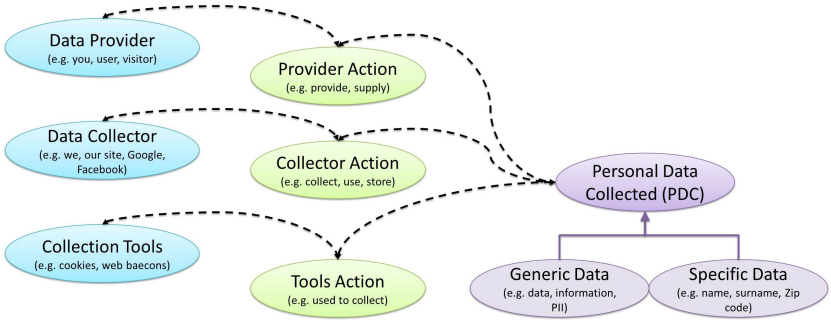

Fig. 4. The Named Entities (NEs) of our system.

or *nationality*. Note that the list of PDC is the one that will be finally presented to the user.

Once the list of NEs has been defined, it is necessary to populate it by adding instances to the concepts, e.g. to list *name*, *surname* and *address* as instances of the entity *Specific Data*, and *you* and *user* as instances of *Data Provider*. The population process is performed by creating files listing instances for each NE (the so called gazetteer lists).

Note that, as suggested in [6, 32], while listing the instances of *Specific Data* we consider as PDC any data belonging to an individual, i.e. his PII (Personal Identifiable Information such as *name*, *surname* or *birthdate*), but also tracking data related to him such as the *current GPS location*, the *clickstream*, or the *friend list*.

4.2 Corpus

A corpus is a set of documents forming the core of an IE system. Its importance is twofold: its analysis drives the acquisition of knowledge necessary to extracting patterns and creating rules, and, once annotated, it is used to test the accuracy of the IE system. According to EU directives (e.g. EU 95/46/EC and the EU 2006/24/EC), privacy policies have to address specific topics of interest for the end user, such as what data they collect, how long they retain the data, whether they share it and so on. Each topic is usually discussed in specific paragraphs of the policy, leading to documents with fixed and similar structures.

In our system we use a corpus containing two types of document: **corpus A** is the part containing 128 paragraphs extracted from the *collection section* of different privacy policies; while **corpus B** is the part formed by 12 complete privacy policies. The use of corpus A helped us to focus on modeling recurrent patterns used to describe collection practices. On the other hand, the use of corpus B lets us test the feasibility of our approach in terms of satisfactory response time, and verify that the accuracy of the system does not decrease when it is applied to complete policies, i.e. extending the analysis to other sections of the policy does not introduce too many false positives.

The privacy policies of our corpus have been collected from websites of different application domains such as e-commerce (e.g. eBay, Paypal, Amazon), searching (e.g. Google, DuckDuckGo), social networking (e.g. Facebook, LinkedIn), and news and

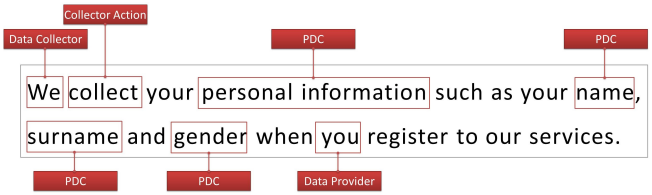

Fig. 5. Example of text annotation.

communities (e.g. WordPress, FileTube, and TripAdvisor). Dealing with different application domains is especially useful to enrich the gazetteer lists. For example, by analyzing social network policies, new instances of *Specific Data*, such as *profile*, *friends list* or *profile picture*, can be detected.

Once the corpus has been selected, and named entities defined, the annotation task can take place. During the corpus annotation, a human annotator reads every document, and tags each occurrence of NE instances. For example, in a document with the sentence "*We collect your personal information such as your name, surname and gender when you register to our services*", '*we*' will be annotated as Data Collector, '*collect*' as Collector Action, and so on, as described in Figure 5. It is important to note that instances of Specific Data (SP) and Generic Data(GD) are annotated as PDC only if the text makes clear that such data will be retained by the website. For example, in a sentence like "*If you do not want to receive e-mail from us, please adjust your preferences*", the term *e-mail* will not be annotated as PDC.

In the following, we refer to the manual annotations as *Standard Gold Set*, and to the annotations resulting by running our IE system as *Response Set*. To evaluate the accuracy of the system, PDC annotations of the *Response Set* will be compared to PDC annotations of the *Standard Gold Set*. Intuitively, the more similar the two sets are, the better the accuracy of the system.

Once the annotation task is completed, each sub-corpus is divided into a training ($\sim$ 70% of the documents of the sub-corpus) and a testing (the remaining $\sim$ 30%) set. After the splitting, corpus A contains 87 paragraphs in the training set, and 34 in the testing set, while corpus B contains 9 complete policies in the training set, and 3 in the testing set. The training sets are used to develop the extraction rules, while the testing sets are kept apart until the very end, when they are used to measure the system's accuracy. The training set of B is especially useful to verify whether patterns extracted by analyzing A do not occurs in other sections of a complete policy as well.

4.3 Extraction Rules

Extraction rules are used to detect specific regularities and patterns in the text. To define such rules, specific declarative languages can be used [33–35]. To develop our IE system we used the Jape language [35]. Jape provides mechanisms to recognize regular expressions in annotations made over a document. For example, it allows to create rules that annotate as a *Person* a word that starts with a capital letter, and that is preceded by the word *Mr*, *Mrs* or *Miss*.

Jape's extraction rules consist of two components: a Left Hand Side (LHS), representing the condition, expressed in form of pattern, and a Right Hand Side (RHS), representing the conclusion, i.e. the action to take once the pattern matches.

Recall that the whole idea behind the development of our system is that privacy policies share a set of fixed patterns. Table 1 presents the patterns that we detected during our analysis. In the table, each pattern is expressed as a sequence of *NE annotations*, but also *Token* and *POS annotations* (like *MODAL*, *SUCH_AS*, *TO*, *INCLUDE*), created by earlier steps of the process (see Figure 2). Moreover, for each pattern, one or more matching sentences are presented. For example, the pattern n.1 of Table 1 will match every sentence where the website is asking the user to provide personal information.

Table 1. List of the *collection* patterns detected by analyzing our corpus.

	Pattern
n.1	**DATA_COLLECTOR** *MODAL* **COLLECTOR_ACTION DATA_PROVIDER** *TO* **PROVIDER_ACTION** *your* **GENERIC_DATA** (*SUCH_AS*)? (**SPECIFIC_DATA**)*. We may ask you to provide your personal data such as name, surname, and gender. The website may request you to provide your financial information.
n.2	**DATA_COLLECTOR** *MODAL* **COLLECTOR_ACTION** *your* **GENERIC_DATA** (*SUCH_AS*)? (**SPECIFIC_DATA**)*. We may collect your personal data such as name, surname, and gender. Google may gather your contact information.
n.3	**DATA_PROVIDER** *MODAL* **PROVIDER_ACTION** *us with your* **GENERIC_DATA** (*SUCH_AS*)? (**SPECIFIC_DATA**)*. You must provide us with your personal data including your name, surname, and gender. The user should supply his personal information.
n.4	*The* **GENERIC_DATA DATA_COLLECTOR COLLECTOR_ACTION** *MODAL INCLUDE your* (**SPECIFIC_DATA**)*. The personal data we collect may include your name, surname, and gender.
n.5	**COLLECTION_TOOLS** *MODAL BE_USED TO* **TOOLS_ACTION** *your* **GENERIC_DATA** (*SUCH_AS*)? (**SPECIFIC_DATA**)*. Cookies may be used to track your browsing data such as your IP address, browser type and pages visited. Web beacons may be used to monitor your browsing activities.

?: zero or one repetition

*: zero or more repetition

italic: terms not relevant for the pattern

ITALIC: temporary annotations

BOLD: named entities

Underline: examples of matching sentences

Once patterns have been defined (LHS), it is necessary to create the output (RHS). Figure 6 describes how a pattern can be translated in a Jape rule, by referring to the first pattern of Table 1. The LHS describes the pattern, and when a match occurs, the RHS is called to annotate *specific* and *generic data* as PDC. Note that in Jape the symbols ?, * and + means respectively zero or one, zero or more, and one or more occurrences.

For each pattern of the table, one or more rules have been created. We created rules accounting for positive and negative (‘*we collect*’ versus ‘*we do not collect*’), and for active and passive (‘*we collect your data*’ versus ‘*your data are collected*’) forms of the patterns. Of course a negative pattern indicates the specified personal data is not collected, so it is not added to the *Response Set*. Moreover, rules have been created in such a way that tokens not relevant for the pattern are ignored (e.g. in a sentences like ‘*we ask you to provide your personal data*’, the presence of the token *your* does not stop the rule to match).

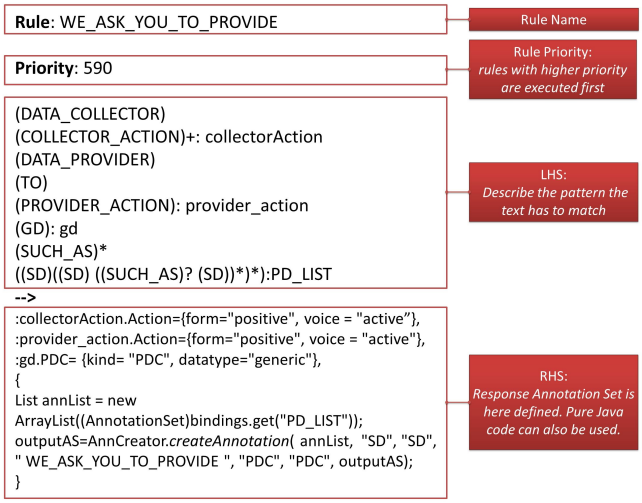

Fig. 6. An example of a JAPE extraction rule.

5 Evaluation and Results

In this section we discuss the methodology used to evaluate the accuracy of the IE we developed, and the results we obtained. We evaluate our IE system by measuring the recall, the precision, and the F_1 score obtained by comparing the *Response Set* to the *Standard Gold Set*.

Recall and precision can be computed using *strict* or *lenient* measures. Such measures are different for the way they treat partial matches. A partial match is obtained when the result of the *Response Set* does not completely match the one of the *Standard Gold Set*, for example the manual annotation is “*e-mail addresses = PDC*” but the *Response Set* only recognizes “*e-mail address= PDC*”. In this cases, *strict* measures would consider the result as a mistake, while *lenient* measures would treat it as a correct result. The observation of partial matches leads us to conclude that they are generally sufficient for our purposes to be considered correct, so we use lenient measures for evaluating the accuracy.

Table 2. Accuracy of the system over the training and the testing sets of corpus A and B.

	Time(sec. per set)		**Precision**		**Recall**		F_1 **score**	
	A	B	A	B	A	B	A	B
Training Set	3.22	9.25	76%	75%	90%	83%	83%	79%
Testing Set	1.28	3.25	78%	80%	81%	87%	**80%**	**83%**

Since the process of creating the rules of an IE system is iterative, and it involves tests over the training set, the evaluation of the accuracy takes place during the whole development phase. The results of this evaluation serve as a guide to indicate whether the addition of new rules, or modification of existing ones, is needed.

Besides rules, gazetteer lists are also modified during the development process: privacy policies are analyzed and, when found, new NEs instances are added to the lists. For example, when the *Facebook* policy is analyzed, the terms *profile* and *profile picture* are added to the *Specific Data* gazetteer list. We noticed that, when the last policies of the corpus are reached, only few new NEs instances are discovered. This suggests that, although gazetteer lists can still be extended, they have already reached a good level of completeness.

The building process terminated when the accuracy of the system was satisfactory and no new patterns were detected. Table 2 shows precision, recall and F_1 score obtained by running the best configuration of our IE system over the training and testing sets of the corpus A and B. This best configuration uses the Annie implementation of the modules shown in Figure 2. The overall accuracy we reached, captured by the F_1 score obtained over the testing set, is 80% for corpus A, and 83% for corpus B. To avoid biased results, the accuracy is evaluated considering the results over the *testing* sets of corpus A and B, since rules have been created without any knowledge of the documents in such sets.

We believe that the obtained results are very promising, and that users could benefit from a system based on our approach. In fact, considering that average users do not read privacy policies, we increase their knowledge on data collection from 0 to 80%. Also, the results are obtained by using a very basic pipeline, and a limited set of rules. Therefore the accuracy could be improved by adding new modules (such as co-reference or ontologies) and enlarging the rule set. Finally, note that the results presented in Table 2 do not take repetition into consideration, i.e., if a personal data item (e.g. *name*, *e-mail*) appears in two sentences, but it is only detected once, we have one correct result and one wrong result, leading to an accuracy of 50%. In our settings, the fact that at least one occurrence of the same data item is correctly identified suffices to have 100% accuracy (for that specific item). Taking this into account, the accuracy can raise from 79% to 92%, if we consider the training set of corpus B.

The main bottleneck for our system is the accuracy of the POS tagger module. Most of the errors in the *Response Set* are due to erroneous results provided by this module in our experimental tests. The POS tagger is in charge of distinguishing whether a word represents a verb, a noun or another lexical category. That means it should be able to say that the word '*place*' represents a noun in the sentence "*we collect information about the place you took your photo*", while it represents a verb in the sentence "*we store your*

Table 3. Comparison of performances obtained by using different POS tagger over the **testing set** of the corpus A and B.

POS Tagger	**Time(sec)**		**Precision**		**Recall**		F_1 **score**	
	A	B	A	B	A	B	A	B
OpenNLP	1.55	4.50	77%	70%	80%	83%	78%	77%
LingPipe	2.45	3.95	79%	75%	68%	84%	73%	79%
Annie	1.02	3.25	78%	80%	81%	87%	**80%**	**83%**

mail address when you place an order". Since our rules are based on the POS tagger annotations (e.g. *provide* is considered a **PROVIDER_ACTION** if and only if it has been annotated as a *verb* by the POS tagger), an error in the tagger's results leads to an error in our system's results.

Let us consider the following sentence: "*LinkedIn **requests** other information from you during the registration process, (e.g. gender, location, etc.)*". If '*requests*' is not recognized as a verb by the POS tagger, no extraction rule will fire, and the words *gender* and *location* will not be annotated as PDC. That means the accuracy of our system strongly depends on the accuracy of the POS tagger, and that it can be improved by decreasing the tagger error rate.

Since multiple implementations of the POS tagger exist, we tested how the accuracy of the system changes according to the tagger used. The results of this comparison, running the system over the testing sets, are shown in Table 3. We used three different POS tagger implementation: the OpenNLP, the LingPipe[5], and the Annie POS tagger. As we can observe, the use of the LingPipe tagger leads to the worst performance over the corpus A, with a difference of 5% when compared with the best results, obtained by using Annie tagger. The results confirm that the choice of the POS tagger has a considerable impact on the performance of the system. Although Annie is the best performing tagger in our setting, it still shows several errors. A possible way to improve the POS performances, and with it our system's accuracy, is by training a POS tagger over the privacy policy domain.

Finally, note that our solution, which was not optimized for time performances, can be executed in near real-time. As reported in the *Time* column of Table 2, the system needs 9.25 seconds to process 9 complete privacy policies (training set of B), therefore the analysis of a single policy requires around one second. This means our approach is applicable to real scenarios, where providing real time responses to the user is an important requirement.

6 Conclusions and Further Work

In this paper we discuss the creation of an IE system, able to extract the list of personal data collected by a website by analyzing the content of its privacy policy. The system shows an accuracy around 80%. Although we believe this accuracy is reasonable and able to benefit the users, improvements can still be made. Adding co-reference

[5] For the LingPipe the *pos-en-general-brown model* was used.

and anaphora resolution modules can increase the accuracy by verifying that, for example, '*we*' refers to the '*website*' when annotating it as DATA_COLLECTOR (so far we assumed this is always true). Also, the accuracy can be improved by using ontologies and thesaurus, to enrich the gazetteer lists with synonyms and close lexical concepts. On the other hand, we believe the system would not benefit of the use of syntactic parsing, since its computational costs would not allow to provide real time responses to the users [6]. It is also important to verify that users find our system beneficial, for example by carrying on a user study. Since the manual creation of extraction rules is a complex and error prone task, the use of systems for the automatic creation of extraction rules, e.g. by applying machine learning techniques, can also be considered (see [36]). Finally, in regards to our more general goal of grading privacy policies, we wish to apply the approach presented in this paper to other sections of privacy policies, e.g. to analyse the *data sharing* practices, and to define a policy grading system based on the results of such analysis.

References

1. Kosta, E., Dumortier, J., Graux, H., Tirtea, R., Ikonomou, D.: Study on data collection and storage in the EU. Technical report, ENISA, European Network and Information Securiy Agency - (2012)
2. Newman, J.: 8 Tools for the Online Privacy Paranoid (2012)
3. Spiekermann, S.: Engineering privacy. Software Engineering, IEEE **35**(1) (2009)
4. Tsai, J., Egelman, S., Cranor, L.: The effect of online privacy information on purchasing behavior: An experimental study. Information Systems Research **21** (2011)
5. Turow, J., Hoofnagle, C.J., Mulligan, D.K., Good, N., Grossklags, J.: The FTC and Consumer Privacy in the Coming Decade. Federal Trade Commission (2006)
6. Tene, O.: Privacy in the Age of Big Data: A Time for Big Decisions. Stanford Law Review Online (2012)
7. Costante, E., Sun, Y., den Hartog, J., Petkovic, M.: A Machine Learning Solution to Assess Privacy Policy Completeness. Submitted (2012)
8. Holtz, L.E., Nocun, K., Hansen, M.: Towards Displaying Privacy Information with Icons. Privacy and Identity Management for Life (2011)
9. Anton, A.I., Earp, J.B., Qingfeng, H., Stufflebeam, W., Bolchini, D., Jensen, C.: Financial privacy policies and the need for standardization. IEEE Security and Privacy **2**(2) (2004)
10. Brodie, C.a., Karat, C.M., Karat, J.: An empirical study of natural language parsing of privacy policy rules using the SPARCLE policy workbench. In: Proc. of SOUPS 2006, ACM (2006)
11. Brodie, C., Karat, C., Karat, J., Feng, J.: Usable security and privacy: a case study of developing privacy management tools. In: Proc. of SOUPS 2005, ACM (2005)
12. Yu, W., Doddapaneni, S., Murthy, S.: A Privacy Assessment Approach for Serviced Oriented Architecture Application. In: Proc. of SOSE'06, IEEE (2006)
13. Yu, W.D., Murthy, S.: PPMLP: A Special Modeling Language Processor for Privacy Policies. In: Proc. of ISCC 2007, IEEE (2007)
14. Cranor, L., Langheinrich, M., Marchiori, M., Presler-Marshall, M., Reagle, J.: The platform for privacy preferences 1.0 (P3P1. 0) specification. W3C (2002)

[6] Preliminary tests show the Stanford Parser needs 123 seconds to process the Google privacy policy.

15. Aïmeur, E., Gambs, S., Ho, A.: UPP: User Privacy Policy for Social Networking Sites. In: Proc. of ICIW 2009, IEEE (2009)
16. W3C: Privacy Enhancing Browser Extensions. Technical report, W3C (2011)
17. Ashley, P., Hada, S., Karjoth, G., Powers, C., Schunter, M.: Enterprise privacy authorization language (EPAL). Technical report, IBM Research (2003)
18. OASIS: extensible access control markup language (xacml) version 2.0. Technical report, OASIS (2008)
19. Schwitter, R.: English as a Formal Specification Language. In: Proc. of DEXA 2002, IEEE Computer Society (2002)
20. Cranor, L., Arjula, M.: Use of a P3P user agent by early adopters. In: Poc. of WPES 2002. (2002)
21. Reagle, J., Cranor, L.: The platform for privacy preferences. Communications of the ACM **42**(2) (1999)
22. Beatty, P., Reay, I., Dick, S., Miller, J.: P3P Adoption on E-Commerce Web sites: A Survey and Analysis. IEEE Internet Computing **11**(2) (2007)
23. Nédellec, C., Nazarenko, A.: Ontologies and Information Extraction. CoRR **abs/cs/060**(July) (2006)
24. Cunningham, H.: Information extraction, automatic. In Brown, K., ed.: Encyclopedia of Language and Linguistics. Volume 5. Elsevier (2005)
25. Turmo, J., Ageno, A.: Adaptive information extraction. ACM Computing Surveys (CSUR) **38**(2) (2006)
26. Hobbs, J.: The generic information extraction system. In: Proc. of MUC 1993. (1993)
27. Deemter, K., Kibble, R.: On coreferring: Coreference in MUC and related annotation schemes. Computational linguistics (2000)
28. Hirschman, L., Robinson, P., Burger, J.D., Vilain, M.B.: Automating coreference: The role of annotated training data. CoRR **cmp-lg/9803001** (1998)
29. Cunningham, H.: GATE, a General Architecture for Text Engineering. Computers and the Humanities **36**(2) (2002)
30. Cunningham, H., Maynard, D., Bontcheva, K.: Text Processing with GATE (Version 6). GATE (2011)
31. Cunningham, H., Maynard, D., Bontcheva, K., Tablan, V.: GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications. In: Proc. of ACL 2002. (2002)
32. Ohm, P.: Broken promises of privacy: Responding to the surprising failure of anonymization. UCLA Law Review **57** (2010)
33. Krishnamurthy, R., Li, Y., Raghavan, S., Reiss, F., Vaithyanathan, S., Zhu, H.: SystemT: a system for declarative information extraction. SIGMOD Rec. **37**(4) (2009)
34. Ashish, N., Mehrotra, S., Pirzadeh, P.: Xar: An integrated framework for information extraction. In: WRI World Congress on Computer Science and Information Engineering. (2009)
35. Cunningham, H., Maynard, D., Tablan, V.: Jape: a java annotation patterns engine (1999)
36. Xu, F.: Bootstrapping Relation Extraction from Semantic Seeds. PhD thesis, Saarland University (2008)