TÉCNICO LISBOA

RSLingo4Privacy: Tools to Improve Consistency Between Privacy Policies and Current Practices

João Paulo de Matos Caramujo

Thesis to obtain the Master of Science Degree in

Engineering and Computer Science

Supervisors: Prof. Alberto Manuel Rodrigues da Silva
Prof. Pável Pereira Calado

Examination Committee

Chairperson: Prof. Ernesto José Marques Morgado
Supervisor: Prof. Alberto Manuel Rodrigues da Silva
Members of the Committee: Prof. Bruno Emanuel da Graça Martins

October 2016

Acknowledgments

It goes without saying that this dissertation and all its work would not have been possible if I had not had the necessary help and support along the way.

I want to thank my supervisor, Professor Alberto Silva, for his remarkable commitment to this project and for introducing me to the new world of research. It was a pleasure being a part of it. Also, a special thanks to my co-supervisor, Professor Pável Calado, for helping me navigate through this complex world of Natural Language Processing and Machine Learning, and to have the patience for answering my questions. Moreover, I also want to thank Professor Travis Breaux for hosting me at CMU during the Summer and allowing me to explore other topics of research.

I must also thank my colleagues, Shaghayegh Monfared and André Ribeiro, for their help and effort in dealing with other aspects of the RSLingo4Privacy project and that were truly an important contribution to this dissertation. The RSLingo4Privacy-Studio, and the background that supports it, are the result of André's work and I do not take credit for it.

Lastly, I have to express the utmost gratitude to my family and true friends, in particular to my girlfriend, Inês, for putting up with a lot of stressful times and always providing the crucial and much appreciated support. I imagine it was not always easy.

Lisboa, October 2016

Abstract

The common operation of popular web and mobile information systems involves the collection and retention of personal and sensitive information about their users. This information needs to remain private and each system should show a privacy policy that describes in-depth how the users' information is managed and disclosed. However, the lack of a clear understanding and of precise mechanisms to enforce the practices described in a policy can constraint the development and adoption of these requirements.

RSLingo4Privacy is a multi-language approach that intends to improve the specification and analysis of such policies, and which includes several processes with respective tools, namely: (P1) automatic classification and extraction of statements and text snippets from original policies into equivalent and logically consistent specifications (based on a privacy-aware specific language); (P2) visualization and authoring of these statements in a complete and rigorous way based on that privacy-aware specific language; (P3) automatic analysis and validation of the quality of these specifications; and finally (P4) policies (re)publishing. This dissertation addresses the first two processes (P1 and P2), through the definition of RSL-IL4Privacy as a privacy requirements specification language that eases the gap between natural and formal languages, as well as the development of the necessary solutions for the automatic text classification and extraction approach.

Keywords

RSLingo4Privacy; RSL-IL4Privacy; Privacy Policies; Privacy Requirements Specification

Resumo

O uso comum de populares sistemas de informação móveis e web envolve a recolha e retenção de informação pessoal e sensível sobre os seus utilizadores. Esta informação tem de permanecer privada e cada sistema deve oferecer uma política de privacidade que descreve em profundidade como é que a informação dos utilizadores é gerida e partilhada. Contudo, a falta de um entendimento claro e de mecanismos precisos para fazer cumprir as prácticas descritas numa política pode restringir o desenvolvimento e adopção destes requisitos.

RSLingo4Privacy é uma abordagem multi-linguagem que tenta melhorar a especificação e análise destas políticas e que inclui vários processos com as ferramentas respectivas, como: (P1) classificação e extracção automática de frases e conteúdos textuais das políticas originais e conversão para especificações equivalentes e logicamente consistentes (baseadas numa linguagem específica de privacidade); (P2) visualização e autorização dessas frases textuais de maneira completa e rigorosa baseada nessa mesma linguagem específica de privacidade; (P3) análise e validação automática da qualidade destas especificações; e finalmente (P4) (re)publicação de políticas. Esta dissertação detalha os primeiros dois processos (P1 e P2), através da definição da RSL-IL4Privacy como linguagem de especificação de requisitos de privacidade que diminui a separação entre a língua natural e linguagens formais, assim como o desenvolvimento das soluções necessárias para a abordagem referente à classificação e extracção de texto automática.

Palavras Chave

RSLingo4Privacy; RSL-IL4Privacy; Políticas de Privacidade; Especifícação de Requisitos de Privacidade

Contents

List of Figures

List of Tables

Listings

Acronyms

BPMN	Business Process Modeling and Notation
CMU	Carnegie Mellon University
DSL	Domain-Specific Language
GATE	General Architecture of Text Engineering
IE	Information Extraction
IR	Information Retrieval
ML	Machine Learning
NER	Named Entity Recognition
NLP	Natural Language Processing
RE	Requirements Engineering
UML	Unified Modeling Language

1

Introduction

Web and mobile information systems increasingly leverage user data that is collected from multiple sources without a clear understanding of data provenance or the privacy requirements that should follow this data. These systems are based on multi-tier platforms in which each "tier" may be owned and operated by a different party, such as cellular and wireless network providers, mobile and desktop operating system manufacturers, and mobile or web application developers. In addition, user services developed on these tiers are abstracted into platforms to be extensible by other developers, such as Google Maps and the Facebook and LinkedIn social networking platforms. Application marketplaces, such as Amazon Appstore, Google Play and iTunes, have also emerged to provide small developers increased access to customers, thus lowering the barrier to entry and increasing the risk of misusing personal information by inexperienced developers or small companies. Therefore, platform and application developers bear increased, shared responsibility to protect user data as they integrate their services into multi-tier ecosystems.

For example, in Canada, Europe and the United States, privacy policies, also called privacy notices (or just "policies" for simplicity), have served as contracts between users and their service providers and, in the U.S., these policies are often the sole means to enforce accountability [3]. In particular, Google has been found to re-purpose user data across their services in ways that violated earlier versions of their privacy policy [4]; and Facebook's third-party apps were found to transfer Facebook user data to

advertisers in violation of Facebook's platform policies [5]. Given the pressure to post privacy policies and the pressure to keep policies honest, companies need tools to align their policies and practices. In this respect, we believe developers need tools to better specify their privacy policies at a requirements and architectural-level of abstraction (i.e., denoting the actors, data types and including restrictions on what data may be collected, how it may be used, to whom it may be transferred and for what purposes) and that privacy policies only present a subset of this view to the general public. The challenge for these companies is ensuring that developer intentions at different tiers are consistent with privacy requirements across the entire ecosystem. To this end, we conducted a series of studies to formalize a set of privacy-relevant requirements captured from privacy policies.

On the other hand, Requirements Engineering (RE) intends to provide a shared vision and understanding of the system to be developed between business and technical stakeholders [6–8]. The adverse consequences of disregarding the importance of the early activities covered by RE are well-known [9, 10]. A privacy policy is a technical document that states the multiple privacy-related requirements that a system should satisfy. These requirements are usually defined as ad-hoc natural language statements. Natural language is flexible, universal, and humans are proficient at using it to communicate. Natural language has minimal adoption resistance as a requirements documentation technique [6, 8]. However, although it is the most common and preferred form of requirements representation [11], it also exhibits some intrinsic characteristics that often present themselves as the root cause of quality problems, such as incorrectness, inconsistency or incompleteness [6, 8, 12].

The main objective of this research is to improve the understanding and quality of privacy policies by providing a set of languages and tools to align those policies with their practices, namely by introducing a privacy requirements specification approach into the regular software development process that would allow to align multi-party expectations across multi-tier applications. The relevance of this approach, called RSLingo4Privacy, and namely its results are of paramount relevance and impact both to the industrial as well as the academic communities by promoting a further rigor related the specification and analysis of privacy requirements and consequently by helping policy authors and developers to avoid the referred inconsistency and better design and implement their systems.

Lastly, the RSLingo4Privacy project itself is within the scope of "Carnegie Mellon Portugal Program"[1], a partnership between Carnegie Mellon University (CMU)[2] and Portuguese research institutions, universities and companies. Due to this program I had the opportunity of taking a ten-week internship at CMU, under the supervision of Professor Travis Breaux, which allowed me to explore more deeply the application of language patterns to the classification of privacy requirements (statements of privacy policies).

[1] http://www.cmuportugal.org/
[2] http://www.cmu.edu/

1.1 Research Problems and Thesis Statement

The main problem to be addressed in this dissertation, which is endorsed by the reviewed State of the Art, and that RSLingo4Privacy intends to solve deals with the fact that:

P: Current formal languages do not provide a complete specification of privacy-related requirements and lack mechanisms to enforce existing privacy policies in which such requirements are described using natural language.

Due to its nature and complexity, this problem can be divided into more specific sub-problems. Each of these sub-problems is explained in the following paragraphs.

P1: Privacy Requirements Representation and Specification. The first problem is the choice of the most appropriate formal language to document privacy-related requirements. There are two major generic requirements documentation perspectives: natural language text and conceptual models [13].

Whether deliberately or not, most of the current formal specification languages, which are restricted to the privacy/authorization domain, lack important elements to model current policies, thus being insufficient for the purpose of our research. Taking into account the importance of natural language for documenting requirements and the fact that privacy policies are written using natural language, a formal specification language that provides a thorough specification of privacy-related requirements in a simple representation format and, at the same time, is flexible enough to provide - if needed - complementary models, improving the comprehension of such requirements by all stakeholders involved, is of great value.

P2: Privacy Requirements Enforcement. As software increasingly leverages platforms and third-party services, development teams need trivial strategies and tools to check their design intent against developer guidelines and privacy policies in the larger ecosystem [14].

Being aware of the preceding problem, and although a clear representation of the privacy requirements is desired, the specification of these requirements must be complete and detailed with all their different elements defined and their associations modeled. The specification language should provide a rigorous description of the sensitive data flow that is expressed in privacy policies and that is mandatory to enable further formal validation and reasoning processes, hence enforcing such policies.

P3: Automatic Information Extraction. Even though the privacy requirements specification language is designed to cope with the specification of new requirements and policies, it should provide the necessary set of constructs for generating explicit and coherent specifications from original natural language policies and through the use of several Natural Language Processing (NLP) techniques. A powerful Information Extraction (IE) approach, capable of providing accurate results, could considerably lessen

the burden of requirements engineers to analyze and enforce privacy policies.

Bearing all these problems, the research work described in this dissertation should demonstrate the thesis statement below:

TS: Through the development of NLP-based solutions it is possible to extract relevant information of current privacy policies and map it into refined and complete privacy-aware specifications, therefore enabling the formal validation of the original policies and enforcement of their requirements.

1.2 Research Methodology

Considering the complexity of the set of problems identified in Section 1.1, namely the formalization of the Domain-Specific Language (DSL) and the proper NLP techniques to process the privacy-related requirements written in natural language text, we decided to use *Action Research* methodology [15] in this dissertation. This methodology is particularly useful because of the aforementioned complexity of the problems and also the fact that this field of study is recent and rather abstract, therefore requiring a certain level of flexibility to frame the proper research problems [16].

Action Research methodology is a cyclical process that considers problems as inputs and the results answering such problems as outputs. Figure 1.1 outlines the five different stages that each iteration of this method encompasses.

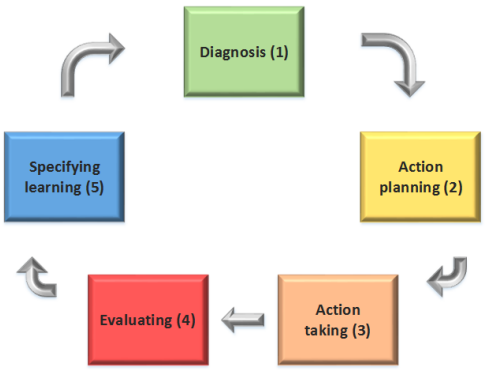

Figure 1.1: The five stages comprising one iteration of the *Action Research* methodology (adapted from [1])

Each stage is defined as follows [1]:

1. **Diagnosing** - In this first phase, the researcher identifies and clearly describes the main problems to be addressed during the dissertation;
2. **Action planning** - After *diagnosing the problem(s)*, the researcher specifies a plan that must guide his work in achieving the solution that is expected to solve the problems;
3. **Action taking** - During this phase, the researcher implements the plan by applying the solution;

4. **Evaluating** - After completing the actions in the previous step, all outcomes are evaluated. The goal is to decide if the solution was adequate and successful;
5. **Specifying learning** - Lastly, the research sums up the knowledge obtained during the iteration. If the solution is suitable for the problem, the research is finished. If, by chance, the solution turned out to be inadequate for the problem, a new iteration needs to take place but the researcher is able to use the new knowledge for refining the strategy.

1.3 Contributions and Publications

The results of the work carried out in this dissertation lead to various contributions, i.e., scientific papers, which I co-authored and that have been published in international conferences, namely:

1. The definition of a privacy-aware profile [17] that was the basis for the definition of RSL-IL4Privacy language.
2. RSLingo4Privacy as a multi-language approach for the specification and analysis of privacy policies [2].

Additionally, a scientific paper refining the RSL-IL4Privacy language [18] was submitted to the *Requirements Engineering* international journal and is currently under revision.

1.4 Dissertation Structure

This dissertation includes nine chapters and two appendices.

Introduction. The initial chapter introduces the main subject of this dissertation, namely the research problems and the thesis statement. It also discusses the research methodology and contributions of this work.

State-of-the-Art. Chapters 2 and 3 overview the related work in the fields of RE, in particular Privacy Requirements Specification, and NLP.

1. Privacy Requirements Specification - This chapter details the work that has been carried out in the field of RE regarding the analysis of privacy policies (i.e., privacy requirements specifications), the existing formal languages for privacy requirements specification and some supporting technologies.
2. Natural Language Processing - The second and final chapter concerning the state-of-the-art reviews some main NLP tasks/problems which are related to this dissertation. In addition, it surveys the software and also some work that applied NLP techniques to privacy policies.

Solution.

1. The RSLingo4Privacy Approach - The overview of the proposed solution. It describes the general approach and outlines the processes which encompass RSLingo4Privacy.
2. RSL-IL4Privacy Language - This chapter describes the language that was defined to deal with the problems highlighted in Section 1.1. It provides the abstract syntax of its constructs and an example using the concrete syntax for each construct.
3. Automatic Classification and Extraction. The first process of RSLingo4Privacy, which includes the classification of statements and extraction of other relevant information for populating the defined RSL-IL4Privacy constructs.
4. RSLingo4Privacy-Studio - Even though it does not fall into the direct scope of the work carried out, RSLingo4Privacy-Studio is the tool that was developed to support RSLingo4Privacy, hence an overview of its features is included in this chapter.

Evaluation. Although the models regarding the classification and extraction tasks use cross-validation (therefore, replicating a real usage situation), the proposed solution is assessed with a specific case study. Both the tasks of classification and extraction are tested separately and together to properly simulate its real scenario. Additionally, the methodology for mapping RSL-IL4Privacy to Eddy (one of the formal languages described in Chapter 2) - and part of the second process of RSLingo4Privacy - is described and its results (process P3) discussed.

Conclusion. This chapter ends the dissertation by providing a more general comparison of RSL-IL4Privacy with the other surveyed formal languages, as well as a final discussion of how RSLingo4Privacy-Studio leverages the approach discussed, in particular when compared to other tools that support such formal languages.

Appendices. The appendices supply the tables and other artifacts with a shape and size that do not make them suitable to appear throughout the aforementioned chapters. Snippets of such artifacts are included in the chapters to provide the reader with the necessary information about them.

1.5 Guidelines

Throughout this dissertation, some guidelines/conventions were followed regarding the concepts used. The terms “privacy requirements”, “privacy-related requirements”, “privacy statements”, “statements” are considered interchangeable since all of them refer to the same notion: the statements of a privacy policy. In addition, “privacy policy”, “privacy requirements specifications” or, for the sake of simplicity, just “policy” are intended to designate common privacy policies.

On the other hand, classification and extraction have a lot in common, including the theoretical concepts. For the classification part we use terms like "classifier", "label" or "statement"; for the extraction part we decided to name the equivalent concepts as "extractor", "tag/annotation" and "entity". We do this to help the reader navigate through the Chapter 6 that describes the classification task immediately followed by its extraction counterpart.

Finally, the RSL-IL4Privacy constructs are written using the `this typewriter font` in Chapter 5 where they are defined. Aside from that Chapter, when discussing them, we use normal font. Additionally both the *types* of a Statement as well as the *tags/annotations* of an Entity use quotation marks ("").

2

Privacy Requirements Specification

The specification and documentation of privacy requirements, as well as its incorporation and subsequent onto the system design [19] has been one of the recent focus in the field of RE. Several studies have been carried out with the objective of creating frameworks that can support the elicitation and support of privacy requirements across various domains [20, 21]. Additionally, definition of models that can represent the scope of a privacy policy and formal languages that can specify such policies and enable further computations have also been a research interest in the community.

This chapter is organized as follows. First, it describes the work that has been done with regard to the analysis and study of privacy policies. After this description, it details and compares the different privacy specification languages that were developed to specify complete policies or privacy requirements. Finally, a comparison of the different technologies and tool support of such languages is also presented.

2.1 Privacy Policies and Requirements Engineering

The analysis of privacy policies with the purpose of understanding the different data practices these documents include and, at the same time, to infer the potential of their use onto the systems' design process has been the focus of some research work in RE.

Even though there is an agreement as to the effectiveness of privacy policies at informing users about

the companies' privacy data practices [22], some authors argue that privacy policies are pointless and with no significant value, if the individuals do not have the ability of making usable and enforcement decisions [23]. To perform such decisions, individuals must understand what is written in a privacy policy in the first place. One of the problems is that, in order to understand a privacy policy, users need to have a better reading skill set than the one they currently have [24]. Reidenberg et al. introduced an approach based on a crowdsourcing task with the goal of assessing if different types of individuals (from experts to regular users) can grasp the necessary information from privacy policies in order to support rational decision-making [25]. The results showed that the ambiguity on some key terms and expressions of the policies lead to disagreements in the participants' answers. In addition, policy experts understood the content and practices described in the policies much more easily than typical as well as informed users. Cranor et al. [26] analyzed 75 policies from online tracking companies with the intent of looking for different kinds of data practices (59 in total). They highlight some vague data practices as a result from the study, namely the practices referring, for example, to data collection, sharing, use, and retention. A separation between *privacy-friendly practices* and *privacy-concerning practices* is used to distinguish between policies with regard to the sensitivity of the information shared. Lastly, this study also overviews the main reasons described in the original policies for such disclosure of information.

However, privacy policies as privacy requirements specifications have been used in the system development process. One of the most important perspectives was the use of goals as a *tool* for extracting privacy and security requirements from privacy policies [27]. The identification of privacy goals which were extracted from the collection of privacy statements that encompass a privacy policy - an uncommon source of requirements in the software design process - and the following refinement of such goals into non-functional requirements allowed one to gain more insight of the importance and advantages of getting requirements engineers to identify conflicts within privacy policies and to align eventual inconsistent policies. On the other hand, the analysis of the privacy policies with the methodology described in this work raised also awareness for the importance of having a more standardized and understandable notion of the different concepts that comprise a privacy policy, such as actors, actions and subject matter (i.e., purpose).

These works fostered the necessity of creating conceptual models or privacy-aware profiles [17] that could provide a common understanding of the concepts that comprise a policy, while giving support for the definition of more formal languages. Such profiles may be important regarding the development of other third-party applications as they can help to easily reveal the existing gap between privacy and system requirements [28] through the definition of the set of all necessary and relevant policy elements, as well as the associations between such concepts.

2.2 Privacy Specification Languages

Several formal languages have been created to specify privacy policies from different perspectives and with different concerns. The definition of formal languages that can specify privacy requirements has been a prominent part of recent research in RE, since the prospect of being able to perform automatic validation of privacy requirements (hence privacy policies) may play an important role in the systems design and integration processes.

2.2.1 Platform for Privacy Preferences

The Platform for Privacy Preferences (P3P) is a XML-based language that allows websites to express their privacy practices in a standard format which intends to provide user agents with the ability to easily access and interpret such practices, hence encoding them in a machine-readable format. Through this mechanism, end-users (i.e., consumers) can be informed about the website practices (in a human-readable format), thus being able to foster their decision-making process when suitable. A *P3P policy* - specification that encloses the encoded data-collection and data-use practices from a given website in a XML format - defines a set of features, such as a standard schema for the data that a website may collect, a standard set of categories like uses, recipients and other privacy attributes, and, as already discussed, a XML format for a privacy policy [29].

P3P gives an exhaustive characterization of a policy by defining a set of elements about the policy itself. It also details statements - defined as practices applied to one or more data types - through the specification of elements like purpose, recipient or retention. Even though that a P3P policy does not control the exchange of data between the website and other third-parties, the classification of the data according to such elements allows one to distinguish in some detail different concerns regarding the nature and management of data.

Other important remark about P3P is the concept of compact policy. A compact policy is a summarized P3P policy that provides hints to user agents enabling them to make quick, synchronous decisions about applying policy [30]. However, one of the downsizes of P3P is precisely the misuse of the compact policy tokens which leads to misrepresentations of websites privacy policies [31]. Other relevant issues regarding P3P, such as the syntax of P3P privacy statements - namely the statements' ambiguity and redundancy due to the unclear separation between the elements described just before - and the vague meaning of a P3P policy - what data is collected and retained, and which part of that data is disclosed to external entities - are two of the most important drawbacks of P3P [32].

The shortcomings of P3P is mainly due to the lack of a formal semantics for this language [33]. The multiple interactions among the different stakeholders expressed various semantic inconsistencies. These conflicts justify the slow adoption of P3P.

2.2.1.1 APPEL Language

APPEL complements P3P by specifying a language that describes collections of preferences regarding P3P policies between P3P agents. Using this language, a user can express her preferences in a set of preference-rules (called a ruleset), which can then be used by her user agent to make automated or semi-automated decisions regarding the acceptability of machine-readable privacy policies from P3P enabled websites [34]. Like for P3P, APPEL has its own semantic inconsistencies. Due to APPEL's syntax-based design, P3P policies that are semantically equal but syntactically different may be treated differently by the same APPEL rule. [35]

2.2.2 Enterprise Privacy Authorization Language

The Enterprise Privacy Authorization Language (EPAL) is a formal language for writing enterprise privacy policies to govern data handling practices in IT systems according to fine-grained positive and negative authorization rights, i.e., an interoperability language for exchanging privacy policy in a structured format between applications or enterprises [36]. Unlike eXtensible Access Control Markup Language (XACML) [37] that is focused on general access control policies, EPAL targets privacy policies only and that makes it not so broadly deployed as XACML [38].

An EPAL privacy policy can be defined as a collection of privacy rules ordered according to a given descending precedence. Each privacy rule specifies a set of different elements such as, for example, a *user* (that belongs to a certain category), a *data category*, an *action* that a user is supposed to perform on such data, and the subsequent *purpose*. A rule also encompasses a *condition* which needs to be met before granting authorization and an *obligation*, a post-condition that must be carried out after granting authorization. The goal behind privacy rules in EPAL is to decide the authorization over a certain request, i.e., if a request is to be allowed or denied. A request is defined over some elements of the privacy rules, namely a user, an action, a data category, and a purpose. Even though EPAL focus access control and privacy, the specification of rules takes into account different concepts which are relevant and share common attributes with other languages described in this Section. However, as for P3P, the lack of a formal semantics does not provide EPAL with the necessary complexity for properly evaluating privacy policies [14].

2.2.3 KAoS

KAoS is a collection of componentized services compatible with popular agent frameworks [39]. These services can be grouped into domain or policy services. KAoS domain services provide one with the possibility of structuring different software-related entities (human resources and components) into domains or sub-domains by grouping them together, hence improving the collaboration between agents and the

management of policies. At the same time, KAoS policy services play a very important role because these services deal with the whole policy life cycle by allowing the specification, management, handling of conflicts and enforcement of such policies within multiple domains.

One of the most important features of KAoS and that is worth discussing is the policy conflict resolution. Whenever a change occurs in one or more policies, thus causing a change in the status of one or more actors, KAoS needs to have tools that can detect which policies are in conflict and eventually solve those conflicts. It is worthy of mention that KAoS language does not provide the necessary means for detecting conflicts within a policy, such as conflicts between rights and prohibitions. The solution to the problem of detecting conflicts between policies relies on a set of *internal* algorithms that, firstly order the policies according to a certain precedence criteria and then, if necessary, construct newly harmonized policies (i.e., policies without conflicts between each other) [39].

The representation chosen to describe the policies and their context largely determines the flexibility, extensibility, and amenability to analysis of a given implementation [39]. KAoS uses Web Ontology Language (OWL) as central policy ontology which allows the definition of the main policy-related concepts but also provides application developers with the possibility of extending and adding application-specific concepts (i.e., specific vocabulary) that may be useful when defining policies.

In order to cope with nowadays needs and new technological requirements, and while being supported by an extensible framework, KAoS evolved into a three-layered architecture [40]:

1. Human interface layer: Graphical interface for specifying policies in natural language. Vocabulary is automatically supplied by necessary ontologies.

2. Policy Management layer: OWL encodes and administrates all data that is related to a policy.

3. Policy Monitoring and Enforcement layer: OWL policies are converted into a format which is suitable for monitoring and enforcement.

2.2.4 Rei

The Rei policy language is a logic-based language, modeled on deontic concepts of rights, prohibitions, obligations and dispensations [41]. Rei is not tied to any particular application and supports the addition of domain-specific information, hence allowing the specification of different kinds of policies (e.g., privacy policies).

The core of the Rei policy language is the constructs that allow the description of the above-mentioned deontic concepts. These constructs are known as *policy objects* and can be defined in terms of a particular domain-dependent action and a set of conditions (i.e., constraints on actor, action and environment). It is also possible to associate a policy object with an entity. At the same time, this language introduces another important concept: the *speech act*. In order to have an effective communication between entities

and policy objects in terms of rights and obligations, Rei contains specifications for four different speech acts, namely delegation, request, cancel and revocation. The use of these capabilities by one or more entities is dependent on the policies that govern the speech acts. Lastly, meta-policies are useful for solving conflicts among policies.

Conflicts in Rei occur if policies overlap in subject, target and action but the policy objects are different [41]. The conflicts identified within Rei can be of two kinds: conflicts of modality and conflicts of obligation and prohibition. The mechanisms that solve the aforementioned conflicts are meta-policies and the policy engine itself in RDF-S, consistent with Rei's ontology [42]. Conflict detection takes place at run-time: the policy engine identifies two conflicting policy objects, declares the conflict and selects the appropriate meta-policy. Meta-policies handle conflicts in two ways: through priorities (either between policy rules or entire policies) and precedence (which modality holds precedence over the other in meta-policies) relations [43].

The Rei framework provides means to reason about policy specifications but it does not provide an enforcement model [42]. Even though it can detect conflicts, Rei does not have the proper tools for enforcing policies by preventing some entities (i.e., subjects) from performing unauthorized actions, for instance.

2.2.5 Eddy

Eddy, a formal language for specifying and analyzing data flow specifications for conflicting privacy requirements [14], is based on a semantic parameterization [44] approach in which requirements written in natural language are depicted as actions and roles in Description Logic (DL) in order to deal with the ambiguity that very often features natural language. The specification of privacy requirements takes into account a given number of concepts (i.e., sets), namely:

- *Req* - The collection of data requirements;
- *Actor* - Different actors to whom the data is disclosed;
- *Action* - Possible actions performed on the data;
- *Datum* - Particular data elements on which actions are performed; and
- *Purpose* - Reasons for acting on data.

Actor, Datum and Purpose can be structured hierarchically using DL subsumption (concept is a subset of another concept). The four main types of actions ("COLLECT", "USE", "RETAIN" and "TRANSFER") highlight the different actions in Eddy and these actions have an association with the other existing concepts, such as Actor, Datum and Purpose. These *relationships* are defined as roles.

Within the scope of this language, a conflict occurs when an inconsistency arises between what is permitted and what is forbidden in a given privacy policy, i.e., a conflict occurs when there is an intersection of what is permitted and what is forbidden. Thus, the detection of conflicts consists in listing all possible individual interpretations of the elements Actor, Datum and Purpose and verify for each interpretation if they are, at the same time, permitted and forbidden. If that does not happen, the privacy requirement specification is *conflict-free* [14]. These conflict-free privacy requirements specifications allow one to trace the data elements from collection requirements to other types of requirements (use, retention or transfer). This is useful for organizations as they need to ensure that their privacy policies are in fact implemented on all the organization.

2.2.6 Comparison and Discussion

The preceding formal languages share multiple features which are in some sort relevant for the context of this dissertation. On the other hand, it is also important to clarify and summarize the differences between all the introduced approaches and, in order to easily comprehend the surroundings of such languages, it is crucial to present a comparison of them. Taking into account the simplicity and readability needed to carry out this comparison, Table 2.1 gives a brief comparison of the most important approaches on the formalization of privacy requirements specification languages.

Table 2.1: Comparison of privacy-aware specification languages

Language	Domain	Semantics
P3P/APPEL	Web Privacy	Declarative
EPAL	Access Control	Declarative
KAoS	Generic	OWL
Rei	Generic	OWL
Eddy	Data Privacy	OWL-DL

The first two formal languages described in this Section (P3P/APPEL and EPAL) have a declarative semantics because both of them have a well-documented syntax (in XML) but lack a formal semantics which compromised the adoption of such languages in general and in the context of this dissertation. At the same time, even though all of these languages have the same semantics, their domain of application is rather distinct. P3P/APPEL, by expressing websites' privacy practices and documenting them in a P3P policy, is the one that has some similarities with the necessary elements to accomplish the objectives of this work. EPAL, on the other hand, deals with authorization concerns.

KAoS and Rei are quite similar on their domain of application and semantics. Both languages can be considered as semantic web languages for representing policies since both KAoS and Rei are based on OWL. Lastly, Eddy language is particularly useful to the scope of this project because it deals with data privacy by providing a specification of privacy requirements with several relevant elements and defining a set of actions which are supported by some phrase heuristics that indicate the type of a statement.

Additionally, Eddy provides a conflict detection process based on Description Logic that is particularly useful to keep track of the different data flows within a policy.

2.3 Technological Support for Privacy Specification Languages

The approaches that include the aforementioned languages are supported by a variety of technologies and tools that provide the necessary means to manage their privacy-related specifications. Table 2.2 overviews such tools with a succinct comparison of the some useful features.

Table 2.2: Comparison of tools that support privacy specification languages

Language	Visualization & Authoring	Publishing	Authorization Enforcement
P3P/APPEL	Yes (IBM P3P, JRC Policy Workbench, P3PEdit)	No	No
EPAL	Yes (Privacy Authoring Editor EPAL Editor)	No	Yes
KAoS	Yes (KPAT)	No	Yes
Rei	Yes (Protégé-Plugin, N3 text editor, RIDE)	No	No
Eddy	Yes (General-purpose text editor)	No	No

IBM P3P Policy Editor[1] is a proprietary tool that permits creating from a template or editing a website's privacy policy using a drag-and-drop graphical user interface (GUI) [30]. JRC Policy Workbench is an open-source tool that provides a GUI for creating, managing and testing P3P policies through a form-based policy editor where the user configures and fills some input fields by following wizards. The JRC Policy Workbench provides an extendable API for building editing and testing environments for other types of XML-based privacy and access control policies like EPAL. P3PEdit website[2] allows the generation of a P3P policy by following a web-based wizard. All these tools abstract the XML syntax, but users still need to properly understand these languages' concepts and how these apply to their website. For this reason these approaches are often considered too complex and difficult to adopt in practice. Also the fact that nor websites neither browsers (only Internet Explorer used) are obliged to use P3P, has contributed to its decreasing use.

KPAT (KAoS Policy Administration Tool) is a graphical tool that allows users to specify, analyze, modify and test policies using KAoS. KPAT offers a set of views of KAoS and detects policy conflicts. Additionally, it also allows managing sets of ontologies. KPAT also offers a wizard to guide the user throughout the policy creation process. Since the policies are specified using the GUI, the corresponding DAML representations are generated automatically using a generic template, avoiding the user to master

[1] https://www.w3.org/P3P/imp/IBM
[2] https://www.p3pedit.com

DAML. Other language-specific templates or domain-specific templates for common classes of policies can be defined.

Besides the general purpose text editors, the specification of Rei policies are supported by a set of tools: a plugin for the Protégé-2000 ontology editor that provides features for creating policies, rules, meta-policies and queries through a custom tab and dialog boxes; a text-based editor that provides content assistance and context information during the specification of Rei policies using the Notation3 (N3) language [45] which makes the policies easier to read; and finally RIDE (Rei Integrated Development Environment) [46]. RIDE is an Eclipse plug-in that uses a wizard-based approach. The creation wizard guides the creation of a policy and in the end generates the corresponding policy file in OWL automatically, based on the user input and selections. Once the policy file is created, the user can launch the test wizard that provides an interface for testing the policy and querying the Rei engine.

Eddy's website provides three examples using an editor where the user can manually specify a privacy policy in a free text area. After finishing the specification of a policy, it is possible to run the Eddy engine to analyze the privacy policy for any possible conflicts or for tracing the flow and showing it in a network chart. The website also provides the user with the option of exporting the equivalent OWL file resulting from such analysis.

3

Natural Language Processing

NLP is a field that has been intensively studied and documented over the last fifty years [47,48]. With the increasingly amount of data available at all times, the problem of making some sort of sense of such data became of the utmost importance. Also, the development of new systems with more computational power gave NLP, as a research field, the opportunity of being applied to a wide range of different areas [49] aside from the timeless field of Artificial Intelligence. With the passage of time, several NLP problems and tasks began to be individualized, studied and with solutions being developed and proposed. These set of problems can be adapted to nowadays' needs, including the analysis of technical documents such as privacy policies.

This chapter is organized as follows. First, it introduces the general classification problem and its fundamental concepts. Second, it provides a refined definition of multi-label classification and a popular category of solutions: problem transformation methods. It also details well-known Machine Learning (ML) algorithms for classification. Third, a description of the Named Entity Recognition (NER) task is given with its underlying concepts explained. Fourth, we overview some of the work that has been carried out in NLP with regard to privacy policies and we finish the chapter by surveying some software for both the multi-label classification and NER problems.

3.1 Natural Language Processing Applications

NLP provide the foundation for a wide of solutions that are used in a variety of situations or tasks. The problems that require NLP-based solutions are deemed NLP tasks. Popular NLP tasks or fields range from the areas of Information Retrieval (IR) or IE to Classification or Word Sense Disambiguation. The boundaries regarding the definitions of both IR and IE are somewhat unclear; however, Gaizauskas and Wilks [50] sum up both fields very concisely: IR is the task of retrieving a set of relevant documents from collection given a query, whereas IE is the field responsible for extracting relevant information from documents. Thus, the NLP tasks addressed in this dissertation are Classification and IE, in particular the NER problem.

3.1.1 Text Classification

Classification, also referred in the literature as document classification or categorization, is one of the most studied problems and main NLP tasks. Even though a document can represent several media formats such as text, images or audio, text is still the primary object for classification. Thus, one can define text classification as the process of assigning textual documents (e.g., paragraphs, sentences or words) to particular topic classes depending on the features the given documents share. In other words, text classification consists in assigning a boolean value to each pair $< d_j, c_i > \in D \times C$, where D is the domain of documents and $C = \{c_1, ..., c_{|c|}\}$ is the set of predefined categories [51] indicating if document d_j belongs to class (i.e., category) c_i.

Taking into account the aforementioned definition of text classification, there is the possibility of having the same document assigned to more than one class (e.g., both $< d_1, c_1 >$ and $< d_1, c_2 >$ give a *true* value). Depending on the number of classes each statement can take, it is possible enrich the former definition by dividing the text classification problem into two major and more specific problems: multiclass and multi-label classification. Multiclass is the classification problem in which a single document or instance can only be assigned to one class or label. On the other hand, multi-label classification is the problem that deals with the possibility of having instances with more than one label. A well-known case of multi-label classification is the task of movie genre identification: a movie can be both an action movie and a comedy. A clear distinction between these two text classification problems is crucial because their resolution employs different methods. The IE approach supporting RSLingo4Privacy includes a multi-label classification problem, therefore we only discuss this type of classification.

3.1.2 Multi-label Classification

The multi-label classification is an important problem in the field of NLP, IR, etc., and it has been fairly studied and used in a variety of solution [52, 53]. To solve this classification problem, the litera-

ture [54] classifies the solutions into two key classes: Problem Transformation Methods and Algorithm Adaptation Methods. On the one hand, problem transformation methods try to solve multi-label classification problems by reducing them into smaller and simpler (multiclass) classification tasks, modifying the data set if needed. On the other hand, algorithm adaptation methods tackle multi-label classification by adapting well-known ML algorithms in order to give them the ability to support this kind of classification. Since the former class is used in many situations and simple to implement, we only describe this class of solutions. In addition, since we refer to multi-label classification, and as opposed to the generic definition of the classification task with documents and classes, we will make reference to labels instead of classes throughout this dissertation ($C \rightarrow L$).

3.1.2.1 Problem Transformation Methods

There is a wide variety of different problem transformation methods that have been studied and applied in many different applications. In this section we provide a brief description of some of the most common methods which allows the reader to get a basic understanding of their properties and differences.

Binary Relevance. The most common problem transformation method [54] is named Binary Relevance (BR). The core idea behind BR is to convert a multi-label problem into a multiclass problem. Then, this *new* learning problem is solved by having L independent binary classifiers, one for each l in L (similarly to the "one-versus-all" approach in multiclass classification). To train this set of binary classifiers, a new train set is generated for each binary classifier, discriminating the documents that are labeled with l or $\neg l$. The prediction(s) for a test instance is given by the positive output(s) of all binary classifiers. However, if all binary classifiers yield negative results, the label with the highest value for the prediction is selected. BR is a relatively straightforward method, has low computational complexity and has provided competitive results to many applications [55]. However, on the other hand, it assumes *label independence*, i.e., it does not take into account eventual correlations among labels.

Classifier Chains. Classifier Chains (CC) [56] provide a solution to the problems of BR. The idea is to have a *chain* of L binary classifiers in which the classifiers along the chain take into account the predictions of the previous classifier(s). This is done by adding to the training set of the following classifiers a new binary feature that represents the results of the previous classifiers ("1" if the document belongs to that previous label, "0" otherwise). Nevertheless, it is reported that the order of the classifiers in the chain has an impact on accuracy, as well as the performance of the classifiers themselves. To address these problems, Ensembles of Classifier Chains (ECC) [56] trains m CC classifiers with a random chain ordering. All CC classifiers give a prediction that is combined to produce a vector of confidence outputs representing the confidences for each label. Additionally, there is the possibility of defining a threshold to discriminate between relevant and irrelevant labels (e.g., a confidence greater than "0.5"

indicates a relevant label).

Label Powerset The Label Powerset (LP) is a method that considers each unique label set (combination of labels) as one class for the multiclass classification task. When classifying a new test instance, the class that is predicted may be actually a combination of labels. Even though it is a basic and effective method, the drawbacks of using LP [57] are the fact that it is only able to predict label sets (combinations of labels) that appear in the training set, as well as that, if a lot of label sets exist, it might not scale in terms of efficiency.

Random k-Labelsets. Random k-Labelsets (RAkEL) [57] can be seen as an ensemble of LP classifiers. The basic idea is to have m LP classifiers, in which a random k-labelset is selected and used to train the LP classifier. For the prediction of new, unseen instances, RAkEL combines the binary outputs for each label in the k-labelset of its "internal" LP classifiers, averages the decision for each label in the original label set and provides a positive output if the decision is above a certain threshold (e.g., "0.5").

Fourclass Pairwise. Fourclass Pairwise (FW) is said to perform well in data sets that have a strong conditional dependence [58]. The underlying idea is to have a binary classifier trained for each pair of labels (similarly to the "one-versus-one" approach in multiclass classification). The fact that it has quadratic complexity [58] may pose a problem to its use with certain data sets.

3.1.2.2 Machine Learning Algorithms for Classification

Regardless of the type of problem at hand (whether it is a basic multiclass or a "transformed" multilabel), the complete solution that implements the classification (i.e., that decides to which class(es) a certain document belongs) is called the "classifier". A classifier includes various sub-processes that play an important role in the classification process and one of the classifier's main components is the ML algorithm it uses. ML is a class of "data-driven" algorithms due to the fact that these types of algorithms learn from data in order to make predictions for new, unseen data (hence the "learning"). The learning process can be supervised or unsupervised. We have a supervised learning algorithm if the data fed to train the classifier is manually labeled beforehand with its examples already classified; on the contrary, an unsupervised learning algorithm deals with the issue of having all data unlabeled. This type of algorithms look for different structural patterns among the data in order to make predictions for unseen data.

The approach regarding text classification in this dissertation uses supervised learning. We synthesize the main ML algorithms that are widely used to deal with the text classification problem using supervised learning.

Decision Trees. The use of Decision Trees (DT) [59] is a process that consists in applying a sequential set of rules with the decision being reached at the end. The possible and different alternatives are represented with probabilities. The training phase begins with the labeled data set and, for each document, the word that better predicts the class of such document is selected. After this step, the data set is divided into two new sets: one with the documents containing the word and other set with the documents that do not. The process is repeated recursively on both subsets, generating new sub-trees, and stops when all the documents in a subset belong to the same category.

Naïve Bayes. One approach to text classification is the use of probabilistic classifiers, namely Naïve Bayes (NB) algorithm. The core idea supporting NB relies in the independence of the features (e.g., words) that characterize a document (e.g., a sentence) with regard to the class it belongs. The conditional probability of a word given a class is assumed to be independent of the remaining "word-class" probabilities. The classification of a new instance is given by a certain probability that arises from the combination of each conditional probability said to be independent. One of the pitfalls of NB is its inability of learning interaction between features [60].

k-Nearest Neighbors. The k-Nearest Neighbors (kNN) algorithm has been intensively applied to pattern recognition [61] specifically in text classification [62, 63]. The description of the algorithm's "procedure" for classification is simple and rather straightforward: for each unlabeled document in a given test set, the kNN algorithm looks for its k-nearest neighbors in the training set and uses their class (i.e., category) to predict the class for the instance in the test set.

Support Vector Machines. The underlying idea of Support Vector Machines (SVM) [64] is to find the optimal hyperplane (in a d-dimensional space) that separates the instances of two classes. Thus, SVM are intrinsically binary classifiers. In order for SVMs to support multiclass classification, there are two common strategies: 1) One-versus-all, in which L classifiers are built and, for a given test instance, the class that gives a positive prediction is chosen; and 2) One-versus-one, in which $L(L-1)/2$ are built and the class that is selected by a higher number of classifiers (like a "competition") is chosen.

3.1.3 Named Entity Recognition

NER (also known as Named Entity Extraction) is a common branch of IE and it is responsible for identifying references to particular sets of entities in a given document. However, it does not check, for example, if two entities are the same. These issues are addressed by other IE tasks such as Coreference Resolution [65]. NER has been widely studied and used for the task of finding names of people, organizations and locations but also dates, time and other numerical measures [66].

There are two major approaches that address the NER task [67]: hand-coded/rule-based methods and statistical models. Rule-based models is based on manually defining a set of rules or patterns (e.g., regular expressions) that can be used to extract entities from a document. Like in the classification problem, unless the domain of the task is very specific with structured documents, the difficulty of the task of writing the necessary rules grows exponentially. On the other hand, statistical models assume that the extraction is a sequence labeling task, i.e., the document is a sequence of tokens with each token being annotated with a specific label (also known as tag). Additionally, the "sequence" suggests that there is a certain degree of dependency between tokens in a document, so the algorithms (as opposed to the classification task) must take *context* into account (features about a word and also its previous/next - "neighboring" - words). There are two different models that can provide a solution for this sequence labeling problem: Hidden Markov Models (HMM) and Conditional Random Fields (CRF). However, recently studies have shown that CRFs yield better results than HMM [68], hence being considered as the state-of-the-art ML algorithm for NER [69].

Conditional Random Fields. Conditional Random Fields (CRF) [70] is a ML algorithm that models a single joint distribution probability over the predicted labels of the tokens in a document [71]. A simple explanation of the training phase of a CRF is the following: it receives any set of features that can model the context of the given words (e.g, if the present word is a noun and its predecessor a pronoun, assign it a value of "1", and "0" otherwise). For each feature, if it is considered important, a weight value can be calculated to emphasize its importance. Next, the score of a label given a sentence is calculated by the sum of all features (with its weights) from the words of such sentence. Finally, the conditional distribution of a label given a certain token is provided by the conditional probability that exponentiates the score and divides it by a normalizing constant.

3.2 Natural Language Processing and Privacy Policies

The algorithms and tools that were described in this chapter help to make the process of analyzing privacy policies using NLP techniques look promising and with interesting potential.

One of the first projects that was carried out consisted in a pilot study focused on the automatic categorization of privacy policies [72] which intended to estimate the ability of extracting important features from privacy policies. The approach consisted in annotating the privacy policies and performing a simple lexical analysis on the data (e.g., stopwords removal), thus identifying some privacy-related concepts. Since the objective relied in discovering the presence or absence of a concept, i.e., assigning documents to two different categories (e.g., "yes" or "no"), the method was to build a classifier and then train it on labeled instances. The process of mapping the privacy policy documents into categories would be done using logistic regression. The results demonstrated the difficulty of the task and the

authors identified three problems of analyzing privacy policies automatically: (1) few annotated privacy policies; (2) if one considers all terms of a privacy policy, data becomes noisy and hard to analyze; and (3) high-frequent words were considered but in the end had no relevance for the objective.

This previous work lead to other projects, such as aligning of privacy policies using a supervised approach with HMMs [73]. In this work, the authors proposed a new data set of over 1000 privacy policies manually divided by certain sections (e.g., policy URL, policy full text). The authors argue that aligning the privacy policies (aligning its sections, content, etc) can enable further automatic analysis and provide support to legal staff to make better recommendations. The HMM - in which the solution was based - uses the similarity across privacy policies when it was evident in the data. Each hidden state corresponded to a topic with distribution over words and bigrams in parts of the policy dealing with that topic. The results of this work proved to work better than previous solutions like clustering.

3.3 Natural Language Processing Frameworks and Libraries

Complementary to the analysis of some of the most common NLP problems that are also present under the scope of this project, as well as some of the well-known ML algorithms, it is also appropriate to overview the most popular NLP software, frameworks and libraries. To provide a clear understanding about such tools, we only discuss software that supports "out-of-the-box" multi-label classification. The remaining libraries are dedicated to the NER problem (however, some of these frameworks can be also used to perform classification tasks).

3.3.1 Multi-label Classification Software

Weka [74] is one the most popular frameworks for data mining and it provides a set of tools for such purpose (e.g., data pre-processing, classification), including a vast collection of ML algorithms. Weka provides an easy-to-use graphic user interface (GUI) but it is also possible to run Weka from the command-line or via API with Java code. All these characteristics make it suitable to be the basis for the definition of new pieces of software that support multi-label classification (Weka only supports multiclass classification).

MEKA. The MEKA framework [75] performs multi-label classification using mainly problem transformation methods and is based on WEKA. It provides a command-line interface and a GUI but it is also possible to use it in Java applications via API. When compared to MULAN, aside from the GUI and the command-line interface, the usage and performance are also different since MEKA does not require any programming skills to begin with.

MULAN. MULAN [76] is a Java library that provides the implementation of the transformation methods defined to cope with multi-label classification. It provides a programmatic API to its users but it does not include a GUI. MULAN (like MEKA) is built on top of WEKA in order to be provided access to its extensive list of resources.

scikit-multilearn. scikit-multilearn [77] is a multi-label classification package for python that provides the implementation of the algorithms/strategies that support the problem transformation methods described earlier. It also includes a wrapper for MEKA giving access to all its features.

3.3.2 Named Entity Recognition Software

Stanford Named Entity Recognizer. The Stanford Named Entity Recognizer [69] is a well-known software for addressing the NER task. Stanford Named Entity Recognizer implements linear chain CRF sequence models which allows the user to build his its own sequence models for NER using previously labeled data. This tool can be used programmatically or through the command-line. To perform tasks such as tokenization, POS-tagging or parsing, Stanford provides additional software that can be used separately or combined.

ANNIE. ANNIE is an IE system included in General Architecture of Text Engineering (GATE) framework [78]. It includes a wide range of modules like, for example, a tokeniser, gazetteer, sentence splitter and POS tagger. This set of models enables the IE functionality, therefore being able to address its most common problems.

RapidMiner. RapidMiner Studio provides an intuitive GUI for designing data mining processes. It comprises a complete set of operations suitable for common NLP tasks and supports the integration with common data sources. RapidMiner also provides an information extraction plugin [79] which includes a useful GUI annotator, hence smoothing the annotation step of the NER task.

3.4 Discussion

This chapter provides an overview regarding the text classification and NER problems of NLP. It provides the *theoretical* concepts and ideas that support the solutions that were developed and evaluated under the scope of this dissertation. Additionally, it summarizes some recent work that has been carried out using NLP approaches to perform some sort of analysis and reasoning about privacy policies according to concerns that were briefly introduced in the previous chapter (e.g., alignment of policies, conceptualization of policies).

4

The RSLingo4Privacy Approach

The State of the Art presented and analyzed in Chapters 2 and 3 suggests and hints at the necessity of having a complete and automated approach for the complete process of specifying and validating privacy policies. Existing specification approaches rely on certain formal languages that do not provide the necessary constructs to specify current policies. Additionally, these languages require a manual and tiring process of mapping between original policies written in natural language and equivalent specifications in such formal languages.

This chapter is organized as follows. First it introduces the background of the RSLingo4Privacy approach, namely its correlation with RSLingo, a general approach for the rigorous specification of software requirements. Finally, this chapter overviews the four processes included in the RSLingo4Privacy approach and provides a clear description of its workflow.

4.1 RSLingo and RSLingo4Privacy

RSLingo is a general approach for the rigorous specification of software requirements that uses lightweight NLP techniques to (partially) translate informal requirements – originally stated by business stakeholders in unconstrained natural language – into a rigorous representation provided by a language specifically designed for RE. The name RSLingo stems from the paronomasia on “RSL” and “Lingo” [80].

On one hand, "RSL" (Requirements Specification Language) emphasizes the purpose of formally specifying requirements. On the other hand, "Lingo" expresses that its design has roots in natural language, which are encoded in linguistic patterns used during by the information extraction process [81–83] that automates the linguistic analysis of Software Requirements Specifications (SRSs) written in natural language.

The language that supports RSLingo and serves the purpose of formally specifying requirements is RSL-IL, in which "IL" stands for Intermediate Language [84]. RSL-IL provides several constructs that are logically arranged into viewpoints according to the specific RE concerns they address, and are organized according to two abstraction levels: business and system levels [84]. Despite sharing the same background and technologies, RSL-IL4Privacy was recently defined independently of the RSL-IL language and with the only purpose to support the rigorous specification of privacy policies with multi-representations.

Additionally, the linguistic patterns that are the foundation of the IE approach in RSLingo are defined in RSL-PL, a language developed for dealing with requirements documented as natural language sentences and the concerns they raise. Therefore, the linguistic patterns supported by RSL-PL act on a sentence-level since the pattern represent the syntactic structure of the natural language sentences. RSL-PL patterns provide the "means" for representing the mandatory linguistic information to specify requirements in natural language. Within RSLingo4Privacy, the focus is the specification of privacy policies instead of SRSs. This particular difference calls for the need of having an IE approach that does not rely on linguistic patterns as the main "technique" for automatically specify and analyze privacy policies. Due to the fact that privacy policies are ad-hoc natural language documents, we focused our approach on a word-level rather than a sentence-level. However, the patterns (which we designated as RSL-PL4Privacy) are used as features themselves along with other different word-level features. A combination of these features is used to train the classifier. We discuss these solutions and their results in Chapter 6.

4.2 RSLingo4Privacy: a Multi-Language Approach

As already introduced in the previous chapters, a privacy policy is a technical document that states multiple privacy-related requirements that websites and mobile apps should show and respective organizations should satisfy. These requirements are usually defined as ad-hoc natural language sentences, meaning that there is not a rigorous and consistent way to specify and validate them. In spite of the advantages of natural language as a flexible, universal, and human proficiency at using it to communicate with each other, there are some well-known restrictions such as the difficulty to automatically analyze and validate the quality of those specifications.

RSLingo4Privacy approach supports the specification of privacy policies giving concrete guidance to improve their quality. RSLingo4Privacy includes several processes (supported by respective tools),

namely:

- P1: automatic text classification and extraction;
- P2: visualization and authoring;
- P3: analysis and quality validation; and
- P4: (re)publishing.

RSLingo4Privacy is a multi-language approach that uses the following privacy-aware languages: RSL-IL4Privacy (introduced and discussed in Chapter 5) and Eddy (previously introduced in Chapter 2). Figure 4.1 overviews the RSLingo4Privacy approach as a top-level Business Process Modeling and Notation (BPMN) diagram.

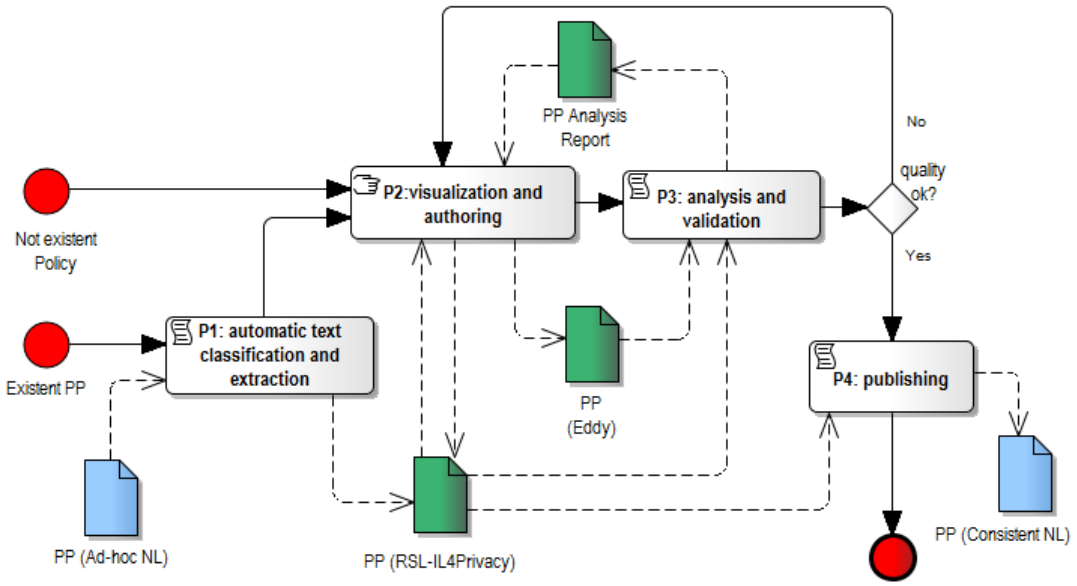

Figure 4.1: The RSLingo4Privacy Approach (adapted from [2])

4.2.1 The RSLingo4Privacy Workflow

If a given ad-hoc natural language policy exists, the process P1 applies complex text classification and text extraction techniques to automatically produce the equivalent specification in RSL-IL4Privacy (P1 is further discussed in Chapter 6). In addition or otherwise, if that policy does not exist, the RSLingo4Privacy approach starts directly with process P2. P2 allows policy authors to visualize and author the new privacy policy in a rigorous and consistent way based on the RSL-IL4Privacy language. Additionally, P2 takes the RSL-IL4Privacy specification and converts it to an equivalent Eddy specification to perform further analysis and validation (P2 is further discussed in Chapter 8).

Process P3 takes as input both RSL-IL4Privacy and Eddy specifications, and provides analysis and validation features, producing, for example, an analysis report with errors and warnings that can be taken into consideration during these authoring and validation processes.

Finally, when the quality of the policy specified in RSL-IL4Privacy is appropriated, the process P4 is responsible for producing an improved version of the policy, specified again in natural language but in a more consistent and high-quality manner.

The entire RSLingo4Privacy approach (including all its processes) is supported by a tool named RSLingo4Privacy-Studio. The RSLingo4Privacy-Studio and the transformations in which the tool relies are discussed in Chapter 7.

The focus of this dissertation is the definition of the RSL-IL4Privacy language for specifying privacy requirements of privacy policies, as well as the development of various NLP-based solutions to support the process P1. Additionally, the mapping method between RSL-IL4Privacy and Eddy (process P2) is detailed and the results for five case studies obtained from the automatic validation using the Eddy engine (a simple run of the process P3) are also discussed.

5

The RSL-IL4Privacy Language

An important activity when developing and integrating complex software systems, such as social networks, is the enforcement of their privacy policies. One can argue that an incomplete and imprecise specification of privacy requirements may compromise the process of analysis and reasoning of such privacy policies.

RSL-IL4Privacy is a language that enables a more rigorous representation and specification of privacy requirements than writing natural language policies alone. RSL-IL4Privacy is defined as a DSL because it allows [85]: (i) the possibility of expressing solutions in the idiom and with the level of abstraction of the problem domain; (ii) increasing the solution's portability, maintainability, reliability and productivity; and (iii) the prospect of reusing domain knowledge in future solutions. In addition, the adoption of a language such as RSL-IL4Privacy allows that its specifications become simpler to read and understand which facilitates the communication between system developers and domain experts [86].

This chapter is organized as follows. First it describes the abstract syntax of the proposed language, including examples for each one of the RSL-IL4Privacy's constructs. Finally, the chapter provide some remarks about the RSL-IL4Privacy's concrete syntax and discusses the overall impact of this language.

5.1 Abstract Syntax

The abstract syntax of the RSL-IL4Privacy is defined with a metamodel based on a Unified Modeling Language (UML) diagram. The use of a graphical notation such as UML provides a straightforward way of representing the attributes that define the elements (i.e., constructs) and the associations between different elements. It provides the reader with a simple view of what the concepts are and how they are related to each other. Additionally, and based on the given metamodel, an Xtext[1] grammar (like a set of BNF-like rules) is defined in order to formally describe the RSL-IL4Privacy's constructs and their relationships. The complete grammar, which enables the concrete textual syntax, is given in Appendix A. We provide an example for each construct using the RSL-IL4Privacy's concrete syntax. To provide a more comprehensive *running example* across the subsections, the content of all textual snippets comes from Facebook's privacy policy, one of the policies that were analyzed under the scope of the RSLingo4Privacy project, specifically for the IE approach.

5.1.1 PrivacyPolicy

The `PrivacyPolicy` construct is the RSL-IL4Privacy's "package" as it outlines the scope for the existence of all the remaining constructs. The metadata that characterizes a privacy policy is specified using this construct. Figure 5.1 illustrates the attributes of the `PrivacyPolicy` construct.

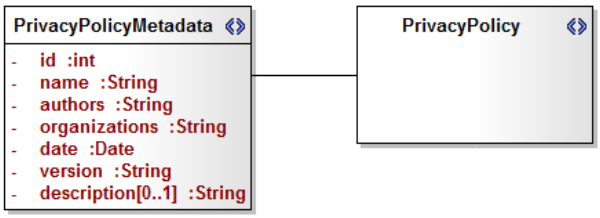

Figure 5.1: The `PrivacyPolicy` construct metamodel

- `id` - The unique identifier of each `PrivacyPolicy`.
- `name` - The name (i.e., the title) of a `PrivacyPolicy`. It is usually the name of the company or service to which the policy applies.
- `authors` - Describes who wrote and maintains a `PrivacyPolicy`.
- `organizations` - The organizations that are governed by a `PrivacyPolicy`. Even though a policy is usually for a specific company, there is also the case when a company (e.g., Microsoft) - that provides various services and applications - chooses to keep a single policy for all its products.
- `date` - The date in which the `PrivacyPolicy` was defined or last updated.

[1] Xtext framework, https://eclipse.org/Xtext

- `version` - The version of the last update is also included with `date` in a given policy.
- `description` - If applicable, the `PrivacyPolicy`'s description. It could be a brief sentence describing the policy or include other more relevant information (e.g., comments on its size). The policy authors have some flexibility to include what they see fit.

To assist the reader with a better understanding of the aforementioned attributes, Listing 5.1 includes a textual snippet documenting the specification of a `PrivacyPolicy` with RSL-IL4Privacy's concrete syntax.

Listing 5.1: Example of a `PrivacyPolicy` specification

```
1  package 1 {
2      metadata=1;
3  };
4
5  privacyPolicyMetadata 1 {
6      name "Twitter"
7      authors "Twitter, Inc."
8      organizations "Twitter"
9      date 18-May-2015
10     version "3.0"
11     description "Twitter's Privacy Policy"
12 };
```

5.1.2 Statement

As described in the previous chapter, a privacy policy is no more than a document consisting of natural language sentences. In RSL-IL4Privacy, we view these sentences as statements. However, each statement can include more than one sentence. If the meaning of a sentence is dependent on the previous or next sentence(s), all sentences are included to encompass a single statement. As a guideline, whenever a sentence begins with or contains a determiner (e.g., "this", "these", "that", "those") or a personal/possessive pronoun (e.g., "it", "its") in the following words, we group all such sentences as only one statement. In addition, sentences beginning with expressions like "For example" or "In particular" follow the same pattern. Since a RSL-IL4Privacy policy is an intermediate specification between natural and more formal language, it is mandatory not to lose the context across the policy. Each statement needs to encapsulate all the information it needs to be understood on its own.

Bearing these properties in mind, a `Statement` describes, in a formal manner, what rules or actions are specified in a `PrivacyPolicy`. Thus, a `Statement` can be seen and called as a *privacy requirement*.

Figure 5.2 depicts the attributes of the `Statement` construct.

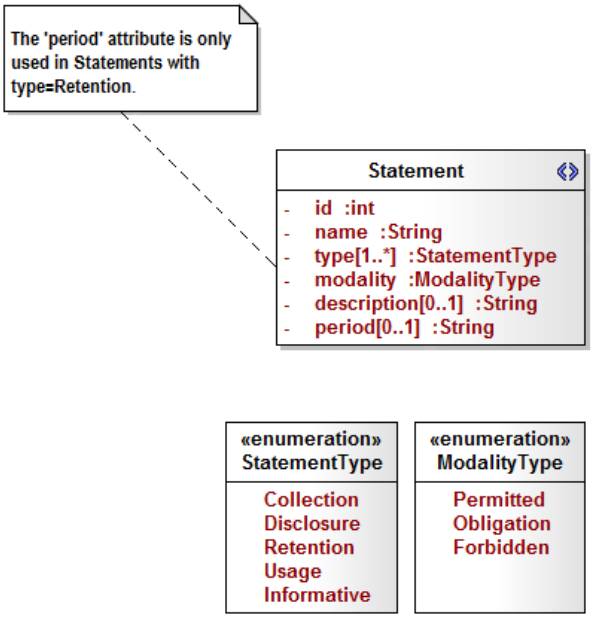

Figure 5.2: The `Statement` construct metamodel

- `id` - The unique identifier of each `Statement`.
- `name` - The "name" of a `Statement`. It can be as simple as the author desires and it can be seen as a `string` identifier of the `Statement` that allows the referral of such statement.
- `type` - Depending on the data practices a `Statement` entails, it is possible to define five different categories, namely:
  1. Collection: "Collection" statements define which data is collected by the service provider.
  2. Disclosure: "Disclosure" statements define which data is disclosed and eventually to what entities.
  3. Retention: "Retention" statements define for how long the data will be stored by the service provider:
  4. Usage: "Usage" statements define the reason for having the data.
  5. Informative: "Informative" statements do not provide critical data practices only generic information.

  It is noteworthy that, according to the information it conveys, a `Statement` can be labeled with more than one type (e.g., "Collection, Disclosure"). On the other hand, and due to the need of maintaining the context, some statements may also be classified as "Informative" aside from one of the other types.

- `modality` - For the purpose of further reasoning about the policies, namely in what concerns *deontic logic* reasoning, each `Statement` is also defined in terms of its deontic modality (i.e., "Permitted", "Obligation" or "Forbidden") depending on the actions they describe.
- `description` - If applicable, the `Statement`'s description. For the purpose of our study, the description of a `Statement` encompasses the natural language text obtained from the original policy (i.e., the original sentence(s)).
- `period` - Only applicable to retention-type statements. It specifies the period of time that a service provider retains data about the users.

The `Statement` is the core construct of RSL-IL4Privacy due to the information it encloses. It is the information present in each `Statement` that call for the definition of the remaining constructs of this section. To assist the reader with a better understanding of the aforementioned attributes, Listing 5.2 includes a textual snippet documenting the specification of a `Statement` with RSL-IL4Privacy's concrete syntax. Note that the `name` of the `Statement` is only the abbreviation "st" concatenated with the `id`. In addition, the reader should also observe the `type` as "Disclosure" and that its `modality` attribute has a value of "Permitted", which allows the disclosure of information. Both the `id` and the `type` are declared in the *header*.

Listing 5.2: Example of a `Statement` specification

```
statement 27 : Disclosure {
    name "S27"
    modality Permitted
    description "We may also share information when we have a good faith belief
                 it is necessary to detect, prevent and address fraud and other
                 illegal activity."
};
```

5.1.3 Recipient

Disclosure-type statements are concerned with the sharing of user data. The improper disclosure of information poses a potential threat to the safety of the users as well as to the company providing the service. Bearing these concerns in mind, it is of the utmost importance to provide a clear specification of the entities that receive the information from the service provider. Also, more generic recipients may aggregate other recipients. The `Recipient` construct describes such entities with their relevant and distinguishable attributes. Figure 5.3 illustrates the attributes of the `Recipient` construct.

- `id` - The unique identifier of each `Recipient`.

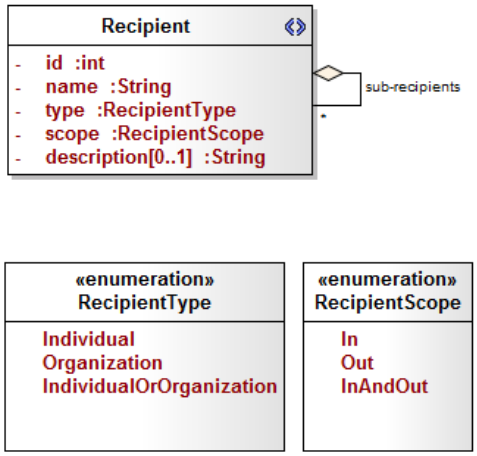

Figure 5.3: The `Recipient` construct metamodel

- `name` - The name of each `Recipient`. Similar to the `Statement` construct, the name of a `Recipient` can be seen as a simple `string` identifier (e.g., "R1") but, on the other hand, it can also convey - in an explicit manner - the name of the entity that receives the information ("e.g., Payment Processor").

- `type` - Aside from the name or identifiers, it is important to enrich the `Recipient` construct with other properties. The `type` of a `Recipient` distinguishes between individuals and organizations. The most common `type` is "Organization" as it designates the third-parties that support directly or not the business of the service provider. Nevertheless, whenever the `type` of a `Recipient` is not explicitly defined in a `PrivacyPolicy` (e.g., sharing data with "anyone"), we assume both types - "IndividualOrOrganization" - for such `Recipient`.

- `scope` - The scope of a `Recipient` discriminates if it is internal ("In") with regard to service provider, external ("Out") or both ("InAndOut"). The `scope` is an important attribute with regard to the detection and analysis of inappropriate disclosure of user data. Usually, an *internal* `Recipient` is an entity (usually a `Recipient` of `type` "Organization") that is part of the family of companies of the service provider, which means that exchanging user data with such companies is less dangerous and common practice. On the contrary, sharing user's personal information with an *external* `Recipient` may lead to serious privacy breaches.

- `description` - If applicable, the `Recipient`'s description. If the author chooses to use the `name` attribute as a simple identifier, the actual "name" of the `Recipient` is given here.

To assist the reader with a better understanding of the aforementioned attributes, Listing 5.3 includes a textual snippet documenting the specification of a `Recipient` with RSL-IL4Privacy's concrete syntax.

Listing 5.3: Example of a `Recipient` specification

```
recipient 3 : Organization {
    name "Third-party"
    scope Out
    description "Third-party organization(s) that - external - support Facebook"
};

recipient 7 : Organization {
    name "service-providers"
    scope Out
    partOf 3
    description "Third-parties that provide services to Facebook"
};
```

5.1.4 PrivateData

The `PrivateData` construct describes the user's personal information that is specified throughout a `PrivacyPolicy`. In other words, each `PrivateData` provides a thorough description of the different bits of private information that a service provider may collect, disclose, retain and use. However, not all user information raises the same kind of privacy concerns, thus it is important to characterize such data in different levels in order to apply a proper . It is also noteworthy that a `PrivateData` can be as simple or as complex as the information that is documented in a `PrivacyPolicy`. The "name" or the "e-mail address" can be themselves a single `PrivateData` or we can have a `PrivateData` with several attributes, including the "name" and "e-mail address". Figure 5.4 depicts the attributes of the `PrivateData` construct.

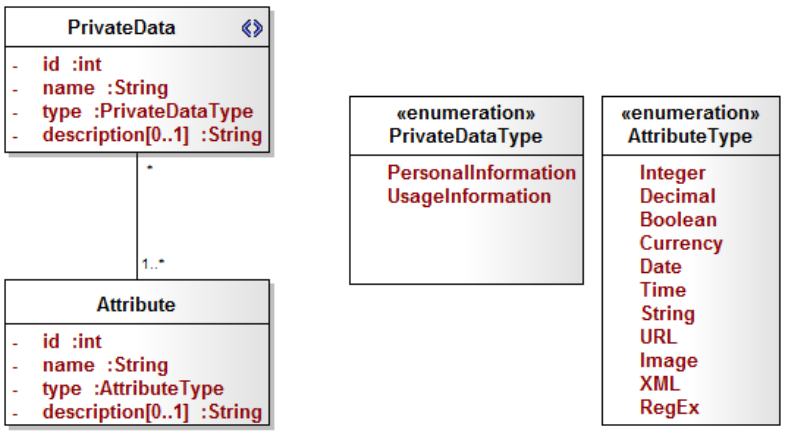

Figure 5.4: The `PrivateData` construct metamodel

- `id` - The unique identifier of each `PrivateData`.

- `name` - Like the `name` attribute of the `Recipient` construct, the `name` of a `PrivateData` can be a basic textual identifier or, if its meaning is rather self-explanatory, it can be the direct designation of the data (e.g., "user name").
- `type` - Even though `PrivateData` constructs encompass personal information about the users, there is a difference in terms of the *sensitivity* of such information. We define two different types for the `PrivateData` construct: "PersonalInformation" and "UsageInformation". "PersonalInformation" includes all information that can identify/contact a given user (otherwise known as "Personal Identifiable Information" [87]) such as name, email or home address, but also other kinds of sensitive information like payment-related data. Additionally, and although such bits of information are usually meaningless on their own, we also combine attributes such as "age" or "gender" as `PrivateData` of `type` "PersonalInformation". On the other hand, "UsageInformation" comprises aggregated data that arises from the use of the service itself (e.g., what content the users view the most) and also technical data from computers or mobile devices (e.g., IP address, computer operating system, app version). This information is often used to infer certain behavior patterns for accommodating marketing and advertising strategies but does not identify or allows the service provider or its partners to contact the users.
- `description` - If applicable, the `PrivateData`'s description. Similar to the `Recipient` construct, if the author chooses to use the `name` attribute as a simple identifier, the actual "name" of the `PrivateData` is given here.

5.1.4.1 Attribute

The `Attribute` construct is a flexible but powerful approach for specifying the attributes of a `PrivateData` construct. There is a wide range of possible types for each one of the `PrivateData`'s attributes, thus there may be the need to enrich an attribute with additional information. With this construct, the set of attributes is specified in a comprehensive and complete manner.

- `id` - The unique identifier of each `Attribute`.
- `name` - The `name` of each `Attribute`.
- `type` - The `type` of each `Attribute`. The possible values for the `type` are: "Integer", "Decimal", "Boolean", "Currency", "Date", "Time", "String", "URL", "Image", "XML" and "RegEx".
- `description` - If applicable, the `Attribute`'s description. It can be used to supplement the `Attribute` with additional information.

To assist the reader with a better understanding of the aforementioned attributes, Listing 5.4 includes a textual snippet documenting the specification of a `PrivateData` with RSL-IL4Privacy's concrete syntax.

To make it different from the previous `Recipient` example, we provide a `PrivateData` specification with the `name` as an identifier. Also, the reader should pay attention to the importance of having a `description` for each `Attribute`. For the examples with `id` 3 and 4 (the "RegEx" `type`) additional information is required (in this case, examples of patterns), so we make use of the `description` field to include that information.

Listing 5.4: Example of a `PrivateData` specification

```
privateData 1 : PersonalInformation {
    name "Account"
    description "The attributes that comprise a basic Facebook account"

    attribute 1 : String {
        name "first name"
    };
    attribute 2 : String {
        name "surname"
    };
    attribute 3 : RegEx {
        name "email"
        description "pretty simple email validation:
                    [A-Za-z0-9_-]+[@][A-Za-z]+[.][a-z]{2,3}"
    };
    attribute 4 : RegEx {
        name "mobile number"
        description "US phone numbers: \d{3}[-]\d{3}[-]\d{4}"
    };
    attribute 5 : String {
        name "password"
    };
    attribute 6 : Date {
        name "date of birth"
    };
    attribute 7 : String {
        name "gender"
    };
};
```

5.1.5 Service

The different data practices and data flows documented in each `Statement` (illustrated by its `type`) usually have a reason, a purpose for them to exist. For instance, a service provider may collect user information to perform simple tasks like enriching the profile of the users or it may need to disclose information to third-parties to be able to support its services.

The `Service` construct is designed to describe these actions performed by the provider. A `Service` can aggregate various smaller sub-services. This allows a more refined specification of the purposes employed by the service provider when handling and managing users' personal information. The reason why the `Service` is designed in this way is due to the fact that further automatic validation and processing (e.g., conflict detection) requires the definition of almost atomic services to ensure quality results. Thus, while it might be useful to look for major services within a `PrivacyPolicy`, a set of smaller services ensure, for example, a proper conflict detection. Statements like "We do not share your data in order to send promotional communications" and "We share your data to track the effectiveness of our ads" are not directly conflicting. Even though they refer to the same high-level `Service` – Advertising –, the purpose for the sharing is clearly different. Therefore, we would have two sub-services: S1 - "send promotional communications" and S2 - "track the effectiveness of our ads". Figure 5.5 illustrates the attributes of the `Service` construct.

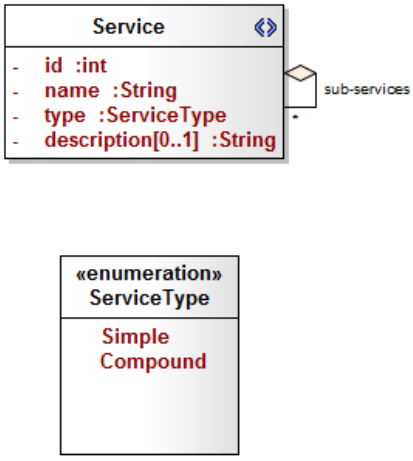

Figure 5.5: The `Service` construct metamodel

- `id` - The unique identifier of each `Service`.
- `name` - Like the `name` attribute of the `Recipient` and `Service` constructs, the `name` of a `Service` can be a simple textual identifier or, if it is a high-level `Service`, it can be its direct designation (e.g., "Advertising").
- `type` - The `type` of each `Service`. This attribute only distinguishes between "Simple" and "Com-

pound" services. We decided to include this attribute (which is not present in the `Recipient` construct even though it can also aggregate smaller recipients) in order to keep track of all the associations between the multiple services and sub-services.

- `description` - If applicable, the `Service`'s description. Similar to the `Recipient` and `PrivateData` constructs, if the author chooses to use the `name` attribute as a simple identifier, the actual "name" of the `Service` is given here.

To assist the reader with a better understanding of the aforementioned attributes, Listing 5.5 includes a textual snippet documenting the specification of a `Service` with RSL-IL4Privacy's concrete syntax. As opposed to the examples regarding the `Recipient` and `PrivateData` constructs, we provide the specification of a `Service` using the `name` attribute as identifier. The "Simple" value for the `type` attribute indicates that this `Service` has no sub-services (the `partOf` demonstrates that this `Service` is itself a sub-service).

Listing 5.5: Example of a `Service` specification

```
1 service 23 : Simple {
2     name "SV23"
3     partOf 16
4     description "conduct audits"
5 };
```

5.1.6 Enforcement

The `Enforcement` construct is one of the novel contributions of RSL-IL4Privacy. The `Enforcement` comprises the description of the mechanisms and tools that are documented in a policy and that allow one to gain insight about how it can be possible to enforce the *privacy requirements* of such privacy policies. On the other hand, it also encompasses rules and specific actions with regard to the use of the system that are important for the enforcement of a given privacy policy. As opposed to what happens in real-world privacy policies where such information is often scattered throughout the policy, this construct provide policy authors with the possibility of clearly identify and describe this *enforcement* information. Figure 5.6 depicts the attributes of the `Enforcement` construct.

- `id` - The unique identifier of each `Enforcement`.
- `name` - The name of each `Enforcement`.
- `type` - The type of each `Enforcement`. The possible values for the `type` are: "Action", "Algorithm", "Config", "Process" and "Tool".

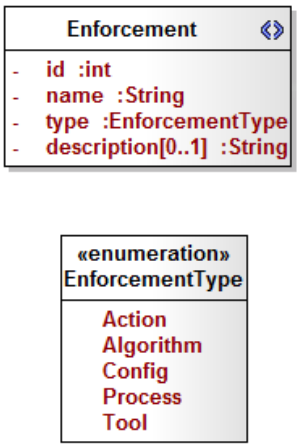

Figure 5.6: The `Enforcement` construct metamodel

- `description` - if applicable, a brief comment regarding the defined `Enforcement` construct.

As opposed to the previous constructs, the `name` attribute is not used as an identifier. It always designates the "name" of an `Enforcement` construct independently of its `type`. The reason that supports this design concern is due to the fact that some of the types (e.g., tools) require an illustrative comment which is provided with the `description` attribute.

To assist the reader with a better understanding of the aforementioned attributes, Listing 5.6 includes a textual snippet documenting the specification of a `Enforcement` with RSL-IL4Privacy's concrete syntax.

Listing 5.6: Example of a `Enforcement` specification

```
1 enforcement 2 : Tool {
2     name "Download Your Information tool"
3     description "Through this tool, users are able to download information
4                  associated with their Facebook account"
5 };
```

5.1.7 The Complete Metamodel

To understand the relationships between the aforementioned constructs, the complete RSL-IL4Privacy metamodel is given in Fig. 5.7. For the sake of simplicity, all the enumerations and "secondary" constructs such as `PrivacyPolicyMetadata` and `Attribute` are omitted.

The `Statement` is the main construct in RSL-IL4Privacy since the information it contains is used to define the remaining constructs afterwards. As a consequence, each `Statement`, in addition to the attributes presented and discussed in Section 5.1.2, also lists the `id` of the other constructs to which it is associated. For disclosure-type statements it also provides the `id` of the recipients linked with those

statements. Listing 5.7 supplements the example presented in Section 5.1.2 (see Listing 5.2) with the missing relationships that were just introduced.

Listing 5.7: Example of a complete `Statement` specification

```
statement 27 : Disclosure {
    name "S27"
    modality Permitted
    description "We may also share information when we have a good faith belief
                 it is necessary to detect, prevent and address fraud and other
                 illegal activity."

    recipient "All"
    privateData "All"
    service 5
};
```

On the other hand, the `Service` and `PrivateData` constructs also have a separate association. The reason for such connection deals with the fact that it is often important to keep track of what personal information is acted on by a certain service. In other words, it may be useful to know what part of the users' private data is needed and why (the "purpose").

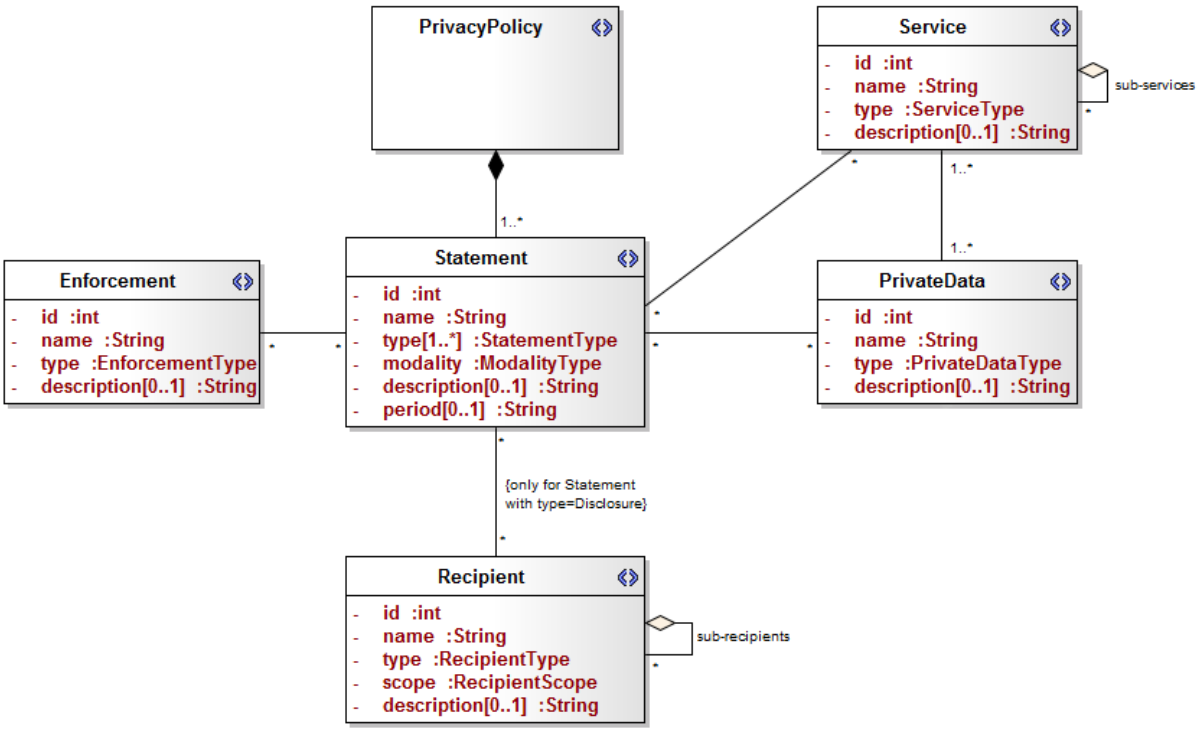

Figure 5.7: The RSL-IL4Privacy metamodel

5.2 Concrete Syntax

The RSL-IL4Privacy language is designed with the final purpose of specifying and documenting the privacy-related requirements (i.e., the statements) of a privacy policy. However, the second process of RSLingo4Privacy - P2: visualization and authoring - also allows policy authors to write new policies, in addition to the case when an existing natural language privacy policy already exists. The first condition emphasizes the importance of having a specification language with a simple syntax and easy to use. Moreover, clear and comprehensive RSL-IL4Privacy specifications can also be provided to end-users, so that they can be able to keep track of the different practices and private data flows of the systems they use.

On the other hand, as observed in the examples discussed throughout the previous Section, the syntax of RSL-IL4Privacy allows an easy mapping to other formal languages in order to enable further validation and reasoning about its own specifications. Also, RSL-IL4Privacy's concrete syntax has some degree of similarity with object-oriented programming languages such as Java that may increase the adoption of RSL-IL4Privacy as a specification language for privacy policies. However, further validation is required to prove or reject such claim. Lastly, it is also noteworthy that RSL-IL4Privacy's concrete syntax can be complemented with different artifacts for other representations, namely graphical and tabular models.

5.3 Discussion

In this section we have presented RSL-IL4Privacy, the language that supports RSLingo4Privacy and that allows the specification and further documentation of the privacy requirements of privacy policies. RSL-IL4Privacy is designed to deal with the gap between natural language and other more formal languages. On the one hand, natural language itself - due to its characteristics - makes it difficult to have validation and reasoning processes with regard to the content that is presented in natural language requirements (from the policies); on the other hand, formal languages enable formal computations but, at the same time, cannot convey the same information as natural language and their concrete syntax may . Statements in RSL-IL4Privacy include the associations with the remaining constructs (therefore, enabling the possibility of performing some kind of validation). In addition, statements also encompass the original text from the privacy policy, hence giving policy authors with the ability of - dynamically - modifying the original policies in order to solve inconsistencies ans conflicts.

Additionally, RSL-IL4Privacy provides the necessary constructs for the specification of current privacy policies. Aside from some common elements (e.g., "recipient", "datum", "service"), the `Enforcement` construct includes important and different information (e.g., rules, tools, etc.) that does not fall into any other constructs but that may have a major role in the enforcement of policies. At the same time, the abstract syntax of RSL-IL4Privacy facilitates the extension and creation of new constructs. Due to

the fact that the values of the attributes that define the constructs are specified using enumerations, the process of enriching the content and scope of such constructs with upcoming and different concerns is relatively easy.

Lastly, alongside with the advantages that come from the use of RSL-IL4Privacy language both for policy authors and even developers that need to be aware of the different data practices and flows that occur in the systems they may need to integrate, the end-users of such systems may also reap benefits. With RSL-IL4Privacy specifications, end-users are able to identify in a clearer way (when compared to natural language) which personal information is, in fact, being managed by the service provider through the use of a language that has a concrete syntax easier to learn than the syntax of other formal languages.

6

Automatic Classification and Extraction

The automatic classification and extraction of relevant information (P1 - the first process of the RSLingo4Privacy approach) aims at accomplishing the mapping between original, existing privacy policies and RSL-IL4Privacy specifications. In other words, it encompasses the automatic task of getting the necessary information from existing policies and populating equivalent RSL-IL4Privacy specifications with their mandatory elements that describe the same information. To fulfill this process we rely on two separate tasks: classification of statements and extraction of information for RSL-IL4Privacy constructs.

As surveyed through the previous chapters, a privacy policy is a type of technical document that comprises a set of privacy-related requirements defined using ad-hoc natural language sentences. In addition, the document itself has no well-established guidelines with regard to its structure and organization. Bearing these characteristics in mind, a solution based on the definition of a language of linguistic patterns such as RSL-PL as the answer for the automatic classification problem is ill-advised: aside from the challenging task of defining the necessary patterns to achieve sufficiently good results, it still might not scale to larger data sets (i.e., a larger number of policies). However, despite these difficulties, the use of patterns is still part of many solutions for text classification problems [88].

The second part of the P1 - the extraction of relevant information - is even harder to carry out using only linguistic patterns. Since we are looking for very specific words and expressions, with its meaning depending on the context (e.g., not all "third-parties" occurrences represent a Recipient), defining the necessary patterns to recognize such entities is truly demanding and impractical.

This chapter is organized as follows. First it describes RSL-PL4Privacy, a *language* that defines patterns on a syntactic-level only under the scope of RSLingo4Privacy and relates this language to its predecessor, RSL-PL. Section 6.2 provides the description of the classification step of P1. It presents various experiments for different solution which build on top of each other using RSL-PL4Privacy, therefore discussing its limitations and improvements. Lastly, Section 6.3 overviews the remaining part of P1, the extraction step. It discusses alternative solutions (with different levels of granularity) for the NER problem of RSLingo4Privacy's P1.

6.1 RSL-PL4Privacy

RSL-PL, the *pattern-language* responsible for extracting relevant information from SRSs and mapping it into RSL-IL specifications, is designed to cope with the eventual variations of the linguistic patterns present in each SRSs' natural language requirement. However, we argue that a similar approach in the case of statements of a privacy policy is hardly feasible both for the classification as well as the extraction steps. The statements of a privacy policy can be extremely long, with no "fixed" structure (e.g., lists and/or tables) and with a wide range of vocabulary. These aspects combined (the length and poor syntactic structure of a statement) prove the complexity of defining a set of rules complete enough for recognizing a) the type of the statements; and b) other relevant elements within statements. Regardless of the aforementioned limitations, we still wanted to assess the potential of using patterns to perform the classification of statements. The definition of such patterns had to be as simple as possible to ensure the solution's scalability.

To come up with a set of patterns that are able to recognize the type of a Statement, we make no attempt at defining rules that can "consume" the entire Statement: we only look for "fragments" that are powerful enough to recognize its type. This means that we search for simple patterns based on the syntactic structure of a Statement (e.g., Part-Of-Speech (POS) tags) along with some specific *action verbs* for each type. Action verbs designate verbs that are associated with specific actions related to some sort of data. For the purpose of this dissertation, these specific actions represent the type of a Statement (e.g., action verb - "collect" $\rightarrow$ type - "Collection"). Breaux et al. [14] identified the most common action keywords and phrases in privacy policies. We define a revised set of six action verbs for each type. These verbs are shown in Table 6.1.

If having an extensive list of action verbs, one might tempted to write boolean rules (i.e., regular expressions) indicating the presence or absence of a given term, and assigning the type of the Statement

Table 6.1: Action verbs for each type of Statement

Collection	Disclosure	Retention	Usage
collect	share	retain	use
receive	disclose	store	access
obtain	transfer	keep	scan
provide	provide	delete	monitor
give	send	record	analyze
send	transmit	preserve	process

based on such results. However, as observed from Table 6.1, there are verbs that are used for both types of statements, hence the need of specific patterns to distinguish between types. Listing 6.1 provides an example that shows how the use of boolean rules, based on even the most common action verbs, might lead to incorrect results. Listing 6.1 includes two statements that contain the word "use", *main* action verb in "Usage" statements.

Listing 6.1: Statements with different types including the action verb "use"

```
1 st1 (Usage): We may also use personal information for internal purposes such
2              as auditing, data analysis, and research to improve Apple's
3              products, services, and customer communications.
4 st2 (Informative): When your personal data is stored by Apple, we use computer
5                    systems with limited access housed in facilities using
6                    physical security measures.
```

Even though both statements contain this word - in fact, both statements also match the pattern "We/we use" which is more than the standalone presence of the action verb - they are of different types. st1, a "Usage" statement, describes the purposes for which the service provider uses the personal information of its users, whereas st2, an "Informative" statement, only gives information about the computer systems it uses.

6.1.1 RSL-PL4Privacy: Patterns

Using the action verbs from Table 6.1, we defined a set of patterns that can be used to identify a statement. These verbs may be in the infinitive form (as shown), or in the present simple/past participle, depending of the pattern. Listing 6.2 provides a couple of pattern examples for statements of type "Collection", "Disclosure", "Retention" and "Usage". Due to space constraints, the complete set of patterns is given in Appendix B. It is worth to mention that, in order to provide a better understanding, we choose to keep our patterns as simple as possible even if that meant writing more patterns. The patterns in Appendix B can be obviously combined, therefore generating a small set of patterns.

Lastly, to provide some sort of consistency with RSLingo, in particular with RSL-PL, we used a similar "notation" for the patterns. For our patterns we adapted RSL-PL for the following necessary

elements: W(...) represents a word (a generic string), W("example") represents a specific word and W(POS) represents a POS tag. The POS tags that were needed are: N - noun, NP - proper noun V - verb, PRP - personal pronoun, PRP$ - possessive pronoun, MOD - modal verb, ADV - adverb, ADJ - adjective and DET - determiner.

Whenever the meaning of a certain pattern is simplified by the use of a specific a word instead of a POS tag, we choose to use the specific word (e.g., negative statements: "not" instead of "ADV"). Additionally, it may be mandatory to discriminate the specific word instead of using the POS tag (e.g., pronoun "we" instead of "you"). If more than one word or POS tag applies, we follow the upcoming convention: W("example1"—"example2") and W(POS1—POS2). Finally, if all action verbs for a certain type can be used in a given pattern, we use W(action-verbs-typeX); if not, we discriminate the verbs explicitly.

Listing 6.2: Examples of RSL-PL4Privacy patterns for each statement type

```
1  // Collection
2  W("We|we") W(MOD) W("collect|receive|obtain")      -> "We may collect"
3  W("You|you") W(MOD) W("provide|give|send") W("us") -> "you can provide us"
4
5  //Disclosure
6  W("We|we") W(MOD) W(ADV) W(action-verbs-Disclosure)    -> "We may also share"
7  W(N|NP) W(MOD) W(v) W("to") W(action-verbs-Disclosure) -> "Facebook may need to
8                                                              disclose"
9
10 //Retention
11 W("We|we") W(V|MOD) W("not") w(action-verbs-Retention) -> "We do/will not retain"
12 W(N|NP) W(action-verb-presentSimple-Retention)         -> "Facebook stores"
13
14 //Usage
15 W("We|we") W(ADV) W(action-verbs-Usage) W(DET|ADJ|PRP$) W(N|NP)  -> "We also scan
16                                                                     your data"
17 W("information" W(MOD) W(V) W(action-verbs-pastParticiple-Usage) -> "information
18                                                                     may be used"
```

From Listing 6.2 it is possible to see examples of different patterns (hence, of different types, in this case "Collection" and "Disclosure" statements) that use the same action verb. The fundamental difference between these two data practices in terms of the definition of patterns is the subject's point-of-view about the action described in the Statement. For "Collection" statements, action verbs such as "provide", "give" and "send" assume the user's perspective, i.e., these statements describe action(s) that a user is supposed to engage in (e.g., "you provide us with personal information"). On the contrary,

"Disclosure" statements encompassing with these action verbs are made with the provider's point-of-view, describing the actions of the provider with regard to an eventual disclosure of information. Thus, the subject of the "Disclosure" statement is a pronoun ("We/we") or a noun/proper noun (the actual name of the service provider).

These are the critical cases regarding the distinction of these different types. Using a small set of action verbs (six for each type) helps to reduce possible classification errors, as the statements with other common action verbs may be more ambiguous and its type difficult to discriminate using these simple patterns. Additionally, as described in the previous section, there is also the chance that these RSL-PL4Privacy patterns for "Collection", "Disclosure", "Retention" and "Usage" statements may conflict with "Informative" statements (for which we cannot define patterns).

6.2 Classification of Statements

Each Statement as a type attribute that can hold more than one value ("Collection", "Disclosure", "Retention", "Usage" and "Informative") as described in the previous Chapter. Thus, the task of classifying the statements of a given policy is a multi-label classification problem. Each Statement will be represented as a vector of statistical features V_s. The challenge of the classification task relies upon using a ML algorithm, which can be seen a "function" f that takes as argument the vector of features V_s and is then able to predict the type(s) $\{T\}$ of each statement: $f(V_s) \rightarrow \{T\}$.

Bearing this in mind, we evaluated our various solutions against the most common problem transformation methods (see Chapter 3): Binary Relevance (BR), Classifier Chains (CC), Ensemble of Classifier Chains (ECC), Label Powerset (LP), Random k-Labelsets (RAkEL) and Fourclass Pairwise (FW). Each method was tested using its default configurations.

Additionally, and for each one of the aforementioned methods, we also tested the solutions with different ML algorithms, namely Naïve Bayes (NB), Decision Trees (DT), k-Nearest Neighbors (kNN) and Support Vector Machines (SVM). All experiments were carried out using MEKA software and 10-fold cross validation. The data set is split in 10 disjoint sets of equal size and, in each trial, 9 sets are used to train and 1 to test. This helps to simulate the performance of the model in a real-world scenario.

6.2.1 The Data Set

As also discussed through Chapter 3, using supervised learning requires a data set labeled beforehand in order to train the classifier afterwards. The experiments carried out in this section were performed using a data set of 1759 manually labeled statements. These statements were retrieved from 20 different privacy policies of well-known companies across various domains (e.g., social networking sites, booking accommodation platforms). The process of gathering the statements followed several considerations:

- We did not include introduction sections also known as "privacy statements";
- We did not include the titles of the sections;
- Some privacy policies also include a cookie policy. We included this sections because they may provide useful information;
- Lists and tables were converted into plain text and its capitalized letters and other symbols were kept.

The distribution of the statements among the five different types, for each service provider, is shown in Table 6.2.

Table 6.2: The statement type distribution for each service provider

	Collection	**Disclosure**	**Retention**	**Usage**	**Informative**	**Multi**	**Total**
Accuweather	6	10	1	0	63	6	86
Airbnb	2	9	1	2	110	10	134
Apple	6	7	1	6	55	6	81
Booking	8	7	0	11	99	0	125
Dropbox	3	2	2	0	28	2	37
Evernote	15	5	4	2	75	2	103
Facebook	10	5	3	8	32	6	64
Google	6	7	45	1	4	2	65
IMDb	4	7	1	3	43	5	63
LinkedIn	11	11	9	7	127	8	173
Netflix	3	4	0	1	48	1	57
PayPal	8	10	2	3	41	3	67
Snapchat	17	7	4	1	39	1	69
Spotify	11	9	0	10	100	5	135
Twitter	13	10	6	9	50	4	92
Uber	12	3	1	1	26	3	46
Walmart	10	6	0	2	75	2	95
WebMD	5	8	3	4	102	4	126
WhatsApp	2	5	11	6	37	3	64
Yelp	5	11	5	1	47	8	77
Total	157	143	58	81	1239	81	1759

As it can be observed from Table 6.2, there is not a significant number of multi-label statements in our data set. We introduce two measures that evaluate the extent to which the data set is multi-label [54]. Label Cardinality (LC) measures the average number of labels of the examples in the data set, whereas Label Density (LD) corresponds to the average number of the examples in the data set divided by the number of labels. For our data set, this accounts for the following values: LC $= 81/1759 = 0.048 \rightarrow 1.048$ average labels for each examples in the data set; and LD $= 1.048/|L| = 1.048/5 = 0.21$. These metrics verify the low "multi-labelledness" [56] of our data set.

6.2.2 Experiment A: Solution with RSL-PL4Privacy only

Our first proposal was a preliminary solution based only on the syntactic patterns defined in RSL-PL4Privacy. The five features used both for training and testing the classifier represent the five possible types for each Statement and the values of the features indicate if a certain statement matched at least one of the patterns for a given type. More specifically, if a statement is a match to one of the patterns, it

gets a value of "1.0" for that type and a value of "0.0" otherwise. Additionally, and since we are not able to come up with specific patterns for the "Informative" type, its feature gets a value of "1.0" if none of other patterns are a match. The effectiveness of this preliminary solution was only measured according to a multi-label Accuracy measure [89] that evaluates how close a true set of labels is to a predicted set of labels, i.e., the proportion of predicted "correct" labels (true positives and true negatives) to the total number of labels. The final Accuracy is the average across all examples from the data set.

6.2.2.1 Results

The RSL-PL4Privacy solution works surprisingly well despite all the discussed limitations of relying solely on patterns for the classification of statements. The performance (Accuracy) for the RSL-PL4Privacy solution is shown in Table 6.3.

Table 6.3: The performance for the classifier with RSL-PL4Privacy only

	BR	**CC**	**ECC**	**LP**	**RAkEL**	**FW**
NB	0.777	0.766	0.77	0.777	0.775	0.775
DT	0.776	0.778	0.775	0.772	0.769	0.774
kNN	0.769	0.778	0.776	0.778	0.77	0.772
SVM	0.776	0.777	0.776	0.778	0.772	0.772

All the methods provide roughly the same accuracy (77%) for all the ML algorithms tested. However, even though these results are quite promising, there is still room for improvement as no features based on the actual words of a statement are being used. At the same time, the classifier trained with this small set of features does not show any variation for the different problem transformation methods.

6.2.3 Experiment B: Solution with RSL-PL4Privacy + Words

Although the solution using RSL-PL4Privacy yields encouraging results, we still wanted to improve the performance of the classifier. Thus, we experimented a new solution that combined RSL-IL4Privacy and words as features in our training and testing sets. Each feature represents the TF-IDF weights [90] of its constituent words. This weight factor demonstrates the importance of a word for a given statement with regard to the entire set of statements. A TF-IDF value increases with the number of times a word appears in a statement (Term Frequency) but it is balanced by the number of times it appears in whole collection of statements (Inversed Document Frequency). For this particular problem, we did not consider words that occur in less than three statements. Additionally, we did not include a particular set of English stop words and words with less than two characters (single letters). Finally, we generated tri-grams for each word (a set of three consecutive words).

The aforementioned data pre-processing results in a total of 7203 features. Incorporating the features based on RSL-PL4Privacy added to a total of 7207 features. This is the number of features used to train and test the classifier.

6.2.3.1 Results

This solution is more complex and with a different set of features. Therefore, the effectiveness of this solution was evaluated according to more performance measures in addition to the Accuracy defined in the previous section. These evaluation measures [91] are considered the standard for multi-label classification:

- Hamming Loss (HLoss) - Hloss evaluates the fraction of classification errors of instance-label pairs (whether a relevant label is not predicted or a irrelevant label is predicted). The performance of a classifier is considered better as the value of HLoss tends to 0.
- Exact Match Ratio (MR) - MR, also known as 0/1 Loss, provides the fraction of instances whose labels match the prediction *exactly*. It is considered a harsh measure as it does not take into account partial-correctness (one or more labels predicted correctly out of a bigger set of labels).
- F1 (macro averaged) - As opposed to the Accuracy measure which is "example-based", "labed-based" measures evaluate the performance of each label individually and average the results across all labels. We use F1 (macro averaged) that is calculated using the same formula as in multiclass classification, except that, in this case (i.e., multi-label classification), both the Prediction and Recall are also obviously "label-based".

The performance for the solution combining RSL-PL4Privacy with word features is shown in Table 6.4.

Table 6.4: The performance for the classifier combining RSL-PL4Privacy + word features without feature selection

Alg.	Perf.	BR	CC	ECC	LP	RAkEL	FW
NB	Acc.	0.507	0.507	0.539	0.344	0.327	0.371
	HLoss	0.297	0.298	0.275	0.387	0.535	0.514
	MR	0.294	0.294	0.347	0.306	0.188	0.167
	F1	0.417	0.417	0.424	0.271	0.281	0.332
DT	Acc.	0.753	0.795	0.807	0.785	0.774	0.777
	HLoss	0.088	0.086	0.08	0.092	0.102	0.102
	MR	0.717	0.779	0.786	0.774	0.682	0.683
	F1	0.755	0.632	0.655	0.598	0.632	0.637
kNN	Acc.	0.721	0.721	Memory	0.721	0.721	0.721
	HLoss	0.12	0.12	out	0.12	0.12	0.12
	MR	0.719	0.719	of	0.719	0.719	0.719
	F1	0.255	0.255	bounds	0.255	0.255	0.255
SVM	Acc.	0.826	0.84	0.83	0.843	0.845	0.768
	HLoss	0.063	0.067	0.072	0.067	0.066	0.117
	MR	0.803	0.828	0.812	0.832	0.817	0.658
	F1	0.696	0.686	0.668	0.677	0.701	0.629

The results of this solution prove the initial theory which suggested that using words as features in combination with RSL-PL4Privacy patterns might provide better results. For certain methods (e.g., CC, LP), algorithms like SVM gave a value of accuracy that showed an increase of approximately 7% when compared to the higher results of the previous solution, for all methods and algorithms. kNN was the only algorithm to have exceeded the memory available. It occurred in the ECC method.

Regarding the algorithms that were evaluated with this set of features, SVM are the one delivering

higher values of accuracy for all the transformation methods, except FW in which DT showed an improvement of about 1%. However, for all the remaining performance metrics and methods (again, except for the values of F1 for the BR and FW methods, in which DT slightly outperforms SVM), SVM provide better results and from considerable distance. This supports the general consensus that considers SVM as the state-of-the-art algorithm for text classification [92].

6.2.3.2 Results with Feature Selection

Due to the high number of features, we performed feature selection to verify if the presence of irrelevant features might be harming the accuracy of the classifier. We used standard feature selection functions [93] such as Chi-Squared (CHI-SQ), Information Gain (IG) and Pearson's Correlation (CORR). For each measure, we experimented different number of features (ranging from 100 to 5000) and recorded the performance. Table 6.5 shows the results. Due to the space constraints, Table 6.5 only presents the most important values (number of features = 100, 500, 1000 and 5000) for all three feature selection measures. Lastly, since SVM provided the best results from afar, we only performed feature selection with this algorithm.

From Table 6.5 it is possible to observe that as feature selection becomes more aggressive (less number of features selected), the performance of the SVM decreases. This corroborates previous studies that claimed SVM are robust, that can handle high dimensions and tend consider a great number of features as *relevant* to perform the classification [94]. However, even though almost all methods perform equally or better without feature selection, for the case of the ECC method, using 5000 features selected through IG resulted in better values of all performance metrics (Acc.: 0.83 $\rightarrow$ 0.856, HLoss: 0.072 $\rightarrow$ 0.06, MR 0.812 $\rightarrow$ 0.835 and F1 0.668 $\rightarrow$ 0.707). All the values that represented an improvement when compared to the original classifier (without feature selection) are highlighted in table using **bold**.

6.2.4 Experiment C: Solution with Words only

The improvement on the results of the previous solution suggest that having a classifier with only word features may actually yield better results. To assess the impact of having only features related to the words - the content - that comprise a statement, we built a classifier with only word features (removing the RSL-PL4Privacy features). This decision yielded training and testing sets with 7203 features.

6.2.4.1 Results

As in the previous solution, the effectiveness of this solution based only on word features was evaluated according to the same set of performance measures: Accuracy, HLoss, MR and F1 (macro averaged). The performance for the solution with word features only is shown in Table 6.6.

Table 6.5: The performance for the classifier combining RSL-PL4Privacy + word features with feature selection measures and using SVM (partial view)

# Features	Measures	Perf.	BR	CC	ECC	LP	RAkEL	FW
100	**CHI-SQ**	**Acc.**	0.78	0.797	0.819	0.796	0.792	0.75
		HLoss	0.086	0.085	0.075	0.086	0.089	0.116
		MR	0.763	0.782	0.798	0.786	0.767	0.634
		F1	0.616	0.612	0.651	0.594	0.634	0.602
	IG	**Acc.**	0.779	0.797	0.823	0.799	0.792	0.75
		HLoss	0.086	0.085	0.074	0.085	0.089	0.115
		MR	0.762	0.781	0.801	0.789	0.767	0.633
		F1	0.611	0.613	0.656	0.598	0.634	0.601
	CORR	**Acc.**	0.78	0.796	0.823	0.791	0.794	0.749
		HLoss	0.086	0.086	0.074	0.087	0.087	0.116
		MR	0.763	0.781	0.8	0.781	0.769	0.628
		F1	0.612	0.613	0.659	0.596	0.627	0.602
500	**CHI-SQ**	**Acc.**	0.806	0.815	0.839	0.808	0.808	0.758
		HLoss	0.078	0.077	0.066	0.081	0.082	0.113
		MR	0.775	0.8	0.816	0.798	0.786	0.644
		F1	0.658	0.655	0.686	0.619	0.65	0.625
	IG	**Acc.**	0.807	0.814	0.839	0.807	0.808	0.758
		HLoss	0.077	0.078	0.067	0.081	0.082	0.114
		MR	0.777	0.799	0.815	0.797	0.786	0.643
		F1	0.659	0.655	0.683	0.619	0.65	0.624
	CORR	**Acc.**	0.804	0.811	0.839	0.803	0.804	0.755
		HLoss	0.078	0.079	0.067	0.083	0.083	0.118
		MR	0.774	0.795	0.814	0.793	0.777	0.641
		F1	0.656	0.649	0.701	0.606	0.638	0.608
1000	**CHI-SQ**	**Acc.**	0.808	0.815	0.84	0.812	0.816	0.758
		HLoss	0.077	0.078	0.066	0.079	0.078	0.116
		MR	0.777	0.799	0.819	0.802	0.785	0.642
		F1	0.657	0.654	0.69	0.627	0.662	0.622
	IG	**Acc.**	0.807	0.815	0.839	0.812	0.816	0.758
		HLoss	0.077	0.078	0.067	0.079	0.078	0.116
		MR	0.778	0.799	0.818	0.802	0.785	0.642
		F1	0.657	0.654	0.688	0.627	0.662	0.625
	CORR	**Acc.**	0.811	0.818	0.844	0.813	0.816	0.754
		HLoss	0.074	0.076	0.065	0.078	0.078	0.123
		MR	0.78	0.805	0.821	0.803	0.785	0.638
		F1	0.67	0.659	0.702	0.624	0.662	0.603
5000	**CHI-SQ**	**Acc.**	0.824	0.831	**0.856**	0.84	0.839	0.763
		HLoss	0.066	0.071	**0.06**	0.068	0.069	0.119
		MR	0.799	0.82	**0.834**	0.83	0.809	0.65
		F1	0.694	0.68	**0.707**	0.674	0.696	0.625
	IG	**Acc.**	0.824	0.831	**0.856**	0.84	0.839	0.763
		HLoss	0.066	0.071	**0.06**	0.068	0.069	0.119
		MR	0.799	0.82	**0.835**	0.83	0.809	0.65
		F1	0.694	0.676	**0.707**	0.674	0.696	0.625
	CORR	**Acc.**	0.826	0.839	**0.854**	0.839	0.839	0.761
		HLoss	0.064	0.068	**0.061**	0.068	0.069	0.124
		MR	0.802	0.826	**0.835**	0.829	0.809	0.649
		F1	0.698	0.689	**0.714**	0.675	0.696	0.611

Once again, SVM are the algorithm that performed better. When compared to the solution using a combination of RSL-PL4Privacy and word features, SVM for all the four performance metrics across the different problem transformation. In addition, the overall results of this solution's classifier are better in comparison to the previous solution (both without feature selection). It is interesting to point out that, by removing the set of four features generated through RSL-PL4Privacy patterns, the Accuracy and remaining metrics for the SVM show higher values than earlier. This seems to verify that, in this problem

Table 6.6: The performance for the classifier with word features only without feature selection

Alg.	Perf.	BR	CC	ECC	LP	RAkEL	FW
NB	**Acc.**	0.507	0.506	0.539	0.344	0.327	0.371
	HLoss	0.297	0.298	0.275	0.388	0.535	0.514
	MR	0.293	0.293	0.347	0.306	0.188	0.167
	F1	0.417	0.417	0.424	0.271	0.281	0.332
DTs	**Acc.**	0.722	0.766	0.793	0.769	0.755	0.753
	HLoss	0.096	0.097	0.085	0.099	0.114	0.114
	MR	0.677	0.75	0.774	0.759	0.644	0.648
	F1	0.587	0.583	0.582	0.564	0.599	0.588
kNN	**Acc.**	0.721	0.721	Memory	0.721	0.721	0.721
	HLoss	0.12	0.12	out	0.12	0.12	0.12
	MR	0.719	0.719	of	0.719	0.719	0.719
	F1	0.24	0.24	bounds	0.24	0.24	0.24
SVMs	**Acc.**	0.829	0.859	0.852	0.86	0.86	0.772
	HLoss	0.061	0.06	0.062	0.06	0.061	0.122
	MR	0.804	0.848	0.833	0.85	0.82	0.662
	F1	0.721	0.711	0.698	0.704	0.737	0.624

of classifying statements, the patterns end up causing trouble to the learning process of the classifier.

6.2.4.2 Results with Feature Selection

The feature selection process for this solution used the same experimental setup as the previous solution that combined RSL-PL4Privacy and word features. We performed feature selection using only SVM since it was the algorithm that, once again, provided better results when performing without feature selection. Table 6.7 shows the results.

Similarly to the previous example, in this solution a lot of features are also still needed in order to deliver good results. The only time when the performance of the classifier slightly increased was the LP method using 5000 features weighted by the CORR measure. Again, to assist the reader with a better understanding, the above-mentioned values of performance are highlighted in **bold** in Table 6.7.

6.2.5 Discussion

In this section we provided several experiments for training and testing a classifier using: 1) RSL-PL4Privacy patterns as features, 2) words as features and 3) a solution combining both *strategies*. The solution delivering better results is actually the one using only words (in this case, trigrams). This appears to demonstrate that these pattern-based features cause more harm than good for the classifier when the actual (pre-processed) words of the statements are also passed as features.

However, despite this issue, both solutions provide exceptionally good results with only subtle differences of performance (approximately 85/86% of Accuracy and a HLoss of 6% for their best results). Additionally, it is also noteworthy that feature selection had low or no impact at all at the performance of the studied classifiers.

Lastly, the experiments carried out in this dissertation were based on having RSL-PL4Privacy as another *source* of features. Other possible course of action could focus on applying the RSL-PL4Privacy

Table 6.7: The performance for the classifier with word features only with feature selection measures and using SVMs (partial view)

# Features	Measures	Perf.	BR	CC	ECC	LP	RAkEL	FW
100	**CHI-SQ**	**Acc.**	0.785	0.818	0.819	0.801	0.792	0.746
		HLoss	0.075	0.075	0.075	0.085	0.093	0.135
		MR	0.75	0.803	0.798	0.794	0.737	0.646
		F1	0.651	0.649	0.651	0.576	0.627	0.556
	IG	**Acc.**	0.783	0.817	0.823	0.809	0.804	0.743
		HLoss	0.075	0.076	0.074	0.081	0.085	0.138
		MR	0.751	0.803	0.801	0.80	0.758	0.64
		F1	0.644	0.649	0.656	0.594	0.629	0.551
	CORR	**Acc.**	0.782	0.815	0.823	0.805	0.801	0.734
		HLoss	0.075	0.076	0.074	0.083	0.087	0.147
		MR	0.751	0.803	0.80	0.794	0.753	0.631
		F1	0.655	0.652	0.659	0.566	0.618	0.525
500	**CHI-SQ**	**Acc.**	0.802	0.837	0.839	0.821	0.829	0.757
		HLoss	0.07	0.069	0.066	0.077	0.075	0.132
		MR	0.764	0.823	0.816	0.811	0.782	0.649
		F1	0.695	0.682	0.686	0.631	0.685	0.581
	IG	**Acc.**	0.801	0.836	0.839	0.821	0.83	0.753
		HLoss	0.069	0.069	0.067	0.076	0.075	0.136
		MR	0.762	0.821	0.815	0.812	0.782	0.645
		F1	0.694	0.683	0.683	0.631	0.687	0.571
	CORR	**Acc.**	0.809	0.83	0.839	0.832	0.83	0.75
		HLoss	0.067	0.073	0.067	0.071	0.075	0.142
		MR	0.77	0.806	0.814	0.821	0.783	0.64
		F1	0.708	0.677	0.701	0.645	0.679	0.557
1000	**CHI-SQ**	**Acc.**	0.807	0.836	0.84	0.823	0.828	0.752
		HLoss	0.067	0.069	0.066	0.076	0.076	0.138
		MR	0.771	0.822	0.819	0.815	0.78	0.643
		F1	0.701	0.685	0.69	0.631	0.683	0.57
	IG	**Acc.**	0.809	0.836	0.839	0.823	0.829	0.753
		HLoss	0.067	0.069	0.067	0.076	0.075	0.137
		MR	0.773	0.821	0.818	0.815	0.782	0.642
		F1	0.70	0.684	0.688	0.631	0.684	0.572
	CORR	**Acc.**	0.814	0.839	0.844	0.837	0.835	0.75
		HLoss	0.065	0.068	0.065	0.068	0.071	0.142
		MR	0.781	0.823	0.821	0.826	0.791	0.64
		F1	0.711	0.689	0.702	0.664	0.704	0.557
5000	**CHI-SQ**	**Acc.**	0.824	0.852	0.856	0.85	0.851	0.766
		HLoss	0.061	0.063	0.06	0.064	0.065	0.126
		MR	0.796	0.84	0.834	0.839	0.811	0.655
		F1	0.714	0.702	0.707	0.684	0.716	0.606
	IG	**Acc.**	0.824	0.852	0.856	0.85	0.851	0.766
		HLoss	0.061	0.063	0.06	0.064	0.065	0.126
		MR	0.796	0.84	0.835	0.839	0.811	0.655
		F1	0.714	0.702	0.707	0.684	0.716	0.606
	CORR	**Acc.**	0.824	0.853	0.854	**0.862**	0.856	0.764
		HLoss	0.062	0.063	0.061	**0.059**	0.063	0.133
		MR	0.795	0.841	0.835	**0.853**	0.815	0.65
		F1	0.721	0.71	0.714	**0.718**	0.725	0.594

patterns directly to the test set, therefore filtering some *easy* statements, and this might lead to better results since the classifier would have fewer examples to classify.

6.3 Extraction of Relevant Information

The goal of the second step of the P1 process of RSLingo4Privacy deals with the extraction of other relevant information from the statements that are, at this point, already classified. We consider as

relevant information the other constructs of RSL-IL4Privacy: Recipients, PrivateData and the Service (its "description"). To achieve this objective, we build the extraction solution on top of the classification task, which means that we only perform extraction on the statements of the most important types: "Collection", "Disclosure", "Retention" and "Usage". The content of "Informative" statements (although providing some valuable information through the Enforcement construct as seen in Chapter 5) is usually very descriptive and has very few mentions to any of these four data practices. Thus, including it in the process would compromise even more the results of a really challenging task. However, the statements with more than one type and that happened to be also of the Informative type were also included.

The NER task is a fairly studied NLP task. However, aside from the work described in Chapter 3, we are not aware of other approaches using NER to extract information from privacy policies. Like in the classification solution, we used supervised learning which means that we have a previously annotated data set with each word carrying a "tag" indicating if such word is an important entity (that must be extracted) or not. Even though the implemented NER solutions are also called "classifiers" in the literature (since they use features to classify each word into a set of types), our solution will be referred to as the "extractor" in order to avoid misunderstandings throughout the reading of this section.

The remainder of this section is as follows. The experimental setup is detailed, namely the corpus that was used, the annotation strategy and its considerations, as well as the features that were used.

6.3.1 Experimental Setup

For the experiments carried out in this section we used the Stanford NER software that provides an implementation of linear-chain CRF as the algorithm for building the probability models used in the extraction process. CRF are considered the best algorithm for handling sequence labeling tasks.

As in the classification task, the evaluation of this proposed solution was performed using the cross-validation technique. We used 5-fold cross validation where in each run roughly 80% of the data is used as the training set and the remaining data is used for testing. The final results are the averaged results of the five trials.

6.3.1.1 Corpus

The corpus used in the process of developing of our extractor is a subset of the data set used for the classification task as we only train our solution with the "Non-Informative" statements (that may include "Informative" statements if the statements are multi-label), hence removing the "Informative" statements. This accounted for a data set of 521 statements that were then tokenized into 956 sentences and finally into words. A final corpus of 27441 words is annotated with the necessary tags.

From the overall 27441 words, 613 are annotated to define 323 named Recipients, 1718 to define 755 PrivateData attributes, 415 to define 208 PrivateData of type "PersonalInformation", 211 to define 80

PrivateData of type "UsageInformation", and finally, 3588 words to define 675 Services. The reader should note that these number of entities does not account for a whole different set of entities. Some annotations refer to the same entity but we do not identify such connection.

6.3.1.2 Tagging Scheme

Most of the entities in our corpus are multi-word entities. There are several tagging schemes for annotating multi-word entities, namely IO, IOB1, IOB2 and more complex schemes like BILOU or SBEIO. We annotated our corpus using the simple IOB2 (Inside-Outside-Begin) format. With this tagging format we annotate the words depending if they are the *beginning* of an entity (B), the *inside* of an entity (I) or otherwise (O) (*outside* an entity). Even though the simple IO format has the advantage of being fairly fast and easy to compute, on the other hand, it is not able to distinguish adjacent entities of the same class (with this annotation scheme, two adjacent entities of the same class are interpreted as one long entity). However, despite occurring very rarely in our corpus and subsequent longer training times, we wanted to accommodate this aspect. Also, for convenience and increase the readibility, whenever we want to refer to a specific tag in this dissertation we use the whole tag (e.g., "TAG") instead of distinguishing between "B-TAG" and "I-TAG".

The Annotation Process. The process of annotating a given corpus is tiring, time-consuming and prone to errors. Therefore, some simple "guidelines" were used while annotating the words, thus allowing us to maintain some consistency during the process. Since the annotation process was done each policy at a time, there are entities with the same designation that are annotated with different tags. This holds for several occurrences of PrivateData:

1. When a list of attributes is given to describe a PrivateData of a certain type, each one of those attributes is annotated with the "PDATT" tag. The PrivateData being described gets the "PDP" or "PDU" tag, depending on the type if the PrivateData is of type "PersonalInformation" or "UsageInformation", respectively;
2. If a previously annotated PrivateData (whether with "PDP" or "PDU") is used in a different privacy policy as an attribute of other PrivateData itself, it gets a "PDATT" tag. Several policies define more extensively the PrivateData constructs, others do not and use the same concept without defining it;
3. When words like "name", "e-mail address" or "phone number" are used separately (whether they are attributes of already defined PrivateData or not), they get the a "PDATT" tag.
4. Some policies define explicitly what they mean by "Personal Information" or ("Personal Identifiable Information"). When this happens, "Personal Information" is tagged as "PDP" and its attributes as "PDATT" (e.g., name, e-mail address, mobile number, etc). The data flows using this entity are confined to this set of attributes. On the other hand, when a policy only uses the term "Personal

Information" vaguely, we annotate it as "PDP" but it encompasses all the other "sub-PDPs" that are eventually defined throughout the policy.

Lastly, we are only interested in knowing the individuals or organizations that may receive the information disclosed by the service provider. Thus, we only annotate Recipients in "Disclosure" statements. We currently do not address co-reference resolution (checking if two entities refer to the same entity), but if there are multiple references to a given Recipient in the same "Disclosure" statement (that may not be describing the explicit action regarding the disclosure of information), we chose to annotate them as well. We acknowledge that this decision leads to several "false positives", since the same entity (i.e., with the same or similar name) may appear in all types of statements, it would not make sense to annotate all those occurrences as we only want to retrieve the ones used in "Disclosure" statements.

6.3.1.3 Features

A NER solution is only as good as the features it is trained with for recognizing the entities in a document. Thus, the definition of a set of features is an important part of the process. There are three "categories" of features [95]: word-level features, list look up features and document-level features. We use a combination of these features as it is provided by the Stanford NER software. Table 6.8 lists the features used by the CRF to train the extractor.

Table 6.8: The features used by the CRF to train the extractor

Feature	Description
Word, prevWord, nextWord	Use the word and its adjacent words.
Tag, prevTag, nextTag	Use the POS tag of the word and its adjacent tags
WordPairs, WordTagPairs	Pairs of words and tags (between the word, the previous and the next word/tag).
BiGrams	The contextual bi-grams.
WordShape	Shape of the word and adjacent words.
4-Word Window	Presence of a word in a window of four words.

Other features were also tested but this list includes the features that delivered better results. As it is common in the literature, we used words and POS tags themselves as features. As it can be observed from the table 6.8, additional features were also obtained through conjunctions of other features (e.g., word pairs, word-tag pairs) as well as some sequence features using a three-window word to both sides of a given word. In the end, the use of this set of features added up to a total of 78275 features.

6.3.2 Results

To evaluate the performance of our solution, we used the three main performance metrics for NER problems that are also used in single-label or multiclass classification: precision, recall and F1-score.

1. Precision - the percentage of correctly extracted entities of all the entities that were found (i.e., true positives divided by the true positives and false positives).

2. Recall - the percentage of extracted entities of all existing entities (i.e., true positives divided by true positives and false negatives).
3. F1-score (also known as F-score or just F1) - equal weighted harmonic mean of precision and recall (i.e., precision and recall have the same importance).

All these metrics assumed an exact-match method that is similar to the Exact-Match Ratio (EMR) metric used for multi-label classification and also widely used in NER tasks [96]. If an entity is not exactly annotated as in the manual annotation, it is considered as wrongly annotated.

6.3.2.1 Proposed Solution

Our proposed solution was the extractor trained with the aforementioned set of features. The extractor must extract all five annotations ("PDATT", "PDP", "PDU", "REC" and "SER") from the whole document. The results for this extractor are given in Table 6.9.

Table 6.9: The performance for the extractor

Entity	Precision	Recall	F1-score
PDATT	0.813	0.594	0.686
PDP	0.891	0.748	0.813
PDU	0.865	0.526	0.654
REC	0.774	0.446	0.566
SER	0.590	0.434	0.500
ALL	0.740	0.531	0.618

As we outlined in the beginning of this Section, the task of extracting the RSL-IL4Privacy constructs is rather difficult. Aside from the guidelines that were already described (e.g., same PrivateData with different tags, only annotating Recipients on "Disclosure" statements), there are other aspects that may influence badly the extractor. The Services or, in particular, the "description" of the Services is usually long with several words. More often than not, the extractor is only able to discover some few, isolated words from those entities. This is also the case for both types of PrivateData ("PDP" and "PDU"). Even though their list of attributes is almost always clear, there are times when very few attributes are used and the definition of the PrivateData itself is long (e.g., "information you give when you contact us").

6.3.2.2 Proposed Solution using a Gazetteer

Due to the fairly low performance of our extractor, one of the possible improvements could be using dictionary look up features [95]. These features are rather self-explanatory. We provide a list (also known as dictionary or gazetteer) of words that are previously defined with the tag of a certain entity and the extractor can use these additional features to identify entities in a document. It is important to point out that matching a word with one of the words of the gazetteer does not automatically means that such words would get the same tag: what it means is that the CRF used in the extractor gets another feature which it can use in its calculations. The use of gazetteers has proven itself useful in several work

in IE including NER tasks [97, 98]. To the best of our knowledge, there is not a gazetteer specifically for privacy policies. Thus, in order to assess the impact of using a gazetteer in our solution, we created a very basic gazetteer that encompassed two sets of entities: "PDATT" and "REC". Our gazetteer has a total of 745 entities, with 504 Recipients and 241 PrivateData attributes. These numbers account for various orthographic combinations (e.g., singular/plural, lower-case/upper-case words) of such entities. Table 6.10 details the performance of the extractor using this new feature.

Table 6.10: The performance for the extractor using a basic gazetteer

Entity	Precision	Recall	F1-score
PDATT	0.803	0.598	0.688
PDP	0.891	0.752	0.815
PDU	0.880	0.544	0.673
REC	0.782	0.533	0.634
SER	0.586	0.435	0.500
ALL	0.740	0.540	0.624

Even though the list of entities provided is really simple and incomplete, the extractor shows an improvement of its performance when using the gazetteer as feature. In particular the recall of the Recipient entity increases substantially with the gazetteer, as well as all the recall values for the other entities. Overall, the extractor delivers a F1-score of 0.624. These results suggest that having a more extensive and complete gazetteer may indeed provide better results for NER tasks dealing with privacy policies.

6.3.2.3 Solutions with one extractor per type and per tag

A possible strategy to improve the performance of an extractor may be *dividing* the problem into smaller sub-problems. Instead of training an extractor that searches a set of statements of all types and tries to identify the entities using the entire set of tags, a solution could rely on having a set of extractors that can only identify one type of entities. Other solution could be training an extractor for each type of statement. In the case of a tie - two or more extractors assigning different tags to the same entity -, the decision would be based on a value like probability score that CRF assigns to each tag (similar to the confidence values in text classification). We carried some preliminary experiments in order to assess these claims and trained one extractor per label. Table 6.11 presents the results achieved for both solutions.

Table 6.11: The performance for one extractor per label

Entity	Precision	Recall	F1-score
PDATT	0.801	0.590	0.680
PDP	0.0.906	0.779	0.838
PDU	0.933	0.505	0.655
REC	0.772	0.519	0.621
SER	0.584	0.429	0.495

Even though this solution seemed to work better in theory, its results shows us otherwise. Almost all individual extractors are outperformed by the initial solution that uses all statements and tags. The

features used in the initial solution and the presence of all types of entities actually help to discriminate between each entities. However, the entity "PDP" demonstrates an interesting improvement for both precision and recall. Further tests are needed to check if a solution using this extractor combined with other eventual extractors may actually provide better results.

6.3.3 Discussion

In this section we outlined the solution that it is expected to extract information from the statements of the privacy policies. Even though the results fall a little behind the results obtained for the classification task, from a high-level perspective, it is better to have to look into half of the data set than to the complete data set.

Regardless, the introduction of a gazetteer and its subtle effect on the performance of the extractor, as well as the fact that one extractor for one of the labels provides better results, show the there are still room for improvement, it is possible to develop more accurate solutions.

7

RSLingo4Privacy-Studio

The RSLingo4Privacy-Studio is the tool that supports the RSLingo4Privacy approach detailed in this dissertation, providing the technological infrastructure for specifying and analyzing privacy policies. It comprises a series of model transformations in order to enable the processes of this approach. The RSLingo4Privacy-Studio is currently under development and this chapter summarizes the particular features that are already implemented at this point.

The RSLingo4Privacy-Studio is built on top of the Eclipse IDE, more specifically leveraging the Eclipse Modeling Framework (EMF). It uses the RSL-IL4Privacy language as an intermediate language to represent the privacy policies and from it to generate more readable representations of such policies (e.g. MS-Excel or MS-Word files), or to check their quality using its own validators or engines of other formal languages such as Eddy (see Chapter 2). As can be seen in Fig. 7.1, RSLingo4Privacy-Studio relies on several transformations to support the multiple privacy policies representations and the mapping between them, namely Text-to-Model (T2M), Model-to-Model (M2M) and Model-to-Text (M2T). RSLingo4Privacy-Studio provides three main features: Import, Export and Check Quality.

The Import feature allows a user to import an Excel file (transformation M2M-1) or an ad-hoc natural language text file (transformation T2M) containing the policy and generate its corresponding RSL-IL4Privacy files. M2M-1 is implemented using the Apache POI library, while T2M is implemented through the execution of automatic text classification and text extraction processes. The Export feature

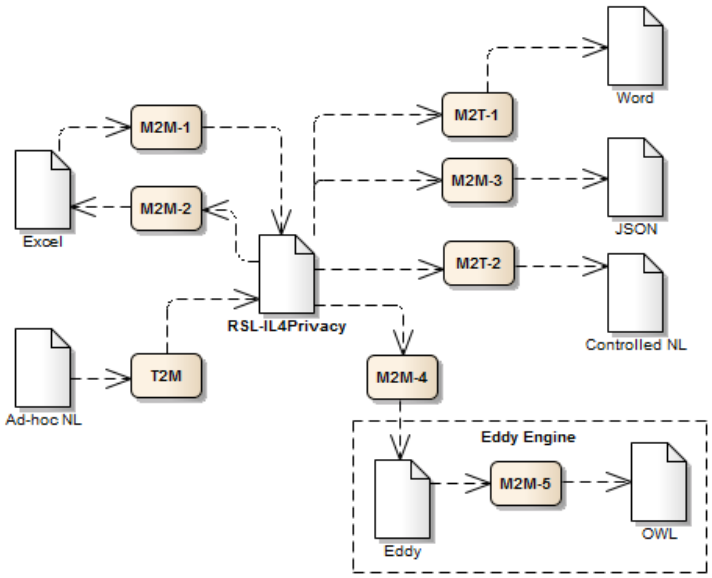

Figure 7.1: RSLingo4Privacy-Studio supported transformations

allows transforming a RSL-IL4Privacy file to other file formats, namely Word, Excel, JSON, Eddy and Text (controlled NL). Transformations M2M-2 and M2M-3 are implemented using the Apache POI library and two template files (one for Word and another for Excel). In turn, the remaining transformations for JSON (M2T-1), Text (M2T-2) and Eddy (M2M-4) are implemented using Xtend. The Check Quality feature is only applicable to Eddy files and allows running the Eddy engine to check if there are any conflicts in the provided policy described in the Eddy file. The output of this process is a log file, containing the possible conflicts, and an OWL (Web Ontology Language) file with the representation of the rules used to describe the policy.

7.1 Multi-transformation Approach

As mentioned before, Studio relies on a multi-transformation approach which includes T2M, M2M and M2T transformations. On the other hand, Studio deals with policies documented with multiple representations, namely: ad-hoc and controlled NL text, Excel, Word, JSON and Eddy. We have considered an Excel file as model, since it is a tabular and highly structured representation. In contrast, we consider that a Word file is similar to a NL text, in the sense that it contains plain text but with just low-level formatting information. However, since it is not as structured as an Excel file we considered it as text and not a model (despite being both internally organized in an archive of multiple XML files). Below we describe each transformation type and then we explain their implementation issues, which are grouped by the technology used to support them.

7.1.1 T2M Transformations

Studio performs a T2M transformation during the import process of an ad-hoc NL text file. This transformation involves the execution of automatic text classification and extraction processes. The classification process identifies the set of statements in the policy provided and classifies them as "Collection", "Disclosure", "Retention", "Usage" or "Informative" (or with multi-labels). The second process extracts the relevant elements from the original statements into their equivalent representation in RSL-IL4Privacy.

7.1.2 M2M Transformations

M2M transformations are used during the import and ex-port of a policy in Studio. The import of an Excel file specifying a policy (transformation M2M-1) generates its corresponding RSL-IL4Privacy file(s), depending on the file structure mode the user has selected (single or multiple). M2M-1 is implemented using the Apache POI library which simplifies the processing of Microsoft Office file formats. M2M-2 performs the reverse transformation (from RSL-IL4Privacy to Excel) and is also implemented using the Apache POI library, but uses an Excel template file. The remaining transformations consist in the export of a privacy policy specified in RSL-IL4Privacy for JSON (M2M-3) and Eddy (M2M-4).

7.1.3 M2T Transformations

These transformations occur when a RSL-IL4Privacy file is exported to Word (transformation M2T-1) and controlled NL text file (transformation M2T-2). The transformations from RSL-IL4Privacy into JSON (M2M-3), Text (M2T-2) and Eddy (M2M-4) are performed using Xtend. A code generator stub written in Xtend is one of the artifacts that is automatically generated from a Xtext grammar definition and automatically integrated into the produced Eclipse plugin. Xtend simplifies the usage and maintenance of the code generator by allowing the definition of code templates, portions of code that contain dynamic parts that change according to the Xtext-based model given as input. On the other hand, the transformations from RSL-IL4Privacy into Word (M2T-1) and Excel (M2M-2) are performed using the Apache POI library and two companion template files (one for each for-mat). We use this library instead of Xtend, because it highly abstracts the complex XML structure that underlies Microsoft Office files. Additionally, we used template files to give more flexibility for a user to customize the style and formatting of the generated files.

Both Word and Excel templates have special tag annotations that represent the dynamic part of the template and identify which property should be placed there during the generation. They are defined using the style (e.g. font type, size or color) that should be reflected in the generated file. The Word template is a document organized in sections, one for each concept of the RSL-IL4Privacy language (e.g.

Statements, Services and PrivateData) and contains subsections for the Services and Recipients, which can contain Sub-Services and Sub-Recipients, respectively. Each section and subsection is delimited by a start tag and an end tag. During the transformation, each section is copied as many times as the number of elements of that type that exist, and the tags are replaced by the value of the respective property of each element.

The Excel template is a workbook organized in sheets, one for each concept of the RSL-IL4Privacy language. Each sheet name identifies the set of elements that describes (e.g. Statements, Services and Recipients) and contains a head row identifying the content of each column and then an example row containing the tags annotations. During the transformation, the example row is copied as many times as the number of elements of that type that exist, and the tags are replaced by the value of the respective property of each element. As can be noted, this process is analogous to the one applied for the sections in the Word template.

7.2 RSL-IL4Privacy Editor

RSLingo4Privacy-Studio provides a textual editor for creating and editing RSL-IL4Privacy files. This editor was developed using the Xtext framework which allows the development of textual DSLs through their definition using Xtext's grammar. From that grammar definition it is possible to automatically generate the full language infrastructure (parser, linker, typechecker and compiler) and a fully customizable Eclipse plugin (more recently also for IntelliJ IDEA or even web browsers) containing the DSL editor with helpful features like syntax highlighting, error checking, auto-completion or source-code navigation [6]. Xtext-based DSLs have Ecore as metamodel. Since Xtext relies on EMF, it can be combined with other popular Eclipse plugins, like Xtend, Sirius or Acceleo. The grammar of a Xtext-based language is composed of rules that describe its key entities and their relations. Each Entity has a name and some properties.

The RSL-IL4Privacy editor is able to deal with two file structure modes to specify a privacy policy: (1) Single file; or (2) Multiple files with one file per element. In the Single file mode, there is only one file that contains all the privacy policy concepts. This strategy is recommended when the privacy policy is small, otherwise will be hard to maintain it. In the Multiple files mode, there is a main file which is used to reference all the other files (one for each RSL-IL4Privacy concept) through "import" statements. The main file also includes metadata that describes the policy like its name, authors, ver-sion and date. This strategy can enhance the maintenance of the policy specification because the different concepts are not mixed in the same file, but instead defined in multiple and isolated files. For instance, if a user needs to add a new state-ment, he only needs to open the file where the statements are specified.

8

Evaluation

The cross-validation methodology gives the ability of assessing the performance of a given model as if it was being tested in a real-world scenario. However, for the purpose of our research work, we wanted to evaluate our developed solutions against a real case study. We chose to use Zynga's privacy policy[1] due to the fact that it had been previously considered for the data set used in the NLP tasks and we still had the necessary files.

After the automatic classification and extraction of information, RSLingo4Privacy encompasses processes that are responsible for the automatic validation of its generated specifications based on formal languages. One of the common claims of formal languages used for the specification of privacy policies is their possibility of performing some complex analysis and reasoning about such requirements and policies. On the other hand, having a specification written in a language with a *controlled syntax* (such as RSL-IL4Privacy) that still preserves natural language to the maximum, provides a better readability and understanding of such requirements but, at the same time, makes it more difficult to automatically analyze these specifications.

One solution for this dichotomy requires combining both types of languages. Using this solution it is possible to present an understandable specification with complementary representations (in comparison to the original policy), while abstracting the reasoning process which is complex and a forerunner for the

[1] https://www.zynga.com/privacy/policy

creation of more formal languages.

This chapter is organized as follows. It provides the Zynga privacy policy as case study for the RSLingo4Privacy approach, namely the evaluation of its first process (P1) which is the primary focus of this dissertation. Additionally, the part of the process P2 that deals with the conversion of RSL-IL4Privacy specifications into equivalent Eddy specifications is also outlined and the preliminary results of the reasoning process described by process P3 are also discussed.

8.1 The Zynga Case Study

This section describes the Zynga case study. It discusses its data set on different levels and applies the NLP solutions developed in Chapter 6 both separately and together.

8.1.1 Characterization of the Data Set

To compare the predicted performance of our models with their *real* performance, we had to follow the same procedure of analyzing the privacy policy beforehand under the scope of RSL-IL4Privacy which meant classifying Zynga's statements according to their type and annotating the statements with the tags representing the services, private data and recipients.

This analysis allowed us to gain insight about Zynga's privacy policy, specifically in terms of size and lexicon (on a more textual level) and also data elements, services and recipients, the constructs of interest. Table 8.1 depicts the number of statements per each type and also presents some usual textual units (e.g., number of characters, words and sentences). These textual information highlight some specifics about the policy at hand.

Table 8.1: The statement type distribution for Zynga's privacy policy

Label	# Statements	# Sentences	# Words	# Characters
Collection	4	4	191	948
Disclosure	11	20	618	3139
Retention	5	6	158	765
Usage	4	5	276	1362
Informative	73	148	2969	14790
Multi	16	41	1642	8285
Total	113	224	5854	29289

To get a more a better sense of this information, we reuse the measures introduced in Chapter 6 (see Section 6.2) that evaluate *how much* multi-label is in fact a given data set. For this privacy policy, the Label Cardinality (LC) = $16/113 = 0.142 \rightarrow 1.142$ average labels for each examples in the data set; and the Label Density (LD) = $1.048/|L| = 1.142/5 = 0.23$. Individually, Zynga's privacy policy is more "multi-labelled" than the data set used to train the previous classifier (see Table 6.2).

On the other hand, when compared to such data set, Zynga's policy is ranked number six of all the 21 policies in terms of its number of statements. Additionally, the 113 statements of this policy encompass

almost two times the number of sentences (224) which results in 5854 words. All these aspects combined leads us to consider the task of analyzing Zynga's privacy policy as non-trivial.

8.1.2 Classification of Statements

To perform the classification of statements from the Zynga privacy policy, we employed the solution that yielded better results with word features only and that was described in Chapter 6, namely SVM for the Label Powerset (LP) transformation method with the top 5000 features selected through Pearson's Correlation (CORR) measure (listed in the Table as LP*). Additionally, we also used the two next better solutions: SVM for Classifier Chains (CC) and Random k-Labelsets (RAkEL) methods without feature selection. Table 8.2 gives the results of the classification solution for this case study.

Table 8.2: The performance for the classifier with word features only applied to Zynga's privacy policy

Perf.	LP*	CC	RAkEL
Acc.	0.764	0.768	**0.777**
HLoss	0.106	0.104	**0.099**
EMR	0.708	0.708	**0.717**
F1	0.569	0.611	**0.614**

The classification results shown in Table 8.2 are interesting and its discussion worthwhile. Despite providing worst results than the solution studied in Chapter 6, in this case study the solution providing the previous best results (LP*) is outperformed by the other two solutions (CC and RAkEL). This underlines the limitations of LP [57], namely its inability for predicting label sets that have not been seen in the training data. When splitting the data for the cross-validation in the previous experiments, it might have happened the case in which the test sets have label sets that are always included in the training set.

On the other hand, this privacy policy - due to the characteristics that were outlined in the previous Section - posed a difficult case study for our model due to the fact that its human analysis (for labeling and annotating) was not trivial. However, the solution providing the best results (RAkEL) is rather good at classifying the "Informative" statements. From the 73 "Informative" statements, 71 are correctly labeled and discarded. On the other hand, several multi-label statements or statements from the remaining four types are labeled only as Informative (therefore being also removed). This occurs for 17 statements.

8.1.3 Extraction of Relevant Information

The next step deals with the extraction of relevant information in which we identify the main RSL-ILPrivacy entities: PrivateData, Recipients and Services. As discussed in Chapter 6, we make use of the previous classification task in order to filter "Informative" statements that do not offer critical data practices. However, due to the fact that the classification solution is not perfect, some "Informative" statements may be incorrectly labeled and some of the remaining types of statements may be labeled

as "Informative", therefore being wrongly discarded. For the reported case of Zynga, the classifier is almost perfect at doing this filtering step. Only 2 "Informative" are passed on, hence not discarded. The problem is the 17 statements that are incorrectly labeled as "Informative" and removed from the corpus. To provide a comparison of both cases, we provide the results of the solution that evaluates the original data set (with its "Informative" statements filtered manually) and the one takes as input the output of the previous classifier.

8.1.3.1 "Individual" Solution

As it can be observed from Table 8.1, excluding the "Informative" statements, we obtain a total of 40 "NonInformative" statements that comprise 76 sentences containing 2885 words. From these 2885 words, 57 are annotated to define 28 named Recipients, 221 to define 88 PrivateData attributes, 49 to define 23 PrivateData of type "PersonalInformation", 14 to define 6 PrivateData of type "UsageInformation", and finally, 362 words to define 82 Services.

We applied our extractor defined in Chapter 6 that uses a gazetteer due to the fact that it was the solution delivering better results. Table 8.3 shows the obtained results.

Table 8.3: The performance for the extractor applied independently to Zynga's privacy policy

Entity	Precision	Recall	F1-score
PDATT	0.589	0.375	0.458
PDP	0.889	0.348	0.500
PDU	0.000	0.000	0.000
REC	0.546	0.429	0.480
SER	0.640	0.390	0.485
ALL	0.612	0.374	0.465

With our extractor we are able to identify almost half of the Recipients described in Zynga's privacy policy. However, there are cases where, for example, a given "Disclosure" statement provides a list of common, similar Recipients and the extractor identifies the general Recipient (e.g., Affiliates) and misses the other entities on the list. Even though this does not provide a complete specification of the Recipients described in the policy, from a reasoning process point-of-view, it may be irrelevant because the action outlined in the policy applies to the general Recipient (which is identified) and subsequently to the remaining Recipients. Additionally, the PrivateData of type "PersonalInformation" ("PDP") has high precision (few false positives) because of the guidelines we elicited in Chapter 6, namely the option of annotating the all the occurrences of the expression "Personal Information" and, if no attributes are given, assume it aggregates all the "PDP" entities found in the policy.

8.1.3.2 "Pipeline" Solution

This solution is applied to a much smaller corpus because some of the aforementioned "NonInformative" statements are wrongly removed and do not reach the extractor. After being classified in the

previous step, we obtain a total 25 statements (40 "NonInformative" statements minus the 17 improperly discard and the two "Informative" statements that are passed). From these 1833 words, 32 are annotated to define 15 named Recipients, 115 to define 47 PrivateData attributes, 30 to define 13 PrivateData of type "PersonalInformation", 6 to define 3 PrivateData of type "UsageInformation", and finally, 272 words to define 63 Services. Using the extractor with the gazetteer we obtain that following results that are described in Table 8.4.

Table 8.4: The performance for the extractor applied using classification results to Zynga's privacy policy

Entity	Precision	Recall	F1-score
PDATT	0.610	0.489	0.541
PDP	1.000	0.462	0.632
PDU	0.000	0.000	0.000
REC	0.500	0.533	0.516
SER	0.714	0.395	0.510
ALL	0.639	0.440	0.521

Using the classifier output turned out to deliver better results for the extraction task. Nevertheless, the reader should be aware that the loss of potential critical information due to the incorrect removal of some "NonInformative" statements represent a bigger problem. For this specific case study, the fact that the "Informative" statements passed on the extractor did not trigger any false positive, therefore being one of the reasons for the improvement of the overall performance. None of the solution was able to identify "PDU" entities. However, since some of their attributes were indeed identified, it would not be difficult to infer the related "PDU" entity.

8.1.4 Discussion

This case study was a real test to the models developed in this dissertation. Even though we - during the period of this dissertations - extended our data set with privacy policies of different domains (providing more scenarios, therefore increasing the probability of applying the defined models to all types of policies), analyzing a really distinct policy is a challenging task.

The long and descriptive statements led to problems in the classification phase of the process P1 of RSLingo4Privacy. For example, this policy links several sentences through common determiners which ends up in considerable multi-label statements where only a fraction of it describes some important data practice (illustrated by the four "Collection", "Disclosure", "Retention" and "Usage" types) with the rest being purely generic, descriptive information. This results in classification errors and subsequently loss of relevant information. All this adds up to the already difficult task of extracting the necessary RSL-IL4Privacy entities from privacy policies.

8.2 Generating Eddy Specifications from RSL-IL4Privacy Equivalent Specifications

The RSLingo4Privacy approach begins with process P1 that allow us to get RSL-IL4Privacy specifications through the automatic analysis of the information conveyed on original policies. Process P2 assumes that RSL-IL4Privacy specifications obtained from P1 (or new specifications, if it the case) are then converted into equivalent Eddy specifications in order to foster formal computational analysis. This section describes the "conversion strategy" and the results of applying Eddy's conflict detection techniques to the converted specifications (i.e., privacy policies) of Facebook, LinkedIn, Twitter, Dropbox and IMDb.

8.2.1 Transform RSL-IL4Privacy into Eddy

A RSL-IL4Privacy to Eddy transformation was defined in the context of the Xtext framework. With this feature it is possible to generate Eddy specifications from equivalent RSL-IL4Privacy specifications. To define this generator we had to find all the common matching concepts between both RSL-IL4Privacy and Eddy grammars.

As discussed in Chapter 5, a policy specified in RSL-IL4Privacy encompasses a set of constructs: Statement, Recipient, Service, PrivateData and Enforcement. The single definition of a Statement (i.e., with its description, modality, and remaining attributes) encloses the various associations with the remaining constructs. A policy in Eddy is represented with a clear specification header ("SPEC HEADER") and the following specification body ("SPEC POLICY"). The header aggregates the prior definition of three elements: "P" for Purpose, "A" for Actor and "D" for Datum. The statements are then described on the body. Each statement has a modality, wherein "P" indicates permission, "O" indicates obligation and "R" indicates prohibition; an action verb; the Datum; the source (following "FROM"), the target (following "TO") and the Purpose (following "FOR"). Based on the description of the different elements and keywords from both languages, it is possible to map the following constructs between languages: the PrivateData can be considered as the Datum, the Service as Purpose and the Recipient as Actor (target). Since the source ("FROM") refers to the service provider, there is not a direct match between concepts in the two languages. The key mappings between both grammars are defined in Table 8.5.

Table 8.5: Matching syntax keywords for RSL-IL4Privacy and Eddy languages

Language	Modality	Action (type)	Datum	Source	Target	Purpose
Eddy	P, O, R	COLLECT, TRANSFER, USE RETAIN	D	FROM	TO	P
RSL-IL4Privacy	Permitted Obligation Forbidden	Collection Disclosure, Usage Retention	privateData	-	recipient	service (description)

8.2.2 Detection of Conflicts using Eddy

As briefly introduced in Chapter 2, Eddy uses Description Logic (DL) to model the different data flows within a policy and find conflicts. A conflict occurs when there is an intersection between permitted and forbidden actions (e.g., sharing data) carried out by an actor for some purpose. Even though an action can be a "Permission", "Prohibition" (i.e., a forbidden action) or an "Obligation" (in terms of its modality), the detection of the "permitted-forbidden" type of conflicts does not require the distinction of allowed actions as "Permitted" or "Obligation". Since the goal of process P3 at this point is just to determine if such conflicts exist, labeling an action as "Permitted" or "Obligation" has no particular relevance to the process. However, "permitted-obligation" conflicts tend to occur within privacy policies but their detection is more complex and requires an extra amount of work. Even though we do classify the statements (i.e., the actions they describe) as "Permitted" and "Obligation", the eventual conflicts of this kind are out of the scope of RSLingo4Privacy approach at this time, thus, their conflict detection process is not discussed.

8.2.3 Results: Conflicting Statements in Detail

We applied Eddy's deontic conflict detection with equivalent RSL-IL4Privacy specifications to assess if RSL-IL4Privacy language had the flexibility and reasoning power to be used as main privacy requirements specification language The results for case studies discussing the policies of Dropbox, Facebook, IMDb, LinkedIn and Twitter are summarized in Table 8.6.

Table 8.6: Results using Eddy conflict detection engine with converted RSL-IL4Privacy specifications

Privacy Policy	**# Conflicts Detected**
Dropbox	0
Facebook	2
IMDb	6
LinkedIn	1
Twitter	12

These results account for the number of conflicts between permitted and prohibited actions. If one statement describes a prohibition regarding the handling of some information (e.g., "We will not use your e-mail address for advertising purposes") and then other statement discusses permitted actions for the same data, a conflict occurs.

Twitter is the system that shows the highest number of deontic conflicts (12). Half of the conflicts occur due to the disclosure of personal information, namely Payment Information (e.g., credit card details) or the user's address. Listing 8.1 depicts some of Twitter's conflicting statements in RSL-IL4Privacy using its concrete syntax.

Listing 8.1: Twitter's conflicting statements S22 with S28 and S22 with S34 (partial view)

```
statement 22 : Disclosure {
    name "S22"
    modality Forbidden
    description "We consider your Payment Information and shipping address private
                 and do not make such information public."

    recipient "All"
    privateData 7, 8
    service "All"

statement 28 : Disclosure {
    name "S28"
    modality Permitted
    description "We share your Payment Information with payment services providers
                 to process payments."

    recipient 5
    privateData 7
    service 6

statement 34 : Disclosure {
    name "S34"
    modality Forbidden
    description "If you buy goods or services through our Services, we may provide
                 the seller, commerce provider or marketplace with your name,
                 email address, shipping address, Payment Information and
                 Transaction Data to help prevent, detect and investigate fraud or
                 other prohibited activities."

    recipient 7, 8, 9
    privateData 7, 8, 10, 18, 19
    service 10
};
```

By stating “We consider your payment information and user shipping address private and do not make such information public” (see the statement S22 in Listing 8.1), one can safely argue that Twitter is concerned with the improper disclosure of information. Therefore, ”not making information public” can mean not sharing any types of personal data with anyone under any circumstances. However, several

other statements – in this case, S28 and S34 - describe the disclosure of the user's personal information (including the discussed types) to multiple recipients and for a variety of reasons (e.g., security, processing of transactions). On the other hand, Twitter's privacy policy also shows slightly different conflicts. Even though this social network clearly states that its "Services are not directed to persons under 13" (a business rule defined as an Enforcement element in RSL-IL4Privacy), the collection and retention of those users' private information is still carried out until the provider becomes aware of such fact. Conflicts occur when this information – that should not be gathered in the first place and is still undetected – is used and processed by Twitter afterwards. More than pointing out eventual conflicts that may be then used to produce revised versions of the policy, these warnings should trigger Twitter to develop mechanisms for preventing children from using their Services or, on the upper hand, to provide a better assessment before processing the users' information.

Facebook policy's conflicts are also due to the inappropriate disclosure of user information. We are told that Facebook does not share personal information (i.e., information "that can by itself be used to contact or identify users") with advertising, measurement or analytics partners, unless the user gives permission. However, its policy states that Facebook shares information (due to a vague definition of this concept, we need to assume that it includes also the users' personal information) with a variety of recipients, such as "partners that support their business". It is safe to say that advertising and performance measurement partners also support the business of Facebook, therefore being eligible for receiving a wide range of information – and with no explicit user consent. At the same time, Facebook also declares that it shares information internally (as one might expect) but also with external third parties for the purposes described in its policy. The unclear definition of the purposes for the disclosure of information leads to inconsistencies. These two examples show conflicts in terms of who is allowed or not to receive information (possibly including users' private information) but also why is necessary for the company to share such information. The conflicts that occur in LinkedIn and IMDb policies are similar to the aforementioned conflicts, therefore being pointless to discuss them.

Lastly, Dropbox is the only policy that showed no conflicting statements during our analysis. Its policy is relatively small when compared to the policies of the other service providers and none of the statements explicitly states any kind of prohibition in terms of data practices (i.e., no statements with "forbidden" modality). Thus, as we pointed out, if the focus of our analysis in this paper is the detection of "permitted-forbidden" conflicts, it makes sense that no inconsistencies were found.

8.3 Discussion

This chapter provided with a very broad assessment and complete evaluation of the RSLingo4Privacy approach. Section 8.1 detailed a new case study, Zynga, which revealed that the domain of a privacy policy - with its very specific terminology and concerns - may play its part in the success on an automatic

classification and information extraction approach (P1). However, the defined solutions yielded good results which suggests that their application on social networking privacy policies, for example, might deliver a better outcome than those provided with Zynga's privacy policy.

On the other hand, Section 8.2 detailed the P2 and P3 processes, namely the conversion from RSL-IL4Privacy to Eddy, the evaluation of five privacy policies using its reasoning engine for conflict detection and a in-depth discussion of the obtained results. With a clear and complete RSL-IL4Privacy specification one is able to generate an equivalent specification in other more formal language, therefore enabling the possibility of performing reasoning and validation processing.

9

Conclusion

The regular use of and increasingly rapid development of platforms such as social networking websites and apps has shed a new light on the "Privacy" topic, raising new concerns and stimulating discussion. If, on the one hand, policy authors, under legal constraints, need to specify privacy policies of the systems their companies provide, on the other hand, the integration of several complex systems with apps fostered developers to be aware of the *what*, *how*, *why* and *when* user information is being handled. Finally, end-users also need to start being a part of this process by being informed and capable of making decisions regarding the privacy and uses of their data and acting accordingly. The multiple private data flows and practices that each privacy policy, whether explicitly or silently, describes are a solid starting point to create formal methodologies of reasoning about these privacy requirements, therefore providing the companies with the necessary evidence for aligning their policies.

The alignment of privacy policies triggered the development of more formal frameworks and languages with the purpose of specifying their privacy requirements using constructs that, in parallel with other methods with their roots on formal logic, allowed one to infer new knowledge and draw conclusion regarding such data flows within policies.

However, despite its ambiguity, natural language is a very powerful strategy of communication (both spoken and written) and is often the cause for the small adoption of other more technical languages. Additionally, these formal languages are driven by the idea and intent of writing new policies and spec-

ifications. They do not include approaches to use or convert existing privacy policies written in natural language. Such mapping, if applicable, has to be done manually which is not feasible when taking it account the change of a paradigm like this.

Bearing all these problems that arise from the state-of-the-art in privacy requirements specification languages and approaches, we argue that a broad perspective delivering the complete workflow, from natural language specifications to their formal counterparts, while providing value to all stakeholders is of the utmost importance.

The RSLingo4Privacy approach includes four major processes that act on different levels of the analysis and specification approach. It is our expectation that the approach described in this dissertation may be considered as valuable and simple approach to deal with a common activity in the RE field, the specification and analysis of privacy requirements.

RSLingo4Privacy is supported by a new privacy specification language that is designed to cope with natural language as an intrinsic characteristic of the requirements specification process, specifically privacy policies.

9.1 RSL-IL4Privacy as Privacy Specification Language

RSL-IL4Privacy provides the fundamental constructs to specify current privacy policies. This revised set of constructs allow the documentation of more useful than other languages, since they deal with the information that appears in a slightly different context and that may be used to enforce these policies (through the use of tools and/or business rules). Additionally, through a simple and well-formed syntax encourages its use by the interesting parties of this process: developers will recognize its concrete syntax as it is similar to other object-oriented programming languages' syntax; policy authors will have the ability for specifying and authoring new or existing policies using natural language all the same; and lastly, end-users will get a privacy policy that clearly divides the various data practices and actions with all its entities outlined. Table 9.1 gives a general comparison of the most important approaches on the formalization of privacy requirements specification languages.

Table 9.1: Comparison of privacy-aware specification languages including RSL-IL4Privacy

Language	**Domain**	**Abstract Syntax,** defined as a...	**Concrete Syntax,** represented by...	**Semantics**
P3P/APPEL	Web Privacy	XML schema	Textual	Declarative
EPAL	Access Control	XML schema	Textual	Declarative
KAoS	Generic	Prolog* constructs	Textual	OWL
Rei	Generic	DAML (XML schema)	Textual	OWL
Eddy	Data Privacy	Grammar	Textual	OWL-DL
RSL-IL4Privacy	Data Privacy	UML Profile + Grammar	Graphic + Textual	Declarative

Taking into account the importance of complementing natural language requirements documentation with conceptual models, to avoid and mitigate ambiguity, a formal specification language that is able to

provide a thorough specification of privacy-related requirements using a concrete syntax with a simple textual representation format but, at the same time, gives the possibility of having the same requirements specification represented by complementary models (tabular or graphical representations, for example) is of great value to better accommodate the needs of all involved. The design and definition of current formal languages do not suit them to this property, as opposed to what we propose with RSL-IL4Privacy.

9.2 Retrieve Information from Existing Policies Automatically

The other contribution the research work in this dissertation provides is a complete approach for automatically extract information from existing policies in natural language. Using state-of-the-art NLP techniques, we provide a two-step procedure to achieve this goal. The first step deals with the proper categorization of a privacy statement.

Not all statements in a privacy policy convey important information regarding critical data practices that might expose the users' private information. Through a precise definition of the differences between existing types of statements, we can then use a solution that automatically classifies and filters the statements of a policy, therefore enabling different uses for each type of statement.

The second phase of this approach is related to the automatic extraction of relevant information. To enrich the RSL-IL4Privacy constructs with the necessary information (hence, generating RSL-IL4Privacy specifications), one has to go through existing privacy policies - that can be extremely long and its analysis time-consuming - and look for the specific bits of information that are important. We provide, as the second contribution related to this process, a model for extracting some RSL-IL4Privacy constructs, namely Recipients, PrivateData and Services.

With this constructs defined in a specification, it is then possible to use other formal languages like Eddy that can provide computational and validation processing, such as conflict detection. We also supply the reader with an interoperability mapping, with the syntax of both languages aligned, and that enables a more formal reasoning of these specifications. We provide the results of this methodology applied to five different privacy policies.

9.3 RSLingo4Privacy-Studio, the tool supporting RSLingo4Privacy

RSLingo4Privacy-Studio provides the necessary technological support to RSLingo4Privacy. It includes features that suit best the processes and concerns of this approach. Table 9.2 compares the features each tool provides for all languages described in this dissertation. Even though all languages have technologies for the visualization and authoring of policies, RSLingo4Privacy-Studio is the only one that gives the ability of publishing privacy-aware specifications. At the same time, RSLingo4Privacy-Studio is the only tool that supports multiple model transformations (e.g., T2M, M2M, and M2T) simultaneously. Finally,

since it is out of the RSL-IL4Privacy scope, RSLingo4Privacy-Studio does not support authorization enforcement. However, future work could focus on extending RSL-IL4Privacy with the core elements needed for the policy authorization enforcement.

Table 9.2: Comparison of tools that support privacy specification languages including RSLingo4Privacy-Studio

Language	Visualization & Authoring	T2M	M2M	M2T	Publishing	Authorization Enforcement
P3P/APPEL	Yes (IBM P3P, JRC Policy Workbench, P3PEdit)	No	No	To HTML	No	No
EPAL	Yes (Privacy Authoring Editor, EPAL Editor)	No	No	To HTML	No	Yes
KAoS	Yes (KPAT)	No	No	No	No	Yes
Rei	Yes (Protégé-Plugin, N3 text editor, RIDE)				No	No
Eddy	Yes (General-purpose text editor)	No	Yes	No	No	No
RSL-IL4Privacy	Yes (Eclipse Xtext-based Plugin)	Ad-hoc NL to RSL-IL4Privacy	Excel, JSON, Eddy	Word and Controlled NL	Yes	No

9.4 Future Work

Future work will focus on the classification of privacy policies against a set of privacy-aware patterns, which would allow us to assign privacy policies a qualitative value (e.g., a grade ranging from 1 to 5) in what privacy is con-cerned. These scores are sometimes called privacy grades[1]. We will also check how RSL-IL4Privacy and the interoperability mechanism perform when dealing with multiple specifications. Finally, we will proceed with the development and enhancement of RSLingo4Privacy-Studio. We also plan on evaluating RSLingo4Privacy-Studio by carrying out usability tests through laboratory-controlled sessions with end-users (e.g. software developers, policy authors and requirement engineers) so that we are also able to receive more precise and direct feedback about the tool.

[1] http://www.privacygrade.org/

Bibliography

[1] G. I. Susman and R. D. Evered, "An assessment of the scientific merits of action research," *Administrative science quarterly*, pp. 582–603, 1978.

[2] A. R. da Silva, J. Caramujo, S. Monfared, P. Calado, and T. Breaux, "Improving the specification and analysis of privacy policies: The rslingo4privacy approach," in *Proceedings of ICEIS'2016 Conference*. SCITEPRESS, 2016.

[3] T. D. Breaux and D. L. Baumer, "Legally "reasonable" security requirements: A 10-year ftc retrospective," *Computers & Security*, vol. 30, no. 4, pp. 178–193, 2011.

[4] C. Farrell, "Ftc charges deceptive privacy practices in google's rollout of its buzz social network," *US Federal Trade Comission News Release, March*, vol. 30, 2011.

[5] E. Steel and G. Fowler, "Facebook in privacy breach," *The Wall Street Journal*, vol. 18, p. 1, 2010.

[6] K. Pohl, *Requirements engineering: fundamentals, principles, and techniques*. Springer Publishing Company, Incorporated, 2010.

[7] I. Sommerville and P. Sawyer, *Requirements engineering: a good practice guide*. John Wiley & Sons, Inc., 1997.

[8] S. Robertson and J. Robertson, *Mastering the requirements process: Getting requirements right*. Addison-wesley, 2012.

[9] K. El Emam and A. G. Koru, "A replicated survey of it software project failures," *IEEE software*, vol. 25, no. 5, pp. 84–90, 2008.

[10] A. Davis, *Just enough requirements management: where software development meets marketing*. Addison-Wesley, 2013.

[11] B. L. Kovitz, *Practical software requirements: a manual of content and style*. Manning Greenwich, 1999.

[12] A. R. da Silva, "Specqua: Towards a framework for requirements specifications with increased quality," in *International Conference on Enterprise Information Systems*. Springer, 2014, pp. 265–281.

[13] K. Pohl and C. Rupp, *Requirements engineering fundamentals: a study guide for the certified professional for requirements engineering exam-foundation level-IREB compliant*. Rocky Nook, Inc., 2011.

[14] T. D. Breaux, H. Hibshi, and A. Rao, "Eddy, a formal language for specifying and analyzing data flow specifications for conflicting privacy requirements," *Requirements Engineering*, vol. 19, no. 3, pp. 281–307, 2014.

[15] K. Lewin, "Action research and minority problems," *Journal of social issues*, vol. 2, no. 4, pp. 34–46, 1946.

[16] R. O'Brien, "An overview of the methodological approach of action research," *Faculty of Information Studies, University of Toronto*, 1998.

[17] J. Caramujo and A. M. R. da Silva, "Analyzing privacy policies based on a privacy-aware profile: The facebook and linkedin case studies," in *2015 IEEE 17th Conference on Business Informatics*, vol. 1. IEEE, 2015, pp. 77–84.

[18] A. R. da Silva, J. Caramujo, S. Monfared, P. Calado, and T. Breaux, "A domain-specific language for the specification of privacy-aware requirements," *To appear in Requirements Engineering*, 2017.

[19] C. Kalloniatis, E. Kavakli, and S. Gritzalis, "Addressing privacy requirements in system design: the pris method," *Requirements Engineering*, vol. 13, no. 3, pp. 241–255, 2008.

[20] Q. He, A. I. Antón *et al.*, "A framework for modeling privacy requirements in role engineering," in *Proc. of REFSQ*, vol. 3, 2003, pp. 137–146.

[21] M. Deng, K. Wuyts, R. Scandariato, B. Preneel, and W. Joosen, "A privacy threat analysis framework: supporting the elicitation and fulfillment of privacy requirements," *Requirements Engineering*, vol. 16, no. 1, pp. 3–32, 2011.

[22] F. H. Cate, "The failure of fair information practice principles," *Consumer protection in the age of the information economy*, 2006.

[23] L. F. Cranor, "Necessary but not sufficient: Standardized mechanisms for privacy notice and choice," *J. on Telecomm. & High Tech. L.*, vol. 10, p. 273, 2012.

[24] M. A. Graber, J. Johnson-West *et al.*, "Reading level of privacy policies on internet health web sites.(brief report)," *Journal of Family Practice*, vol. 51, no. 7, pp. 642–646, 2002.

[25] J. R. Reidenberg, T. Breaux, L. F. Cranor, B. French, A. Grannis, J. T. Graves, F. Liu, A. McDonald, T. B. Norton, and R. Ramanath, "Disagreeable privacy policies: Mismatches between meaning and users' understanding," *Berkeley Tech. LJ*, vol. 30, p. 39, 2015.

[26] L. F. Cranor, C. Hoke, P. G. Leon, and A. Au, "Are they worth reading? an in-depth analysis of online advertising companies' privacy policies," in *2014 TPRC Conference Paper*, 2014.

[27] A. I. Antón, D. Bolchini, and Q. He, "The use of goals to extract privacy and security requirements from policy statements," in *Proc. of the 26th IEEE Int'l Conf. on Software Engineering (ICSE'03)*. Citeseer, 2003.

[28] P. Kumari *et al.*, "Requirements analysis for privacy in social networks," in *8th International Workshop for Technical, Economic and Legal Aspects of Business Models for Virtual Goods, Namur*, 2010.

[29] L. Cranor, M. Langheinrich, and M. Marchiori, "A p3p preference exchange language 1.0 (appel 1.0): W3c working draft 15 april 2002," *World Wide Web Consortium (W3C), URL: http://www. w3. org/TR/P3P-preferences*, 2002.

[30] L. F. Cranor, "P3p," 2013.

[31] P. G. Leon, L. F. Cranor, A. M. McDonald, and R. McGuire, "Token attempt: the misrepresentation of website privacy policies through the misuse of p3p compact policy tokens," in *Proceedings of the 9th annual ACM workshop on Privacy in the electronic society*. ACM, 2010, pp. 93–104.

[32] M. Schunter, E. Van Herreweghen, and M. Waidner, "Expressive privacy promises-how to improve the platform for privacy preferences (p3p)," in *Position paper for W3C Workshop on the Future of P3P (available at www. w3. org/2002/p3p-ws/pp/ibm-zuerich. pdf)*, 2002.

[33] T. Yu, N. Li, and A. I. Antón, "A formal semantics for p3p," in *Proceedings of the 2004 workshop on Secure web service*. ACM, 2004, pp. 1–8.

[34] M. Langheinrich, L. Cranor, and M. Marchiori, "Appel: A p3p preference exchange language," *W3C Working Draft*, 2002.

[35] G. Hogben, T. Jackson, and M. Wilikens, "A fully compliant research implementation of the p3p standard for privacy protection: Experiences and recommendations," in *European Symposium on Research in Computer Security*. Springer, 2002, pp. 104–125.

[36] C. Powers and M. Schunter, "Enterprise privacy authorization language (epal 1.2)," *W3C Member Submission*, vol. 10, 2003.

[37] T. Moses *et al.*, "Extensible access control markup language (xacml) version 2.0," *Oasis Standard*, vol. 200502, 2005.

[38] W. Han and C. Lei, "A survey on policy languages in network and security management," *Computer Networks*, vol. 56, no. 1, pp. 477–489, 2012.

[39] A. Uszok, J. Bradshaw, R. Jeffers, N. Suri, P. Hayes, M. Breedy, L. Bunch, M. Johnson, S. Kulkarni, and J. Lott, "Kaos policy and domain services: Toward a description-logic approach to policy representation, deconfliction, and enforcement," in *Policies for Distributed Systems and Networks, 2003. Proceedings. POLICY 2003. IEEE 4th International Workshop on.* IEEE, 2003, pp. 93–96.

[40] A. Uszok, J. M. Bradshaw, J. Lott, M. Breedy, L. Bunch, P. Feltovich, M. Johnson, and H. Jung, "New developments in ontology-based policy management: Increasing the practicality and comprehensiveness of kaos," in *Policies for Distributed Systems and Networks, 2008. POLICY 2008. IEEE Workshop on.* IEEE, 2008, pp. 145–152.

[41] L. Kagal, T. Finin, and A. Joshi, "A policy language for a pervasive computing environment," in *Policies for Distributed Systems and Networks, 2003. Proceedings. POLICY 2003. IEEE 4th International Workshop on.* IEEE, 2003, pp. 63–74.

[42] G. Tonti, J. M. Bradshaw, R. Jeffers, R. Montanari, N. Suri, and A. Uszok, "Semantic web languages for policy representation and reasoning: A comparison of kaos, rei, and ponder," in *International Semantic Web Conference.* Springer, 2003, pp. 419–437.

[43] E. C. Lupu and M. Sloman, "Conflicts in policy-based distributed systems management," *IEEE Transactions on software engineering*, vol. 25, no. 6, pp. 852–869, 1999.

[44] T. D. Breaux and A. I. Antón, "Deriving semantic models from privacy policies," in *Sixth IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY'05).* IEEE, 2005, pp. 67–76.

[45] N. Shadbolt, T. Berners-Lee, and W. Hall, "The semantic web revisited," *IEEE intelligent systems*, vol. 21, no. 3, pp. 96–101, 2006.

[46] A. B. Shah, A. B. Shah, L. Kagal, and A. Joshi, "An integrated development environment for policies," *organization*, 2005.

[47] T. Winograd, "Understanding natural language," *Cognitive psychology*, vol. 3, no. 1, pp. 1–191, 1972.

[48] D. G. Bobrow, "Natural language input for a computer problem solving system," 1964.

[49] M. Krauthammer and G. Nenadic, "Term identification in the biomedical literature," *Journal of biomedical informatics*, vol. 37, no. 6, pp. 512–526, 2004.

[50] R. Gaizauskas and Y. Wilks, "Information extraction: Beyond document retrieval," *Journal of documentation*, vol. 54, no. 1, pp. 70–105, 1998.

[51] F. Sebastiani, "Machine learning in automated text categorization," *ACM computing surveys (CSUR)*, vol. 34, no. 1, pp. 1–47, 2002.

[52] K. Trohidis, G. Tsoumakas, G. Kalliris, and I. P. Vlahavas, "Multi-label classification of music into emotions." in *ISMIR*, vol. 8, 2008, pp. 325–330.

[53] J. P. Pestian, C. Brew, P. Matykiewicz, D. J. Hovermale, N. Johnson, K. B. Cohen, and W. Duch, "A shared task involving multi-label classification of clinical free text," in *Proceedings of the Workshop on BioNLP 2007: Biological, Translational, and Clinical Language Processing*. Association for Computational Linguistics, 2007, pp. 97–104.

[54] G. Tsoumakas and I. Katakis, "Multi-label classification: An overview," *Dept. of Informatics, Aristotle University of Thessaloniki, Greece*, 2006.

[55] O. Luaces, J. Díez, J. Barranquero, J. J. del Coz, and A. Bahamonde, "Binary relevance efficacy for multilabel classification," *Progress in Artificial Intelligence*, vol. 1, no. 4, pp. 303–313, 2012.

[56] J. Read, B. Pfahringer, G. Holmes, and E. Frank, "Classifier chains for multi-label classification," *Machine learning*, vol. 85, no. 3, pp. 333–359, 2011.

[57] G. Tsoumakas, I. Katakis, and I. Vlahavas, "Random k-labelsets for multi-label classification," *IEEE Transactions on Knowledge and Data Engineering*, vol. 23, no. 7, pp. 1079–1089, 2010.

[58] J. Read, "Work on multi-label classification," 2011.

[59] S. R. Safavian and D. Landgrebe, "A survey of decision tree classifier methodology," 1990.

[60] S. Ceri, A. Bozzon, M. Brambilla, E. Della Valle, P. Fraternali, and S. Quarteroni, *Web Information Retrieval*. Springer Science & Business Media, 2013.

[61] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE transactions on information theory*, vol. 13, no. 1, pp. 21–27, 1967.

[62] E.-H. S. Han, G. Karypis, and V. Kumar, "Text categorization using weight adjusted k-nearest neighbor classification," in *Pacific-asia conference on knowledge discovery and data mining*. Springer, 2001, pp. 53–65.

[63] O.-W. Kwon and J.-H. Lee, "Text categorization based on k-nearest neighbor approach for web site classification," *Information Processing & Management*, vol. 39, no. 1, pp. 25–44, 2003.

[64] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[65] V. Ng and C. Cardie, "Improving machine learning approaches to coreference resolution," in *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, 2002, pp. 104–111.

[66] D. M. Bikel, R. Schwartz, and R. M. Weischedel, "An algorithm that learns what's in a name," *Machine learning*, vol. 34, no. 1-3, pp. 211–231, 1999.

[67] S. Sarawagi, "Information extraction," *Foundations and trends in databases*, vol. 1, no. 3, pp. 261–377, 2008.

[68] S. Sarawagi and W. W. Cohen, "Semi-markov conditional random fields for information extraction," in *Advances in neural information processing systems*, 2004, pp. 1185–1192.

[69] J. R. Finkel, T. Grenager, and C. Manning, "Incorporating non-local information into information extraction systems by gibbs sampling," in *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, 2005, pp. 363–370.

[70] J. Lafferty, A. McCallum, and F. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," in *Proceedings of the eighteenth international conference on machine learning, ICML*, vol. 1, 2001, pp. 282–289.

[71] H. M. Wallach, "Conditional random fields: An introduction," 2004.

[72] W. Ammar, S. Wilson, N. Sadeh, and N. A. Smith, "Automatic categorization of privacy policies: A pilot study," 2012.

[73] R. Ramanath, F. Liu, N. Sadeh, and N. A. Smith, "Unsupervised alignment of privacy policies using hidden markov models," 2014.

[74] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: an update," *ACM SIGKDD explorations newsletter*, vol. 11, no. 1, pp. 10–18, 2009.

[75] J. Read, P. Reutemann, B. Pfahringer, and G. Holmes, "MEKA: A multi-label/multi-target extension to Weka," *Journal of Machine Learning Research*, vol. 17, no. 21, pp. 1–5, 2016. [Online]. Available: http://jmlr.org/papers/v17/12-164.html

[76] G. Tsoumakas, E. Spyromitros-Xioufis, J. Vilcek, and I. Vlahavas, "Mulan: A java library for multi-label learning," *Journal of Machine Learning Research*, vol. 12, no. Jul, pp. 2411–2414, 2011.

[77] P. Szymański, "Scikit-multilearn: Enhancing multi-label classification in python," *Manuscript in preparation*, 2014.

[78] H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan, "GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications," in *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics (ACL'02)*, 2002.

[79] F. Jungermann, "Information extraction with rapidminer," in *Proceedings of the GSCL Symposium'Sprachtechnologie und eHumanities*. Citeseer, 2009, pp. 50–61.

[80] D. de Almeida Ferreira and A. R. da Silva, "Rslingo: An information extraction approach toward formal requirements specifications," in *Model-Driven Requirements Engineering Workshop (MoDRE), 2012 IEEE*. IEEE, 2012, pp. 39–48.

[81] S. Bird, E. Klein, and E. Loper, *Natural language processing with Python*. " O'Reilly Media, Inc.", 2009.

[82] H. Cunningham, "Information extraction, automatic," *Encyclopedia of language and linguistics,*, pp. 665–677, 2005.

[83] D. Ferreira and A. Silva, "Rsl-pl: A linguistic pattern language for documenting software requirements," *Proceedings of RePa*, vol. 13, 2013.

[84] D. de Almeida Ferreira and A. R. da Silva, "Rsl-il: An interlingua for formally documenting requirements," in *Model-Driven Requirements Engineering (MoDRE), 2013 International Workshop on*. IEEE, 2013, pp. 40–49.

[85] A. Van Deursen, P. Klint, and J. Visser, "Domain-specific languages: An annotated bibliography." *Sigplan Notices*, vol. 35, no. 6, pp. 26–36, 2000.

[86] M. Young *et al.*, *Technical writer's handbook*. University Science Books, 2002.

[87] P. M. Schwartz and D. J. Solove, "Pii problem: Privacy and a new concept of personally identifiable information, the," *NYUL Rev.*, vol. 86, p. 1814, 2011.

[88] R. Snow, D. Jurafsky, and A. Y. Ng, "Learning syntactic patterns for automatic hypernym discovery," *Advances in Neural Information Processing Systems 17*, 2004.

[89] S. Godbole and S. Sarawagi, "Discriminative methods for multi-labeled classification," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2004, pp. 22–30.

[90] J. Ramos, "Using tf-idf to determine word relevance in document queries," in *Proceedings of the first instructional conference on machine learning*, 2003.

[91] M.-L. Zhang and Z.-H. Zhou, "A review on multi-label learning algorithms," *IEEE transactions on knowledge and data engineering*, vol. 26, no. 8, pp. 1819–1837, 2014.

[92] S. M. Mohammad, S. Kiritchenko, and X. Zhu, "Nrc-canada: Building the state-of-the-art in sentiment analysis of tweets," *arXiv preprint arXiv:1308.6242*, 2013.

[93] G. Forman, "An extensive empirical study of feature selection metrics for text classification," *Journal of machine learning research*, vol. 3, no. Mar, pp. 1289–1305, 2003.

[94] T. Joachims, "Text categorization with support vector machines: Learning with many relevant features," in *European conference on machine learning*. Springer, 1998, pp. 137–142.

[95] D. Nadeau and S. Sekine, "A survey of named entity recognition and classification," *Lingvisticae Investigationes*, vol. 30, no. 1, pp. 3–26, 2007.

[96] E. F. Tjong Kim Sang and F. De Meulder, "Introduction to the conll-2003 shared task: Language-independent named entity recognition," in *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*. Association for Computational Linguistics, 2003, pp. 142–147.

[97] Z. Kozareva, "Bootstrapping named entity recognition with automatically generated gazetteer lists," in *Proceedings of the eleventh conference of the European chapter of the association for computational linguistics: student research workshop*. Association for Computational Linguistics, 2006, pp. 15–21.

[98] A. Smith and M. Osborne, "Using gazetteers in discriminative information extraction," in *Proceedings of the Tenth Conference on Computational Natural Language Learning*. Association for Computational Linguistics, 2006, pp. 133–140.

A

The RSL-IL4Privacy Grammar

Listing A.1: The RSL-IL4Privacy Grammar

```
PrivacyPolicy:
    'package' name=QualifiedName '{'
        importPackages+=ImportPackage*
        (metadata=PrivacyPolicyMetadata)?

        privateData+=PrivateData*
        recipients+=Recipient*
        services+=Service*
        enforcements+=Enforcement*

        statements+=Statement*
    '}';

QualifiedName:
    ID ('.' ID)*;

ImportPackage:
```

```
    'import' importedNamespace=QualifiedNameWithWildcard;

QualifiedNameWithWildcard:
    QualifiedName '.*'?;

PrivacyPolicyMetadata:
    'privacyPolicyMetadata' name=QualifiedName '{'
        'name' nameAlias=STRING
        'authors' authors=STRING
        'organizations' organizations=STRING
        'date' date=Date
        'version' version=STRING
        ('description' description=STRING)?
    '}';

Date:
    day=INT '-' month=Month '-' year=INT;

Month:
    name=('Jan'|'Feb'|'Mar'|'Apr'|'May'|'Jun'|'Jul'|'Aug'|'Sep'|'Oct'|'Nov'|'Dec');

/** Enforcement */
Enforcement:
    'enforcement' name=ID ':' type=EnforcementType '{'
        'name' enforcementName=STRING
        ('description' description=STRING)?
    '}';

enum EnforcementType: Action | Algorithm | Config | Process | Tool | Unknown;

/** Service */
Service:
    'service' name=ID ':' type=ServiceType '{'
        'name' serviceName=STRING
        ('partOf' partOf=[Service])?
        ('description' description=STRING)?
        ('refersPrivateData' (refPrivateData=RefPrivateData | refPDAll='All'))?
    '}';
```

```
56
57 enum ServiceType: Simple | Compound | Unknown;
58
59 /** Recipient */
60 Recipient:
61     'recipient' name=ID ':' type=RecipientType '{'
62         'name' aliasName=STRING
63         'scope' scope=RecipientScope
64         ('partOf' partOf=[Recipient])?
65         ('description' description=STRING)?
66     '}';
67
68 enum RecipientType: Individual | Organization | IndividualOrOrganization | Unknown;
69
70 enum RecipientScope: In | Out | InAndOut | Unknown ;
71
72 /** PrivateData */
73 PrivateData:
74     'privateData' name=ID ':' type=PrivateDataType '{'
75         'name' aliasName=STRING
76         ('partOf' partOf=[PrivateData])?
77         ('description' description=STRING)?
78         attributes+=Attribute*
79     '}';
80
81 Attribute:
82     'attribute' name=ID ':' type=AttributeType '{'
83         'name' aliasName=STRING
84         ('description' description=STRING)?
85     '}' ;
86
87 enum PrivateDataType: PersonalInformation | UsageInformation | Unknown ;
88
89 enum AttributeType:
90     Integer|Decimal|Boolean|Currency|Date|Time|String|URL|Image|XML|RegEx|Unknown;
91
92
93 /** Statement */
```

```
94  Statement:
95      'statement' name=ID ':' type+=StatementType (',' type+=StatementType)* '{'
96          'name' aliasName=STRING
97          'modality' modality=ModalityType
98          ('description' description=STRING)?
99          ('period' period=STRING)? // only for Retention Statements
100
101         ('recipient' (recipients=RefRecipient | refRecipientAll='All'))?
102         // recipient is only for Disclosure Statements
103         ('privateData' (privateData=RefPrivateData | refPrivateDataAll='All'))?
104         ('services' (services=RefService | refServiceAll='All'))?
105         ('enforcement' (enforcements=RefEnforcement | refEnforcementAll='All'))?
106     '}';
107
108 enum StatementType: Collection | Disclosure | Retention | Usage | Informative ;
109
110 enum ModalityType: Permitted | Obligation | Forbidden ;
111
112
113 RefRecipient: refersRecipient+=[Recipient](',' refersRecipient+=[Recipient])* ;
114
115 RefPrivateData:
116     refersPrivateData+=[PrivateData](',' refersPrivateData+=[PrivateData])* ;
117
118 RefService: refersService+=[Service](',' refersService+=[Service])* ;
119
120 RefEnforcement:
121     refersEnforcement+=[Enforcement](',' refersEnforcement+=[Enforcement])* ;
```

B

RSL-PL4Privacy Patterns

Listing B.1: The complete set of RSL-PL4Privacy patterns

```
1  // Collection
2  W("We|we") W("collect|receive|obtain")
3  W("We|we") W(MOD) W("collect|receive|obtain")
4  W("We|we") W(MOD) W(ADV) W("collect|receive|obtain")
5  W("We|we") W(ADV) W("collect|receive|obtain")
6  W("We|we") W(MOD) W(V) W(TO) W("collect|receive|obtain")
7  W("We|we") W(V|MOD) W("not") W("collect|receive|obtain")
8  W("We|we") W(V|MOD) W("not") W(ADV) W("collect|receive|obtain")
9  W("You|you") W("provide|give|send") W("us")
10 W("You|you") W(MOD) W("provide|give|send") W("us")
11 W("You|you") W(MOD) W(ADV) W("provide|give|send") W("us")
12 W(N|NP) W("collects|receives|obtains")
13 W(N|NP) W(MOD) W("collect|receive|obtain")
14 W(N|NP) W(MOD) W(ADV) W("collect|receive|obtain")
15 W(N|NP) W(ADV) W("collects|receives|obtains")
16 W(N|NP) W(MOD) W(V) W(TO) W("collect|receive|obtain")
17 W(N|NP) W(V|MOD) W("not") W("collect|receive|obtain")
```

```
18 W(N|NP) W(V|MOD) W("not") W(ADV) W("collect|receive|obtain")
19 W(N|NP) W("provide|give|send") W("us")
20 W(N|NP) W(MOD) W("provide|give|send") W("us")
21 W(N|NP) W(MOD) W(ADV) W("provide|give|send") W("us")
22
23 //Disclosure
24 W("We|we") W(action-verbs-Disclosure)
25 W("We|we") W(MOD) W(action-verbs-Disclosure)
26 W("We|we") W(MOD) W(ADV) W(action-verbs-Disclosure)
27 W("We|we") W(ADV) W(action-verbs-Disclosure)
28 W("We|we") W(MOD) W(V) W("to") W(action-verbs-Disclosure)
29 W("We|we") W(V|MOD) W("not") W(action-verbs-Disclosure)
30 W("We|we") W(V|MOD) W("not") W(ADV) W(action-verbs-Disclosure)
31 W(N|NP) W(MOD) W(v) W("to") W(action-verbs-Disclosure)
32 W(N|NP) W(V|MOD) W("not") W(action-verbs-Disclosure)
33 W(N|NP) W(V|MOD) W("not") W(ADV) W(action-verbs-Disclosure)
34 W(N|NP) W(MOD) W(action-verbs-pastParticiple-Disclosure)
35 W(N|NP) W(MOD) W(ADV) W(action-verbs-pastParticiple-Disclosure)
36 W(N|NP) W(V) W(action-verbs-pastParticiple-Disclosure)
37 W(N|NP) W(MOD) W(V) W(action-verbs-pastParticiple-Disclosure)
38 W(N|NP) W(action-verbs-presentSimple-Disclosure)
39
40 //Retention
41 W("We|we") W(action-verbs-Retention)
42 W("We|we") W(MOD) W(action-verbs-Retention)
43 W("We|we") W(MOD) W(ADV) W(action-verbs-Retention)
44 W("We|we") W(ADV) W(action-verbs-Retention)
45 W("We|we") W(MOD) W(V) W(TO) W(action-verbs-Retention)
46 W("We|we") W(V|MOD) W("not") W(action-verbs-Retention)
47 W("We|we") W(V|MOD) W("not") W(ADV) W(action-verbs-Retention)
48 W(N|NP) W(action-verb-presentSimple-Retention)
49 W(N|NP) W(V|MOD) W("not") W(action-verbs-Retention)
50 W(N|NP) W(V|MOD) W("not") W(ADV) W(action-verbs-Retention)
51 W(N|NP) W(V) W(action-verbs-pastParticiple-Retention)
52 W(N|NP) W(MOD) W(V) W(action-verbs-pastParticiple-Retention)
53 W(N|NP) W(action-verbs-Retention)
54 W(N|NP) W(MOD) W(action-verbs-Retention)
55 W(N|NP) W(MOD) W(ADV) W(action-verbs-Retention)
```

```
56 W(N|NP) W(ADV) W(action-verbs-Retention)
57 W(N|NP) W(MOD) W(V) W(TO) W(action-verbs-Retention)
58
59 //Usage
60 W("We|we") W(action-verbs-Usage) W(DET|ADJ|PRP$) W(N|NP)
61 W("We|we") W(action-verbs-Usage) W(DET|PRP$) W(ADJ) W(N|NP)
62 W("We|we") W(MOD) W(action-verbs-Usage) W(DET|ADJ|PRP$) W(N|NP)
63 W("We|we") W(MOD) W(action-verbs-Usage) W(DET|PRP$) W(ADJ) W(N|NP)
64 W("We|we") W(ADV) W(action-verbs-Usage) W(DET|ADJ|PRP$) W(N|NP)
65 W("We|we") W(ADV) W(action-verbs-Usage) W(DET|PRP$) W(ADJ) W(N|NP)
66 W("We|we") W(MOD) W(ADV) W(action-verbs-Usage) W(DET|ADJ|PRP$) W(N|NP)
67 W("We|we") W(MOD) W(ADV) W(action-verbs-Usage) W(DET|PRP$) W(ADJ) W(N|NP)
68 W("We|we") W(V|MOD) W("not") W(action-verbs-Usage) W(DET|ADJ|PRP$) W(N|NP)
69 W("We|we") W(V|MOD) W("not") W(action-verbs-Usage) W(DET|PRP$) W(ADJ) W(N|NP)
70 W("We|we") W(action-verbs-Usage) W("information")
71 W("We|we") W(action-verbs-Usage) W("information")
72 W("We|we") W(MOD) W(action-verbs-Usage) W("information")
73 W("We|we") W(MOD) W(action-verbs-Usage) W("information")
74 W("We|we") W(ADV) W(action-verbs-Usage) W("information")
75 W("We|we") W(ADV) W(action-verbs-Usage) W("information")
76 W("We|we") W(MOD) W(ADV) W(action-verbs-Usage) W("information")
77 W("We|we") W(MOD) W(ADV) W(action-verbs-Usage) W("information")
78 W("information" W(MOD) W(V) W(action-verbs-pastParticiple-Usage)
```