GaiusT 2.0: Evolution of a Framework for Annotating Legal Documents

Nicola Zeni[1], Luisa Mich[2(✉)], and John Mylopoulos[3]

[1] Department of Information Engineering and Computer Science, University of Trento, Trento, Italy
nicola.zeni@unitn.it

[2] Department of Industrial Engineering, University of Trento, Trento, Italy
luisa.mich@unitn.it

[3] School of Electrical Engineering and Computer Science, University of Ottawa, Ottawa, CA, Canada
jmylopou@eecs.uottawa.ca

Abstract. Semantic annotation technologies support the extraction of legal concepts, for example rights and obligations, from legal documents. For software engineers, the final goal is to identify compliance requirements a software system has to fulfill in order to comply with a law or regulation. That implies analyzing and annotating legal documents in prescriptive natural language, still an open problem for research in the field. In this paper we describe GaiusT 2.0, a system for extracting requirements from legal documents. GaiusT 2.0 is the result of the evolution of GaiusT, and has been designed and implemented as a web-based system intended to semi-automate the extraction process. Results of the application of GaiusT 2.0 show that the new version improves performance of the extraction process and also makes the tool more usable.

Keywords: Semantic annotation · Legal documents · Legal requirements · Natural language processing · Linguistic resources · User needs

1 Introduction

Recent trends in technologies, as well as social, economic and political issues, have greatly increased the importance of ensuring compliance of software systems with regulations, laws, policies and other types of legal documents. Autonomous vehicles, the Internet of Things, cloud services, augmented reality, videogames, online auctions, e-commerce and e- related sectors, and social networking platforms are only some of the many technologies [1] challenging the elicitation of legal requirements [2] from a variety of legal documents. That in turn has driven researchers to design and develop tools and systems to support requirements analysts in eliciting compliance requirements from regulatory documents [3, 4]. Existing systems range from editors for manually annotating (markup) legal documents to frameworks integrating linguistic tools and natural language processing (NLP) features. NLP systems constitute essential components, as they fully or partially automate requirements extraction, demanding an in-depth semantic and pragmatic analysis, an objective that is far from being reached [5]. In this context, the first version of GaiusT represented a multi-phase framework to support the different tasks in which legal requirements extraction can be decomposed [6]. The framework was implemented as a modular system with a semantic annotation system core [7] coded in TXL, a structural transformation system [8].

The applications and experiments with GaiusT exposed a number of issues that required a deep reengineering of the system. The reengineering project started in 2013 and the first steps included (a) an analysis of existing tools and linguistic resources that could be adopted to address the high level requirement for an improved version of GaiusT, but still preserving its nature of a lightweight system [9]; (b) an investigation of usability issues for a tool supporting the extraction of requirements from legal documents [10].

In this paper we describe the architecture of the new version of GaiusT, called GaiusT 2.0. The evolution project moved from the output of those steps. In addition to the above, the project aimed to address new challenges that are: (a) how to deal with legal documents in different domains and cultures; (b) how to support user intervention in all the steps of the requirements extraction process. GaiusT 2.0 is a completely revised system supporting requirements extraction (SSRE) where all the modules have been re-implemented and new ones have been added. To evaluate the new system, preliminary results obtained with GaiusT 2.0 are reported and compared with those obtained with GaiusT for the HIPAA (Health Insurance Portability and Accountability Act) [6, 11].

This paper is structured as follows. Section 2 describes the problems to be addressed by a system supporting requirements elicitation annotating legal documents and the requirements for such systems. Section 3 illustrates the architecture of GaiusT 2.0 comparing it with that of GaiusT. Section 4 presents the main results of the evaluation of GaiusT 2.0. Section 5 illustrates the related work and finally, the conclusions are drawn in Sect. 6.

2 Extracting Legal Requirements from Textual Documents

2.1 Supporting Requirements Elicitation for Regulatory Compliance

The core task a SSRE for regulatory compliance has to support is the identification of deontic concepts described in legal documents, from compound concepts such as rights and obligations to requirements concepts, such as goal, actor, action, resource. A detailed description of the characteristics of an SSRE is given in [6]. Here we review the issues most relevant to the evolution of our system described in the next section. First of all, automatic annotation and processing of legal documents is particularly difficult because they are written in “legalese”, a specialized prescriptive language. Moreover, from a linguistic point of view an SSRE from legal documents has to deal with knowledge in different domains, or rather the specialized language and knowledge of laws for specific domains. A SSRE has also to take into account the structure of legal documents which generally respect a multi-level hierarchical structure. Such structure constitutes the basis for internal and external references [12]; for example, describing a right, the involved actors may be found defined in another document or section; exceptions can refer to actions defined in another subsection, etc. As a result, at least two models are required for annotating a given regulation: (a) a conceptual model, including the deontic concepts that represent prescribed behaviors – rights and obligations (descriptive metadata) – and (b) a structural model of the text of the regulation (structural metadata). The second model is particularly important to delimit the scope of concepts identified in the text according to the conceptual model. Still, a comprehensive SSRE had to support many other activities including the very first steps in which (a) the conceptual model for a law is designed, or adapted from an existing one, for a given regulation and (b) indicators corresponding to the concepts are defined in an annotation schema. The variety of tasks and elements involved also requires the creation and management of a large knowledge base; multi-language annotation is also in place in some projects. Existing systems usually focus only on one of the tasks described in this section that is the annotation of deontic concepts, or the definition of a conceptual model for a given law [13], or of a structural model. The goal of the new version of GaiusT is to deal with all these tasks, integrating and consolidating the results of previous researches but also adding new modules.

As regards the implementation of a SSRE, a variety of programming languages and technical frameworks could be used and it is quite common that some of the linguistic activities (e.g. parsing, word frequency analysis) or editing features (both on texts and on the graphical diagrams for the conceptual models) are developed by customizing open source programs or looking for existing libraries. These modules have then to be integrated into the framework taking into account transportability and inter-operability issues. All these issues were critical to the success of the reengineering project for GaiusT 2.0; the complexity of the system implied a high risk of worsen its performances relative to the less automatic process supported by GaiusT.

2.2 Requirements for a System Supporting Requirements Extraction from Legal Documents

There are a number of users involved in an SSRE - requirements analysts, business analysts, lawyers, experts on standards, experts on linguistics, developers, domain experts (e.g. health experts for laws like HIPAA) to name but a few. Users of the SSRE can be classified into three main categories: researchers, developers and end users. Each of these classes have specific needs and requirements that an SSRE from legal documents has to satisfy and are described in the following adding to those given in [10].

Researchers. SSRE from legal documents are often developed by researchers working on large-scale international research projects (e.g. European projects involve research groups in different countries) which often last years. This is indeed the case with GaiusT 2.0. The system evolved from a research project to design and implement semantic annotation tools where the final framework, called Cerno, was a large-scale semantic annotation system [7]. The Cerno project started about ten years ago [14]; from its inception the researchers were scattered in different organizations and countries. The application of Cerno to legal documents required a re-design of some of its modules and the development of new ones and resulted in the first version of GaiusT [6]. These evolutions revealed further high-level requirements to be satisfied by a new version of GaiusT:

- to deal with conceptual models other than the ones used in GaiusT, describing laws at a different abstraction level [10], for example the model used in the Nómos project [15] (for another model see [16]);
- to allow researchers in different fields - compliance with legal documents implies a multi-disciplinary approach - to use the modules of the system;
- additional features for choosing the concepts to be annotated in legal documents for a given research project;
- management of multi-accesses, spaces and tools for different projects;
- support while testing the impact of changes in the annotation schema on the output of the annotation steps;
- storing all the artifacts of a project for traceability;
- interfaces to the core modules to support the definition of annotation rules for new conceptual models;
- modules for the analysis and comparison of output produced in all the steps of the requirements extraction process.

Developers. Developers of an SSRE have not only to implement solutions for a variety of tasks but they also have to choose from a large number of platforms, technologies, libraries, programming languages and standards. The most critical decisions made for the GaiusT 2.0 project are related to the need to integrate modules, libraries and new software and, whenever possible, to use open source resources. Mashing them up is often more challenging than expected. For example, available off-the-shelf modules frequently turn out to be incompatible, or have low performance levels, or do not implement all the needed features. An analysis of the available tools and resources (and of their adaption needs) is given in [9]. The main developers' requirements identified for GaiusT 2.0 are the following:

- to address the limitations of TXL, which had resulted less scalable than expected to process long documents, and to support an evolutionary approach to the implementation of prototypes for the modules of GaiusT 2.0;
- to adopt the most suitable technologies to implement GaiusT 2.0 as a full-fledged web-based system, a high level requirement shared by all the user classes;
- to port available open source linguistic tools and resources that often are in Java [17–19] in C#, the language adopted for the Gaius project.

End users. Prospective users of GaiusT 2.0 are requirements engineers or analysts, and lawyers. A particular class of users of SSRE are the participants in experiments, often students whose role and needs are similar to those of (junior) requirements analysts. For these users, needs are also related to the support of the manual analysis and annotation of legal texts (necessary to create a gold reference for comparison with the GaiusT 2.0 results). For end users, requirements are mainly related to:

- ease of use and learnability, which are hampered by the trade-off between the need of the developers to implement new solutions in an effective and efficient process, often with throwaway prototypes, and the need of the researchers to test their analysis and annotation approaches without the help of a developer;
- usability of the annotation schema and engine modules (Table 1), that turned out to be critical also to allow researchers and developers to gather users' feedback and data to compare different solutions to a specific problem;

Table 1. GaiusT component comparison matrix.

Component/Module	GaiusT	GaiusT Version 2.0
Annotation schema generator		
Conceptual model parser	–	Parses XMI models
Word frequency list (WFL) generator	Generates WFLs	+ Generates n-grams
Lexical database manager	Uses WordNet to validate WFL entries	+ Uses Google n-grams to validate n-grams
Part of Speech (PoS) manager	External (TreeTagger)	Embedded (porting Java libraries in C#)
Template Library	–	Generates templates for different types of laws (domains) and projects
Annotation engine		
Text extractor	Extracts text from different file formats	+ Parallel processing and web downloader
Normalizer	Clear texts and puts 1 sentence per line	+ Parallel processing
Document structure analyzer	TXL rules	Regular expressions
Annotation rules generator	External (TXL rules)	Embedded (regular expressions)
Annotation generator	Annotates concepts	Annotates concepts and relationships
Repository		
Relational database	Annotated items	Structured artifacts
Evaluation module	Provides data for basic statistics	Provides HTML statistical reports, graphical traceability, document browsing
Graphical model generator		
Graph generator	–	Generates graphical models of annotations
Graphical User Interface		
Interface	Client application	Web application (responsive, multi-user)

- support to group work as requirements analysts and lawyers are sometime working in different places and have to cooperate on interdisciplinary projects;
- to annotate legal documents at different granularity levels, that is associating texts of different size - ranging from single words to the entire document - to a given concept;
- a specific requirement for lawyers is that of having legal citations for external references according to the Bluebook [12].

3 The Framework

3.1 GaiusT

The explanation of GaiusT given here is abbreviated to what is necessary to understand this paper. A full description can be found in [6]. GaiusT was designed as a multi-phase framework to semi-automatically identify deontic concepts in legal documents. It has been successfully applied to the annotation of documents both in English and in Italian. Starting from a meta-model of legal concepts - a conceptual model - GaiusT identifies instances of complex deontic concepts, such as rights and obligations. From the meta-model one can derive an annotation schema which specifies the rules for identifying the instances of legal concepts in a document. For each concept, the annotation schema specifies its identifiers with their syntactic roles, and patterns are used to represent complex concepts. Indicators can be single words, phrases, or references to previously parsed basic entities while patterns are collection of concepts related by regular expressions. For each legal document, the output of GaiusT is an XML file that lists all the annotations generated by the system.

The first version of GaiusT was about 50 k lines of code and more than 130 MB in size.

3.2 GaiusT 2.0

Based on the requirements gathered for the different classes of users described in Sect. 2.2 and the limitations identified from the experiments and applications of GaiusT, the SSRE evolved into GaiusT 2.0, a web-based system with new and improved modules. Most of the requirements are related to the need to deal with different domains (privacy for health data, accessibility and ICT, etc.) and with different legal systems where general principles determine different conceptual models (descriptive metadata) and different document structure (structural metadata). The architecture of GaiusT 2.0 includes a number of components and is represented in Table 1, where each of them is compared to the corresponding module of GaiusT.

The Annotation schema generator supports the process to semi-automatically create annotation schemas i.e. to associate each concept in a conceptual model to a list of indicators to be used to identify the concepts in a legal document. For this component, two of the modules – the WFL generator and the Lexical database manager – have been fully reengineered. In particular, the statistical analyses of the input documents provided by the WFL generator module have been integrated with the calculation of TF/IDF (term frequency/inverse document frequency). The Lexical Database Manager has been extended by adding Google n-grams [20] (consisting of 1-grams to 5-grams; the last available version contains more than 346 million bigrams and more than 2,616 million trigrams) and the WordNet ontology [21]. The part of speech (PoS) module has been replaced by porting in C# sharpNLP (in Java [22]), and expanding it, as it only covers part of the English language and does not deal with Italian. A new Template library module has been added to support the creation of annotation templates and the reusability of items (concepts, indicators, patterns, structural items). This module helps the users in the definition of annotation schemas according to the legal documents to be analysed. For example, a user can define a template for the semantic annotation of American laws, including structural patterns, legal concepts and patterns to annotate any American legal document; as a result compliance requirements extraction from a specific American law does not have to start from scratch as items defined in the template can be reused. Besides, to support different users groups and annotation works, a project template was added.

The Annotation engine supports a number of steps in order to: (1) extract text from different file formats (Text extractor), (2) normalize the input document by removing leading and unprintable characters and trailing spaces to produce a text document where each line represents a phrase (Normalizer); (3) annotate text units with tags for structure and cross references identification (Document structure analyzer); (4) generate rules for annotating concepts; (5) identify concepts, applying the annotation schema i.e. indicators and patterns defined for the conceptual model; find relationships between identified concepts by using the heuristic patterns (Annotation generator). In GaiusT, the Annotation rules generator was in TXL [8]; to overcome its limitations, GaiusT 2.0 uses regular expressions. These expressions allow to effectively capture grammar and syntactic rules for the chosen languages. The design and implementation of this component has been the most demanding in the entire project.

The redesign of the system included the expansion of the Database component, a relational database that increased from 10 to 71 tables. A new module deals with unstructured documents and artifacts of the annotation process.

A new component, the Graphical model generator, has been added to create graphical models with the instances of the deontic concepts extracted from the annotated texts (a specialized module for the conceptual model used for Nómos is described in [23]).

The GUI component is the result of the migration of GaiusT to a web-based system. An explorative web prototype was developed using NodeJs [24]. Later, for maintenance and integrability reasons, it was migrated to ASP.NET MVC [25]. Mobile and multi-users accesses are also supported.

The actual version of GaiusT 2.0 is about 120 k lines of code, a more than two-fold increase, and 415 MB in size of libraries.

4 Evaluation

GaiusT 2.0 satisfies most of the requirements described in 2.2. It is a full-fledged web-based system and it is used by researchers working in groups and on different projects. The system deals with multi-accesses, storing artefacts for projects which adopt different annotation models and schemas. Researchers and end users utilize it for new experiments and applications. Besides HIPAA, the results of which are reported in the next subsection, a new project is now applying GaiusT 2.0 to the English version of German [26] and Italian privacy law [27] i.e. laws whose documents have a different structure and are in a different domain. For the developers, technologies deployed for implementing GaiusT 2.0 offer better support to extend and adapt the system. Although quantitative results are not yet available, the experiences gained so far confirm that GaiusT 2.0 usability has largely improved thanks to the web interface that has reduced the time needed to start using the system. Requirements not yet implemented regard the future work outlined in the conclusion.

To evaluate the new version of GaiusT we applied it to the HIPAA (with the same assessment methodology described in [11]) in particular to Sect. 164.524 and Sect. 164.526, thus allowing us to compare the results obtained from GaiusT 2.0 with those reported in [6]. We adopted experts' manual annotation as gold standard as an upper bound of what automatic annotation can do. Moreover, the performance of the tool was calibrated with the degree of disagreement among experts in the experiment (an average of 23 %). We used standard information retrieval metrics[1] - precision, recall, fallout, accuracy, error and F-measure – to compare the performance of the tool to that of humans. The results of the experiment are presented in Table 2.

Table 2. Evaluation rates of experts and the two versions of GaiusT annotating HIPAA.

	Human	GaiusT	GaiusT 2.0
Precision	0.83	0.84	0.86
Recall	0.92	0.87	0.90
Fallout	0.49	0.42	0.73
Accuracy	0.78	0.78	0.74
Error	0.22	0.22	0.28
F-measure	0.86	0.85	0.87

[1]
- Recall is a measure of how well the tool performs in finding relevant items TP/(TP + FN);
- Precision is a measure of how well the tool performs in not returning irrelevant items TP/(TP + FP);
- Fallout is a measure of how quickly precision drops as recall is increased FP/(FP + TN);
- Accuracy is a measure of how well the tool identifies relevant items and rejects irrelevant ones (TP + TN)/N;
- Error is a measure of how much the tool is prone to accept irrelevant items and rejects relevant ones (FP + FN)/N;
- F-measure is a harmonic mean of recall and precision 2 × Recall × Precision/(Recall + Precision)

where TP is the number of items correctly assigned to the category; FP is the number of items incorrectly assigned to the category; FN is the number of items incorrectly rejected from the category; TN is the number of items correctly rejected from the category; and N is the total number of items N = TP + FP + FN + TN.

The new system has improved recall without diminishing precision. It should be noted that, taking into account the disagreement among experts, GaiusT 2.0 does a little bit better with respect to precision and F-measure than human annotators.

GaiusT 2.0 has also improved the process of structure identification and the identification of structure elements is performed with a recall of 100 %. The tool is able to correctly identify all cross-references but, for example, the cross reference "… in paragraphs (b)(1) (ii)(A) or (B)" is partially matched as "paragraphs (b)(1)" because in the input text there is an extra-space between "(b)(1)" and "(ii)(A)" so that the regular expression fails to catch the entire occurrence.

Further experiments need to be carried out to evaluate the productivity of the new system. To this end the effort to adapt it to the analysis of new laws had to be taken into account: by providing a larger support to the activities in the annotation process, GaiusT 2.0 should help to reduce the time needed for the annotation. Real time support (or nearly real time) would be useful to analyze online documents; for example, for moderating posts according to given social networking rules.

5 Related Work

The extraction of requirements from legal documents is a challenging task and several approaches have been presented to tackle the problem, ranging from manual analysis to the use of NLP techniques to semi-automate the process.

Heuristic rules for extracting rights and obligations from regulations are given in [28]; in this field Breaux proposed a framework to acquire legal requirements using a systematic frame-based requirements analysis methodology. The framework uses an upper ontology (which describes general concepts that are the same across all knowledge domains) to classify regulatory statements, for context-free mark-up and to handle the structural organization of regulatory documents. Recently a Hit (Human Intelligence Tasks) was called for on Amazon Mturk [29] to have people apply their approach to the annotation of legal documents.

To automatically support the most critical steps of the semantic annotation process some researchers propose the application of NLP tools and machine learning approaches to achieve full, domain-independent analysis of legal documents. The purpose of these approaches includes activities like the extraction of a particular kind of concept and the identification of text fragments relevant for a given goal or for document classification. In [30] authors propose a name entity recognition methodology for the automatic identification of actors relevant for the analysis of a regulation or a law. Lesmo et al. [31] present TULSI, a system for the extraction of semantic annotations from legal documents, a core task of the GaiusT 2.0 architecture. GATE (General Architecture for Text engineering) [32] provides a rule-based language to build annotation patterns with an approach similar to that used by GaiusT 2.0, however GATE is not web–based and none of its modules could be integrated into it.

As regards the analysis of the structure of legal documents and the management of cross references - seemingly easy tasks but a complete, affordable, robust system is still lacking - it is worth citing two recent papers [33, 34]. Both propose a systematic approach based on regular expressions and linguistic techniques to identify cross references in legal text: a necessary step, but unfortunately insufficient to fully address this task. GaiusT 2.0 uses regular expressions to capture formal partitions (the document structure) and cross-references, achieving good performances as an evolution of the approach described in [35].

Other authors have focused on the problem of managing knowledge in legal documents, to support lawyers in searching for legal cases and sentences related to their professional activities. Among the tools used to manage legal knowledge, EuNomos [36] supports the identification and classification of legal documents through an ontology-based process. Documents are analyzed and annotated in XML format, according to the Akoma Ntoso standard [37], using a combination of linguistic and machine learning techniques (text classification, text similarity and pattern matching) to extract concepts and structure from the legal texts. GaiusT 2.0 also uses conceptual and structural models - small ontologies intended to capture domain semantics - to identify main concepts and relevant clues in the structure of the texts. Another knowledge management tool for lawyers is SALEM [38], which uses linguistics technologies and machine learning (a technique similar to SMV, support vector machine) to extract provisions from legal documents. An interesting result is reported in [39] where authors compare machine learning classification of legal sentences versus pattern based classification and highlight how a pattern based classifier resulted more robust in the categorization of legal documents than a SVM classifier. Gaius T 2.0 is not directly comparable with any of the existing tools, even though the techniques used for semantic annotation are shared by many of the existing approaches. GaiusT 2.0 differs from all them as it is a comprehensive large scale system which includes modules to deal with most of the tasks related to requirements extraction from legal documents. It uses a semi-automatic pattern-based approach to analyze legal documents dealing with the needs related to the application of the system to different domains and legal systems. Worth citing are the (complementary) approaches addressing the problem the other way around, that is starting from software requirements and checking if they comply with a given law; see for example [40]. A methodology for reasoning and modeling of obligations is described in Hashmi [41]. Reasoning would be useful to check if a given event breaks a law (rule) and is one of the goals of Nómos [15].

6 Conclusions and Future Work

Regulatory compliance for software systems is a complex problem. In this paper we described GaiusT 2.0, a large-scale system supporting the extraction of legal requirements from textual documents. The design of the system started in 2013 and evolved from GaiusT by improving and including new modules. The design of the modules was based on the users' needs gathered in different projects and on the analysis of available linguistic resources and development technologies. GaiusT 2.0 addresses most of the requirements described in Sect. 2.2. However there are a number of issues that have to be further investigated and features to be added. The following are prospective points for future study: (1) add reasoning mechanism to improve the annotation step; (2) adopt one of the standard mark-up languages for legal documents for example, LegalRuleML [42]; (3) add a layer to transform legal concepts into legal requirements, a step that none of the existing systems support yet.

Acknowledgements. Nicola Zeni's work was supported in part by the ERC advanced grant 267856 'Lucretius: Foundations for Software Evolution'. Our thanks go to the experts and users who made themselves available to test GaiusT 2.0.

References

1. IEEE: IEEE Computer society predicts Top 9 technology trends (2016). http://www.computer.org/web/pressroom/Technology-Trends-2016
2. Systems Engineering Body of Knowledge (SEBoK) v.1.5.1. (2015). http://sebokwiki.org/wiki/Guide_to_the_Systems_Engineering_Body_of_Knowledge_(SEBoK)
3. RELAW workshops. http://gaius.isri.cmu.edu/relaw
4. Jurix conferences. http://jurix.nl/proceedings
5. Davis, E.: The singularity and the state of the art in artificial intelligence: the technological singularity. In: Ubiquity symposium (Ubiquity 2014), 12 pages (2014)
6. Zeni, N., Kiyavitskaya, N., Cordy, J.R., Mich, L., Mylopoulos, J.: GaiusT: supporting the extraction of rights and obligations for regulatory compliance. Req. Eng. **20**(1), 1–22 (2015). (online 2013)
7. Kiyavitskaya, N., Zeni, N., Cordy, J.R., Mich, L., Mylopoulos, J.: Cerno: light-weight tool support for semantic annotation of textual documents. Data Knowl. Eng. **68**(12), 1470–1492 (2009)
8. Cordy, J.R.: The TXL source transformation language. Sci. Comput. Program. **61**(3), 190–210 (2006)
9. Zeni, N., Mich, L., Mylopoulos, J., Cordy, J.R.: Applying GaiusT for extracting requirements from legal documents. In: 6th International Workshop on Requirements Engineering and Law (RELAW 2013), pp. 65–68. IEEE (2013)
10. Zeni, N., Mich, L.: Usability issues for systems supporting requirements extraction from legal documents. In: 7th International Workshop on Requirements Engineering and Law (RELAW 2014), pp. 35–38. IEEE (2014)
11. Health Insurance Portability and Accountability Act – HIPAA. http://www.hhs.gov/ocr/privacy/hipaa/understanding
12. Bluebook. https://www.legalbluebook.com
13. Souza, V., Zeni, N., Kiyavitskaya, N., A, P., Mich, L., Mylopoulos, J.: Automating the generation of semantic annotation tools using a clustering technique. In: Kapetanios, E., Sugumaran, V., Spiliopoulou, M. (eds.) NLDB 2008. LNCS, vol. 5039, pp. 91–96. Springer, Heidelberg (2008). doi:10.1007/978-3-540-69858-6_10
14. Kiyavitskaya, N., Zeni, N., Cordy, J.R., Mich, L., Mylopoulos, J.: Applying software analysis technology to lightweight semantic markup of document text. In: Singh, S., Singh, M., Apte, C., Perner, P. (eds.) ICAPR 2005. LNCS, vol. 3686, pp. 590–600. Springer, Heidelberg (2005). doi:10.1007/11551188_65
15. Siena, A., Jureta, I., Ingolfo, S., Susi, A., Perini, A., Mylopoulos, J.: Capturing variability of law with *Nómos* 2. In: Atzeni, P., Cheung, D., Ram, S. (eds.) ER 2012. LNCS, vol. 7532, pp. 383–396. Springer, Heidelberg (2012). doi:10.1007/978-3-642-34002-4_30
16. Winkels, R., Hoekstra, R.: Automatic extraction of legal concepts and definitions, pp. 157–166. IOS (2012)

17. OpenNLP. https://opennlp.apache.org
18. Stanford CoreNLP. http://nlp.stanford.edu/software
19. NLP toolkit. http://www.nltk.org
20. Google n-grams. http://storage.googleapis.com/books/ngrams/books/datasetsv2.html
21. WordNet. https://wordnet.princeton.edu
22. SharpNLP. https://sharpnlp.codeplex.com
23. Zeni, N., Seid, E.A., Engiel, P., Ingolfo, S., Mylopoulos, J.: Building large models of laws with NómosT. In: 35th International Conference on Conceptual Modeling (ER 2016). Springer (2016, to appear)
24. NodeJs. http://nodejs.org
25. ASP.NET MVC. www.asp.net/mvc
26. Germany's Federal Data Protection Act - Bundesdatenschutzgesetz – BDSG. https://www.loc.gov/law/help/online-privacy-law/germany.php
27. Data Protection Code - Legislative Decree no. 196/2003. http://194.242.234.211/documents/10160/2012405/DataProtectionCode-2003.pdf
28. Breaux, T.D., Antón, A.I.: Analyzing regulatory rules for privacy and security requirements. IEEE Trans. on SW. Eng. **34**(1), 5–20 (2008)
29. Amazon Mechanical Turk. https://www.mturk.com
30. Dozier, C., Kondadadi, R., Light, M., Vachher, A., Veeramachaneni, S., Wudali, R.: Named entity recognition and resolution in legal text. In: Francesconi, E., Montemagni, S., Peters, W., Tiscornia, D. (eds.). LNCS (LNAI), vol. 6036, pp. 27–43Springer, Heidelberg (2010). doi:10.1007/978-3-642-12837-0_2
31. Lesmo, L., Mazzei, A., Palmirani, M., Radicioni, D.P.: TULSI: an NLP system for extracting legal modificatory provisions. Art. Intell. Law **21**(2), 139–172 (2013)
32. GATE. https://gat.ac.uk
33. Sannier, N., Adedjouma, M., Sabetzadeh, M., Briand, L.: An automated framework for detection and resolution of cross references in legal texts. Req. Eng., 1–23 (2015). http://link.springer.com/journal/766/onlineFirst/page/1
34. Oanh Thi, T., Bach Xuan, N., Le Minh, N., Akira, S.: Automated reference resolution in legal texts. Artif. Intell. Law **22**(1), 29–60 (2014)
35. Zeni, N., Kiyavitskaya, N., Mich, L., Mylopoulos, J., Cordy, J.R.: A lightweight approach to semantic annotation of research papers. In: Kedad, Z., Lammari, N., Métais, E., Meziane, F., Rezgui, Y. (eds.) NLDB 2007. LNCS, vol. 4592, pp. 61–72. Springer, Heidelberg (2007). doi:10.1007/978-3-540-73351-5_6
36. Boella, G., di Caro, L., Humphreys, L., Robaldo, L., van der Torre, L.: NLP challenges for EuNomos a tool to build and manage legal knowledge. In: 8th International Conference on Language Resources and Evaluation (LREC 2012). European Language Resources Ass., Istanbul (2012)
37. Akoma Ntoso standard. http://www.akomantoso.org
38. Soria, C., Bartolini, R., Lenci, A., Montemagni, S., Pirrelli, V.: Automatic extraction of semantics in law documents. In: 5th Legislative XML Workshop (2007)
39. de Maat, E., Krabben, K., Winkels, R.: Machine learning versus knowledge based classification of legal texts. In: 23rd Conference on Legal KW and IS (JURIX 2010), pp. 87–96. IOS Press (2010)
40. Massey, K.: Legal requirements metrics for compliance analysis. Ph.D. dissertation, North Carolina State University (2012)
41. Ingolfo, S., Siena, A., Mylopoulos, J., Susi, A., Perini, A.: Arguing regulatory compliance of software requirements. Data Knowl. Eng. **87**, 279–296 (2013)
42. LegalRuleML. https://www.oasis-open.org/committees/legalruleml