Information Science

Master Thesis

Ontology Based Information Extraction in the License Domain

Author: Ken-Thomas Nilsen

Supervisor: Andreas Lothe Opdahl

June 1, 2015

Abstract

All computer users needs to deal with End User License Agreements(EULA). Every time we install software or sign up for a web service we are expected to read and accept such a legal agreement. For most users this is only a slightly annoying step in the process, and we have been conditioned through many years just to accept these texts unwittingly. These texts are often long and filled with legal jargon, and hence almost impossible for an interested lay person to understand. In this thesis I have explored the use of common natural language processing and knowledge extraction techniques in the domain of EULAs and license agreements. My project have included the development of an artifact that use these techniques, and then makes the data available through the usage of semantic technology. It extracts document structure, named entities, binary relations and definitions. I have built a classifier that use topic modeling to find binary relations. These topics are then used by the classifier to decide in what topic a given binary relation belongs. I have also experimented with the use of text search in ontologies to try and find the realization of a given binary relation in a specified ontology. The artifact is run on a specific EULA, and I evaluate the knowledge extracted from each of the techniques investigated. I have not tried to find the best existing implementation of a technique, but instead evaluated the kind of data extracted and what specific needs that arise in the domain of licenses.

The extraction and representation of the structure of the license were a success, and I have used that extraction as a basis for a vocabulary that describes my extracted data. All extractions are related directly back to the text were it was extracted. This is because of the legal documents role in a judicial system. As the text decide the results in court, it is important to keep a reference back to the source document. Because of this my system can be viewed as a system that semantically enrich a text, but without reasoning about higher levels of knowledge. I conclude that extracting knowledge using common NLP and knowledge extraction tools is feasible and opens up for research into its use in document summarization and in facilitating comprehension of such legal texts. I also conclude that my classifier for binary relations has weak performance, but list a set of changes and prerequisites that would warrant further experimentation. I also conclude that we will need to take special steps in the construction of our ontologies for my experiment with using the built in comments and labels in an ontology to be viable.

Acknowledgments

I would like to thank my supervisor Andreas Lothe Opdahl for his help during the work on this thesis. Without his belief in the project and the resulting boosts of motivation and inspiration from our meetings would this not have been possible. I would also like to thank everyone who has shared frustrations and ideas during our lunch breaks, and made this year memorable. Sometimes does a patient ear listening to me fumbling through an explanation help more than the most well written text book.

Contents

List of Figures

List of Tables

Listings

Chapter 1

Introduction

Software and web applications often demand that you sign an *End User License Agreement* before you can start using the product. These licenses are seldom read, and a user will inevitably accept a contract without knowing the content or scope of the license agreement. By using techniques from *Information extraction*, *Ontology Based Information Extraction* and *Natural Language Processing* I wish to make this kind of legal text more accessible. A successful application of techniques from these disciplines should result in RDF triples representing the license. Such a representation will give us the tools we need to visualize the licenses, compare them automatically and also make it easier for the user to define what kind of restrictions they will and will not accept. The adoption of ontologies has increased across many domains such as document retrieval, image retrieval, bioinformatics, manufacturing, industrial safety, environment, disaster management, e-government, e-Commerce, tourism and most interesting in this context, law(Wong, Liu, & Bennamoun, 2012, p. 2). Shadbolt, Berners-Lee, and Hall (2006) describe the need for standardization, and express how the success of the web of data builds on not only technology, but is also dependent on a social effect. When we get enough users, and social traction for this kind of technology, the users will demand machine readable licenses. Until then this project can contribute to such an development.

There is done a lot of research in the legal domain, but not that much in the sub domain of licenses. I haven't found any papers that discuss adoption of a license agreement ontology, and most interesting use cases require an existing community adopted ontology. As this were not available have my project focused on the use of existing extraction techniques to describe what is available in such a legal text.

1.0.1 Research questions

My project builds on several important themes like *Natural Language Processing*, *Machine Learning* and *Information Extraction* in different forms. The different techniques are interrelated and covered in several of these research domains. This has given me a big toolbox to choose from during my project. My research questions are:

- Can information extraction using NLP and machine learning techniques describe an EULA in a satisfying manner?
  - What parts of an EULA can be extracted using existing techniques?
- Can information extraction using NLP and machine learning techniques extract normative knowledge?
  - Can knowledge be mapped from text to an existing ontology?

My project results in an artifact that extracts information from an EULA, and makes this available using semantic technology. As shown in Section 2.2.5 must a legal ontology cover several layers of information. This stack of layers describe different kinds of knowledge that is contained in and related to a legal text. My artifact extracts knowledge directly from text, and keeps the EULA intact. The legal text is what is used by the judicial system. That means that my extractions will be unable to reason about higher layers of legal knowledge, but it can be used as part of a tool for understanding an EULA. The value of this become obvious as the alternative is to wait for a consensus and adoption from the service and software providers. This is complicated both by the inherent complexity of legal systems, international differences and legal language, but also by the fact that software and service providers got no real incentive to create machine readable licenses until users demand them.

Chapter 2

Theory

I have based this project on theory from many different fields of study. This way I have covered a broad range of techniques and tried to cherry pick techniques applicable to my project.

2.1 Ontology Based Information Extraction

Wimalasuriya and Dou (2010, p. 309) defines Ontology-Based Information Extraction as

> a system that processes unstructured or semi-structured natural language text through a mechanism guided by ontologies to extract certain types of information and presents the output using ontologies.

This definition is broad, but the authors argue that it encompass the most important factors that make an OBIE system different from an Information Extraction system. They also define a set of functionality that a system needs to count as a OBIE system.

- Process unstructured or semi-structured natural language.
- Present the output using ontologies.

- Use an IE process guided by an ontology.

An OBIE system needs to use an ontology to guide its extraction, but such a system might create its own ontology or take an ontology as a parameter. In this project I will consider systems that take an ontology as a parameter, because of my choice of performance measure.

Wimalasuriya and Dou (2010, p. 309) lists these use cases were OBIE systems show a lot of promise:

- Automatically processing information contained in natural language text.
- Creating semantic contents for the Semantic Web.
- Improving quality of ontologies.

License texts are formed in natural language, even though legalese might be a limited form of natural language. This creates semantic contents in that it is not hard to imagine a central repository of extracted information. And lastly by using the OBIE system as an evaluation of the ontology quality, we might be able to deduce good representations and weaknesses with the different ontologies. They also list the main extraction techniques used in the systems from the survey(Wimalasuriya & Dou, 2010, p.312).

- Linguistic rules represented by regular expressions.
- Gazetteer lists.
- Classification techniques.
- Construction of partial parse trees.
- Analyzing HTML/XML tags.
- Web-based search.

All techniques are not used by all systems, and in this survey they use the differing techniques to classify the different OBIE systems. Shah and Jain (2014) reference many of the same sources as Wimalasuriya and Dou (2010), and conclude that there are several directions for future work with OBIE Systems, and mention:

- Improving the IE process
- Generating semantic content for the semantic web
- implementing Ontology Based web services for results.

And they then reiterate that the possibility of generating content for the semantic web is one of the major factors that make OBIE an interesting field of research.

2.1.1 Knowledge Extraction tools

Gangemi (2013) states that:

> In the last years, basic NLP tasks: NER, WSD, relation extraction, etc. have been configured for the Semantic Web tasks including ontology learning, linked data population, entity resolution, NL querying to linked data.

Based on this he presents a landscape analysis of the current tools for Knowledge Extraction from text, when applied for use on the Semantic Web. He concludes that tools used for Knowledge Extraction provide good results for all the basic tasks tested, but the evaluation and meassures differ across different tasks and tools. This paper should therefor be viewed as a first step into integrating measurements of different tasks from the perspective of providing useful analytic data out of text.

2.1.2 Ontology population

Cimiano (2006, p.26) defines Ontology population as

> learning the extensional aspects of a domain.

This means finding instances of concepts and also relations, where an instance is an objects set-membership in a class. In general this is a difficult task and would in practice require full knowledge of NLP(Cimiano, 2006, p.232). Ontology Population

is closely related to *Named entity recognition*(NER). In this way he limits the scope to a fixed and small number of class, since that is the domain of NER. He then provides an overview of common approaches used in ontology population:

Lexico-syntactic Patterns can extract both instances and relations, and is based on an assumption that the *Noun Phrase* at the hyponym position is an instance.

Similarity-based Classification use the context of a phrase to disambiguate its sense. With these techniques data sparseness becomes a problem.

Supervised Approaches predict the category of an instance with a model induced from training data using machine-learning techniques. This includes NER tasks and using templates. This kind of annotation demands a big trainingset, and hence have problems scaling. It also rely on an assumption that documents have a similar structure and content.

Knowledge-based and Linguistic Approaches see the population task as a disambiguation problem and this as a byproduct of natural language understanding. By using linguistic evidence and domain-specific ontology to weight, discard and refine hypotheses iteratively, they find the most satisficing hypothesis in the hypothesis space.

He then concludes that supervised techiques is not a feasible option for the task of classifying entities with respect to a large ontology(Cimiano, 2006, p.237).

2.2 Law and License ontologies

License text can be viewed as a subdomain of legal texts. There is a lot of research done in the legal domain, and the knowledge obtained might be applicable to licenses to. I start with the Legal domain in general, and then follow up with some of the work done in license ontologies and license management.

2.2.1 Legal and Law domain

Ontology building is an active research field and, as noted above, the adoption of these techniques are starting to become more widespread. The use of ontologies also grows in the legal domain(Lenci, Montemagni, Pirrelli, & Venturi, 2009, p. 76). Legal ontologies have been developed before the invention of the semantic web, were they were used for knowledge management and as knowledge bases(Benjamins, Casanovas, Breuker, Gangemi, & Marx, 2005, p. 9). In the EU IST project Semantic Knowledge Technologies(SEKT), they used Ontology learning to automate extraction of an ontology for use in a decision support system for newly appointed judges in the Spanish legal system(Völker, Langa, & Sure, 2008, p. 1).

Lenci et al. (2009) describes their use of the T2K(Text to knowledge) tool for automatically extracting ontological knowledge from Italian legislative texts. They claim that all ontology learning experiments carried out in the legal domain are mainly focused on concept extraction as a primary step of ontology development. This is corroborated by Wong et al. (2012, p. 30) who also stress that most ontologies are in the lightweight spectrum. Lenci et al. (2009, p. 77) splits the automated extraction of ontologies in the legal domain into two different approaches. The first one using *frequency of definition* and the second one analyzing *term extraction*.

2.2.2 Frequency of definitions

The first approach use the frequency of definitions in the legal text as a source of information to learn domain relevant concepts as well as relations. This approach are exemplified through Walter and Pinkal (2009). They use a rule-based approach for extracting and analyzing definitions from parsed text. Using this technique they managed to get high *precision* in their returned results, but their *recall* is still lacking. They also argue that the role of definitions in legal texts are particularly important, because of their explicit and precise nature. Their technique might have limited recall, but in combination with other techniques, it can help in text-driven ontology learning.

2.2.3 Term extraction

The other approach focuses on term extraction as a fundamental prerequisite for the identification of concepts and relations in normative texts. This approach follows the basic assumption that domain-specific concepts are typically phrased as terminological units. Lenci et al. (2009)'s paper are in this category. They conclude that there are great potential for this kind of technology.

In the context of Ontology Learning, we can define Walter and Pinkal (2009) as a linguistics-based system, were they first parse the text using the *Preds-parser* and then use rules to extract the definitions. The tool described by Lenci et al. (2009) is a hybrid system. The tool is called T2K, and uses both statistical, NLP and machine learning techniques to extract knowledge.

Venturi (2010) investigates the peculiarities of legal language. He claims that little attention has been devoted to the peculiarities of legal language, and NLP-techniques for identifying and analyzing these. It compare results obtained from parsing legal text from parsing ordinary language, and he finds a broad bias towards *nominal realization of events* rather than verbal in legal text. He claims that this nominal realization of events poses a serious challenge for knowledge-representation systems.

McCarty (2007) makes the case that the technology of natural language processing has advanced in such a way recently that many barriers have been crossed. This paper describes the use of a tagged corpus as a training set, and a *Lexicalised Head-Driven Statistical Model*. There is no such corpus specifically for the legal domain, but it conjectures that *Penn Wall Street Journal Treebank*[1] got enough legal terminology that most of the statistical information is available there. Such an annotated corpus would open up for the use of supervised learning techniques.

2.2.4 License ontologies

There has been done some work on licenses, but not from a users perspective.

[1]The Penn Treebank Project annotates naturally-occuring text for linguistic structure. They produce skeletal parses showing rough syntactic and semantic information,a bank of linguistic trees. https://www.cis.upenn.edu/~treebank/

Zhao and Perry (2008) sees a license agreement as the knowledge source for a license management system. They argue that a machine readable representation of a license agreement is highly desirable. Creating an ontology will both fill the role of such an representation and also get the other benefits of an ontology, like knowledge reusability. In their example they construct their own example license agreements, and use them to show the use of an ontology for the license agreements.

Anjomshoaa, Weippl, Tjoa, and Asfandeyar (2009) develop an abstract license ontology that contains common features found in different license agreements. Then they model three real world licenses using this ontology. They describe a use case were a university needs a new software product for its students. They have a set of requirements to this product which is formalized and used to validate their approach. This formalism is used to create SPARQL queries which return the licenses that fit the requirements. Their idea is to create a basic ontology that is easy for a company to extend if it is missing any requirements, which makes it possible for a user to query a database that returns the products that fit the users requirements.

Ahmed, Anjomshoaa, Asfandeyar, Tjoa, and Khan (2010) have the same goal as Anjomshoaa et al. (2009), which is to help an end user make decisions about terms associated with a license agreement. Here they extend the original idea to a full license agreement management system that fits into the *SemanticLIFE* framework. They conclude that such a centralized system for license management saves the users of the time reading the license agreements and finding desired information.

Lastly Asfand-e-yar and Tjoa (2013) extend the ontology from Anjomshoaa et al. (2009) with a sub ontology to elaborate a section of domain knowledge. The proposed use case is a semantic web model for Online shopping license agreements. They show the ease of wich you might extend the ontology to describe domain knowledge. They claim that by using similar methods it could be used to adopt other license agreements, for example Airline's Terms and Condition.

2.2.5 Legal document ontology structure

Palmirani, Cervone, and Vitali (2011, p. 167) describe a legal resource as a complex multi-layer informative architecture that includes several perspectives of analysis. The first layer is the text, which is officially approved by a legal authority. This text

got a document structure that organizes the text. Then there is metadata involving the document, such as keywords, procedural steps, lifecycle of the document and identifiers. On top of this we got knowledge about the reality in which the document act a role, such as previous judgments. Then lastly we got the interpretation and modeling of the meaning of the text under legal perspective.

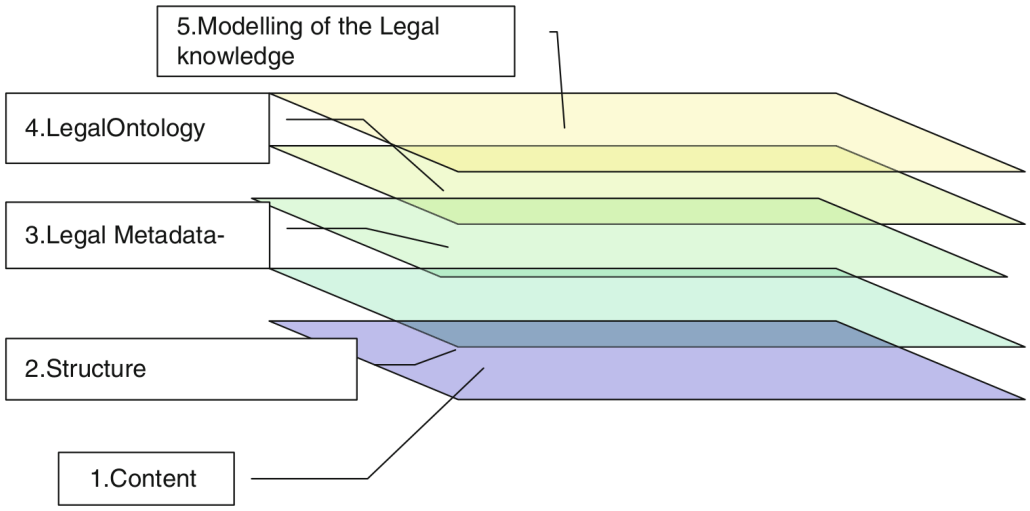

Figure 1: Layers of representation in legal document modelling(Palmirani, Cervone, & Vitali, 2011, p. 168)

2.3 Adoption

The need for and interest in machine readable licenses exists, and as shown in Section 2.2, researchers have no problem describing good use cases. Legislators have in the begun to adopt XML standards for the formal sources of law they manage. Since these standards have been developed locally there are several different XML legal standards for handling legal resources(Boer, Winkels, & Vitali, 2008, p. 21). *CEN MetaLex*[2] is an open XML interchange standard made with this in mind. It impose a standardized view on legal documents for the purposes of information exchange and interoperability, but it also protects users of existing standards from *vendor lock-in*. MetaLex is on its way to becoming a formal and de facto standard for legislation in XML(Boer et al., 2008, p. 22). *LKIF* is part of a deliverable of the European project for Standardized Transparent Representations in order

[2]http://www.metalex.eu/

to Extend Legal Accessibility (ESTRELLA[3]). The Legal Knowledge Interchange Format (LKIF), is a Semantic Web based language for representing legal knowledge in order to support modeling of legal domains and to facilitate interchange between legal knowledge-based systems. These are both accessible as OWL and RDF, and interfaces with each other(Boer et al., 2008, p. 32). According to Palmirani et al. (2011, p. 168) *MetaLex* only covers the first, second and partially the third layer illustrated in Figure 1. That means that there is a need for build ontologies on top of these representations that model the much more complex levels four and five. *LKIF* serves two purposes, both as a interchange format comparable to MetaLex, but it also serves as a reusable and extensible core ontology, application programmer interface, and inference engine specification for legal decision support systems, knowledge management systems and argumentation support systemsBoer et al. (2008, p. 31). As a knowledge representation language it combines existing Semantic Web(See Section 4.3) technology with a new LKIF Rules language that expands the semantics of RDF and OWL.

2.3.1 Creative Commons

Creative Commons is a standard for machine-readable expression of copyright licensing. They provide a simple and standardized way to give the public permission to use or share your work(Commons, 2014). Their mission statement is

> Creative Commons develops, supports, and stewards legal and technical infrastructure that maximizes digital creativity, sharing, and innovation.

Their focus is on creative works on the Internet, and giving a user the chance of defining in what way he find it acceptable that others use his works. This is different from EULAs, because we know that the companies use EULAs to limit rights, and not to open up for use. What Creative Commons does that is applicable in our context is the use of a *Three layer license*. The license consists of:

Legal Code Which employs regular legal language for use in the legal system.

[3]http://www.estrellaproject.org/

Human Readable Textual representation motivated by the fact the most people is not lawyers and need an explanation.

Machine Readable Which means semantic annotation directly in web sites or included in the works for search engines and users to see.

This layered license is an interesting way to view licenses. We know that the use of legalese is motivated in a clear and accurate language that can be used with as little ambiguity as possible. This is important in legal context to secure equality before the law. That said it can, as most terminology, feel hostile for the user. Some people argue that legal text should be readable for normal people, claiming that the verbosity and complicated syntax of a document has negative impact on legal certainty(Myška, Smejkalová, Šavelka, & Škop, 2012, p.274). To make legal language more clear means to modify it in order to enhance its comprehensibility. While this makes the text more friendly to the user, it can make important information contained in the original disappear. In this simplification that happen between the layers it can occur problems that lead to copyright infringements(Myška et al., 2012, p. 279). A core part of this problem lies in a users incorrect comprehension of the tool. Words like *work*, *attribution* or *non-commercial* are complicated, and the use of these words can be confusing for the user of CC-licenses which might lack legal literacy. This inaccuracy in a words use, and its simplification, is done on the expense of the preciseness of legal language(Myška et al., 2012, p. 283). Even though there still is problems, could accepting such a layering be a pragmatic solution as Creative Commons have provided public with a system that is able to facilitate the demanding process of licensing a work.

2.4 End User License Agreement

An EULA or a Software license agreement is a legal contract between a software or service provider and a user. An EULA can vary in form, but in it you will find rights and restrictions that apply to the software. Typical components are definitions, a grant of license, limitations on use, a copyright notice and a limited warranty. These can be both invasive and restrictive, and there exists examples were the developer or vendor are permitted to search the user's system without prior notification, or that prohibit the user from complaining publicly about the product. Most users passively accept their EULAs because the alternative in many cases is to surrender

the right to use a piece of software that they already have paid for. This kind of license is called a *shrink-wrapped license* or a *click-wrap license*. This imply that the license is bundled inside the software or packaging and is not attainable before purchase and that you sign a contract by clicking yes to continue using the product. The validity and enforceability of EULAs are sometimes criticized, but at least we can argue that the software providers feel these offer them an additional level of security since they keep using them. The fact that they are in use, and possibly is enforceable makes it important that the user gets to know what the EULAs contain upon signing(LINFO, 2006)(PCMagazine, 2014)(HowToGeek, 2014).

2.5 System Development

There has been a lot of research in the field of software development methodologies. What most of these have in common is a focus on giving management a way of controlling and predicting the production, and the team a way to structure their work. A single person team is a programmer who works on his own, as opposed to a programmer working as part of a team.

2.5.1 Waterfall

The waterfall model dates back to the 1970s, and it introduced a way to impose order and control into what had been an informal and chaotic development processes(Weaver, 2004, p. 55). Waterfall style development divides a project into activities that need to be completed in succession(Fowler, 2004, p. 20). This means that a project can start with a 2-month analysis phase, followed by a 4-month design phase, followed by a 3-month coding phase, followed by a 3-months testing phase. The name and content of each step can vary, but the process doesn't change. Each step is subject to inspection, and only when the activity is done is it signed off as being complete. This way will each step in the development start from a previously decided upon baseline. No stages are repeated unless there are big or radical changes to the projects scope(Weaver, 2004, p.55).

2.5.2 Iterative development

Fowler (2004, p. 19) describes the debate between waterfall and iterative development as one of the biggest in development process. When doing iterative style development you divide your project by subsets of functionality. Each iteration complete its set of functionality by doing all the steps of a software life cycle; the analysis, design, coding and testing. Iterative development can have different names and can be categorized by various distinctions, but such distinctions are specializations of an iterative development system, and is not as well-defined as the iterative/waterfall distinction.

2.5.3 Agile

Agile or *lightweight* methodologies turn away from the old heavyweight processes. A lightweight process is low in ceremony, where heavyweight process got lots of documents and control points during the project. It is an umbrella term that covers many processes that share a common set of values and principles described in the *Manifesto for Agile Software Development*(Fowler, 2004, p. 24). The manifesto say that the signatories value: *Individuals and interactions* over processes and tools. *Working software* over comprehensive documentation. *Customer collaboration* over contract negotiation. *Responding to change* over following a plan.

They also published *Twelve Principles of Agile Software* that the signatories say to follow(Beck et al., n.d.). Examples of these processes is Extreme programming(XP), Scrum, Feature Driven Development(FDD), Crystal, and DSDM(Dynamic Systems Development Method)(Fowler, 2004, p. 24).

Chapter 3

Research Method

3.1 Design Science Research

In this project I have used the Design and Creation research strategy. My focus has been to create an *instantiation artifact*. An instantiation is a working system that demonstrates that an idea can be implemented in a computer-based system(Oates, 2006, p.109).

Hevner, March, Park, and Ram (2004, p.83) defines seven guidelines for Design Science. These guidelines have been described as being integral to ensure top quality design science(Hevner & Chatterjee, 2010, p.19), and the purpose of establishing these guidelines where to assist researchers, reviewers, editors, and readers to understand the requirements for effective design-science(Hevner et al., 2004, p.82). These guidelines are not meant to be mandatory, but Hevner et al. (2004) claim they should be addressed for design-science research to be complete.

3.2 Development Methodology

In this project I have followed an iterative development process. I have not followed a specific methodology 100% but I have based my methodology on the *Incremental Model* described in Weaver (2004, p. 59). This means that the system is developed in a succession of self-contained iterations, but built on a common set of requirements.

Guideline	**Description**
Design as an Artifact	Design-science research must produce a viable artifact in the form of a construct, a model, a method, or an instantiation.
Problem Relevance	The objective of design-science research is to develop technology-based solutions to important and relevant business problems.
Design Evaluation	The utility, quality, and efficacy of a design artifact must be rigorously demonstrated via well-executed evaluation methods.
Research Contributions	Effective design-science research must provide clear and verifiable contributions in the areas of the design artifact, design foundations, and/or design methodologies.
Research Rigor	Design-science research relies upon the application of rigorous methods in both the construction and evaluation of the design artifact.
Design as a Search Process	The search for an effective artifact requires utilizing available means to reach desired ends while satisfying laws in the problem environment.
Communication of Research	Design-science research must be presented effectively both to technology-oriented as well as management-oriented audiences.

Table 1: Guidelines for Design Science(Hevner, March, Park, & Ram, 2004, p. 83)

I have continuously written and updated lists of remaining tasks. I will refrain from calling this a scrum or kanban board, since I have had the responsibility, as the sole developer, for each task all the way through its life cycle. I have prioritized tasks, and updated them as the project progressed. This has facilitated effective development and have helped me organize remaining tasks in a succinct way.

3.2.1 UML

Larman (2005, p. 30) writes that:

> The purpose of modeling(sketching UML, ...) is primarily to *understand*, not to document.

He then writes that the act of modeling or *doing UML* is not to write a specification that could be handed over to a programmer, but to quickly explore alternatives. This is in accordance with Fowler (2004, p. 2) that say that there are three different ways to use UML. Either as a sketch, a blueprint or as a programming language. When used as a sketch you can use it to communicate some aspect of a system, either to explain some existing code or to hash out some alternatives before you write some more code.

I have used a whiteboard and diagrams in the sketch mode. This way I have drawn a lot of small diagrams continuously through the project that have facilitated my development. The effect of this is that I early could settle on a representation of my domain, and most refactoring have been for readability, and not because of bad structural decisions. My IDE(See Section 4.1) can export my finished Class-diagrams, so my whiteboard has served as a way of exploring options, and then my IDE later exports the finished diagrams for use when needed. This way have both uses of a diagram been covered.

3.3 Evaluation

To evaluate the performance measure of an information extraction or a OBIE system is not a clear cut or easy task. Different metrics are often used for evaluating a system's success against an already existing gold standard. As discussed in Section 2.2.4 there is no clear ontology that is in broad use or ready for adoption. Such metrics will only find a score based on correctly identified instances and property values. This means that there is not possible to build a gold standard for use in calculation of success without choosing a specific ontology to use for annotation. My evaluation of the system, and hence my degree of success is therefore measured against the data I hope to extract by using a specific extraction method and the data extracted this way. This is in itself a kind of gold standard analysis, but not formalized in a system for automatic analysis of data. I will count correct and incorrect identifications, and evaluate the success in terms of expected and unexpected outcomes. An analysis using some sort of metric on a gold standard is necessary at some point for a system to be deemed successful, but that is not in the scope of this thesis. In Section 3.3.1 I describe the different metrics used in OBIE systems that would be relevant for such an analysis.

I have chosen to use *Jetbrains License Agreement for Education*(Appendix A) for my evaluation. This is an EULA with few pages and language that seems clear and concise. This is a real license document, so the data extracted is not artificial. What is untypical, or at least not the rule is the document being clear and concise and consisting of few pages. This way the resulting data will be real results from a real setting, but I got no way of testing if the test document is a good match to the reality or an average license agreement. My gain from this decision is twofold. A smaller file is less expensive for my artifact to handle, both when it comes to run time and memory. More important is the fact that it doesn't contain too much confusing language or too much structural complexity that risk to obscure any results. For the system to be valuable it should also handle these kinds of license agreements, but I have chosen to see the overly complicated EULAs as a special case, and leave such licenses to further research. As I claim no expert knowledge of the license domain, and the extracted knowledge will only be based off of analysis of the text, the value of understanding a license is higher than researching and finding the perfect middle ground. This means that I use *Jetbrains License Agreement for Education* as an illustration of feasibility and scope for my project without claiming a wider correlation to other documents.

3.3.1 Metrics

By using metrics to compare the results from different EULAs against an gold standard we can find out how good the system performs, not just on a specific studied file, but by automatically count all documents in a corpus of EULAs. This is not applicable for this project, but I have chosen to highlight these methods because of their importance in this field.

Precision and *Recall* are known metrics from information retrieval. Precision is the proportion of correctly returned items, and recall is the proportion of correct items returned. Most IE systems face a trade-off between improving precision and recall(Wimalasuriya & Dou, 2010, p.318). Recall can be increased by increasing the amount of returned items, but that might hurt precision because more might not be correctly returned. The same the opposite way, you can increase precision by only returning things that are obviously correct, but then only a few relevant would be returned.

$$Precision = \frac{|correct\ answers|}{|all\ answers|} \qquad Recall = \frac{|correct\ answers|}{|all\ answers\ in\ the\ gold\ standard|}$$

Figure 2: Precision and Recall equations (Wimalasuriya & Dou, 2010, p.318)

The *F-Measure* is also often used. This equation makes it possible to weight if precision or recall is the most important metric. For example would precision be much more important in a web search than recall, because a searcher rarely care about how many are not returned, but would be annoyed if the returned were not relevant.

$$F\text{-}Measure = \frac{(\beta^2 + 1)P * R}{(\beta^2 R) + P}$$

Figure 3: F-Measure (Maynard, Peters, & Li, 2006, p.2)

One of the problems that occur if you use Precision and Recall is that the score is either correct or it is not. Maynard, Peters, and Li (2006) say that a performance measure for a OBIE system should allow scalar values of correctness.

Cimiano, Ladwig, and Staab (2005) use a performance measure called *Learning Accuracy*, which measure the closeness of the assigned class label to the correct class label from the ontology. They use the distance between the nodes in term of edges as a measure of distance. Maynard et al. (2006, p.3) summarize it as measuring;

> the degree to which the system correctly predicts the concept class which subsumes the target concept to be learned.

It is calculated by finding the shortest path from the root to the key concept. The shortest path from the root to the predicted concept. The shortest length from root to the most specific common abstraction, which is the least common super concept. The shortest length from this super concept to the predicted concept. Then the learning accuracy is calculated as the equation in Figure 4.

According to Maynard et al. (2006) the biggest problem with learning accuracy as a performance measure is that it does not take into account the depth of the key concept in the hierarchy and only considers the distance to the least common super concept.

$$Learning\ Accuracy = \frac{CP}{FP + DP}$$

Figure 4: Learning Accuracy (Maynard, Peters, & Li, 2006, p.3)

Maynard et al. (2006) propose another metric called *Augmented Precision* and *Augmented Recall*, where they combine precision and recall scores with a cost-based component. This is a extension of the Learning accuracy measure, and is based on using a *Balanced Distance Metric*. This way we also consider weighted semantic distance in addition to the binary notation of correctness.

Chapter 4

Development

The development of my application have been done in an iterative fashion with several specific increments and goals. I have been able to examine several different techniques and tools in the domain of licenses. This chapter will describe the tools used and the development of my artifact.

4.1 Development Tools

In the development of the artifact I have used Java, Eclipse, git and Maven. These are common tools, that are used in system development. The Java programming language is a general-purpose, concurrent, class-based, object-oriented language(Gosling, Joy, Steele, Bracha, & Buckley, 2015, p. 1) and is, according to the TIOBE index, the second most used programming language in the world(TIOBE, 2015). The combination of its prevalence and the existence of NLP packages and tools make this a good choice for my project. When developing in Java it is usual to use an IDE(Integrated Development Environment). Eclipse is a commonly used IDE, and it has good integration with both Git and Maven. There are many alternatives available, but I chose eclipse because of familiarity. Git is a version control system. The job of a version control system is to record and save changes to a document during development so that it can be access later if the need should arise. This way I have not had to worry about making changes that might break the system during the development. This facilitated the project's exploratory nature. It is a *Distributed Version control* system which means that all developers keep all

changes and all history locally. This is in contrast to a centralized system where you only check out the latest version of the software. As this is a solo project haven't the effects on collaboration been in focus, but it have given me a way to concisely back up my project to an external server. Maven is a software project management and comprehension tool. One of the big gains from using Maven is that it eases collaboration through standardization of project structure and it encourages a best practice for the project. As this is a solo project is that not an important part for me. That said it will still be important if there is to be any further development of the system. My main reason for using Maven is that it makes downloading and organizing packages easier. Most big projects host a maven version of their project, and this makes it easy to maintain, download and use. Protégé is a free and open-source ontology editor and framework for building intelligent systems. Protégé is made for development of ontologies and makes imposing restrictions on properties and classes an easier task. My use of Protégé has mainly been as a tool to explore existing ontologies, and for exploration when developing my own vocabulary specific for this project.

4.2 Dataset

In the research article *Learning to detect spyware using end user license agreements* by Lavesson, Boldt, Davidsson, and Jacobsson (2010) they theorize that vendors need to inform users about inclusion of spyware to avoid legal repercussions. This is done through the systems EULA during the installation of a program, and that hence the EULA will contain long and unwieldy sentences to camouflage that fact. They compare 17 different text classification algorithms that try to classify EULAs as Good or Bad. For that purpose they compiled a dataset of 996 EULAs divided into a Good class or a Bad class, which they used as a training set. They obtained the good instances from Download.com and the bad instances from links found on SpywareGuide.com. This dataset is made public(Lavesson, 2014), and I have use that dataset in this project. I have not been interested in the licenses characterized as bad, since these add a complexity that should be discussed on its own.

4.3 Semantic Web

Semantic web, or Web 3.0, is an extension to other web technologies that adds machine readable knowledge to the web. The basic building block in semantic web is the concept of a graph. In such a graph we use URIs to identify specific things, and the vertices that connect these. Such a thing can be an author, but as we know that names very seldom are unique to one specific person, a name is not a good identifier. URIs are unique in that you don't risk getting the wrong web page as long as you enter the correct web address. This makes them suited for use as an identifier, and the job of keeping track of unique addresses is already built into the structure of the internet. This way we can all agree who is who, without risking talking about different things. Knowledge encoded this way is what makes up a graph. We call anything represented with a URI for a *resource*. The vertices connecting things together we call *properties*, and they are in themselves resources and the data fields we call *literals*. A *triple* is a statement that consists of an *Subject*, *Predicate* and an *Object*. One such triple is two nodes, the subject and the object, and the vertex connecting these, the predicate. There are few rules except that a literal can not be the subject of a triple. A graph can be expressed in several different ways expressing the same content. These are called different *serializations* and translation between these will not change the knowledge, only the syntax used to express it. RDF, RDFS and OWL are the basic representation languages of the Semantic Web(Allemang & Hendler, 2011, p. 27). *RDF*(Resource Description Framework) is a W3C standard based on XML that can describe such a graph[1]. It is also a small vocabulary that together with *RDFS*(RDF Schema) and *OWL*(Web Ontology Language) has the expressive power of defining classes, things' membership in these classes and how these things interact. Such an abstraction is called a *model*(Allemang & Hendler, 2011, p. 13) and is used to communicate knowledge, reason over the facts expressed and mediate knowledge among different representations of a concept. In this thesis I use the word *vocabulary* to describe what I call a *lightweight ontology*. This is to distinguish between the level of expressivity modeled into an ontology. The word ontology has a lot of different meanings and is used inconsistently(Hepp, 2007, p. 3). The word came originally from philosophy, and in computer science we define an ontology as: "Ontologies are the *vocabulary* and the formal specification of the vocabulary"(Hepp, 2007, p. 6). In Hepp (2007, p. 6) this is meant as a contrast to a *knowledge base* which is the specific knowledge expressed using an ontology. As both a knowledge base and an ontology is possible to express using

[1]http://www.w3.org/TR/rdf11-concepts/

the basic representation languages I will not make this distinction explicit. The use of vocabulary instead of ontology will therefore imply the level of expressivity, even if the current ontology being discussed contains more expressive knowledge.

4.4 Iterations

This project has gone through several phases, and several iterations. In this section a phase will mean a direction for the project. When the project change direction this leads to a new phase, which in turn can have several iterations. An iteration will here mean an iteration in the sense used in system development, even though a phase also is an iteration of the *generate/Test cycle* described in Section 3.1. This is to distinguish between an iteration in the development and an iteration of the project. I have chosen *JETBRAINS LICENSE AGREEMENT FOR EDUCATION*(Appendix A) as my source document for evaluating the system. In this section I will show the data extracted using the system on this specific document.

In statistics we talk about two different kinds of errors. A *Type 1* error is a incorrect rejection of a true null hypothesis, a false positive, and a *Type 2* error is is the failure to reject a false null hypothesis, a false negative. In this chapter I will use these expressions to discuss the resulting extractions in my system. By Type 1 errors I will mean data extracted that shouldn't been, and by Type 2 I will mean data not extracted that should have been.

4.4.1 Inception

My inception phase consisted of researching the systems and tools described in the theory. This way I hoped to find existing OBIE systems that already implements tools and techniques needed. This search gave little fruition. Little software described is freely available, but many of these systems were built using GATE(*General Architecture for Text Engineering*) which is *Open Source* and freely available. My limited access to alternative systems led to an change in the topic for the project. From a project exploring the differences between existing systems used in the domain of License agreements, it now turned into a project investigating the feasibility of such a system in the license domain.

4.4.2 Phase 1 - GATE

My first phase were used focusing on exploring GATE and its existing functionality. Since GATE both is open source and seem to have a big user base it seemed a good fit for my project. This process started by following tutorials and reading the User guide. Through this work I got some experience both with the GUI-interface bundled with the system, and using the system in my own code. This led me to experiment with other existing modules to see what exists, and try to get a feel for what should be part of my system. Despite initial success, it proved to complex for me to effectively use. All examples worked as expected and as described, but my own attempts to change and adapt the existing pipelines proved unfruitful. After having exploited the available sources of documentation, tutorials and questions online, I chose to abandon GATE as a platform for my project. This proved to be a false start and the time invested gave little returns.

4.4.3 Phase 2 - Build my own system

My experience with GATE led to a new direction for my project. Instead of using an existing general framework, and extend upon this functionality, I chose to implement my own domain specific system. The plan was to add functionality and information extraction methods in an iterative fashion. The result of this was planned be a system that extracts knowledge and output these as triples which then facilitates my analysis of their usefulness in the domain of licenses.

4.4.4 Iteration 1 - Exploration

The first iteration consisted of researching NLP libraries, their existing functionality and their use. I also formalized a list of required functionality needed by a system. My choice boiled down to two different systems, *Apache OpenNLP* and *Stanford CoreNLP*. Both are open source and freely available, and also seemed to have good documentation. This made both these systems into good alternatives and I chose to use some time trying out both, by following guides for getting started and tutorials.

In the end I chose to use the software libraries from *The Natural Language Processing Group* at Stanford University. This is *"a team of faculty, research sci-*

entists, postdocs, programmers and students who work together on algorithms that allow computers to process and understand human languages"(The Stanford NLP Group, 2015). *Stanford CoreNLP* is an application where they have collected a lot of the published applications into a pipeline that do a lot of analysis(Surdeanu et al., 2014). As both systems seemed to overlap in functionality I made that choice with ease of use as the only measure of success. I also chose not to use the pipeline functionality from the Core-NLP package, but instead use the specific applications included in Core-NLP. The requirements for the system were developed as a reaction to my experiences from using both these libraries and my earlier work with GATE. I decided that the different extractions should be low coupled, and that I should tie the results together in the last stage of the project. By investigating each of the different extraction techniques individually I risked to loose some efficiency, since each technique might do the same kind of analysis. Had I focused on building a pipeline where each preceding analysis feeds its results into the next then that eventuality had been minimized, but with a big increase in complexity. This way I didn't need to write a system that translate the system's specific representations of data, but instead handle the extractions through an object.

Requirements
Should extract URLs
Should extract Named Entities
Should keep the document structure
Should extract Definitions
Should use a triple store as a backend
Should use existing ontologies, if available
Existing ontologies should be used to extract knowledge
Should be easy to exchange ontologies
Should follow good practices in object oriented design

Table 2: Initial requirements

Existing ontologies describing document structure was easy to find through *http://lov.okfn.org*[2]. I have chosen to use the *Document Components Ontology(DoCO)* to describe the structure of a document. DoCO is a part of *Semantic Publishing and Referencing Ontologies (SPAR)*[3] which consists of eight ontologies describing all parts of semantic publishing. DoCO describe document components, both structural like block, inline, paragraph, section and chapters and rhetorical con-

[2]http://lov.okfn.org/dataset/lov/vocabs/doco
[3]http://sempublishing.sourceforge.net/

cepts like introduction, discussion, acknowledgement, reference list, figure and appendix(Shotton & Peroni, 2014). This is a lot more than I need for the project, and I only use a few concepts from this vocabulary. I have built the system, with the possibility to exchange vocabularies in a trivial and controlled way. This means that if I discover a vocabulary describing these concepts in a better way, very little code will have to be changed, and only changed in a trivial way.

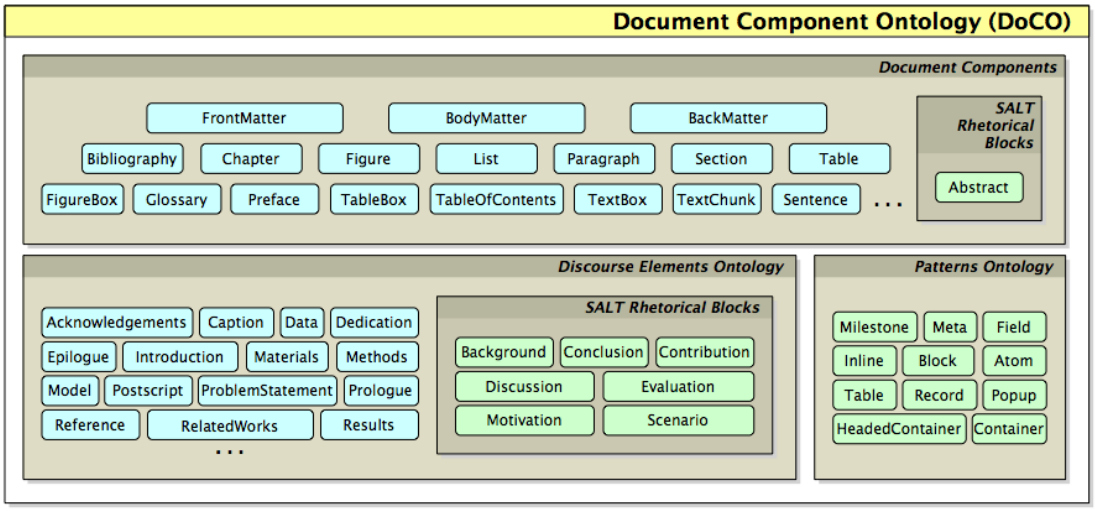

Figure 5: DoCO architecture retrieved from http://www.essepuntato.it/lode/http://purl.org/spar/doco

Finding ontologies describing the domain of licenses were harder. I found that *The Software Ontology* included a module for describing licenses. The Software Ontology is a resource for describing software tools, their types, tasks, versions, provenance and associated data. I also discovered that there are ontologies for describing copyrights. Because of the availability, and also the relation to license agreements, I chose to look closer on two ontologies of this kind. I also chose to check out *LKIF-Core*. As described in Section 2.3, this is an interchange format for legal resources. This means that there are no direct focus on licenses, but I still chose to investigate its use in my system.

Name	**URI**
The Software Ontology	http://theswo.sourceforge.net/
Semantic Copyright	semanticcopyright.org/
Copyright Ontology	http://rhizomik.net/html/ontologies/copyrightonto/
LKIF-Core ontology	http://www.estrellaproject.org/lkif-core/

Table 3: License, copyright and legal ontologies

4.4.5 Iteration 2 - Document Structure

As discussed in section 2.2.5 it is important to keep the structure of the legal text, as the linguistic analysis and extracted data will be connected to the legal document. I had to decide on an object-oriented representation of a license document. I chose to use the structure in DoCO as basis, and decided to represent a Document as a list of Document Parts, where a document part is an abstract utility class. Then I sub-classed the document part into a Title, Section, Section Label and a Section Title. A legal document is read section by section and gets typed using regular expressions expressing what part of the document it is. Each section then were split into sentences using a Sentence-Tokenizer. I chose to keep sentences as the smallest part in my model. This means that specific extracted knowledge like a named entity will use indices to point to a location in a sentence. This means that every triple will point back to the document, and that the ordering and structure of the document is preserved for later use. As most of the analysis methods I planned to use takes a sentence as an argument this makes sense as a pragmatic representation of the document.

My system uses regular expressions to decide between what are the different parts of a document. A new document is first split into paragraphs by newlines. Then each paragraph is added to the document. If the part matches a regular expression(Listing 4.1) describing a *Label* and is short it is added to my model as a section title. If it contains a label, but it is not short, then it is a section with a title. Else it is a section without a label. I have chosen *Section* from the DoCO ontology(described in Section 4.4.4). This concept is defined as *A logical division of the text, usually numbered and/or titled, which may contain subsections.* This fits my use of the concept.

```
Pattern labelPattern = Pattern
        .compile("^\\(\\w+\\)|^\\d+\\.\\d*\\.*\\d*\\.*\\d*\\.*");
```

Listing 4.1: Regular expression matching labels

4.4.6 Iteration 3 - Named Entities

Named entity recognition is the task of labeling words or phrases that are the names of things. Named entities is an important part of an EULA. This extraction should extract the involved parties mentioned in the text. I chose to use *Stanford Named*

Entity Recognizer(NER) in my system(Finkel, Grenager, & Manning, 2005). The Stanford NER package is distributed with three models in English. They are trained on different data, and for extraction of different classes of Named Entities.

4 class: Location, Person, Organization, Misc Trained for CoNLL[4]

7 class: Time, Location, Organization, Person, Money, Percent, Date Trained for MUC

3 class: Location, Person, Organization Trained on both datasets

Named Entity	3 Class	4 Class	7 Class	Interesting
licensor		•		✓
jetbrains s.r.o.	•	•	•	✓
na		•		
prague	•	•	•	✓
czech republic	•	•	•	✓
commercial register		•		✓
municipal court of prague	•	•		✓
licensee		•	•	✓
jetbrains educational program	•	•	•	✓
third party software		•		
jetbrains account		•		✓
united states copyright law	•	•	•	✓
international treaty		•	•	✓
united states	•	•	•	✓
clients		•		✓
evaluation period		•	•	
license		•		
license agreement for education		•		✓
negligence		•		
licensee and licensor		•		✓
either licensor or licensee		•		✓
licensee			•	✓
licensor 's license agreement for education			•	✓
us	•		•	✓

Table 4: NERs extracted from JetBrains License Agreement(Appendix A)

[4]http://www.cnts.ua.ac.be/conll2003/ner/

As seen in Table 4 it varies what is extracted as Named Entities when using the different models. I have marked things I want to have returned as a Named Entity with a checkmark. I make this distinction because a false positive could be as bad as a false negative in this case. Words like *Licensee* and *Licensor* is important, and should be included. They aren't entities in the same way as a *United States*, but these labels are used through the document. A clear error is the words *evaluation period*, *license* and *negligence*. These should be counted as wrongly extracted. I chose not to take any special considerations for words like *Licensee* and *Licensor*, but these might warrant special treatment in a later stage.

4.4.7 Iteration 4 - Binary Relation Extraction

Binary relationships lend themselves to semantic technologies by extracting triples, consisting of two arguments and a relation connecting these. This is similar to a triple used in semantic technologies and by extracting these triples then try to couple these relations to concepts in a ontology is an interesting task.

This iteration started by me researching the *Stanford Dependency Parser*(DepParser). The DepParser was designed to provide a simple description of the grammatical relationships in a sentence. The goal was that it should be easily understood and effectively used by people without linguistic expertise who want to extract textual relationships(Marneffe & Manning, 2013, p. 1). This representation of relationships are extracted from *Phrase Structure Parses* using rules, and the result is triples of relations between pairs of words(Marneffe, Maccartney, & Manning, 1999, p. 1). This proved to be a dead end since I could not locate a grammar to describe the kind of extractions I wanted.

I discovered that there are two different approaches to Binary Relation ExtractionBRE. The first approach uses part-of-speech tagging and chunking(dividing a sentence into its noun, verb, and prepositional phrases). The other approach also uses dependency parsing(Corro & Gemulla, 2013, p. 1) in its extraction. *ClausIE* is a system using the second approach, and I moved my efforts to trying to use that package. When run from the command line it gave me results, but I had problems using it inside my own code. Because of the time constraints in this project I decided that I could not waste any more time on this package. Corro and Gemulla (2013, p.1) mentions alternative systems, both using Dependency Parsing and not.

Out of these I then chose to use *ReVerb*. ReVerb is a system using the first approach of BRE. This approach is faster than using Dependency Parsing, and it has good precision, but with lacking recallCorro and Gemulla (2013, p. 1). As my vision of this kind of system is based on modular use of packages, I chose to use ReVerb without a critical discussion of pros and cons of the two approaches.

I remove the URLs from the text before running the ReVerb analysis on my data. The reason is that the package had some trouble with sentences containing lots of special characters. I therefore chose to remove them prior to analysis, and instead include an numbered placeholder. This makes it possible to analyze sentences containing URLs, and it also keeps the form of the sentence complete. Removing the URL altogether would lead to me trying to analyze malformed sentences, and that is not a better solution because of ReVerbs dependence on POS-tags.

Arg1	**Relation**	**Arg2**	**Confidence**
Software	is protected by	United States Copyright Law and International Treaty provisions	0.9755597201513483
Software	is the property of	Licensor	0.9018802574024445

Table 5: Example binary relations extracted with ReVerb from Jetbrains License Agreement

An argument in a binary relation might be a Named Entity. In Table 5 this is the case. In row 2 we can see the that *Arg2* is *Licensor*. This relation is easy for us to read and understand, and we intuitively understand that *Licensor* is the same Named Entity as extracted in Section 4.4.6, and the definition extracted in Section 4.4.10. All these extractions then should be represented as the same *Resource* in the resulting dataset.

4.4.8 Iteration 5 - Topic extraction

As I started this iteration, I thought that among the binary relations extracted from a corpus of text there would be some repeating relations. I wanted to cluster these relations extracted from my corpus of licenses and use these clusters as a classifier. I started this iteration by running ReVerb and extracting binary relations from my corpus of licenses. Even though ReVerb use *shallow syntactic analysis*, it still took 36 hours. I used Apache Derby to persist the extracted relations. Derby is an open source relational database. It is fully implemented in Java, which made it easy to

set up and use[5]. As this should not be a part of the resulting system, but instead work as a temporary storage for experimenting with the relations, this were the only requirements for my choice of database.

The result were a SQL-database consisting of 41314 rows. Each extracted relation were saved with a given ID-number, the name of the document it were extracted from, and the relation and arguments in each own column(Listing 4.2).

```
CREATE TABLE reverbRels
(
REL_ID INTEGER NOT NULL GENERATED ALWAYS AS IDENTITY (START WITH
   1, INCREMENT BY 1),
DOC_NAME VARCHAR(24) NOT NULL,
REL VARCHAR(50) NOT NULL,
ARG1 VARCHAR(20) NOT NULL,
ARG2 VARCHAR(20) NOT NULL);
```

Listing 4.2: SQL Create statement used in Iteration 5 - persists binary relations

I wanted to use these relations to build a classifier. My problem were that they were unlabeled. To label this data would be a lengthy process. I could have manually picked relations, and try to cherry pick the ones were I see a clear pattern. This tactic is flawed in that I would have to construct a dictionary for all important terms and their corresponding labels. That way I could have used a standard vector-space model to classify using the corresponding labels, but I would have lost the chance to find patterns that were not clear to me as a layman. This was not an interesting pursuit for my project, even though I adopted such a technique later for definition extraction(See Iteration 4.4.10).

I instead explored using Topic Models as basis for my classifier. Topic models provide a way to analyze large volumes of unlabeled text. A *topic* in this context is a cluster of words that frequently appear together. This is done through algorithms for discovering the themes of a document and annotating collections of documents with these themes(Blei, 2012, p. 77). This is statistical methods that discovers themes that run through the text. This requires no prior labels since the topics emerges from the original texts. There are different topic models, but I chose to use the Java library *MALLET - MAchine Learning for LanguagE Toolkit* as they implement *Latent Dirichlet Allocation*. The basic intuition that is a basis of *LDA* is that a document consists of multiple topics. It is a statistical model of document

[5]http://db.apache.org/derby/

collections that tries to capture this intuition(Blei, 2012, p. 78). The algorithms got no information about subjects and are not labeled with topics or keywords, and the topics discovered can aid tasks like information retrieval, classification and corpus exploration(Blei, 2012, p. 79).

MALLET is a Java-based package for statistical natural language processing, document classification, clustering, topic modeling, information extraction, and other machine learning applications for text. I have used its Topic modeling functionality. This package also contains a set of command-line tools that I used to explore the basic functionality available. By training the algorithm on my corpus, and only extracting one topic i get the words in Table 6. The command-line tool limits the amount of printed words, but there is no limitation when used in my own code.

Words in Topic
software
license
agreement
product
terms
rights
including
damages
agree
warranty
user
provided
copy
limited
copyright
services
computer
conditions
information

Table 6: Test Topic Modeling. One topic extracted from corpus.

My first task were to use Topic Modeling to extract topics from my binary relation dataset. My intuition were that by extracting topics from the relations, I could use a standard Vector Space model to compare the topics to new binary relations extracted at a later point. If I could find a topic that resulted in a specific kind of relation, I could have assumed a connection between that topic and that

concept in my ontologies. Then I could use such a topic as a label in my classifier. I built my classifier using *Apache Lucene*[6] to compare documents. Lucene is an open source search engine built in Java that gives me the possibility to index documents and query these documents directly from my system. I extracted topics, and then created one document per topic. These topics now were indexed using Lucene. A query then returns the document representing a topic that is the most alike according to Lucenes vector space compare mechanism. I also experimented with removing the named entities from the relations before running the Topic Modeling algorithm. In a sentence would naturally a named entity be important, and I were afraid that named entities from the corpus would end up in a topic. This could make my algorithm classify all relations from a specific company into the same topic. I ran the algorithm on my relation dataset after removing named entities, but this led to topics being represented by just a few words. This were not ideal, and I chose to proceed with the named entities still present in the relations.

I also needed a way to check if the results from using the classifier on my corpus would result in uniform classifications. I chose to use *Levenshtein distance* as my measure which is the minimal number of insertions, deletions and substitutions to make two strings equal. This distance is implemented in *StringUtils*. This implementation gave poor results because two strings containing the same words but in different order will demand a lot of changes for the strings to match each other. I rewrote the distance calculator such that it first picks the part of the sentence with the least distance, and then use Levenshtein distance to find how those token compare. By adding the distances, I get a number representing the difference between the present tokens instead of between the symbols placement in the complete string representing the sentence. This way I can compare the sentences *cat dog* and *dog cat* and get the distance 0 instead of the distance 6 which is what I want. Low numbers would mean that the classified binary relations got grouped together with other relations containing a lot of the same words. I then normalized the values, and I made 1 to be perfect classification and 0 to mean no pattern. I will call this measurement for *Adapted Levenshtein*.

[6]http://lucene.apache.org/

Data	#	Topics	ReVerb Conf	Score > 0.70	Average
Rels	1	200	>0.0	1	0.5825438471367728
	2	200	>0.75	11	0.5933647584814938
	3	80	>0.75	0	0.5526640838705178
	4	300	>0.00	6	0.5910004632652717
	5	300	>0.50	17	0.6003900271145889
Corpus	6	200	na	0	0.5555795578048943
	7	80	na	0	0.51998184623614
	8	300	na	0	0.5727156732657103
Cutoff	9	300	0.5	101	0.6989226298408402
	10	200	0.75	62	0.7136612126691609

Table 7: Classification using LDA from the Mallet package.

In Table 7 you can see that the quality of the relations extracted affect the quality of the classification. I first trained my Topic Model using the relations extracted from my EULA corpus. ReVerb assigns a confidence value to their extraction, and I filtered these values. I also tried different amounts of topics, calculated average adapted levenshtein score and I counted how many topics got an adapted levenshtein score higher than 0.70. A topic with just one matching relation is a special case that gets a Levenshtein score of 1, and these are not counted. I also run some analysis using the whole corpus of licenses. After that I tried the best results from earlier, with an cutoff level of 0.500 on the classifier, where I threw away all classifications where the Lucene score were lower than the cutoff. In the *Rels* classifications I used the same data both as training set for Mallet and as the relations being classified. This might introduce some errors since data in the wild might not be properly represented. This means that the results in Table 7 might be skewed. Since my idea is based on sentences being similar having the same meaning, a classification using the same training set still makes sense and a bad classification on that set imply a bad classifier.

I chose to count the amount of topics that classify with an adapted levenshtein higher than 0.70 since the average score is misleading. A good total average score is skewed if some topics is good, and some is bad. It also says very little of the quality of using these settings in an extraction task. By counting the amount of good classifications I see how many topics could be used in such a way. If it is possible to find a property that describe the relation expressed in a topic, we could assume that a binary relation from a new document being classified to the same topic, is expressing the same data.

On inspection it turned out that these concepts had a varying degree of success. In Table 8 we see that several sentences that limits the amount of usage to one is classified as alike. In Table 9 we can see that sentences containing a lot of the same words is classified into the same topic even though they have contrary meaning. This is because the word *not* has no special meaning. We know there is a difference between *being freeware* and *not being freeware*, but that is not captured in this classification. Despite this we can at least claim that this is important knowledge that should be captured. That means that a property describing something about the status of the software in context of being freeware might be possible to find.

Sentences classified as alike
only one copy is used at a time
only one copy is used at a time
NOT serves ONLY
only one copy is used at a time
THIS IS THE ONLY WARRANT
NOT serves ONLY
THIS IS THE ONLY WARRANT
this Software is sold only with Restricted Rights
only one copy is used at a time
only one copy is used at a time
only one (1) copy is used at a
only one copy is used at a time

Table 8: Sentences classified as belonging to same topic.

Sentences classified as alike
This version is Freeware
The Software Product is a freeware
) are provided as freeware
This software is not freeware
This Software is not freeware
FlashTask is a freeware
GSN is FREEWARE
This copy of the SOFTWARE is a freeware

Table 9: Sentences classified as same topic with differing meaning.

4.4.9 Iteration 6 - Ontology comments and labels

As I now had a classifier that can classify different binary relations, it now followed that these classes should be connected to a specific resource in an ontology. I realized that the ontologies I had chosen to explore contained both comments and labels explaining their different properties and concepts. These comments and labels could contain similar language to what the extracted binary relations does. This way I should be able to compare source sentences with a comment from the ontology, using a standard vector model. I first explored extracting the comments and adding these to their own documents, and this way use Lucene in the same way as earlier. The continuation of this were the indexation of RDF files, and that led me back to the framework I planned to use in a later iteration.

Apache Jena[7] is a free and open source Java framework for building Semantic Web and Linked Data applications. It has a built in triple store, reasoner, *sparql* capabilities, and methods for building models in your source code. It is also distributed with a set of vocabularies and ontologies that can be used by the reasoner. What I discovered was that it also has built-in support for text search through sparql with Lucene as a back end used for indexing. This was exactly what I needed, and I proceeded by using this instead of my initial plan of parsing the ontology comments out into other documents and then index these.

I managed to get it up and running quite quickly, but the fact that I wanted to index both comment and labels complicated the process. This module has little documentation and I had to dig through the source code to find out how to do this. I also had to write a system to load and index all my chosen ontologies into the same index. This should make sure a single query return the most fitting resource available in the licenses, and I could get a quick overview over what failes to help me change my tactic.

```
String pre = StrUtils.strjoinNL("PREFIX : <http://example/>",
    "PREFIX text: <http://jena.apache.org/text#>",
    "PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>",
    "PREFIX schema: <http://schema.org/>",
    "PREFIX lkif:
       <http://www.estrellaproject.org/lkif-core/legal-action.owl#>",
    "PREFIX copyOnt:
       <http://rhizomik.net/ontologies/copyrightonto.owl#>",
```

[7] https://jena.apache.org/

```
    "PREFIX swo: <http://www.ebi.ac.uk/swo/>",
    "PREFIX swoLicense: <http://www.ebi.ac.uk/swo/license/>",
    "PREFIX lkifRole:
       <http://www.estrellaproject.org/lkif-core/legal-role.owl#>",
    "PREFIX semCopy:
       <http://www.semanticweb.org/ontologies/2009/10/11/>");
String qs = StrUtils.strjoinNL("SELECT * ",
    " { ?s text:query ('"+ queryString +"' 5) ;"
    + " OPTIONAL { ?s rdfs:label ?match } "
    + " OPTIONAL { ?s rdfs:comment ?match } }");

Query q = QueryFactory.create(pre + "\n" + qs);
QueryExecution qexec = QueryExecutionFactory.create(q, ds);
QueryExecUtils.executeQuery(q, qexec);
```

Listing 4.3: Text search through sparql. Checks both comments and labels if these are indexed.

I chose a test query to make sure I got expected results by choosing a random concept from one ontology. The results were promising(See Table 10), since it returned a mix of concepts from several of my ontologies, and all seemed relevant.

#	s	match
1	copyOnt:LegalPerson	"Legal Person"@en
2	lkif:Private_Legal_Person	"Legal Person (Private Law)"
3	lkifRole:Professional_Legal_Role	"A professional legal role is a legal profession of some person, examples: lawyer, judge etc."
4	schema:Person	"Person"@en

Table 10: Result from test query "legal person". Typing removed.

When the sparql text search(STS) were up and running I tried three different techniques for choosing appropriate concepts from the ontologies. First I tried to match the sentences directely against the ontologies. That means that I extracted the binary relations from the Jetbrains EULA, before classifying these extracted relations. I chose to remove relations with low confidence given by ReVerb, as I did in Section 4.4.8. I then tried to find a way to obtain the scoring used for ordering the results. This is not implemented in the framework, so I was not able to set a cutoff value. Setting such a cutoff value would remove any matches where there were insufficient lucene score between the ontology and the text and might make it possible to remove some results. Then I tried to use the classifier. I chose that only

topics with score higher than 0.70, as shown in Table 7, should be collected. Then I used those binary relations to query the ontologies resulting in the closest property being returned. This way the classifier is used as a filter to choose properties available. Lastly I chose to use the words in the topic that the sentence is classified as in the query-text instead of the binary relations extracted from the data set.

I count a classification as successful if the extraction returns a relevant property among the returned properties. In Table 11 is the results from running the classifier with the different settings.

	Technique	**Correct**	**Errors**
No cutoff	Just binary relation	5	18
	Associated Topic	0	21
	Binary relations only good topics	16	5
cutoff	Binary relations only good topics	0	3
	Associated Topic	0	3

Table 11: Results using SPARQL-text search.

4.4.10 Iteration 7 - Definition extraction

Definitions are important in a legal text(See section 2.2.2). A definition redefines a word, and this modifies or clarifies the meaning of the term throughout the legal document. Definition extraction is a hard topic, with many nuances. My implementation use definition connector phrases to find sentences that are potential definitions. This is not a foolproof tactic, but this way I have been able to extract definiens and definiendum for some kind of definitions.

I used the list of connector phrases in table 12.

The list in Table 12 is then used in combination with a regular expression to find occurrences in the text. This is a rudimentary technique, and even though it does have some success in this system, I still think a fully implemented definition extraction system would be better. The first step of getting better performance in such a system should be to check *part of speech* tags associated with the word to remove all cases where a word has the wrong POS.

Sentences containing definitions as extracted by my system is rendered in full in Appendix C. All definitions extracted by the word *mean*(also matches the word

Connector phrases
use to / for
consist of
mean
call
contain
describe / define as
stands for
is directed to
concerns
mean
comprise
take care of
mean by
speak of
define
used to / for
call
formed by
make possible
is possible
there are
function as
provides

Table 12: Verb Connector phrases(Westerhout, 2010, p. 243)

means) is correctly extracted. Some sentences are also matched by the connector phrases and extracted even though they aren't definitions. This is because of my rudimentary implementation of a definition extraction system as explained.

Definiendum	Definiens
Licensor	JetBrains s.r.o., having its principal place of business at Na hrebenech II 1718/10, Prague, 14700, Czech Republic, registered with Commercial Register kept by the Municipal Court of Prague, Section C, file 86211, ID.Nr.: 265 02 275.
"Software"	any software program included in JetBrains Educational Program at https://www.jetbrains.com/student and any third party software programs that are owned and licensed pursuant to Section 5 of this Agreement by parties other than Licensor and that are either integrated with or made part of Software (collectively, "Third Party Software").

Table 13: Definitions correctly extracted by my system.

Among the binary relations extracted in Section 4.4.7 we see that the definitions is also extracted. In Table 14 we see that the *Relation* part of the extraction match one of our connector phrases from Table 12, but the *Arg2* is not a complete sentence and only part of the definition. This illustrates the need for a specific definition extraction module, even though some connection might be desirable.

Arg1	Relation	Arg2	Confidence
Software	means	any software program	0.03230696269027211

Table 14: Definitions in binary relations.

4.4.11 Iteration 8 - Lifting data

In the last iteration I focused on lifting the data available from the other iterations. That consisting of developing a small vocabulary that describe the extraction tasks I have done, the lifting of these data and storing them in my triple store. As I have strived for modularity, I have placed all Triple store(TDB) and Jena operations in its own utility class. This ensures low coupling between my domain representation and the final construction of the triples. Low coupling will make it possible to change my triple store, and change my choice of framework without changing anything in the domain representation or the extraction tasks. In Table 15 you can see the short forms with their correlating URI. My vocabulary is lightweight and is not supposed to add any legal knowledge. As I use the DoCO vocabulary to describe the document structure, I only had to chose a few concepts to describe the data

that gets extracted by the system.

qname	URI
rdfs:	http://www.w3.org/2000/01/rdf-schema#
dc:	http://purl.org/dc/terms
ex:	http://eulaThesisExample.org/
doco:	http://purl.org/spar/doco/

Table 15: Short form of URIs used in the text.

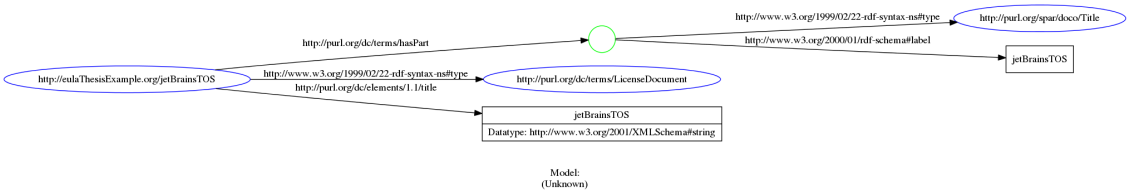

Figure 6: Triple graph: Initial information in triple store.

I start the process by extracting some initial data. As you can see in Figure 6, I have chosen to represent a specific EULA as a resource. This resource has a type dc:LicenseDocument. This resource also has a dc:title which in this case is derived from my local filename. This name is not extracted and should be either supplied by hand or composed by a product name and number. The title of the document is also represented as an anonymous node having type doco:Title, and a label. The use of dc:hasPart which points to an anonymous node is synonymous to the way I represent all parts of the document.

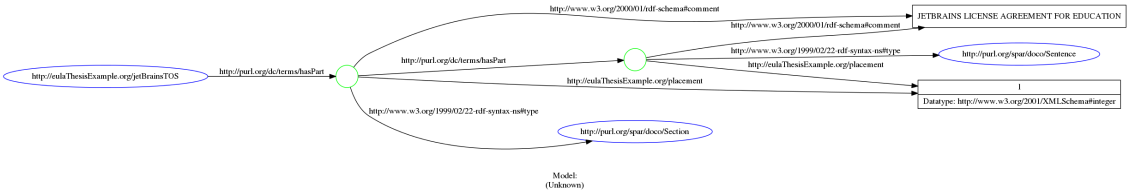

Figure 7: Triple graph: Organization of the Sections

Figure 7 show how I organize a document. The EULA resource is built up of doco:Sections. A section got a rdfs:comment which is the whole text of the section. It also has a ex:placement which is an index that orders that section in relation to the other sections. A section also dc:hasPart an anonymous node of type doco:Sentence. These sentences also got a ex:placement representing its ordering, an rdfs:comment representing the specific string. In Figure 7 we see a section of only one sentence, and hence both the section and the sentence has the same rdfs:comment. Figure 13 in the Appendix show a section with several sentences.

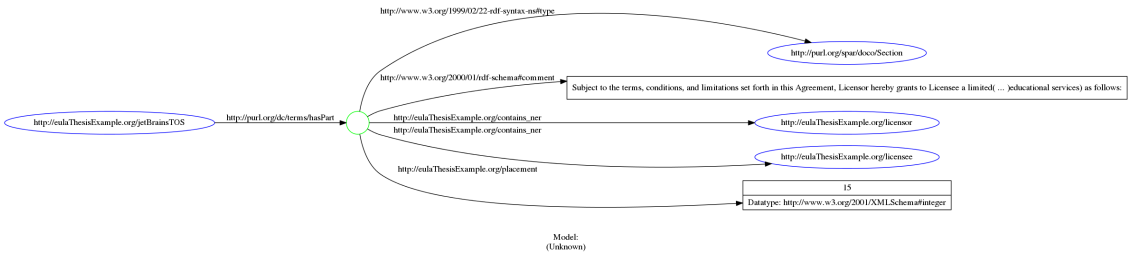

Figure 8: Triple graph: Sections contains NERs

A ex:Section can contain named entities. A named entity is represented as a resource, and the property ex:contains_ner connects it to the section as shown in Figure 8.

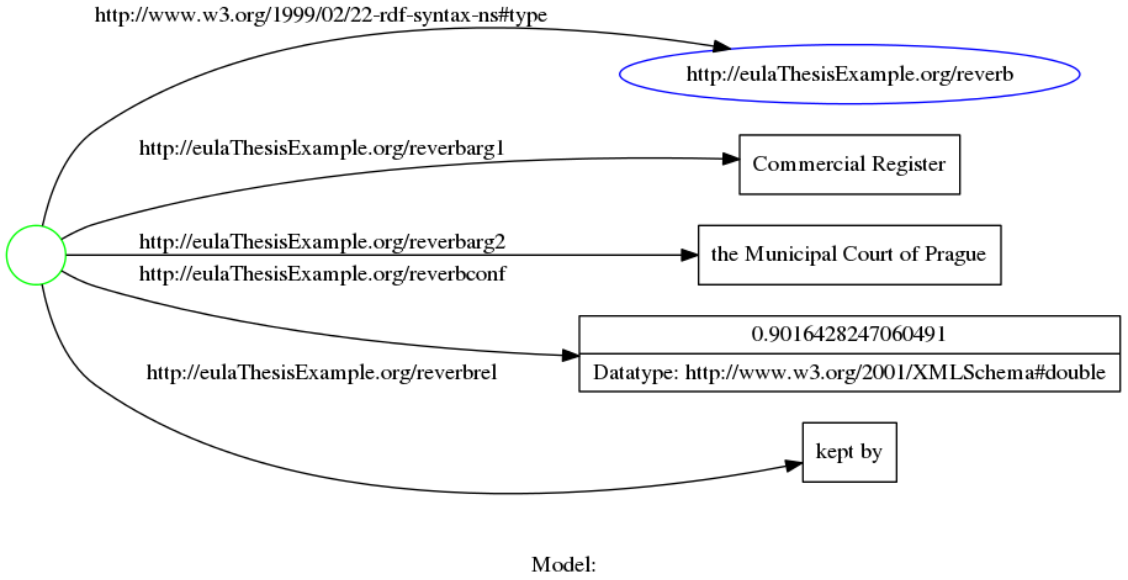

Figure 9: Triple graph: Binary relation represented as triples.

A binary relation is represented as an anonymous node with type ex:reverb. It has four properties that represent the data extracted ex:reverbarg1, ex:reverbrel, ex:reverbarg2 and ex:reverbconf. A section can have a property ex:contains_reverbs, which points to a list of binary relations. This is illustrated in Figure 14 in the Appendix. I chose to use package specific names because the data should be traceable back to how it was extracted. Even though this is not as evident with the other types of extraction should they also be read as extraction using the specific technique. An alternative to this is discussed in Section 5.0.12.

A definition is connected to an EULA resource with an ex:contain_definition property(See Figure 10). A definition is also a regular sentence and is also represented as a part of a section(See Figure 12). The use of a definition is illustrated

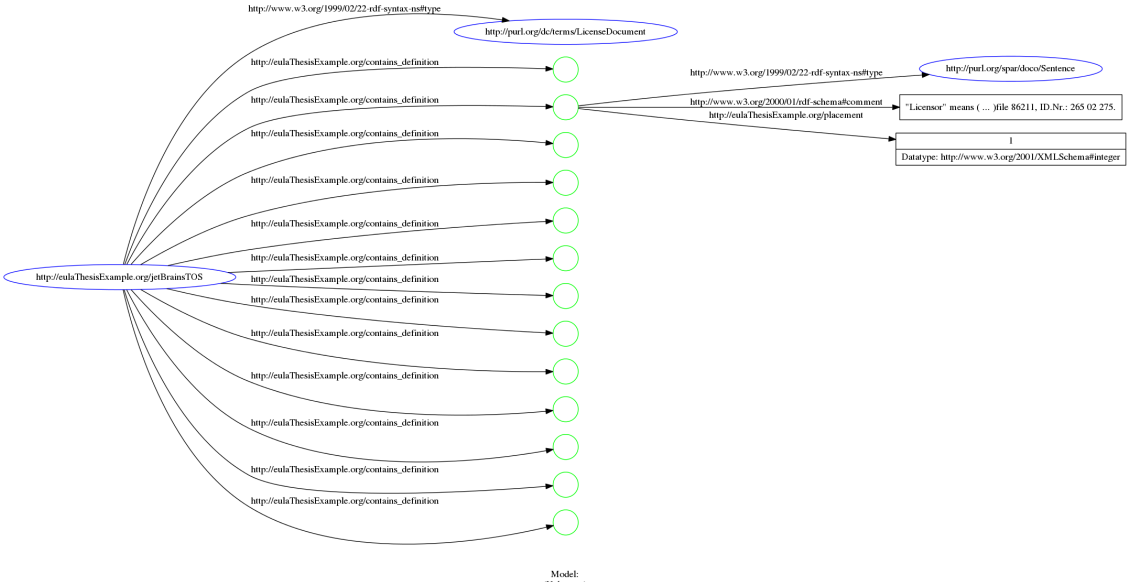

Figure 10: Triple graph: EULA contains definitions.

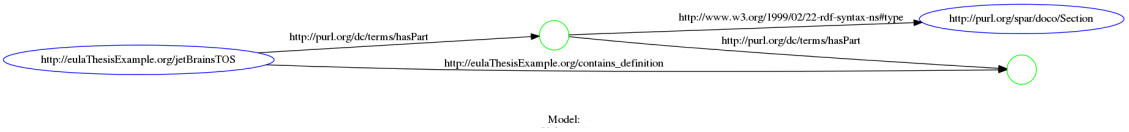

Figure 11: Triple graph: Definitions are sentences.

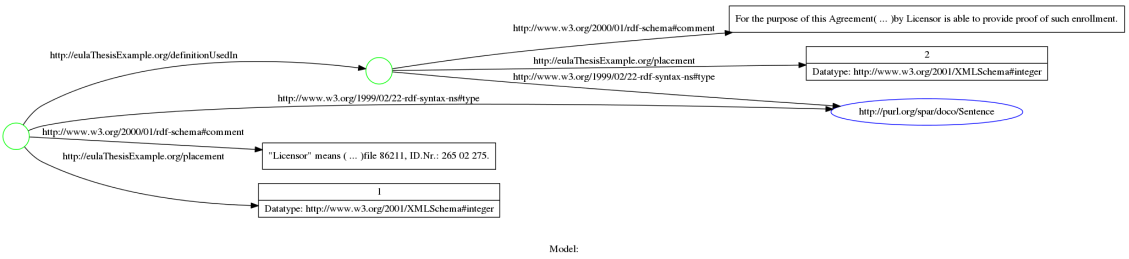

Figure 12: Triple graph: Definitions point to the places it is used.

with ex:definitionUsedIn which point to a sentence that contain the definition.

Chapter 5

Discussion

In this project I have strived to follow the Design Research Guidelines (Hevner et al., 2004, p. 83) and I have tried to use the Design Science research checklist (Hevner & Chatterjee, 2010, p. 20). The Generate/Test cycle have been a key piece in the development of my artifact, and has facilitated my exploration of the different techniques. The project has resulted in an artifact, that investigates a relevant and interesting problem. My contributions to research is hence my exploration of the different techniques and the artifact itself. The development process has utilized knowledge as this were uncovered. This is in accordance with the "Design as a Search Process" guideline, and have made me able to change directions when problems, that I could not predict, occurred.

5.0.12 Techniques investigated

My choice of techniques to investigate have been guided by the theory. In Section 2.1.1 I quote Gangemi (2013) which lists a set of NLP tasks that are relevant for the semantic web. Among these are named entity recognition and relation extraction which I have chosen to investigate. I chose not to investigate *word sense disambiguation*, the task of identifying the sense of a word when the word has several meanings. This decision were grounded in the fact that most WSD systems use a dictionary[1] to find a sense and I did not have access to a dictionary in the legal domain. Such a dictionary could also have been used as a basis for a Gazetteer list,

[1] http://aclweb.org/aclwiki/index.php?title=Word_sense_disambiguation

which is a list of important things and concepts in a text that should be highlighted. Construction of such a list would need specific domain knowledge and have been left for future work. Wimalasuriya and Dou (2010, p. 312) lists a set of main extraction techniques used in OBIE systems. I have used what they call "linguistic rules represented as regular expressions", "classifications techniques" and "partial parse trees"(See Section 2.1). This shows that I have chosen relevant techniques to investigate.

Wimalasuriya and Dou (2010, p. 309) lists a set of functionality that a system needs to be defined as an OBIE system(See Section 2.1). My system process unstructured data, and present the results using semantic technology. The last requirement for an OBIE system is that it will have to use an information extraction process guided by an ontology. My application have partially managed to follow this specification. My classifier tries to connect the extracted data to a concept in my ontology, but I have chosen to not include these data among the data that gets lifted into my triple store because of its poor performance. This makes my system an OBIE system in spirit, even though it technically doesn't qualify.

The document structure is inherently important, and is specifically mentioned as one of the layers of knowledge in a legal text(See Section 2.2.5). This means that it should be preserved through the extraction process. My extraction method separates between titles and sections. It also preserves the *label* of the section or title, where the label is any notation signifying an ordering. I have not observed any errors in this process, but this apparent robustness would break if it comes across a way of labeling that I did not see while investigating this. A section gets split into sentences as the smallest constituent. The effect of this is that I will have to represent extracted knowledge in relation to the sentence. I have chosen to do this through the use of indices. This increase the complexity of the extracted data, but the alternative is to split a sentence into even smaller constituents and that complexity is in my mind not a better solution.

Named Entities Recognition is important in most information extraction(see Section 2.1), and it is also important in the domain of licenses. In addition to the clear importance of Organizations and Locations present in a License agreement, we see that it is able to extract even more named entities than this. In the domain of licenses I have found that such a method should extract the special terms *Licensor* and *Licensee*. These should when present be handled as named entities since they give us knowledge about the different parties in the document. These concepts

should maybe be treated as a special case that could be caught using a gazetteer list, but this should be treated in future work with the cooperation of domain experts.

Binary Relations have proved to contain a lot of interesting applications. It can in its current form function as a way to connect a lot of different extracted knowledge. A named entity in the context of a relation could give an indication of importance. A relation between two named entities would mean that these named entities are somehow connected, and hence should be investigated. The same is true about definitions. A definition occurring in a relation, should be treated as important since the legal text have decided to specifically express interest in that concept.

Binary relations are interesting in itself, but I allocated a lot of time to build a classifier for such relations. My hunch were that these relations contains important information and that legal language is quite uniform in expressing meaning. This would make classification of these relations an important task, and if I could find a way to collect relations containing the same words, then all relations belonging to such a class of relations would have the same representation in an ontology. My choice of generating a corpus of relations from my corpus of EULAs were motivated by this uniformness. I chose to use topic modeling for the task of dividing the text into categories that a relation could belong to. My use of topic modeling were also motivated by a desire to filter out the names of the companies mentioned in such relations. This were not a success and as described in Section 4.4.8, I tried to experiment on removing named entities from a relation after it were extracted. This were not successful, and I think constructing the relations corpus again is warranted. This time I would remove named entities before running the relations extraction algorithm. By replacing a named entity with a place holder name it should still keep the structure of a sentence and thus not inhibit the extraction of the relations. This were not done out of time restraints as such an extraction would add to an already extensive run time of 36 hours. This should remove the chance of my classifier being confused by a topic containing a named entity. Sentences with named entities are different from sentences without, so we would want to keep a reference to a named entity if that is applicable, and this should remove the risk of errors introduced by named entities being extracted as important in a topic.

Even with the possibilities for improvement mentioned over, it still were partially successful in its grouping of relations. I had to adjust the settings of the classifier

to strict levels for the algorithm to produce usable results. This means that I chose to sacrifice some recall to increase precision. As I don't have an annotated training to use as a test set, this score is set by counting changability in the resulting classifications through my adapted levenshtein algorithm. This algorithm seemed to work in that it helped me see what classes had the most uniform relations among the topic classes. As I used a vector model for comparing relations to topics it doesn't take into account the *sentiment* of the sentence. By sentiment I mean if a sentence is positiv or negativ, or if something is expressed as allowed or not allowed. Using some sort of sentiment analysis, even in a basic form were we check for a predefined set of sentiment modifiers could improve such classification further. Existence of a training set would give me a lot of other means of training a classifier. When this is not available, and I think it wont be until we decide on a specific ontology to use in this domain, I think that further study of unsupervised machine learning techniques should continue to be an interesting field of study.

When a relation is classified into a topic, then that topic should say something about the sentences it contains. The most direct route is to check each topic and try to find a property or concept in a specific ontology that fits. This is time consuming manual labor, and not interesting in this context as this prerequisite domain knowledge. I decided to see if I could automate such a task using the labels and comments included in my test ontologies. This is based on the same idea that a relation containing the same words as another having the same meaning. For this to be a success I knew that the same words that are used in the text would have to be used inside the ontology. I realized that how we describe the use of a concept and how the same thing is used does not prerequisite the use of the same words. Despite this I still thought this idea worth exploring. This were eased by the existence of built in text search through sparql in Jena. It took some time configuring the system, since the documentation only demonstrates a minimal example. It did not give the user a possibility to attain the scoring used by the Lucene in the search query which also means that it wasn't possible for me to filter queries that such bad scores that they should be counted as not matching anything.

Definition extraction is an interesting task. As discussed in Section 2.2.2, has definitions a particularly important role in legal texts. A definition is explicit and precise and therefor important to grasp in the context it is used. My application is a very basic implementation of such a system. The task of definition extraction proved to be a much more complex matter then I first thought. That said it still catch and use at least some definitions and made it possible to illustrate how these

relate and belong in such a system. I chose to represent the whole sentence as a definition. That means that a sentence that is a definition is not treated different from other sentences, but it is easy to extract all sentences that are definitions, and also where these are used. A definition also got a pointer to all sentences where it occurs. I chose not to represent a definition with indices in the sentence that it occur since that the word occurs is what make the sentence interesting, and not the placement of the word in the sentence.

My vocabulary does not concern itself with representation of legal data. Its job is to express the knowledge extracted and tie these concepts together with the document representation. This means that it specifically represent how I think the data should be organized. This will not conflict with an ontology that described the legal concepts in the same text. This means that a future system doing more higher level analysis could use my triple store to get the text out from my system, and then enrich the resource representing the document with the knowledge they have extracted. By having a resource that represent the EULA and also keeping all the sections of the EULA available I have facilitated more extensive extraction at a later step. If I had managed to use my classifier to decide to what concept in an ontology a binary relation belongs to. Then my vision where connecting the resource representing the binary relation to the ontology. This way it would be made clear that the analysis is based upon the knowledge extracted in the relation and not on analysis of the section or the specific sentence. I chose to use the DoCO vocabulary to describe a legal document. This vocabulary describes the parts of a document in a fitting manner for this kind of application. It consists of a lot of other things not used in my system. If I were to decide that it is to elaborate or there is developed a vocabulary that better describes a legal document it should be trivial to change. My choice of representing a document using anonymous nodes that has a comment I think is a logical way to model the document structure in my system. If a resource points to a part, and this part is a literal string then that string can't be used as a subject in any other triples. This way we would lose a lot of flexibility. I contemplated defining a type ex:extraction, where a node representing an extraction has type ex:extraction and then all extractions has a ex:extractionMethod that describe what the extraction consists of. This would have made it possible to easily query for things like "all things extracted" for a specific license. As it is now you will have to know how the specific extraction methods are represented. This is not as modular as the other type, but it is flexible in that every property in this case can be seen to represent that specific extraction

method. A user is free to say at what level an extraction happen. Is it extracted from a sentence, a section or from the whole document.

5.0.13 Advantages, Disadvantages and Problems with my system

My system is quite simple in form. This simplicity is good in that the addition of new techniques for extraction should be independent from the already existing extraction methods. A disadvantage of this is that it doesn't provide a standard way of extending the system. Creating rules and deciding in advance how we should handle new extractions I think would be ideal. It demands a higher step-in phase for new developers, but from that point we got a standard way of understanding the internal representation. My system in its current form require a change to my representation of a document. This represent the simplicity in that I allow changes, but it creates complications in that everyone are free to complicate as needed and with no rules to ensure consistency. I think that moving all extracted data out into classes implementing an *interface* or by sub classing and that way enforcing what is the systems requirements for use should be a natural next step. This way adding a new extraction method would only demand writing a domain representation of the data extracted, and then writing a corresponding handling of this in the modules that are responsible for lifting the data. I did a similar choice with my vocabulary and chose not to include any strict rules for how it should be used. As with the source code, this makes it possible to represent knowledge in several different ways. Even my own extraction is not perfectly consistent. By imposing some more rules and restricting how extraction should be represented as triples I think it would make my system easier to use.

Another problem is the connection of the different knowledge layers. In my system I have made an assumption that a sentence with the same words means the same. This is a simplification that creates some problems. First we got the fact the several sentences should be associated with the same predicate. This is not a problem as long as the system classify both sentences into correlating classes. The problem is visible in my use of text search to try and connect properties to sentences. Since we can express a lot of the same things using different words, would my technique only find connections where the words used in the ontology and the words used in the text correlate. This could in turn be mediated somewhat if we

demand that an ontology contains not only one definition, but several covering the same topic. Though that might help my algorithm it could undo any gain won through simplification if not done in a cautious manner.

The system as is doesn't connect named entities that are arguments in a binary relation to the specific resource representing that named entity. As the text in a binary relation argument might also contain more than just the named entity I chose to postpone this decision. I think the right way to do it, would be to create a property from the named entity resource to the string literal. This way we could still keep the complete string that is the argument as extracted by the system, and also use the named entities to point to where it is used in the text. Because of time constraints I did not implement this or explore the effect this should have other then the informal definition given over.

Chapter 6

Conclusion and Further Work

In this project I have tried to evaluate existing information extraction techniques in the new domain of Licenses. I have built an Artifact that use these techniques, lifts the resulting knowledge using semantic technologies and persist them in a triple store. My choice of techniques is grounded in the theory and I have shown that I have chosen relevant techniques to investigate. This artifact is a small piece in the big field of semantic web and Law, but fills its purpose as illustrating some aspect needed of an information extraction system in this domain. Because of inherent complexity in the legal domain is the extraction of knowledge from legal text a hard task. The sub-domain of licenses inherit all these complexities, and my initial intuition that this limits some of the complexity were not true. Because of the levels of knowledge inherent in a legal text can my system only hope to say something about what is extracted directly from the text, and leaving the complex higher levels of knowledge to future work. With the future addition of a good and extensive legal ontology covering the higher layers of knowledge could my project be seen as a step towards connecting legal language expressed in a license agreement and such an ontology.

6.1 Research questions

I have shown in Chapter 4 what kinds of data a set of common techniques can extract. These extraction have shown that it is possible to extract interesting knowledge in this domain using no specific tuning or training of the methods. These

techniques are not in itself enough to satisfyingly describe a license agreement. Without a higher layer ontology that describe how the things made available in the text is realized it will never be able to reason about anything that a lay person doesn't have insight into. Since the text in the legal document always will have to be available, as discussed in Section 2.2.5, will this kind of extraction always need to be grounded in the legal text. It will still be beneficial to extract this kinds of data, as this could improving knowledge discovery and comprehension for a user. What this shows is that there is a good amount of knowledge available for extraction even without adding any domain knowledge. Domain knowledge would in an OBIE system be expressed in an ontology that is used actively in the extraction of data. The development of my classifier using topic extraction and binary relations(See Section 4.4.8), and then using text search through sparql, was my attempt to bridge the layers of knowledge. The classifier might have some promise, in that it seems to classify some kinds of relations, but this classifications is done by using the same data as the learning task and hence needs thorough testing. I have also highlighted some changes that should be made to try and increase its performance. The use of text search to find matching comments in an ontology might be an interesting venture, but as of now I did not manage to connect binary relations to specific properties in the ontologies investigated. First of all I think that an ontology describing the domain of licenses in a thorough way would help this process. Such a specific ontology might have its own problems, but if we suppose the existence of this, then could the combination of a classifier for binary relations and text search be a viable option and at least warrant more scrutiny. A second thing to consider is the inclusion of "sample phrases" in such an ontology. A domain expert should be able to point out how important concepts are realized in legal texts. The inclusion of such specialized knowledge like sample words, or even maybe a machine learned set of words that describe that specific topic, would facilitate an extraction task, but as discussed earlier should this not be at the expense of clarity and simplification.

6.2 Further work

An important and prioritized task should be to create an ontology that describes the concepts of a license or adding this domain to more general ontologies. The construction of such an ontology is a big project that will have to be done by

people with intimate knowledge of the legal system. Since Wong et al. (2012, p. 30) conclude that the majority of ontologies developed are lightweight ontologies might the gap between the legal knowledge layers not be closed any time soon. Describing the normative layer of a legal text is a complicated task in itself. This is complicated even further by differing judicial systems with different interpretations and different judgments. I think that trying to create a cohesive general system is an intractable problem, but by using the mechanics of the semantic web could such a system be built in an iterative and distributed manner. This might be a lengthy process, but as technology and development progress should patterns and definitive systems emerge.

Even with an ontology that describes the domain perfectly is the task of connecting concepts from this ontology to sentences in the legal text a hard problem. This problem is further complicated by the fact that something that could or should be designed as a concept in such an ontology might not be succinctly and specifically expressed in the text. Finding a way to extract such *emergent concepts* could prove to be a hard task. Another big problem is synonymous with the problems in all Legal Ontologies in that these concepts should be connected to their realization in a legal system. This will be different across continents and countries and might modify specific parts of a document into more specific word meanings and usage.

A system following the form of Creative Commons distributed by the Licensees themselves could prove to be the solution to some of the problems of knowledge layers and understanding. As consumers get used to more and more fine grained information available, I think a demand for machine readable licenses will be inevitable. The problem doesn't just start in the representation of a legal text using semantic technology(that is expressing knowledge in an ontology). When the technical details are figured out an important theme would be the problem of trust. As the legal text got the final say in a legal case, then how can we trust what is distributed in a machine readable format? There are different ways this could be solved. All links between these layers could be controlled by a third party organization that can certify the knowledge. Another alternative is a vision of an crowd sourced online repository that can correlate the texts and the semantics before consumption by users. This is only viable with broad adoption, and we also got to trust the editors and commentators of this service. Even with such a thing in place will the problem of simplifying a text without loosing clarity(See Section 2.3.1) be a problem that needs to be solved to stop users inadvertently breaking the agreements terms.

Another interesting outlook would be to check my systems effectiveness in context of *Document summarization* and if it helps *comprehension* of a license agreement. Despite its problems outlined in Section 5.0.13 I still think these extractions can help a user to understand the underlying text better. For a legal professional it might be viewed as metadata that could be made available, and where the linking structure might help navigation of large documents. For a lay user interested in knowing what is said, but not assuming any legal knowledge I also think it could have a positive effect in comprehension. I would propose an experiment where the comprehension is tested, through an application that graphically provide highlighting of concepts in the text and that helps a user navigate a legal document. Then through an interview or a questionnaire designed by domain experts could we measure what level of comprehension is attained. A test group being exposed to an summarized version extracted would also be interesting of the same reason, and also to see the difference between expert users and lay users. Such an experiment is dependent upon development of specialized software to present these kinds of data.

The use of a gazetteer list also warrants further focus. Such a list might be generated from an ontology, or an ontology could be seen as a specialization of such a list. In its nature should a domain ontology describe what is important. This is synonymous to a gazetteer list in that it highlights important concepts. The construction of such a list would need to be in accordance with legal experts because of the layered nature of legal knowledge. I have focused on what is possible to extract directly from the text without specialized knowledge about how these things are interpreted in court. A gazetteer list is most interesting if what is described as important is because of the normative extension of the text. This way we will be able to extract what is meant by the text and how this meaning is expressed in a legal system. Just as a list of words without any connection to further knowledge or an ontology I think it would not increase knowledge in any influential way.

This project have shown me the level of complexity needed in such a solution. Even with the continuous complexity of user adoption I still doesn't think the outlook is bleak. As shown in Chapter 2 there are a lot of research in the legal domain and semantic technology. This research is mostly guided towards helping experts do their job better, but I also think this technology will gain lay users. This kind of technology might be a middle ground in simplifying and increasing comprehension from regular users. Simplifications has its problems as discussed in Section 2.3.1, but it doesn't take to much inspiration to envision an ontology describing the concepts already present in a detailed legal ontology in a simpler

manner. Such a description could illustrate concepts and correlations both for professionals in training and for lay users trying to navigate the complicated domain of legal language.

Bibliography

Ahmed, M., Anjomshoaa, A., Asfandeyar, M., Tjoa, a. M., & Khan, A. (2010, February). Towards an Ontology-Based Solution for Managing License Agreement Using Semantic Desktop. *2010 International Conference on Availability, Reliability and Security*, 309–314. doi:10.1109/ARES.2010.104

Allemang, D. & Hendler, J. (2011, July). *Semantic Web for the Working Ontologist: Effective Modeling in RDFS and OWL* (2nd ed.). Morgan Kaufmann Publishers Inc. Retrieved from http://dl.acm.org/citation.cfm?id=2016697

Anjomshoaa, A., Weippl, E., Tjoa, A., & Asfandeyar, M. (2009). Blending the sketched use case scenario with license agreements using semantics. *Knowledge Science, Engineering ...* 275–284. Retrieved from http://link.springer.com/chapter/10.1007/978-3-642-10488-6%5C_28

Asfand-e-yar, M. & Tjoa, A. M. (2013). LNCS 7804 - Using Semantic Web to Enhance User Understandability for Online Shopping License Agreement, 233–242.

Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., ... Thomas, D. (n.d.). Manifesto for Agile Software Development. Retrieved April 30, 2015, from http://agilemanifesto.org/

Benjamins, V. R., Casanovas, P., Breuker, J., Gangemi, A., & Marx, V. (2005). Law and the Semantic Web, an Introduction. *3369*, 1–17.

Blei, D. (2012). Probabilistic Topic Models. *Communications of The ACM*, *55*(4), 77–84.

Boer, A., Winkels, R., & Vitali, F. (2008). MetaLex XML and the Legal Knowledge Interchange Format. In P. Casanovas, G. Sartor, N. Casellas, & R. Rubino (Eds.), *Computable models of the law* (pp. 21–41). Springer Berlin Heidelberg. doi:10.1007/978-3-540-85569-9-2

Cimiano, P. (2006). *Ontology learning and population from text: algorithms, evaluation and applications*. doi:10.1007/978-0-387-39252-3

Cimiano, P., Ladwig, G., & Staab, S. (2005). Gimme'the context: context-driven automatic semantic annotation with C-PANKOW. *... of the 14th international conference on ...* 332–341. Retrieved from http://dl.acm.org/citation.cfm?id=1060796

Commons, C. (2014). Creative Commons. Retrieved May 28, 2014, from http://creativecommons.org/

Corro, L. D. & Gemulla, R. (2013). ClausIE: clause-based open information extraction. *... of the 22nd international conference on World ...* (1). Retrieved from http://dl.acm.org/citation.cfm?id=2488420

Finkel, J. R., Grenager, T., & Manning, C. (2005). Incorporating non-local information into information extraction systems by gibbs sampling. *Proceedings of the 43nd Annual Meeting of the Association for Computational Linguistics*, (1995), 363–370. doi:10.3115/1219840.1219885

Fowler, M. (2004). *UML Distilled: A Brief Guide to the Standard Object Modeling Language* (Third Edit). Addison-Wesley. Retrieved from http://books.google.no/books/about/UML%5C_Distilled.html?id=nHZslSr1gJAC%5C&pgis=1

Gangemi, A. (2013). A Comparison of Knowledge Extraction Tools for the Semantic Web. In *Lecture notes in computer science (including subseries lecture notes in artificial intelligence and lecture notes in bioinformatics)* (Vol. 7882 LNCS, pp. 351–366). doi:10.1007/978-3-642-38288-8-24

Gosling, J., Joy, B., Steele, G., Bracha, G., & Buckley, A. (2015). The Java® Language Specification.

Hepp, M. (2007). ONTOLOGIES: STATE OF THE ART, BUSINESS POTENTIAL, AND GRAND CHALLENGES. In M. Hepp, P. D. Leenheer, A. de Moor, & Y. Sure (Eds.), *Ontology management: semantic web, semantic web services, and business application* (pp. 3–22). Springer.

Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Design Science in Information Systems Research. *MIS Quarterly*, *28*(1), 75–105. doi:10.2307/25148625. arXiv: /dl.acm.org/citation.cfm?id=2017212.2017217 [http:]

Hevner, A. & Chatterjee, S. (2010). Design Science Research in Information Systems. In *Design research in information systems* (Vol. 22, pp. 9–22). doi:10.1007/978-1-4419-5653-8

HowToGeek. (2014). The Software License Agreement (or EULA) Dissected and Explained. Retrieved May 31, 2014, from http://www.howtogeek.com/53107/the-software-license-agreement-or-eula-dissected-and-explained/

Larman, C. (2005). *Applying UML and Patterns: An Introduction to Object-oriented Analysis and Design and Iterative Development*. Prentice Hall PTR. Retrieved from http://books.google.com/books?id=tuxQAAAAMAAJ%5C&pgis=1

Lavesson, N. (2014). End User License Agreements (EULA) Database. Blekinge Tekniska Högskola. Retrieved May 28, 2014, from http://www.bth.se/people/nla.nsf/sidor/resources

Lavesson, N., Boldt, M., Davidsson, P., & Jacobsson, A. (2010, January). Learning to detect spyware using end user license agreements. *Knowledge and Information Systems*, *26*(2), 285–307. doi:10.1007/s10115-009-0278-z

Lenci, A., Montemagni, S., Pirrelli, V., & Venturi, G. (2009, July). Ontology learning from Italian legal texts, 75–94. Retrieved from http://dl.acm.org/citation.cfm?id=1563987.1563995

LINFO. (2006). EULA definition by The Linux Information Project. Retrieved May 31, 2014, from http://www.linfo.org/eula.html

Marneffe, M.-c. D., Maccartney, B., & Manning, C. D. (1999). Generating Typed Dependency Parses from Phrase Structure Parses.

Marneffe, M.-c. D. & Manning, C. D. (2013). Stanford typed dependencies manual.

Maynard, D., Peters, W., & Li, Y. (2006). Metrics for Evaluation of Ontology-based Information Extraction.

McCarty, L. T. (2007). Deep semantic interpretations of legal texts. *Proceedings of the 11th international conference on Artificial intelligence and law - ICAIL '07*, 217. doi:10.1145/1276318.1276361

Myška, M., Smejkalová, T., Šavelka, J., & Škop, M. (2012). Creative Commons and Grand Challenge to Make Legal Language Simple. In *Ai approaches to the complexity of legal systems. models and ethical challenges for legal systems, legal language and legal ontologies, argumentation and software agents* (pp. 271–285). Springer Berlin Heidelberg.

Oates, B. J. (2006). *Researching Information Systems and Computing*.

Palmirani, M., Cervone, L., & Vitali, F. (2011). A Legal Document Ontology: The Missing Layer in Legal Document Modelling. In *Approaches to legal ontologies* (pp. 167–178). Springer Netherlands. doi:10.1007/978-94-007-0120-5_10

PCMagazine. (2014). EULA Definition from PC Magazine Encyclopedia. Retrieved May 31, 2014, from http://www.pcmag.com/encyclopedia/term/42799/eula

Shadbolt, N., Berners-Lee, T., & Hall, W. (2006, May). The Semantic Web Revisited. *IEEE Intelligent Systems*, *21*(3), 96–101. doi:10.1109/MIS.2006.62

Shah, R. & Jain, S. (2014). Ontology-based Information Extraction: An Overview and a Study of different Approaches. *87*(4), 6–8.

Shotton, D. & Peroni, S. (2014). DoCO, the Document Components Ontology. Retrieved May 6, 2015, from http://www.essepuntato.it/lode/http://purl.org/spar/doco

Surdeanu, M., Bauer, J., Finkel, J., Bethard, S. J., McClosky, D., & Manning, C. D. (2014). The Stanford CoreNLP Natural Language Processing Toolkit. *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, 55–60. Retrieved from http://www.surdeanu.info/mihai/papers/acl2014-corenlp.pdf

The Stanford NLP Group. (2015). The Stanford NLP Group. Retrieved April 27, 2015, from http://nlp.stanford.edu/

TIOBE. (2015). TIOBE Software: Tiobe Index. Retrieved April 8, 2015, from http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html

Venturi, G. (2010). Legal Language and Legal Knowledge Management Applications, 3–26.

Völker, J., Langa, S., & Sure, Y. (2008). Supporting the construction of Spanish legal ontologies with Text2Onto. *Computable Models of the Law*, 105–112. Retrieved from http://link.springer.com/chapter/10.1007/978-3-540-85569-9%5C_7

Walter, S. & Pinkal, M. (2009, July). Definitions in Court Decisions –Automatic Extraction and Ontology Acquisition, 95–113. Retrieved from http://dl.acm.org/citation.cfm?id=1563987.1563996

Weaver, P. (2004). *Success in Your Project: a guide to student system development projects*. Prentice Hall; retrieved from http://www.pearson.ch/HigherEducation/Management/ResearchMethodsin/1471/9780273678090/Success-in-Your-Project-a-guide-to.aspx

Westerhout, E. (2010). *Definition extraction for glossary creation* (Doctoral dissertation).

Wimalasuriya, D. C. & Dou, D. (2010, March). Ontology-based information extraction: An introduction and a survey of current approaches. *Journal of Information Science*, *36*(3), 306–323. doi:10.1177/0165551509360123

Wong, W., Liu, W., & Bennamoun, M. (2012, August). Ontology learning from text. *ACM Computing Surveys*, *44*(4), 1–36. doi:10.1145/2333112.2333115

Zhao, Q. & Perry, M. (2008, March). An Ontology for Autonomic License Management. *Fourth International Conference on Autonomic and Autonomous Systems (ICAS'08)*, 204–211. doi:10.1109/ICAS.2008.12

Appendices

Appendix A

JETBRAINS LICENSE AGREEMENT FOR EDUCATION

JETBRAINS LICENSE AGREEMENT FOR EDUCATION

Version 1, Effective as of September 23, 2014

IMPORTANT! READ CAREFULLY: THIS IS A LEGAL AGREEMENT. BY DOWNLOADING, INSTALLING, COPYING, SAVING ON YOUR COMPUTER, OR OTHERWISE USING THIS SOFTWARE, YOU (LICENSEE, AS DEFINED BELOW) ARE BECOMING A PARTY TO THIS AGREEMENT AND YOU ARE CONSENTING TO BE BOUND BY ALL THE TERMS AND CONDITIONS OF THIS AGREEMENT. IF YOU DO NOT AGREE TO THE TERMS AND CONDITIONS OF THIS AGREEMENT, YOU SHOULD NOT DOWNLOAD, INSTALL AND USE THE SOFTWARE.

1. PARTIES

(a) "Licensor" means JetBrains s.r.o., having its principal place of business at Na hrebenech II 1718/10, Prague, 14700, Czech Republic, registered with Commercial Register kept by the Municipal Court of Prague, Section C, file 86211, ID.Nr.: 265 02 275.

(b) "Licensee" means a student or an instructor. For the purpose of this Agreement, a student is an individual who is enrolled at a recognized tertiary educational institution (university or college) that grants degrees requiring not less than the equivalent of two years of full-time study, and upon request by Licensor is able to provide proof of such enrollment. For the purpose of this Agreement, an instructor is an individual who conducts lectures and/or seminars at a recognized tertiary educational institution (university or college), and upon request by Licensor is able to provide proof of such involvement.

2. DEFINITIONS

(a) "Client" means a computer device used by Licensee for running Software.

(b) "Software" means any software program included in JetBrains Educational Program at https://www.jetbrains.com/student and any third party software programs that are owned and licensed pursuant to Section 5 of this Agreement by parties other than Licensor and that are either integrated with or made part of Software (collectively, "Third Party Software").

(c) "JetBrains Account" means profile record on https://account.jetbrains.com, which identifies Licensee and licenses for the Software provided by Licensor to Licensee. Sharing credentials for a JetBrains Account with any other person is not permitted.

3. OWNERSHIP

(a) Software is the property of Licensor or its suppliers. Software is licensed, not sold. Title and copyrights to Software, in whole and in part and all copies thereof, and all modifications, enhancements, derivatives and other alterations of Software regardless of who made any modifications, if any, are, and will remain, the sole and exclusive property of Licensor and its suppliers.

(b) Software is protected by United States Copyright Law and International Treaty provisions. Further, the structure, organization, and code embodied in Software are the valuable and confidential trade secrets of Licensor and its suppliers and are protected by intellectual property laws and treaties. Licensee agrees to abide by the copyright law and all other applicable laws of the United States including, but not limited to, export control laws.

4. GRANT OF LICENSE

Subject to the terms, conditions, and limitations set forth in this Agreement, Licensor hereby grants to Licensee a limited, non-exclusive, non-transferable license to use Software for non-commercial, educational purposes only (including conducting academic research or providing educational services) as follows:

(a) Licensee may: (i) (i) install, register with JetBrains Account, and use the licensed edition and version of Software listed at https://www.jetbrains.com/student on any number of Clients and on any operating system supported by Software; (ii) use Software for non-commercial, educational purposes only, including conducting academic research or providing educational services; and

(iii) make one back-up copy of Software solely for archival purposes.

(b) Licensee may not: (i) sell, redistribute, encumber, give, lend, rent, lease, sublicense, or otherwise transfer Software, or any portions of Software, to anyone without the prior written consent of Licensor; (ii) reverse engineer, decompile, disassemble, modify, translate, make any attempt to discover the source code of Software, or create derivative works from Software; or (iii) use Software for any commercial purpose.

6. THIRD-PARTY SOFTWARE LICENSE

Software includes certain Third-Party Software. Licensee agrees to comply with the terms and conditions contained in Third-Party Software license agreements. List of such Third-Party Software is available on Licensor's website at www.jetbrains.com. Licensee agrees and acknowledges that Sections 8 and 9 of this Agreement shall also govern Licensee's use of the Third-Party Software. Licensor will have no responsibility with respect to any Third-Party Software, and Licensee will look solely to the licensor(s) of the Third-Party Software for any remedy. Licensor claims no right in the Third-Party Software, and the same is owned exclusively by the licensor(s) of the Third-Party Software.

LICENSOR PROVIDES NO WARRANTY, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE, AND NON-INFRINGEMENT, WITH RESPECT TO ANY THIRD-PARTY SOFTWARE.

7. RESTRICTED USE DURING EVALUATION PERIOD

(a) Licensee is granted a right to use Software for evaluation purposes without charge, subject to the evaluation terms and conditions with respect to particular Software available at www.jetbrains.com ("Evaluation Period").

(b) Upon expiration of Evaluation Period, Licensee must obtain License for perpetual use of Software or cease using Software. Software contains a feature that will automatically disable Software upon expiration of Evaluation Period. Licensee may not disable, destroy, or remove this feature of Software, and any attempt to do so will be in violation of this Agreement and will terminate Licensee's rights to use Software. 7. UPGRADES

(a) During the term of this Agreement (as defined in Section 11(a) below), Licensor will provide generally available new versions of Software to Licensee free of charge and pursuant to the optional upgrade terms published on Licensor's website at www.jetbrains.com.

(b) If not agreed otherwise in writing between Licensor and Licensee, upon upgrading to new version of Software the relationship between parties shall be governed and amended (if applicable) by the terms and conditions of Licensor's License Agreement for Education related to Software available at www.jetbrains.com on the day of upgrade purchase or download.

8. LIMITED WARRANTY

SOFTWARE IS PROVIDED TO LICENSEE "AS IS" AND WITHOUT WARRANTIES. LICENSOR MAKES NO WARRANTY AS TO ITS USE OR PERFORMANCE. TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, LICENSOR, AND ITS AFFILIATES, SUPPLIERS AND RESELLERS, DISCLAIM ALL OTHER WARRANTIES AND CONDITIONS, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE, AND NON-INFRINGEMENT, WITH REGARD TO SOFTWARE, AND THE PROVISION OF OR FAILURE TO PROVIDE SUPPORT SERVICES.

9. DISCLAIMER OF DAMAGES

(a) TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW,

IN NO EVENT WILL LICENSOR OR ITS AFFILIATES, LICENSORS, SUPPLIERS OR RESELLERS BE LIABLE TO LICENSEE UNDER ANY THEORY FOR ANY DAMAGES SUFFERED BY LICENSEE OR ANY USER OF SOFTWARE, OR FOR ANY SPECIAL, INCIDENTAL, INDIRECT, CONSEQUENTIAL, OR SIMILAR DAMAGES (INCLUDING, BUT NOT LIMITED TO, DAMAGES FOR LOSS OF PROFITS OR CONFIDENTIAL OR OTHER INFORMATION, FOR BUSINESS INTERRUPTION, FOR PERSONAL INJURY, FOR LOSS OF PRIVACY, FOR FAILURE TO MEET ANY DUTY INCLUDING OF GOOD FAITH OR OF REASONABLE CARE, FOR NEGLIGENCE, AND FOR ANY OTHER PECUNIARY OR OTHER LOSS WHATSOEVER) ARISING OUT OF THE USE OR INABILITY TO USE SOFTWARE, OR THE PROVISION OF OR FAILURE TO PROVIDE SUPPORT SERVICES, EVEN IF LICENSOR HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES, AND REGARDLESS OF THE LEGAL OR EQUITABLE THEORY (CONTRACT, TORT OR OTHERWISE) UPON WHICH THE CLAIM IS BASED.

(b) IN ANY CASE, LICENSOR'S ENTIRE LIABILITY UNDER ANY PROVISION OF THIS AGREEMENT WILL BE LIMITED TO THE AMOUNT OF FIVE (5) US DOLLARS.

10. EXPORT REGULATIONS

Licensee agrees and accepts that Software may be subject to import and export laws of any country, including those of the European Union and the United States (specifically the Export Administration Regulations (EAR)). Licensee acknowledges that it is not a citizen, national, or resident of, and is not under control of the governments of Cuba, Iran, North Korea, Sudan or Syria and is not otherwise a restricted end-user as defined by applicable export control laws. Further, Licensee acknowledges that it will not download or otherwise export or re-export Software or any related technical data directly or indirectly to the above-mentioned countries or to citizens, nationals, or residents of those countries, or to any other restricted end user or for any restricted end-use.

11. TERM AND TERMINATION

The license is granted for a period of one (1) year. To maintain a license, Licensee may submit to Licensor renewal application for another one (1) year. Licensor reserves the right to reject such application without cause.

(b) If Licensee fails to comply with the terms and conditions of this Agreement, this Agreement and Licensee's right and license to use Software will terminate immediately. Licensee may terminate this Agreement at any time by notifying Licensor. Upon the termination of this Agreement, Licensee must delete Software from its Clients and archives.

(c) LICENSEE AGREES THAT UPON TERMINATION OF THIS AGREEMENT FOR ANY REASON, LICENSOR MAY TAKE ACTIONS SO THAT SOFTWARE NO LONGER OPERATES.

12. MARKETING

Licensee agrees to be identified as a customer of Licensor and agrees that Licensor may refer to Licensee by name, trade name and trademark, if applicable, and may briefly describe Licensee's business in Licensor's marketing materials, on Licensor's website, in public or legal documents. Licensee hereby grants Licensor a license to use Licensee's name and any of Licensee's trade names and trademarks solely pursuant to this marketing section.

13. GENERAL

(a) Licensor reserves the right at any time to cease the support of Software and to alter prices, features, specifications, capabilities, functions, licensing terms, release dates, general availability or other characteristics of Software.

(b) This Agreement, including the Third-Party Software license agreements, constitutes the entire agreement between the parties concerning Licensee's use of Software, and supersedes any and all prior or contemporaneous oral or written representations, communications, or advertising with respect to Software. No purchase order, other ordering document or any hand written or typewritten text which purports to modify or supplement the printed text of this Agreement or any schedule will add to or vary the terms of this Agreement unless signed by both Licensee and Licensor.

(c) A waiver by either party of any term or condition of this Agreement or any breach thereof, in any one instance, will not waive such term or condition or any subsequent breach. The provisions of this Agreement which require or contemplate performance after the expiration or termination of this Agreement will be enforceable notwithstanding said expiration or termination.

(d) This Agreement will be governed by the laws of Czech Republic, without reference to conflict of laws principles. Licensee agrees that any litigation relating to this Agreement may only be brought in, and will be subject to the jurisdiction of, any Court of Czech Republic.

(e) Titles are inserted for convenience only and will not affect in any way the meaning or interpretation of this Agreement. If any provision of this Agreement is held invalid, the remainder of this Agreement will continue in full force and effect. Either Licensor or Licensee may assign this Agreement in the case of a merger or sale of substantially all of its respective assets to another entity. This Agreement will be binding upon and will inure to the benefit of the parties, their successors and assigns.

For exceptions or modifications to this Agreement, please contact Licensor at:

Address: Na hrebenech II 1718/10, Prague, 14700, Czech Republic Fax: +420 241 722 540 E-mail: sales@jetbrains.com

Appendix B

Binary Relations

Binary relations extracted using ReVerb with confidence higher than 0.75 with context text.

IMPORTANT! READ CAREFULLY: THIS IS A LEGAL AGREEMENT. BY DOWNLOADING, INSTALLING, COPYING, SAVING ON YOUR COMPUTER, OR OTHERWISE USING THIS SOFTWARE, YOU (LICENSEE, AS DEFINED BELOW) ARE BECOMING A PARTY TO THIS AGREEMENT AND YOU ARE CONSENTING TO BE BOUND BY ALL THE TERMS AND CONDITIONS OF THIS AGREEMENT. IF YOU DO NOT AGREE TO THE TERMS AND CONDITIONS OF THIS AGREEMENT, YOU SHOULD NOT DOWNLOAD, INSTALL AND USE THE SOFTWARE.

Arg1	Relation	Arg2	Confidence
THIS	IS	A LEGAL AGREEMENT.	0.8080483769098364
DEFINED BELOW)	ARE BECOMING A PARTY TO	THIS AGREEMENT	0.8361206564311017

"Licensor" means JetBrains s.r.o., having its principal place of business at Na hrebenech II 1718/10, Prague, 14700, Czech Republic, registered with Commercial Register kept by the Municipal Court of Prague, Section C, file 86211, ID.Nr.: 265 02 275.

Arg1	Relation	Arg2	Confidence
Commercial Register	kept by	the Municipal Court of Prague	0.9016428247060491

Software is the property of Licensor or its suppliers. Software is licensed, not sold. Title and copyrights to Software, in whole and in part and all copies thereof,

and all modifications, enhancements, derivatives and other alterations of Software regardless of who made any modifications, if any, are, and will remain, the sole and exclusive property of Licensor and its suppliers.

Arg1	Relation	Arg2	Confidence
Software	is the property of	Licensor	0.9018802574024445

Software is protected by United States Copyright Law and International Treaty provisions. Further, the structure, organization, and code embodied in Software are the valuable and confidential trade secrets of Licensor and its suppliers and are protected by intellectual property laws and treaties. Licensee agrees to abide by the copyright law and all other applicable laws of the United States including, but not limited to, export control laws.

Arg1	Relation	Arg2	Confidence
Software	is protected by	United States Copyright Law and International Treaty provisions	0.9755597201513483
Software	are	the valuable and confidential trade secrets of Licensor	0.7564821600277878
the valuable and confidential trade secrets of Licensor	are protected by	intellectual property laws and treaties	0.8609062079391654

Subject to the terms, conditions, and limitations set forth in this Agreement, Licensor hereby grants to Licensee a limited, non-exclusive, non-transferable license to use Software for non-commercial, educational purposes only (including conducting academic research or providing educational services) as follows:

Arg1	Relation	Arg2	Confidence
limitations	set forth in	this Agreement	0.8806351795513172

(i) install, register with JetBrains Account, and use the licensed edition and version of Software listed at URL0 on any number of Clients and on any operating system supported by Software;

Arg1	Relation	Arg2	Confidence
the licensed edition and version of Software	listed at	URL0	0.8369310285744885

sell, redistribute, encumber, give, lend, rent, lease, sublicense, or otherwise transfer Software, or any portions of Software, to anyone without the prior written consent of Licensor;

Arg1	Relation	Arg2	Confidence
sublicense	otherwise transfer	Software	0.8262168566121267
the source code of Software	create derivative works from	Software	0.8329844416682104

Software includes certain Third-Party Software. Licensee agrees to comply with the terms and conditions contained in Third-Party Software license agreements. List of such Third-Party Software is available on Licensor's website at URL2. Licensee agrees and acknowledges that Sections 8 and 9 of this Agreement shall also govern Li. Licensor will have no responsibility with respect to any Third-Party Software, and Licensee will look solely to the licensor(s) of the Third-Party Software for any remedy. Licensor claims no right in the Third-Party Software, and the same is owned exclusively by the licensor(s) of the Third-Party Software.

Arg1	Relation	Arg2	Confidence
Software	includes	certain Third-Party Software	0.9390894667018188
List of such Third-Party Software	is available on	Licensor 's website	0.9731974582444354
Licensor	claims no right in	the Third-Party Software	0.9375432980258913

Upon expiration of Evaluation Period, Licensee must obtain License for perpetual use of Software or cease using Software. Software contains a feature that will automatically disable Software upon expiration of Evaluation Period. Licensee may not disable, destroy, or remove this feature of Software, and any attempt to do so will be in violation of this Agreement and will terminate Licensee's rights to use Software.

Arg1	Relation	Arg2	Confidence
Licensee	must obtain License for	perpetual use of Software	0.8451362130624644
Licensee	remove	this feature of Software	0.9012987057050333

LIMITED WARRANTY SOFTWARE IS PROVIDED TO LICENSEE "AS IS" AND WITHOUT WARRANTIES. LICENSOR MAKES NO WARRANTY AS TO ITS USE OR PERFORMANCE. TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, LICENSOR, AND ITS AFFILIATES, SUPPLIERS AND RESELLERS, DISCLAIM ALL OTHER WARRANTIES AND CONDITIONS, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE, AND NON-INFRINGEMENT, WITH REGARD TO SOFTWARE, AND THE PROVISION OF OR FAILURE TO PROVIDE SUPPORT SERVICES.

Arg1	Relation	Arg2	Confidence
SOFTWARE	IS PROVIDED TO	LICENSEE	0.9597843186585014

EXPORT REGULATIONS Licensee agrees and accepts that Software may be subject to import and export laws of any country, including those of the European Union and the United States (specifically the Export Administration Regulations (EAR)). Licensee acknowledges that it is not a citizen, national, or resident of, and is not under control of the governments of Cuba, Iran, North Korea, Sudan or Syria and is not otherwise a restricted end-user as defined by applicable export control laws. Further, Licensee acknowledges that it will not download or otherwise export or re-export Software or any related technical data directly or indirectly to the above-mentioned countries or to citizens, nationals, or residents of those countries, or to any other restricted end user or for any restricted end-use.

Arg1	Relation	Arg2	Confidence
Licensee	accepts	that Software	0.9111447197045949
Licensee	export	laws of any country	0.7746359356723698
resident	is not under	control of the governments of Cuba	0.951231153918689

TERM AND TERMINATION The license is granted for a period of one (1) year. To maintain a license, Licensee may submit to Licensor renewal application for another one (1) year. Licensor reserves the right to reject such application without cause.

Arg1	Relation	Arg2	Confidence
The license	is granted for	a period of one (1) year	0.9730538824171511

If Licensee fails to comply with the terms and conditions of this Agreement, this Agreement and Licensee's right and license to use Software will terminate immediately. Licensee may terminate this Agreement at any time by notifying Licensor. Upon the termination of this Agreement, Licensee must delete Software from its Clients and archives.

Arg1	Relation	Arg2	Confidence
Licensee	may terminate this Agreement at	any time	0.8037586010780811

This Agreement will be governed by the laws of Czech Republic, without reference to conflict of laws principles. Licensee agrees that any litigation relating to this Agreement may only be brought in, and will be subject to the jurisdiction of, any Court of Czech Republic.

Arg1	Relation	Arg2	Confidence
This Agreement	will be governed by	the laws of Czech Republic	0.9711855186851363

Titles are inserted for convenience only and will not affect in any way the meaning or interpretation of this Agreement. If any provision of this Agreement is held invalid, the remainder of this Agreement will continue in full force and effect. Either Licensor or Licensee may assign this Agreement in the case of a merger or sale of substantially all of its respective assets to another entity. This Agreement will be binding upon and will inure to the benefit of the parties, their successors and assigns.

Arg1	Relation	Arg2	Confidence
Titles	are inserted for	convenience only	0.9254999139862493
This Agreement	will inure to	the benefit of the parties	0.8556357677504458

Appendix C

Definitions

"Licensor" means JetBrains s.r.o., having its principal place of business at Na hrebenech II 1718/10, Prague, 14700, Czech Republic, registered with Commercial Register kept by the Municipal Court of Prague, Section C, file 86211, ID.Nr.: 265 02 275.

"Licensee" means a student or an instructor.

"Client" means a computer device used by Licensee for running Software.

"Software" means any software program included in JetBrains Educational Program at https://www.jetbrains.com/student and any third party software programs that are owned and licensed pursuant to Section 5 of this Agreement by parties other than Licensor and that are either integrated with or made part of Software (collectively, "Third Party Software").

"JetBrains Account" means profile record on https://account.jetbrains.com,

Software contains a feature that will automatically disable Software upon expiration of Evaluation Period.

During the term of this Agreement (as defined in Section 11(a) below), Licensor will provide generally available new versions of Software to Licensee free of charge and pursuant to the optional upgrade terms published on Licensor's website at www.jetbrains.com.

Titles are inserted for convenience only and will not affect in any way the meaning or interpretation of this Agreement.

Appendix D

RDF Graphs

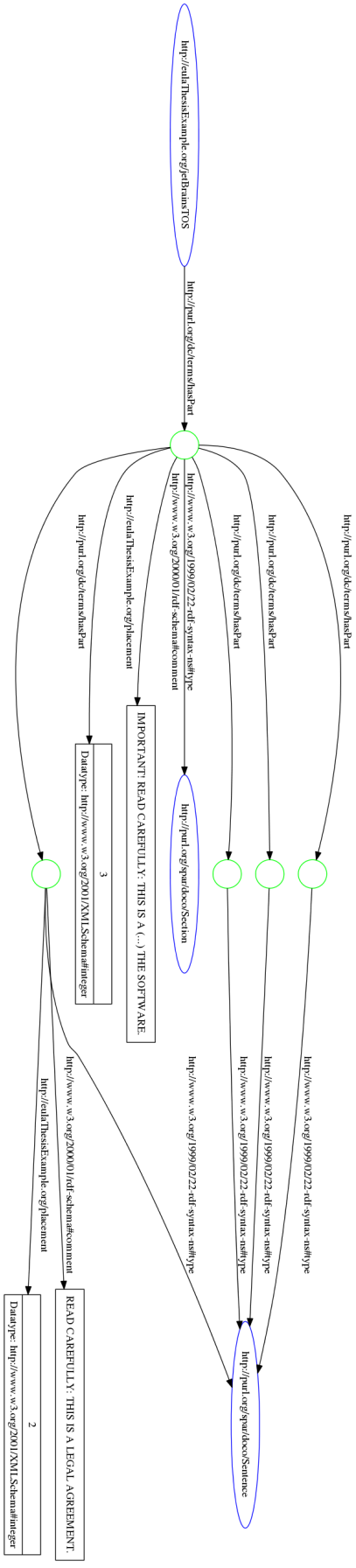

Figure 13: Graph of section with several sentences

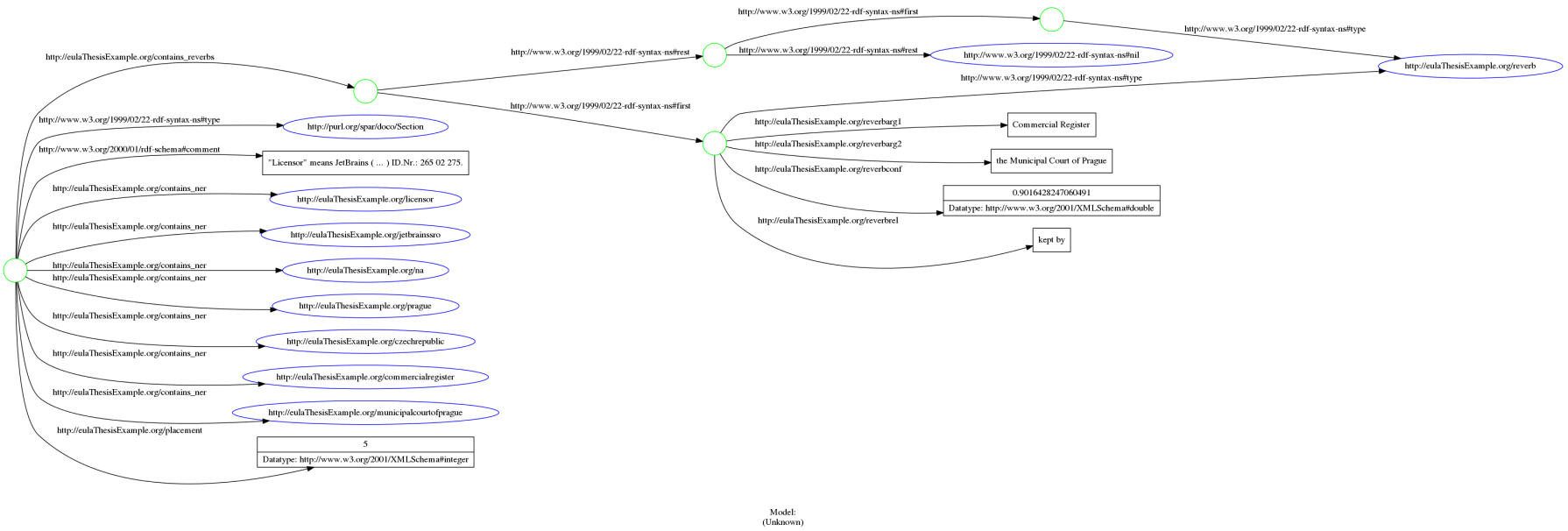

Figure 14: Named entities are put in a list.