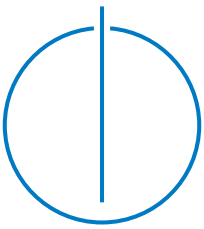
Fakultät für Informatik
Der Technischen Universität München

Bachelor's Thesis in Wirtschaftsinformatik

Ein Ansatz für eine semantische Informationsextraktion von Urteilen des Bundesgerichtshofs

Tobias Eyl



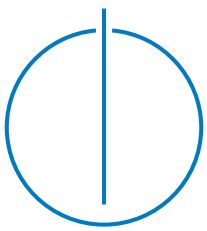
Fakultät für Informatik
Der Technischen Universität München

Bachelor's Thesis in Wirtschaftsinformatik

Ein Ansatz für eine semantische Informationsextraktion von Urteilen des Bundesgerichtshofs

An Approach for a Semantic Information Extraction of Decisions of the Federal Court of Justice

Erstbetreuer:	Prof. Dr. rer.nat. Florian Matthes
Zweitbetreuer:	M.Sc. Ingo Glaser
Tag der Einreichung:	16.09.2019



Erklärung

Ich versichere, dass ich diese Bachelor's Thesis selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel verwendet habe.

I assure the single handed composition of this bachelor's thesis only supported by declared resources.

Garching b. München, den 16. September 2019

Tobias Eyl

Zusammenfassung

Während die Menge an Dokumenten auch in der Rechtsbranche stetig steigt, fehlt es an geeigneten Möglichkeiten, diese Vielzahl an Dokumenten effizient nutzen zu können. Gerichtsentscheidungen, Fachliteratur der auch Gesetzestexte werden in fast ausschließlich in unstrukturierer Form veröffentlicht wie meistens als PDF.

Im Rahmen dieser Arbeit, wird ein Ansatz entwickelt, um aus einem Urteilstext die involvierten Parteien sowie deren rechtliche Beziehungen untereinander zu extrahieren und als Graph visuell darzustellen, um den Rechtsexperten eine schnellere Analyse der Urteile zu ermöglichen. Um dies zu erreichen, wird zu Beginn eine linguistische Analyse von Urteilsexten durchgeführt, um die sprachlichen Besonderheiten von Urteilen zu erfassen. Hierbei steht vor allem im Schwerpunkt, welche Schlüsselbegriffe gerichtlich verwendet werden, um bestimmte rechtliche Beziehungen auszudrücken und desweiteren, ob strukturelle Ähnlichkeiten vorhanden sind in Bezug auf wie einzelne rechtliche Konzepte dargestellt werden. Nachfolgend wird eine Ontologie entwickelt, deren Zweck die Modellierung der definierten rechtlichen Beziehungen ist. Diese bildet im Folgeden die Grundlage für die Annotation der semantischen Beziehungen im Rahmen der Erstellung eines Trainingsdatensatzes für spaCy's Dependency Parsers. Anschließend werden Extraktionsregeln auf Quelltext-Ebene definiert, um den Abhängigkeitsbaum zu traversieren und die erforderlichen Informationen zu extrahieren. Schließlich wird eine minimalistische Frontend-Applikation implementiert, die illustrativ das Endergebnis einer Informationsextraktion auf Grundlage des dargestellten Ansatzes darstellen wird.

Abstract

While the amount of documents is also continuously growing in the legal sector, there are lacking possibilities for effectively using these resources compared to other sectors like finance sector. Most legal documents like court decisions,

legal literature or the law texts itself are mostly published in plain text with little or without any additional metadata that might enable a more efficient usage.

Within the scope of this work, an approach is developed which extracts the legal parties and their legal relations among them and finally displays the extracted data in a graph-like form enabling the legal professional to conduct a more efficient research. In order to achieve this, at the beginning, a linguistic analysis will be performed to elicitate judgment specific linguistic features and subsequently to build a set of legal keywords indicating certain legal relations like a specific contractual agreement between two parties. Following, an ontology representing all the required semantic information within the sentences containing these keywords is built. In order to do this, a broad literature research is conducted and its results will be analyzed in the next. The developed ontology will then be implemented on the basis of a NLP-Technique called Dependency Parsing. For this, a model for spaCy's neuronal-network based dependency parser is trained which subsequently is applied to the respective section of judgments. On the basis of the semantic dependency model, extraction rules for every defined legal relation are implemented to enable the extraction of the information once the annotations has been set correctly. Finally, a visual representation will be implemented providing a well-arranged overview of the extracted semantic information.

The results of the evaluation show that this approach delivers remarkable high precision results despite being based on a relatively small set of training data with 38 training sentences and 25 sentences for evaluation.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Verzeichnis der Listings

1 Introduction

1.1 Motivation

While the digitization has already transformed many sectors and industries, the German legal sector is still working in a rather traditional, analog, way. There are various reasons for this finding like little incentives for law firms to make use of technological solutions to enhance the efficiency of their work processes as their revenue model is based upon the principle of billable hours.[1][2] Another reason is the lack of larger public data sources for legal documents. For example court decisions are rarely published and if they are, they are mostly published in an unstructured format like PDF or when published in XML, the used XML scheme provides only very little structural elements. At the same time, court decisions only consist of a limited variation regarding the used syllabus and sentence structure in comparison to other legal documents like contracts.[HSN04] One reason for this situation is that the structure and content of court decisions are regulated by law, for civil judgments, these are defined in § 313 ZPO (German Code of Civil Procedure). Based on these circumstances, the chances of a good suitability of court decisions to be used as a data set for further analysis and the retrieval of semantic information seem to be promising and form one of the reasons for using court decisions in this thesis project. Independent of regarding the work of lawyers, judges or legal scientists, a substantial part of legal work is research. Legal professionals have to analyze contracts, law texts, court decisions and many other legal documents. From the perspective of a legal professional, it is very time consuming to research and analyze judgments with regard to whether the judicial constellation in the judgment matches the one of the current matter. Especially, in more complicated cases with more than two or three parties and in which

[1] https://www.business.hsbc.uk\T1\guilsinglrightfinancing-investments-in-legal-tech-2018

[2] https://www.bucerius-education.de/fileadmin/content/pdf/studies_publications/Legal_Tech_Report_2016.pdf

the validity of many legal actions is relevant for the solution of the case, it often becomes challenging to keep track of the situation over a long judgment text. Currently, there exists no solutions providing summarized or aggregated versions of court decisions that could enable the legal professionals to conduct a more efficient research. There only exists solutions publishing judgments in full text with only basic keyword based search options.

When looking at the technical site, both the possibilities to extract relevant semantic information and their quality have remarkably risen over the last few years as Hirschberg et al.[HM15] only recently analyzed.

Facing an ever growing amount of data while at the same time, the tools or the technical procedures to build them are already available, the clients are currently putting more pressure on the law firms to make use of these possibilities to provide more efficient - and hereby cheaper - legal services.[2] As especially legal research is very time consuming from the perspective of a legal professional, further research in this area seems to be promising.

1.2 Structure

In this chapter, a short motivation was presentend. In chapter two, a desription of related work about how to model semantic information and which techniques are used for information extraction. In chapter three, the research method and the research questions are layed out. Afterwards, the actual system design is presented to be followed by the implementation in chapter 5. Finally, in chapter 6 the evaluation results are provided before the thesis concludes with a summary of the limitations and a look at future work.

2 Related Work

This chapter summarizes the main research papers on which this work is based on and refers to. Its structure follows the order of the research questions by firstly presenting existing approaches regarding how a metamodel might look like to enrich the content of unstructured legal documents with annotations to provide structured semantic information. Afterwards, research follows on how techniques of *Natural Language Processing* (NLP) in general can be used to extract such semantic information before, finally, existing approaches on how sorts of *Dependency Parsing* are used to extract semantic information.

2.1 Concepts for Modeling Semantic Metadata in Legal Documents

Before it is possible to extract semantic information from unstructured legal documents, one has to define a structured representation of the required semantic metadata necessary to aggregate the demanded semantic information. This structured representation is also called an *ontology*. According to Wyner, an ontology is an ëxplicit, formal, and general specification of a conceptualization of the properties of and relations between objects in a given domain".[Wy08] This means it is necessary to define an abstract concept for a certain demanded information within the application domain as well as necessary types of metadata with which the raw text has to be annotated in the next step. Metadata in general can be grouped in categories like administrative metadata, provenance metadata, structural metadata or finally semantic metadata. [Sl] For developing an approach to extract the involved parties and their legal relations among each other, only semantic metadata is relevant within this work.

For developing domain specific concepts, there already exists a respective amount of research with regard to the legal domain. The Nomos framework extends GaiusT and its key concepts are shown in Fig. 2. Sleimi et al. [Sl] developed a metadata model based on both frameworks for traffic laws of Luxembourg. Since both, GaiusT and Nomos, are intended to serve for different types of legal documents and even law systems, neither of them directly fits as a basis for this work. Due to the lack of a fitting concept for German legal documents, and especially court decisions, one goal of this work is to develop a metamodel for legal concepts fitting to the section Tatbestand"within court decisions of the Federal Court of Justice in Germany making use of the approaches followed by the mentioned existing solutions.

2.2 Semantic Information Extraction

The term Semantic Information Extraction describes a wide field of work and can be divided in several categories. Jurafsky et al. [JM09] categorize the extraction if semantic information in *named entity recognition* (NER), *relation extraction* and *event extraction* while event extraction itself is further split in the categories *temporal expression*, *temporal normalization* and *template filling*.

NER describes the process of annotating proper names with a term describing the kind of the proper name. [RN203]. An example for these proper names could be names of companies and persons or also domain specific terms like protein names [BCDF14]. While NER is also actively used in the legal domain, it will not be used for the research within this work. This thesis will only concentrate on the categories of relation and event extraction.

The term relation extraction can either describe the extraction of semantic relations between two named entities or also between two text tokens each representing an entity from a domain specific ontology. [JM09] As described in 4.1, in this work a domain specific ontology is used to annotate certain entities. For each of defined semantic relation, extraction rules are developed.

Within the field of NLP, implementing a structured way of extracting the domain specific entities and the semantic relations among them requires a certain sort of parsing method as a basis to develop a set of rules (grammar) for

extracting semantic metadata like entities or relations. The two most common ones are shortly explained in the following paragraphs, *Constituency Parsing* and *Dependency Parsing*.

2.2.1 Constituency Parsing

According to Jurafsky et al. [JM09] the term Constituency Parsing describes the task of recognizing a sentence and assigning a syntactic structure to itänd is thereby also called *Syntactical Parsing*. This means the sentence is split in several predefined units, the constituents. The definition of a sentence's syntax is done in a declarative formal way, mostly by using a *Context-free Grammar* (CFG). CFGs represent a class of a formal grammar that is not only applied within the field of NLP but also for modeling computer languages. [JLM] As this set of formal rules only describe the structure a sentence might have, no implications are made concerning in which order the rules have to be applied. Therefor, constituency parsing is considered to be an intermediate step for a later step, the *Semantic Parsing*. Bhatia et al. [Bh] describe in their research an approach for extracting regulated information types from privacy policies based on a domain specific ontology representing the constituents of the sentence. For each of these constituents, a grammar for automatically finding and extracting them in other privacy policies has been developed. The evaluation conducted by Evans et al. [Ev] shows they reached, based on a data set of 30 policies, an average precision of 0.72 and an average recall of 0.74 compared to the pairs identified by analysts. Concerning court decisions, Wyner et al. [Wy10] discussed approaches how to extract legal arguments from judgments of the the European Court of Human Rights based on constituency parsing and a context-free grammar in specific, and also in general, how ontologies and NLP might be suitable to identify semantic information like case factors and participant roles.

2.2.2 Dependency Parsing

Opposed to constituency parsing, syntactical structures do not play a role for Dependency Parsing and so a sentence's only syntactical structure is the order of the single words itself. The formal grammar, dependeny parsing is based on,

consists of a set of grammatical functions describing the directed binary grammatical relations among the words. [JM09, chapter 13] Today, there already exists a de-facto standard for these grammatical annotations, the *Universal Dependencies* project. [Nib] As explained more detailed in section 4.2.2, resulting its benefits, dependency parsing brings with it [JM09], it is widely used across several domains for relation extraction from natural language.[ZCL] Afzal et al. [AMF] use dependency trees to represent relations between named entities and subsequently train a machine learning model based on these trees for an unsupervised relation extraction enabling the automatic generaton of multiple-choice questions. Especially, in the biotech domain extensive research is conducted about that Shahab provides a compressed overview. [Sh17] With regard to the legal domain, Dell'Orletta et al. [DF12] generally research the need for an adaption of dependeny parsing approaches to the a specific domain, like the legal domain is. Based on the findings Gildea presented in research [Gi], according to which the quality of results dependency parsing brings on texts different from the texts the dependency parser has been trained on, Dell'Orletta et al. propose basic and consistent criteria in respect to task definition and evalution. Sleimi et al. [Sl] developed a dependency grammar in combination with a grammar for constiuents from which each of them represents a defined legal concept. The grammatical dependencies are customized to the special legal language and is used to extract the found legal concepts witin a statement of the traffic laws of Luxembourg.

3 Research Method

3.1 Research Questions

A first objective of this thesis is to investigate how an ontology might look like that provides a structured representation for sentences within the section Tatbestandöf judgments of the German Federal Court of Justice that describe the undisputed legal relations among the involved parties. On the basis of the developed ontology, research follows regarding how NLP-techniques can be leveraged to automatically extract the necessary sentences as well as the relevant parts of them. For testing the found concepts questions, a prototype will be implemented based on the NLP library *spaCy*. All the used spacy modules as well as all other technologies will be fully explained in chapter 4.

All the stated research goals are split into the following three research questions:

1. How an ontology for representing semantic information of court decisions can look like?
2. How the key information of a court decision can automatically be extracted using NLP?
3. How a prototype for a semantic analysis of court decisions can be implemented?

3.2 Research Method

At the beginning of the work, a broad literature review was performed covering the following parts to obtain the theoretic concepts as these build the basis for the following concept development and implementation:

1. Linguistic Analysis of a set of German judgments of the FCJ concerning potential linguistic specialties which might be valuable for developing an ontology,
2. Literature review of existing research on metamodels and ontologies for (German) legal documents,
3. Literature review of existing research on existing approaches for modeling legal sentences,
4. Research on existing technical tools, libraries in the field of NLP which might be used to implement a prototype.

The following evaluation of the prototype's results consists of a qualitative and quantitative part. For the qualitative evaluation, a legal expert manually analyzed a set of court decisions with respect to the defined type of semantic information to be automatically extracted. These results were consequently compared with the results generated by the prototype. With regard to the quantitative evaluation, the results of common statistical methods tailored to the used NLP-techniques were produced and interpreted.

4 System Design

This chapter introduces important underlying concepts of the implementation as well as an overview of the actual system architecture before the prototypical implementation is explained in the next chapter. In section 4.1 different types of ontologies are described which have already been used in related research work. Next to this, also existing metamodels are shown which have been developed with the intention to harmonize the field of legal ontologies. In section 4.2 useful existing parsing methods are discussed which enable to extract the semantic information described by the annotations that have been applied to the text in accordance to the ontology. In the last part of this chapter, section 4.3, the prototype's architecture is layed out.

4.1 Types of Ontologies and Metamodels

In general, an ontology is an explicit, formal and general specification of a concept describing the objects and structural relations between those objects in a certain domain.[Wy08] As a result of this, ontologies used for specific solution might largely differ from those used in other implementations, although there might not be a reason for this. To prevent a too large variety among ontologies used in the legal domain with the aim to reach a better comprehensibility and quality for every of these ontologies, there exist solutions that are considered to serve as a metamodel for developing legal ontologies. The three major ones are described in the following sections. As commonly defined, a metamodel defines the valid element types a model can consist of, and how the elements can be related to each other.[Se03] In other words, metamodels provide a framework for models with that a model can be prooved valid.

4.1.1 Types of Metamodels for Modeling Semantic Metadata in the Legal Domain

4.1.1.1 GaiusT

One of the most sophisticated metamodels for models of semantic metadata occuring within legal documents, is the *GaiusT tool*.[Ze15] It is based on and enhances the *Cerno* information extraction framework developed by Kiyavitskaya et al..[Ki] The concepts used by GaiusT follow the approach of the *Deontoic Logic*. According to the definition of the *Stanford Encyclopedia of Philosophy*, deontic logic is a type of symbolic logic that consists of the notions shown in Table 4.1 which are describing "what follows from what".[3]

Concept	Concept type and its indicators
Right	May, can, could, permit, to have a right, should be able to
Anti-right	Does not have a right to
Obligation	Must, requires, should, will, would, which is charged with, may not, can not, must not
Anti-obligation	Is not required, does not restrict, does not require

Tabelle 4.1: Syntactic for deontic concepts[Ze15]

Based on this, GaiusT focuses on the following legal concepts: *Actors*, *prescribed behaviors*, *resources*, *actions* and *constraints*. The complete conceptual model can be seen in Fig. 4.1.

Looking at top left corner of Fig. 4.1, one can recognize, the concept "Goal". GaiusT is one of a so called *goal-oriented* framework. A goal-oriented framework takes a project's goals and objectives as the focus of the whole model and hereby enable a practice-oriented design method.[GAP]

4.1.1.2 Nomos

Nomos provides an even more goal-oriented approach that serves as a metamodel for models of semantic metadata within legal documents.[JM09] It focuses on five main concepts: roles, duties, rights, situations and associations.[Si]

[3] `https://plato.stanford.edu/entries/logic-deontic/`

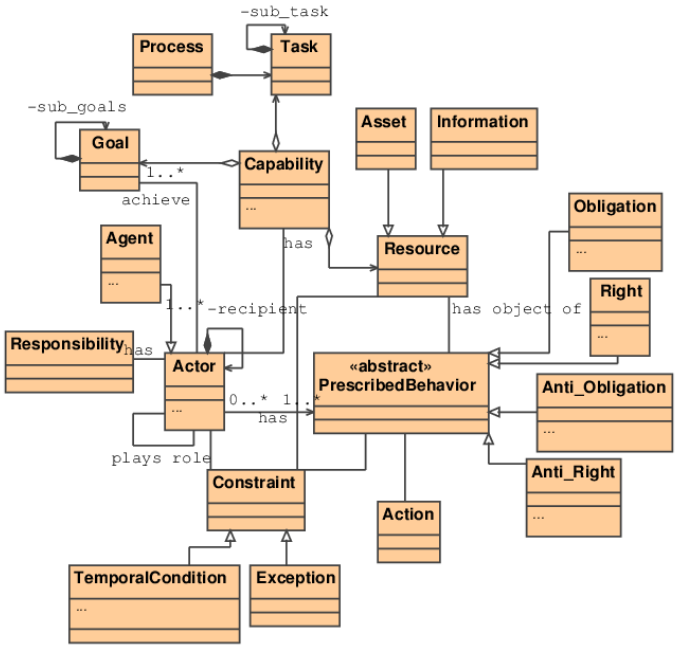


Abbildung 4.1: The GaiusT conceptual model[Ze15]

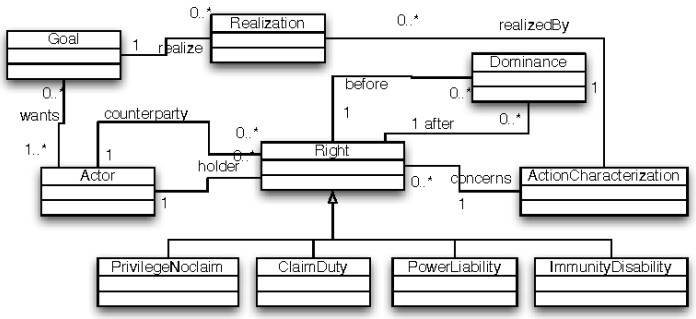


Abbildung 4.2: The Nomos metamodel[Si]

Similiar to GaiusT, also with the Nomos framework, one can see "Goalïn the upper left corner of Fig. 4.2 as the root of the entire metamodel. In addition to GaiusT, several subtypes of rights were introduced next to the concept ÄctionCharacterization". In its newest version, *NomosT*, that bases on both, Nomos and GaiusT, introduces additional helper concepts to enable an automatic extraction of the five key concepts.[Ze] These additional concepts are: *Actor*, *Resource*, *Exception*, *Antecedent*, *Consequent*, *PositiveVerb*, *NegativeVerb*, *Holder* and *Beneficiary*.

4.1.1.3 LegalGRL

In contrast to the both GaiusT-based metamodels mentioned before, the *Legal Goal-oriented Reuirements Langugage*(LegalGRL), is based on the *Hohfeldian System*. While Deontic Logic is tailored around two concepts, permissions and obligations, the Hohfeldian System is built around eight types of legal rights.[Ho17] These are *rights* (or claims), *privileges*, *powers*, *immunities* and their respective opposites *no-rights*, *duties*, *disabilities* and *liabilites*. Now, for developing a Legal GRL model, one has to categorize each statement of the legal document based on these Hohfeldian concepts.[Ho17] Additionally, for a Legal GRL model, also the concepts *subject*, *verb*, *actions*, *preconditions*, *exceptions* and *cross-references* are introduced. Afterwards, these conducted annotations have to be transformed into deontic goals of type *Permission* and *Obligation*. Usually, these steps have to be applied iteratively.

4.1.2 Types of Legal Ontologies

Building upon the ideas of the mentioned metamodels, also several legal ontologies exist.[BVW04] All of these ontologies served as general ideas for answering the first research question of how an ontology for German court decisions might look like. However, there will be no detailed discussion of these ontologies since the scope of the thesis is limited and, hence, only a small part of judgments of the German Federal Court of Justice can be considered as data set. Therefor, the development of a complete ontology for German civil court decisions is not a goal of this work.

4.1.2.1 OWL

The *Ontology Web Language*(OWL) is a machine-readable ontology developed to serve as a common basis for ontologies within different application domains.[4] It is part of the *Semantic Web* development. Wyner [Wy08] uses OWL to implement an ontology for legal case-based reasoning. The ontology consists of six main classes. All of them may have several subclasses. According to the

[4] `https://www.w3.org/OWL/`

OWL definition, all classes also can be built of a sum of subclasses just as like they may have specified conditions and properties that have to be set.

4.1.2.2 LKIF

LKIF[5], the *Legal Knowledge Interchange Format*, has been developed as part of the *Estrella* project with the goal to unify previously existing legal ontologies.[Ho] By using the standards OWL-DL and SWRL, which are both part of the family of Semantic Web standards, LKIF ensures to be also compliant to Semantic Web standards. LKIF defines over 50 terms while its top structure is based on the CYC[6] ontology. At the very basic level, LKIF is separated by three types of layers: the *top* level, the *intentional* level and the *legal* level. Fig. 4.3 shows the concepts defined by the top layer. While this top layer might appear self-explaining, it is crucial for the fundamental parts of any legal concept like the location, time, parthood or change.

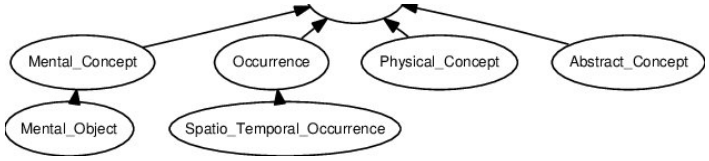


Abbildung 4.3: The LKIF top layer[Ho]

Fig. 4.4 shows the part of the LKIF ontology concerning actions. This part is particularly interesting as actions represent any legal transaction with all its associations and serve either as the basis or are partly reused for the ontologies described section 4.1.2.3 and 4.1.2.4.

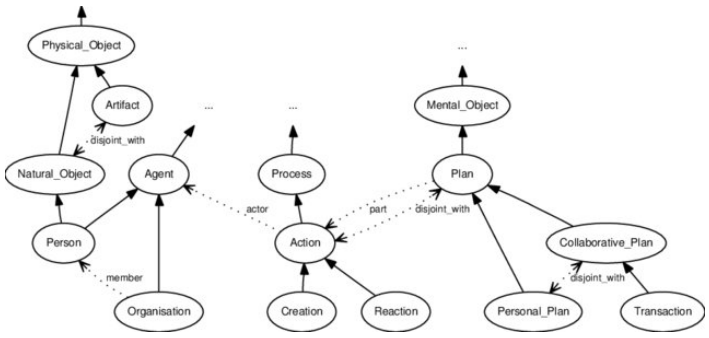


Abbildung 4.4: The LKIF concepts: Actions, agents and organisations[Ho]

[5] http://www.estrellaproject.org/lkif-core/
[6] http://www.cyc.com

4.1.2.3 LegalRuleML

Another major ontology for the legal domain is *LegalRuleML*[7]. LegalRuleML is an extension of the general and widely known *RuleML*[8] web-rule specification. While incorporating the in section 4.1.1.1 explained deontic logic, LegalRuleML provides a set of formal descriptions for legal norms, policies or reasoning.[At15] These provided structures are technically represented by a XML-schema flavored with the XML-language *RelaxNG*[9]. As LegalRuleML mainly focuses on modeling legal norms, its concept details are omitted. However, it should be noted that LegalRuleML is indeed one major player in the field metamodels for legal documents. Due to its high detail grade and extensiveness, it nonthelees provides one with useful ideas also for other types of legal documents like its listing of a good metamodel's characteristics which is also one of the reasons it is mentioned in this section. So it is emphasized to keep the number of defined legal concepts as low as possible and to try to use pattern whenever it is possible to ensure the concepts are independent and thereby can be combined to model larger concepts.

4.1.2.4 UFO-L

One of the most recent works is the one by Griffo et al.[GAG], that presents an ontology specifically designed to model legal relations and by doing so, improving one of the major impediments of former legal ontologies. The ontology is based on the *Unified Foundational Ontology* (UFO) and thus is named *UFO-L(egal)*. UFO-L incorporates the in section 4.1.1.3 described Hohfeldian System and combines these concepts with the one developed by Alexy's[Al02], the *relational theory of constitutional rights*. In addition to the Hohfeldian System, Alexy's theory provides the concept of a possibility to deny a legal relation's object, for example a right. Thus, the concept of an *omission* brings a major benefit, especially for modeling legal relations. While Griffo et al. use the specific example of e-mail service contracts which often contain a duty to *omit* sending the same message to a large number of recipients, the general thought beyond is also valid with regard to research within this work. Also in the German civil law system exist duties to omit a certain action like within

[7] `https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=legalruleml`
[8] `http://wiki.ruleml.org/index.php/RuleML_Home`
[9] `https://relaxng.org/`

the field of the German copyright law[10] (UrhG), for example § 97 Sec. 1 UrhG grants a person whose copyright was infringed a claim against the infringing person to omit the action violating the copyright in the future. But of course, this applies also to individual contracts signed under the German Civil Law system as these contracts might also contain all different kinds of duties to omit. One of UFO-L's key elements is the use of a *legal relator*. A legal relator aggregates externally dependent *legal moments*. A legal moment is the super class for one of UFO-L's legal core concepts, a *right*, *duty*, *no-right*, *permission*, *legal power*, *legal subjection*, *disability* or *immunity*. Fig. 4.5 shows a part of UFO-L. In this figure, one can also recognize that a legal relator is either a *simple* or a *complex* one.

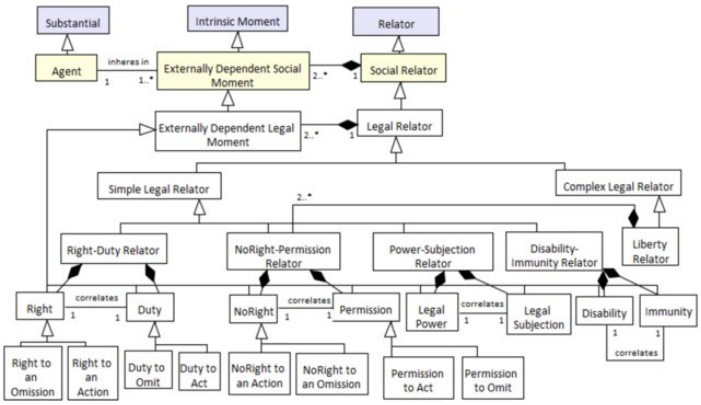


Abbildung 4.5: Main part of UFO-L[GAG]

4.1.3 Discussion

As the scope of this work is only about extracting involved parties and the basic type of the legal relations among them from court decisions, all of the mentioned ontologies are in fact too sohisticated for the limited application area. However, these ontologies deliver useful ideas for the development of the ontology used for the purpose of this work. So, the ontology of this work strives to follow the principles of the existing ones, in particular of the ones based on the Hohfeldian system of law and the ones based on OWL an there by also UFO-L.

A general result is that metamodels and ontologies following the principle of deontic logic fit more for tasks about modeling legal norms as deontic rules

[10] `https://www.gesetze-im-internet.de/urhg/`

only consider rules and obligations but no external dependencies like actors, in particular persons, but also deontic rules do not consider bilateral relations like UFO-L does.

Sleimi et al. [JM09] developed a narrow metamodel for modeling traffic laws of Luxembourg which can be seen in Fig. 4.6. They only distinguish between models on a statement-level and such on a phrase-level. In this context, a statement represents a group of sentences while a phrase in fact represents one sentence. This work is highly oriented to the structure of this metamodel and the subsequent ontologies. As for the scope of this work, only single phrases describing a legal relation are considered, thus especially the phrase-level metamodel serves as a reference for the model developed in this work.

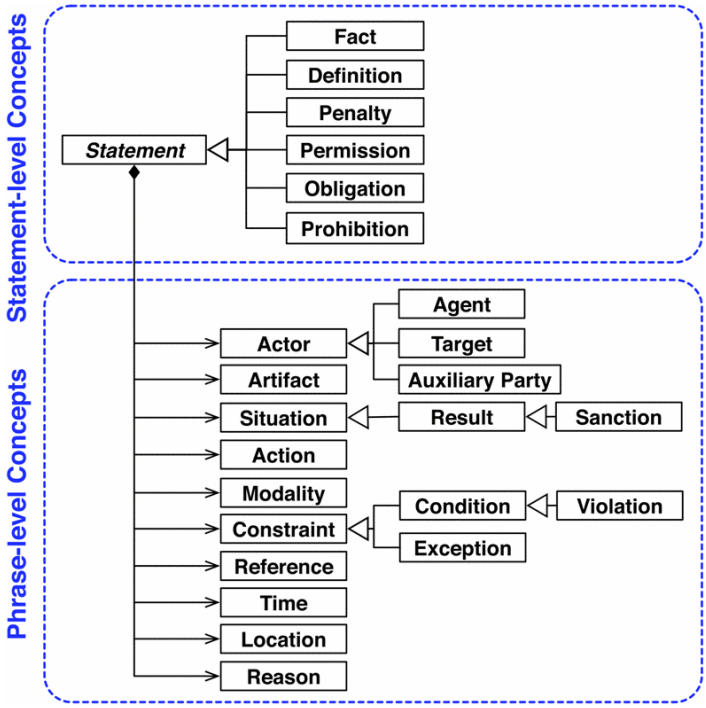


Abbildung 4.6: Metamodel for luxembourgian traffic laws from: [JM09]

4.2 Underlying NLP-Concepts for Information Extraction

When talking about information extraction, there exists a large variety of technical methods to be distinguished. In fact, the term information extraction itself is not a sharp technical definition but rather commonly describes a

pipeline containing several steps to transform unstructured text into a structured representation of the text by applying NLP-techniques to finally be able to filter out the required information. The most common NLP-techniques for the purpose of information extraction are *Named Entity Recognition*, *Part-of-Speech Tagging* and different parsing techniques like in particular *Constituency Parsing* and *Dependency Parsing*. This work is solely focused on elaborating parsing techniques, researching which one is the most suitable one for applying the previously developed ontology for extracting the involved legal parties and the legal relations among them.

4.2.1 Constituency Parsing

The current state of research mainly distinguishes between two large groups of parsing techniques, Constituency Parsing nad Dependeny Parsing. Basically, Constituency Parsing consists of two main parts. The first one is about defining a grammar to define which syntactic components exist and also which component consists of other syntactic components. The other main part of constituency parsing is defining an algorithm to specify how, meaning in which order, the syntactic components are processed. Hence, constituency parsing gives an unstructured sentence a syntactic syntax and thereby is also often called *Syntactic Parsing*.

Context-free grammar The most common grammar type used to define the syntactic parts of a sentence is a context-free grammar(CFG). CFGs use declarative rules to specify which words or symbols of the original text build a certain syntactical compoment. Therefor, it necessary to define a fixed amount of words that serves as a dictionary to look up the associated s yntactic component. The symbols of CFG can be grouped in two types: *Terminals* and *Non-terminals*. Terminals of the grammatic rules match the actual words used in the original language of the text that is up to be analyzed. On the other side, non-terminals are self-defined names for the syntactic components. After a non-terminal, either a terminal may follow or another non-terminal. Thus, a CFG is a type of grammar that allows components to be built out of other components. Every CFG starts with a unique start symbol. While the syntactical components defined by a CFG often represent grammatical structures of

the respective text language, the concept of CFGs can also be used to formuate custom components modeling sentence parts specific to a certain application domain. Wyner et al.[Wy10] developed a CFG to provide a formal description of the structure of a legal argument how it is mainly used court decisions of the European Court of Human Rights (ECHR). The formal description of this grammar and its explanation can be seen in Fig. 4.7.

$$T \Rightarrow A^+D \quad (1)$$
$$A \Rightarrow \{A^+C|A^*CnP^+|Cns|A^*sr_cC|P^+\} \quad (2)$$
$$D \Rightarrow r_cf\{v_cs|.\}^+ \quad (3)$$
$$P \Rightarrow \{P_{verbP}|P_{art}|PP_{sup}|PP_{ag}|sP_{sup}|sP_{ag}\} \quad (4)$$
$$P_{verbP} = sv_Ps \quad (5)$$
$$P_{art} = sr_{art}s \quad (6)$$
$$P_{sup} = \{r_s\}\{s|P_{verbP}|P_{art}|P_{sup}|P_{ag}\} \quad (7)$$
$$P_{ag} = \{r_a\}\{s|P_{verbP}|P_{art}|P_{sup}|P_{ag}\} \quad (8)$$
$$C = \{r_c|r_s\}\{s|C|P_{verbP}\} \quad (9)$$
$$C = s^*v_cs \quad (10)$$

(a) Formal CFG description

T	General argumentative structure of legal case
A	Argumentative structure that leads to a final decision of the factfinder $A = \{a_i, ..., a_j\}$, each a_i is an argument from the argumentative structure
D	The final decision of the factfinder $D = \{d_i, ..., d_j\}$, each d_i is a sentence of the final decision
P	One or more premises $P = \{p_i, ..., p_j\}$, each p_i is a sentence classified as premise
C	Sentence with a conclusive meaning
n	Sentence, clause or word that indicates one or more premises will follow
r_c	Conclusive rhetorical marker (e.g. therefore, thus, ...)
r_s	Support rhetorical marker (e.g. moreover, furthermore, also, ...)
r_a	Contrast rhetorical marker (e.g. however, although, ...)
r_{art}	Article reference (e.g. terms of article, art. x para. x, ...)
v_p	Verb related to a premise (e.g. note, recall, state, ...)
v_c	Verb related to a conclusion (e.g. reject, dismiss, declare, ...)
f	The entity providing the argument (e.g. court, jury, commission, ...)
s	Sentence, clause or word different from the above symbols

(b) Explanation of CFG

Abbildung 4.7: CFG describing structure of a legal argument: [Wy10, Fig. 1 of]

While CFGs are the most used technique for defining syntactical structures in sentences, the *Cocke-Kasami-Younger* (CKY) algorithm[Co69, Yo67] is the standard method for creating a parsing tree out of the CFG, meaning the CKY algorithm controls in which order the CFG rules are applied. But as constituency parsing is only used as a related work useful to determine which parsing technique is the most suitable one for this work, it will not be explained in detail. While the quality and quantity of CFG rules can be quite high, a grammar stays a finite set of rules. However, the structure of natural language as it is used in practice also exceeds the scope of a complete CFG that models a whole language grammar, a so called *treebank*[Sa03]. As a consequence, the main painpoint of constituency parsing is ambiguity. In practice, it is possible that a certain partial set of the CFG rules perfectly matches the syntax of two sentences which semantic meaning, however, might be completely different. This is the reason why often CFGs are used in combination with some sort of a statistic method. Hence, for reducing the amount of ambiguity within a CFG, the CFG mostly gets extended to a *Probabilistic context-free grammar* (PCFG). In addition to a CFG, every rule is associated with a probability

that has been calculated by determining the occurence of a certain syntactical structure within an annotated training data set.[Xu11] At the end, the parse tree to be used out of the matching ones, is determined by the probability of the whole parse tree that was calculated by multiplying all the probabilities from the rules which has been applied for reaching all the non-terminals in the parse tree. But as PCFGs are not part of the chosen technical concept for the implementation, the functioning of a PCFG will not be further explained in greater detail.

4.2.2 Dependency Parsing

Contrary to Constituency Parsing, developing descriptive rules to define which group of words represent a certain syntax element is not a part of Dependency Parsing at all. Dependency Parsers uses the individual words or even tokens of a sentence itself and bases on directed, binary relations between two words. Every sentence has exactly one root element. Starting from this root word, all other relations are derived. These relations mostly describe a grammatical notion between the two words. One of the most intuitive examples is the grammatical subject of a sentence derived from the sentence's root element. The de-facto standard framework for syntactical dependency structures is called *Universal Dependencies*[11].

4.2.2.1 Semantic Role Labeling vs. Syntactical Grammar Functions

However, similiar to Constituency Parsing, it is possible to use domain specific semantic dependencies between the words instead of formal grammatical functions of the language.[Ni05] This concept is known as *Semantic Role Labeling*(SRL) and describes the process of finding and annotating so-called predicate argument structures in sentences with a type of semantic frame and role labels.[AL] As semantic annotations are modeling specific contextual relations, they are by design highly dependent on domain the text is from. Generally, one could validly argue that thereby using syntactic dependencies might be more promising to be used as dependency grammar as it is more likely syntactical relations can reused for several domains. To counter this disadvantage,

[11] `https://universaldependencies.org/`

Palmer et al.[PGK05] developed *PropositionBank*(PropBank), an annotated corpus of semantic roles that should serve as an additional layer of the *Penn Treebank*[12]. PropBank follows a verb-based approach, meaning it defines a set of semantic roles for each verb of the Penn Treebank. While the Penn Treebank is focused on the English language, the its backing general idea of being verb-focused serves as the basis of the ontology developed within this work and which explained in section 4.3.1.1. Additionally, while the less general structure of semantic role labeling might lead to more effort on annotating a larger amount of training sentences, this impediment does not directly apply to domain specific sentences. Li et al.[LZY] summarized the characteristics of domain specific sentences as follows:

1. a limited syllabus
2. word usage has patterns
3. little semantic ambiguities
4. frequently used domain jargon

Li et al. use these characteristics as the foundation for the development of a machine-learning based approach to automatically annotate domain specific sentences with semantic role labels. That these special features in particular can also be found in German legal documents has been shown by Busse[Bu98] and Hansen-Schirra et al.[HSN04]. More specifically, Hansen-Schirra et al. analyzed German court decisions. Concerning the sylabus used in court decisions, they built a reference corpus with jargon found across different types of legal documents and measures a value of 53.38%. With 38.70%, court decisions interestingly reached the lowest value, meaning court decisions provided the relatively smallest syllabus of all legal document types compared to the reference corpus. Basically one can say, by using semantic labels it is intended to reach better precision results with a lower amount of training data. This is further illustrated in section 5.1.2.1.

Although all of the following algorithms of dependency parsers originally were designed focused on dependency grammars representing syntactical strucutures, they can also be applied to work on semantic dependencies as long as the following general formal requirements of a *Dependency Grammar* are met:

[12] `https://catalog.ldc.upenn.edu/LDC99T42`

1. In every sentence is exactly one sentence token functioning as the single root node, meaning a sentence token without any incoming arcs.
2. Besides the root node, each node has exactly one incoming arc.
3. There exists a unique path from the root token to each other token of the sentence.

Every dependency grammar fulfilling these requirements, produces a connected and directed graph with one distinct root. Next to these requirements there exists another constraint for dependency grammars that is not strictly a mandatory one like three criteria mentioned before, but rather a fourth optional one - the *Projectivity* criteria. A dependency tree is considered to be projective when all its arcs are projective. An arc between the head node and its dependent is projective when all the nodes between the head and its dependent also can be reached from the head node. While also all non-projective trees can represent completely valid dependency trees, projective dependency trees enable to formulate more efficient algorithms for information extraction on the basis of context-free grammars. In the following two main concepts for dependency parsers are introduced.

4.2.2.2 Arc-factored Dependency Parsing

Arc-factored dependency parsers are closely related to constituent parsers despite both types largely differs in the information both parse trees do model. As already shortly mentioned in 4.2.2, it is possible to develop a context-free grammar for projective dependency trees. Arc-factored dependency parsers make use of this rule and also bases on approaches following a dynamic programming style. All algorithms of this family have in common that they all implement bottom-up approaches to calculate the propability of a dependency tree by building the sum of all the propabilities the individual arcs have. The propability of an individual arc is calculated by building the sum of the weighted *features* of an arc. As an arc's feature can be chosen the criteria whether the head is a noun or also a combination like whether the head is a noun and the dependent is a verb or also a criteria like the length of an arc. The following formulas show the according formal mathematical definition for a tree t and an arc a with its features f and related weight w.

$$prob(t) = prob(a_1) + ... + prob(a_n)$$
$$prob(a) = f_1 w_1 + ... + f_n w_n$$

Collins Algorithm The Collins algorithm[Co03] is the first algorithm that is based on the arc-factored scoring model. Making use of the similiarity between finding the constituent parse tree with the hightest probability and finding the most probable dependency tree, the Collins algorithm is in fact an extension of the CYK-algorithm. Due to its complexity class of $O(n^5)$, the Collins algorithm itself is not used in practice but only serves as the basis for more efficient algorithms, today. As the algorithm is basically an adaption of the CYK algorithm to dependency parsing and the CYK algorithm can be considered well-known, for the sake of the brievity, the algorithm is not explained in further detail.

Eisner's algorithm The most known and used algortihm for arc-factored dependcy parsing is the one developed by Eisner[Ei]. The Eisner algorithm improves the Collins algorithm by reducing its complexity to n^3. This is accomplished on the back of this idea: For drawing a left-to-right arc - or in other words - connecting two subtrees, the Collins algorithm uses five position variables to add this arc in one step - besides of the subtree's heads, also the start and end positions of the left subtree's interval are considered. Now, the Eisner's algorithm only uses 3 position variables and splits adding an arc in three steps. When thinking of a subtree as a triangle, in the first step the Eisner algorithm both subtrees in half for now only working with right part of the left subtree (LR) and left part of the right subtree (RL). By doing so, only the both head positions and the end position l' of the left subtree's interval needs to be known. This results of the fact that only projective dependency trees are considered and, thus, the start position of the right subtree's interval is defined by the next position after l', $l' + 1$. Depending on which of both subtrees functions as the head of the other one, only the remaining side of the dependent subtree needs to be parsed if there are still missing dependents within this subtree. Although the Eisner algorithm provides the basis for today's arc-factored oriented parsing solutions in practice, also the details of this algorithm are omitted in this thesis since the implementation finally bases

on a transition-based parser and the findings relating to arc-factored oriented approaches only served as foundation for decision making.

4.2.2.3 Transition-based Dependency Parsing

Transition-based dependency parsers follow a different approach compared to the ones based on an arc-factored model in order to reduce the runtime complexity and thereby make algorithms usable for larger sentences and texts in practice.

The arc-standard algorithm Since arc-factored algorithms calculate a tree's probability based on the arcs' probabilities themselves, for determining the most probable tree, each of these algorithms has to be applied simultaneously on many different trees. Consequently, by applying an algorithm from the greedy family (transition-based)only one tree needs to be built since the algorithm determines the next arc with the highest probability based on a defined decision method. This leads by design to a linear runtime complexity within $O(n)$. For determining which transition follows, different machine-learning techniques are considered. Such techniques are *decision trees*, *support-vector machines (SVM)* or *memory-based learning*. As only explaining this part would take a large amount of space, the scope will be reduced to the ones actually used by the implementation solution which will follow in the section XXX. One of the main representatives of transition-based parsers for projective dependency trees is the *Arc-standard algorithm* by Nivre[Nia]. As mentioned above, the arc-standard algorithm is a greedy algorithm and is in fact a modification of the well-known *Shift-Reduce-algorithm* for context-free grammars. This means, it follows a simple left-to-right bottom-up strategy for parsing list of tokens as input. As it is the case with the shift-reduce-algorithm, also the arc-standard algorithms works with a buffer, a stack and a proper data structure to store the current state of the so far constructed dependency graph. Next to this, there exist three valid operations: *Shift*, *Left-Reduce*, *Right-Reduce*. The shift operation pops the next input token from the buffer and pushes it to the stack. A left-reduce operation takes the most top token t_1 and second most top token t_2 of the stack, and adds an arc from t_2 as the *head* to t_1 as the *dependent* and reduces both to t_2. The right-reduce operation works accordingly with head

and dependent element swapped. By adding the new relation to the dependency graph configuration, the graph is transisted to the next state. Within the initial state, the stack and the current dependency graph are empty while all the words are stored in the buffer. The parser terminates when the buffer is empty and the stack only contains a single word. When the finally constructed dependency graph meets the requirements defined in section 4.2.2 and thereby is in fact a valid projective depedency tree, the algorith terminates successfully, otherwise throws an error indicating an invalid string as input. For the sake of brievity within this thesis, a formal description of the algorithm is omitted at this part of the work as the *arc-eager algorithm* described in the next paragraph isan extension of the arc-standard algorithm whose description contains a formal writing of the algorithm that incorporates the arc-standard algorithm.

The arc-eager algorithm [Nia] As mentioned above, the arc-standard algorithm strictly follows a bottum-up strategy. While this might not be seen as a problem when theoretically assuming that one intends to build a complete dependency tree for a certain sentence at everytime, it however becomes one when only some of the dependencies are sufficient for providing the wanted semantic information. To resolve this issue in practice and provide a practically, not theoretically more efficient transition-based algorithm, Nivre[RN504] combined this bottom-up approach with top-down concepts to reach a more practicable variant of incrementality. In order to achieve this, preconditions are added to the operations *left-arc* and *reduce*.

1. The next token from the buffer cannot already be a dependent, meaning cannot already be in the stack

2. Tokens can be temporarly stored on the stack for later processing

The Left-Arc operation is basically the same as the left-reduce operation of the arc-standard algorithm. However, instead of using the two tokens on top of the stack only the first token on top of the stack and the next input token from the buffer is used. A larger change occured from the former right-reduce operation to the current right-arc operation. As illustrated in Fig. 4.8, the operation for adding arcs to dependents on the right is the one that directly affects the functioning for determining whether it is required to add the dependencies for all the nodes on a lower level before adding the arc to the upper node. In

order to allow such partially completed dependency trees, the reduce operation is implemented in a standalone variant within the arc-eager algorithm. By doing this, it is now possible to delay the reduce operation for arbitrary many operations and thereby create long chains of right-dependent tokens.

Initialization	$\langle \mathbf{nil}, W, \emptyset \rangle$	
Termination	$\langle S, \mathbf{nil}, A \rangle$	
Left-Arc	$\langle w_i \vert S, w_j \vert I, A \rangle \rightarrow \langle S, w_j \vert I, A \cup \{(w_j, w_i)\} \rangle$	$\neg \exists w_k (w_k, w_i) \in A$
Right-Arc	$\langle w_i \vert S, w_j \vert I, A \rangle \rightarrow \langle w_j \vert w_i \vert S, I, A \cup \{(w_i, w_j)\} \rangle$	$\neg \exists w_k (w_k, w_j) \in A$
Reduce	$\langle w_i \vert S, I, A \rangle \rightarrow \langle S, I, A \rangle$	$\exists w_j (w_j, w_i) \in A$
Shift	$\langle S, w_i \vert I, A \rangle \rightarrow \langle w_i \vert S, I, A \rangle$	

Abbildung 4.8: Transitions of arc-eager dependency parsing: [RN504, Fig. 5 of]

Enabling failure correction The above described concepts for arc-eager dependency parsing only describe the fundamental basis. In addition to that, there have been applied a number of modifications which finally lead to the underlying concept of the dependency parser provided by *spaCy*[13], the library used to conduct the prototypical implementation.

On the way to the most recent used dependency parser the first modification is the introduction of additional *non-monotonic* operations proposed by Honnibal et al.[HGJ]. Non-monotonicity's counterpart, *monotonicity*, refers to the single head requirement explained in section 4.2.2. It describes the feature of enforcing that once an action has been performed, all the following ones have to be compliant with it.[HGJ] Making every head assignment binding might be favorable with regard to the simplicity of an algorithm, however, this benefit comes at the price of not being able to correct false assignments. Since even the best machine learning technique cannot guarantee a 100 % correctness, false head assignments at one place within the sentence will at least result in a wrong annotation, in the worst case scenario, it will even result in an error as the sentence might not be parsable at a later point anymore. By allowing also some non-monotonic operations, the algorithm is potentially able to recover from a previously conducted wrong dependency annotation. In order to be able to recover from a failure, a set of possible wrong annotations must be provided

[13] `https://spacy.io`

as part of the training data set for building the parser's gold tree. Further explaining these detailed adaptions would require a considerable amount of space and thereby this is omitted and it is only refered to the original paper [HGJ].

Nivre et al.[NFG14] go a step further by introducing the new operation **Unshift** and are thereby developing a way to guarantee that the parser will always terminate with a valid - not necessarly the gold standard fitting - projective dependency tree. According to the general constraints of transition-based dependency parsers, the parser terminates when the buffer is empty and there is only one element left on the stack. So far, in other case the parser would not terminate or finally propagate an error. Now, if the buffer is empty and there is more than one element left on the stack, this parser variation deterministically chooses between a Reduce and the new Unshift operation. If the stack's top element already has a head, reduce is applied as usual. But once there is an element left with a head the unshift operation is chosen. The rest equally functions as the previous version. As long as there is at least one element in the buffer, the same statistical model is used to make a non-deterministic choice between Right-Arc and Left-Arc or Reduce.

Honnibal and Johnson[HJ] finally combines the two previously mentioned modifications to build a non-monotonic dependency parser that guarantees for a higher percentage of inputs to terminate with a valid projective dependency tree. In comparison to the directed attachment accuracy of the base version of a monotonic arc-eager parser, they reach with this combined approach 91.85%. This result correlates with 6.25% of error reduction. At its core, the combination of the two approaches consists of integrating the Unshift operation into the first non-monotonic dependency parser by Honnibal et al.[HGJ] which uses a statistical model to determine the next operation while the original parser by Nivre et al.[NFG14] makes use of a deterministic approach for the unshift operation.

4.2.3 Discussion

When deciding between using constituency parsing and dependency parsing for information extraction, formost, it is worth mentioning that there is no exclusivity between both concepts in the first place. As done by Sleimi et al.[Sl],

one can first define rules for constituents representing legal concepts and afterwards use a grammatical dependency parser to finally extract the semantic information. With an overall precision value of 79.4% for automatically identified spans, the result is promising. However, as also stated in their evaluation, defining rules for constituents is time-expensive and almost never can cover all variatons occuring in practice. Thereby, errors during constituency parsing propagates to the dependency parser leading more likely to false dependency trees. Similiar results have been reached by Evans et al.[Ev] who developed a constituency-based approach for extracting semantic information modeled in the form of hyponoms from privacy policies. In their evaluation, they come to the conclusion that using a dependeny parser would allow a deeper analysis of the relationships between every single word. They are confident that using a dependency parser in combination with a machine learning algorithm to train dmaon specific models of hyponymy might be promising. On the basis of these findings, it has been decided to focus within this thesis on developing a dependency parser-only approach that instead of defining constituents to represent legal concepts intends to directly use dependencies between two nodes to represent a certain legal concept within a sentence. This approach is explained in full detail during the next section and chapter.

Regarding a decision between dependency parsers based on an arc-factored model and the transition-based ones, it can be made relatively clear in favor of the transition-based ones due to their significantly lower runtime complexity class of $O(n)$ compared to $O(n^3)$ of the Eisner algorithm. Hoewever, the decision between the different variants of transition-based approaches is not so evident. In the above section, the main variants for transition-based parsers were described with regard to the kind of operations each of them supports for adding new dependencies between two nodes. While these techniques can be considered as common ground of transition-based parsers, the result of a concrete implementation can still vary since the output also highly depends on the technique used to determine the next operation. A common grouping of all the possible concepts is the separation between so called *static oracles* and *dynamic oracles*. The group of static oracles contains all the solutions that relies on a gold tree, meaning on the basis of the current dependency graph's configuration and the next input token, the optimal next operation is looked up in the gold tree. The biggest advantage of static oracles is clearly that they are deterministic. On the other side however, static oracles are only able to

provide a valid projective dependency tree for inputs that fit the gold tree training set. Mostly, some kind of classifier is trained based on a treebank to serve as static oracle but it also possible to use heuristic disambiguation rules like the original arc-eager parser by [Nia] or a type of formal grammar like a CFG.[Ni08] Particular to the legal domain, Wyner et al.[Wy10] used a CFG for this purpose, but not a grammatical one but one specifically tailored to court decisions of the ECHR as already shortly explained in section XXX. However, within their evaluation, they state that one of their approache's major impediment was that due to using a CFG based approach, their solution is not able to properly extract legal arguments that are structured as defined. As a potential solution for this, they explicitly suggest making use of machine-learning techniques. SpaCy, the NLP-library used to implement the prototype uses a dynamical oracle for that its basic functioning is explained in section 4.3.1.4.

4.3 Architecture

After laying out the theoretical concepts which back the prototypical implementation, its architecture is explained in this section. This will consist of a general description of the processing pipeline for generating a graphical representation of the judgment's legal facts from the original publication format to its final visual representation. As part of this, it will also be discussed which of these steps can be performed automatically, semi-automatically or has to be done manually. In the following, a technical description of the software architecture is been given as well as an explanation of how the semantic metamodel looks like and which how the parsing of the annotated text has been conducted. Also this work follows the common architecture for NLP-based software projects by modeling the different processing steps as components of a *pipeline*. As illustrated in Fig. 4.9, at the beginning of the processing pipeline the original court decisions are taken to conduct a manual linguistic analysis on them. Based on identified linguistic features, a customized data pre-processing is performed which generates a data set of exemplary court decisions suitable to be further used as foundation to annotate the text with the defined dependency types during the linguistic analysis. Once the training data set consisting of all the annotated sentences has been created, it is used to train

the machine-learning based dependency parser upon. Once the depedency parser model has been trained, it is applied on a test data set which results will finally be visually presented as a graph by the frontend application.

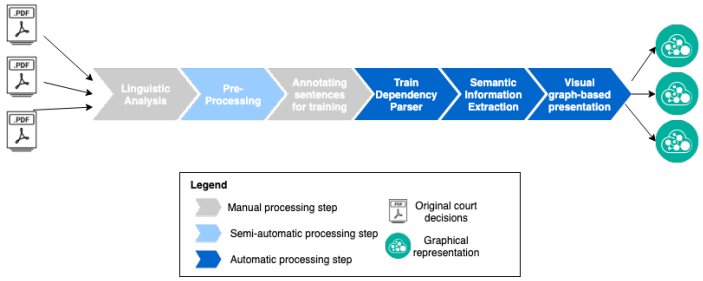


Abbildung 4.9: Complete processing pipeline of the information extraction

Source: Own illustration

4.3.1 Mapping NLP-pipeline steps to software components

4.3.1.1 Linguistic Analysis and Ontology Development

The linguistic analysis conducted as part of this work were focused on a defined set of 15 types of legal relations to keep the scope managable. The considered legal relations can be seen in Table 4.2 together with their keywords whose roles is also explained in this section. When speaking of analyzing court decisions, as for the purpose of this work, it is meant as analyzing the section Tatbestandöf judgments. This can be taken as granted due to the fact that § 313 ZPO regulates that this specific section contains all the required information to rertrieve the involved legal parties and the legal relations among them. For determining which legal relations should be considered for this thesis, two basic thoughts were most important. First, to be as comprehensible as possible, it is intended to support well known relations like a sales or a rental agreement. But second, with regard to the use case of such an application in practice, also legal relations which involve companies or more than two persons in general are part of the set as a graphical representation of a case's underlying legal facts has the more value for the legal professional the more parties and relations play a role and by that the more challenging it gets to keep track of the legal situation. Therefor, also shareholder relations and assignments of claims are covered by

the research. For each of the relation types, it was researched, first, which information actually needs to be extracted to enable a graphical representation and send, how this information is described within a sentence. Referring back to section 4.2.2.1 in which the linguistic characteristics of court decisions were described, the most notable one for this work is the limited syllabus used in court decisions. A concrete result of this specialty is the fact, one can identify *keywords* for each legal relation. Concerning this work, the availability of keywords had to major implications. First, these keywords helped to reduce the effort finding sentences which describe a legal relation so they can be further analyzed to be used to train the dependency parser model in a following step. Second, concerning the ontology development itself, these keywords could also be used as aggregation points for modeling a legal relation with all its subparts which are technically represented by semantic dependencies annotated to the text. The entire ontology is based on the idea of a **Legal Root**. This is a result of the existing concept used when working with syntactical dependencies that is also used by the spaCy dependency parser. This concept is a verb-centric one, meaning the root element of a sentence's dependency tree is always either the main verb or the auxiliary verb of the sentence. While defining such a general rule for which words represents a sentence's root is possible for syntactial dependencies, this approach does not work anymore for semantic dependencies, at least not for those whose semantic roots are not the tense indicating verb at the same time. One major finding of this linguistic analysis is that with one sentence of a judgment more than one legal relations can be and actually are described. This finding also relates to the general characteristics of legal texts, from which one of them is above-average usage of longer relative clauses. Therefor, a special dependeny was added to the ontology, the *legalroot* dependency label. This dependency is used when the syntactial root of the sentence is not also the only legal root of the sentence. One often occuring example is the one when the sentence's syntactial root would be the auxiliary verb. In this a *legalroot* arc goes from the auxiliary verb to the corresponding full verb if it is one of the keywords.

All the cases in which the previously mentioned legal relations mostly occur are assigned to the second, seventh and eighth civile senate of the Federal Court of Justice in Germany according to the court's current organizational

chart.[14] The linguistic analysis and thereby also the following steps of the implementation is limited to judgments assigned to these senates.

Legal Concept	Keywords
Abtretung	abtreten, treten (ab), Abtretung, Abtretungvertrag
Darlehen	aufnehmen, Darlehen, Darlehensvertrag
Gesellschaftsgründung	gründen, errichten, Gründungsgesellschaft, Gründungskommanditist, Gründungs- und Treuhandkommanditist
Gesellschaftsbeteiligung	beteiligen, Kommanditbeteiligung, Kapitalerhöhung, Anteil, Gesellschafter
Insolvenzverwaltung	Insolvenzverwalter, Insolvenzantrag
Kaufvertrag	verkaufen, veräußern, Verkauf, Veräußerung, Kaufvertrag, kaufen, erwerweben, Kauf, Erwerb
Klagebegehren	begehren, fordern, nehmen (in Anspruch)
Kündigung	kündigen, Kündigung
Mietvertrag	mieten, vermieten, Mietvertrag, Mieter
Rechtsnachfolger	Rechtsnachfolger
Schadensersatz	Schadenseratz
Stellvertretung	Stellvertreter, Bevollmächtigter, Geschäftsführer, Prokurist
Widerruf	widerrufen, Widerruf
Generic Auxiliary Concept	Keywords
Erklärung	erklären
Vereinbarung	vereinbaren, schließen, Vereinbarung, erklären

Tabelle 4.2: Concepts of the legal ontology with their keywords

At the top, concepts representing legal relations indicated by specific legal terms. At the bottom concepts, describing a generic relation indicated by a generic keyword. These are words from which legalroot dependencies might go to children words.

4.3.1.2 Pre-processing

Before being able to search for and annotate key sentences based on the developed ontology, there had to be done a significant amount of pre-processing to get the sentences in a format spaCy is able to process. The lack of not

[14] `https://www.bundesgerichtshof.de/DE/DasGericht/Geschaeftsverteilung/Geschaeftsverteilungsplan2019/Zivilsenate2019/zivilsenate2019_node.html`

only machine-readable data sets for legal documents but also the general lack available resources is a well-known impediment of using NLP in the legal sector. The official bodies, the FCJ[15] itself or the German Federal Ministry of Justice[16], only publish court decisions either in the completely unstructured format PDF[17] or in a very limited structured XML[18] format. Although meanwhile, there exists a private open legal data platform[19], also the data within the used JSON[20] format incorporates additional characters like HTML[21] syntax. While non of these available sources provide court decisions in a clean machine-readable format, the Open Legal Data platform comes close and formost is the only one providing an API and also possibility to download the whole set of available court decisions in their database at once. Hence, this platform is used the data source for court decisions.

As base data, the dump with all decisions from the Open Legal Data platform was downloaded.[22] After that, all the decisions were imported as a database in the MongoDB instance to allow faster processing than working with JSON-files. For a description of the actual pre-processing implementation, please refer to section 5.1.1.

4.3.1.3 Annotation of court decisions

Once the pre-processing has been finished, one can finally start to annotate the key sentences of the training data set with the dependencies. While a sentence itself represents the type of legal relation accordingly to the keyword it contains, the dependencies represent either a relation to an involved legal party or to a certain type of information which specifies the legal relation, e.g. a date. For the annotaton process, a tool was used, named *INCEpTION*[23]. INCEpTON is a tool developed by *Technical University Darmstadt*[24] that intends to

[15] https://www.bundesgerichtshof.de/DE/Home/home_node.html
[16] https://www.rechtsprechung-im-internet.de/jportal/portal/page/bsjrsprod.psml
[17] http://wwwimages.adobe.com/www.adobe.com/content/dam/acom/en/devnet/pdf/pdfs/PDF32000_2008.pdf
[18] https://www.w3.org/standards/xml/schema
[19] http://openlegaldata.io/
[20] https://tools.ietf.org/html/rfc8259
[21] https://html.spec.whatwg.org/multipage/
[22] https://static.openlegaldata.io/dumps/de/2019-02-19_oldp_cases.json.gz
[23] https://inception-project.github.io/
[24] https://www.tu-darmstadt.de/

ease the process of annotating not only dependencies but various types. For any details, it is refered to the extensive documentation of the tool which consists of a *user documentation*[25], an *administrator documentation*[26] as well as a research paper[Kl].

While the INCEpTION tool works with numerous input formats, spaCy requires a special json format to train its machine-learning based model. To keep the implementation as close as possible to existing standards, the *CoNLL-U*[27] format is used. The CoNLL-U format is the successor of the CoNLL-X format[RN606] that was a first approach towards a unified framework of annotations for multilingual dependency parsing. As a CoNLL format also the CoNLL-U format is part of the *Universal Dependencies*[28] framework that intends to provide a consistent scheme to annotate grammatical structures. While the developed model of the dependency parser is indeed not grammar-based, one can still use the common CoNLL-U format as input format.

As the INCEpTION tool is only an external helper tool and the focus of this work is on the underlying ontology itself and on methods how for extracting semantic information modeled by the ontology, the INCEpTION specific annotation process is only shortly described by its main steps. For further details, it is refered to the official documentation.

First, the previously extracted and selected sentences for training the dependency parser model are converted from a line-separated text file to the mentioned CoNLL-U format by using the a from the spaCy ecosystem, called *spacy_conll*.[29] The CoNLL file is afterwards imported to the INCEpTION tool to conduct the actual annotation. Once the annotation is finished, the exported CoNLL file now containing the semantic dependency annotations is converted to the *JSON training format* used by the spaCy.[30] For the conversion, spaCy's built-in converter is used.[31]

[25] https://inception-project.github.io//releases/0.11.0/docs/user-guide.html
[26] https://inception-project.github.io//releases/0.11.0/docs/admin-guide.html
[27] https://universaldependencies.org/format.html
[28] https://universaldependencies.org/
[29] https://spacy.io/universe/project/spacy-conll
[30] https://spacy.io/api/annotation#json-input
[31] https://spacy.io/api/cli#convert

4.3.1.4 Training of Dependency Parser Model

SpaCy's dependency parser is built on the concept of a transition-based parser explained in section 4.2.2.3. However, this concept is extended by combining it with *bidirectional long short-term memory* (BiLSTMs)[KG16], a technique incorporating neural-networks (Deep Learning). BiLSTM takes a token's context into consideration for calculating its vector which is then used to make a prediction about the next parsing step. Referring back to what was said in the discussion about the different dependeny parser concepts and here in particular regarding the characteristics of static and dynamic oracles for deciding which dependency should be added next and with which label, this BiLSTM-technique functions as the dynamic oracle of spaCy's statistical model. BiLSTM is an extension of LSTM which itself is a special variant of *RNN*.

RNN stands for *recurrent neural networks* and represents a statistical method for learning how to model sequential data. While RNN calculates a token's vector by using the ones of all its pre-successors, *BiRNN* also incorporates the following tokens. For calculating a token's individual vector, RNN-based approaches use a manually defined set of so called *feature* functions. Common features used by transition-based parsers are lexical characteristics like a token's lemma value next to part-of-speech (POS) tags of a certain number of words in the buffer (the following tokens"), the left-most and right-most tokens on the buffer and the stack (which are mostly the syntactical modifiers[32]), the number of modifiers' modifiers, the parents of the words on the stack and the length of the spans built by the stack tokens.[KG16, p. 3] Now, instead of using manually defined feature functions, LSTM-based approaches only minimally define feature functions, in specific, only the POS-tags and as an extension also the left- and right-most modifiers of the three top-most tokens on the stack next to the left-most modifier of the next token in the buffer. [KG16, p. 7]

4.3.1.5 Extraction of Semantic Legal Information

Once the dependency parser model has been trained and has been applied to a sentence, the necessary semantic information for the visual presentation has

[32] `seehttps://universaldependencies.org/u/dep/index.html`

to be extracted and stored properly. For extracting information, the generated dependency tree for a key sentence is parsed by source code based logic. For details please refer to section 5.1.2.2.

Legal Concept	Dependencies
Abtretung	zed (Zedent), zes(Zessionar), anspr(Anspruch), ansprattr(Anspruch-Attribut)
Darlehen	dnehmer (Darlehensnehmer), dgeber (Darlehensgeber), darlattr (Darlehensattribut)
Gesellschaftsgründung	ae (Anteilgseigner), aeattr (Anteilseigner-Attribut), ges (Gesellschaft), gesattr(Gesellschaft-Attribut), bform (Beteiligungsform), bformattr (Beteiligungsform-Attr), bsum (Beteiligungssumme), bsumattr (Beteiligungssumme-Attribut)
Gesellschaftsbeteiligung	ae (Anteilgseigner), aeattr (Anteilseigner-Attribut), ges (Gesellschaft), gesattr(Gesellschaft-Attribut), bform (Beteiligungsform), bformattr (Beteiligungsform-Attr), bsum (Beteiligungssumme), bsumattr (Beteiligungssumme-Attribut), treuh (Treuhänder), treug (Treugeber)
Insolvenzverwaltung	insverw (Insolvenzverwalter), insschu (Insolvenzschuldner)
Kaufvertrag	kaeufer, verkaeufer, kpreis (Kaufpreis), ksache (Kaufsache)
Klagebegehren	sbeteil (Streitbeteiligter), kbeg (Klagebegehren), rgrund (Rechtsgrund)
Kündigung	kuendigender, kuendgeg (Kündigungsgegner), kuendgrund (Kündigungsgrund), kuendattr (Kündigung-Attribut)
Mietvertrag	vmieter (Vermieter), mieter, mieth (Miethöhe)
Rechtsnachfolger	rnach (Rechtsnachfolger), rvor (Rechtsvorgänger)
Schadensersatz	setyp (Schadenseratz-Typ), rgrund (Rechtsgrund), schuldv (Schuldverhältnis)
Stellvertretung	Stellvertreter, Bevollmächtigter, Geschäftsführer, Prokurist
Widerruf	wirufndr (Widerrufender), wirufgeg (Widerrufsgegner), wirufgstand (Widerrufsgegenstand)
Generic Auxiliary Concept	Dependencies
Erklärung	erkl (Erklärender), erklempf (Erklärungsempfänger), erklgrund (Erklärungsgrund), legalroot
Vereinbarung	vpartner (Vereinbarungspartner), legalroot

Tabelle 4.3: Legal concepts and their dependencies

5 Implementation

As a complete description of the implementation specifics would by far exceed the scope of the thesis, and is in fact also not neccessary for understanding, only the main parts (NLP) are described and regarding the other parts, it is referred to the source code.

5.1 Backend

5.1.1 Data Set and Pre-processing

As already mentioned, due to the lack of data sources that provide legal documents as raw data in a machine-readable format, pre-processing must not be underestimated when one intends to apply NLP-techniques on judgments. Specific for the used data source, the first step was to erase all the HTML-tags from the downloaded base data as no raw data-only data dump was available. For this task, the source code of an existing implementation provided by a tool from the Open Legal Data initiative's ecosystem is used.[33] This implementation is extended by customized Regex-based cleaning routines. As this step is specific to the used data set, its details are omitted within this thesis. For details, pleaser refer to the source coude. Afterwards, the required section Tatbestandïs extracted from the judgment together with the judgment's docket number since only this part is relevant. In order to do so, spaCy's *Token Matcher*[34] is used, a rule-based matching technique to extract text based on defined text patterns. As patterns were defined, first, the expression Tatbestandänd second, the expression Ëntscheidungsgründe". Here, Ëntscheidungsgründeïs the name of the section following the Tatbestandßection. Both expressions are uniquely

[33] `https://github.com/openlegaldata/legal-ner/blob/master/legal_ner/preprocessing.py`

[34] `https://spacy.io/usage/rule-based-matching#matcher`

used within a judgment so there is no risk to find duplicates. The matcher returns the integer position of both terms wihin the text. Afterwards a new *Doc*[35]-element is created with the span in between representing the Tatbestandßection. A Doc-element is spaCy's wrapper implementation to model a document's text as a sequence of tokens and also sentences.

The initialization of the Doc-element is a major point when using spaCy. It is here, where all the steps of the general well-known pre-processing tasks within the field of NLP are applied.[36] Fig. 5.1 shows the entire pre-processing pipeline for transforming an unstructured text to one with structured lexical, morphological, syntactic and semantic information. Below each phase, its single subtasks are written. Those marked bold and in red are the ones used within this work. As one can see, the parts of the syntactic and semantic phase like in particular dependency parsing are already included. However, concerning the current section of the thesis, the first two phases are relevant for the extraction of key sentences suitable for being annotated and used as training data.

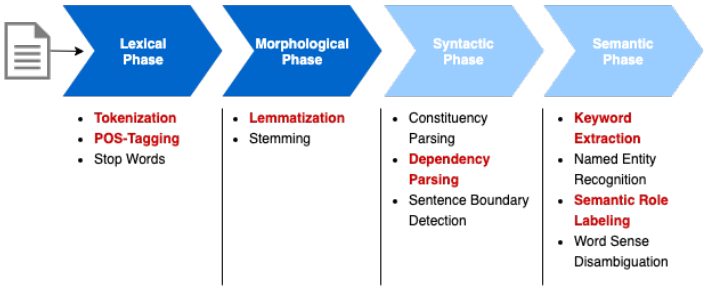


Abbildung 5.1: Pre-processing pipeline with used components bold and in red

Source: Own illustration based on [VGN]

Now that we can work only on the required part of a judgment, all judgments ruled by one of the considered senates are searched for the defined keywords of which each represents one of the legal concepts thas has previously been defined. Also for this task, spaCy's token matcher is used. For the matching, not the actual textual representation of a word is used by the lemmatized one. By using the word's, or more precisly, token's lemma, we actually compare a word's base form and thereby are able to cover a larger variety of sentences. For example, the lemma of both words, *"kaufen"* and *"kaufte"*, is *"kaufen"*. (German for to sell)

[35] `https://spacy.io/api/doc`
[36] `https://spacy.io/usage/linguistic-features#tokenization`

During pre-processing several issues came up which results in the fact that this used pre-processing procedure can not be conducted automatically but rather requires a manual control at the end. For once, this is caused by the fact that this implementation only is considered to be a prototypical one. But the issues' remaining reasons one also faces when implementing a more sophisticated NLP-pipeline, arise from the text's characteristic itself. Table 5.1 provides a summarized overview of two major constraints.

Issue	Description	Examples
Abbreviations of legal parties' names (especially company names)	The spaCy tokenizer splits these space-separated abbreviations in individual tokes. Due to the large variety of abbrevation types, it is difficult to find an exhaustive set of rules to properly merge them to one token which is mandatory to apply correct dependency annotations	Examples: A. B. C. GmbH & Co. KG, Herr A. B.
References to legal documents introduced by one of the parties (mostly referenced and quoted contracts)	Due to the variety, it is difficult to find rules to automatically ignore these parts for finding key sentences for training data.	Example: ..., was die Parteien vertraglich auf Seite 10 des Vertrages vereinbart haben: ... VERTRAGSTEXT...

Tabelle 5.1: Summary of arised issues during pre-processing and the resultung constraints

At the end of this pre-processing, the sentences are tokenized in a way so they can further be processed to the annotation phase.

5.1.2 Dependency Parser

Describe the parameters of spaCy's dependency parser

5.1.2.1 Annotation and Training of the dependency parser model

The pre-processed sentences have been pre-annotated with syntactical dependencies based on the TIGER corpus.[37] The developed dependencies of every legal concept are not designed to annotate a whole sentence with only semantic legal dependencies but rather just use semantic label for the dependencies to tokens which actually have some sort of legal semantic meaning. While the remaining dependencies do not get semantic labels, these non-semantic ones still need to be rearranged so the new semantic structure actually builds a valid dependency tree again. Concerning this work, the developed ontology was designed in a way to allow the new semantic dependency structure to also be projective.

Concerning the exisiting syntactic labels, by theory, one possibility could have been to completely delete all non-semantic label and replace them just with one like ". But instead of choosing this way, it was decided to use the existing syntactical labels and limit the refactoring scope of the syntactial relations to only adjusting the arcs direction and heads. Fig. 5.2 illustrates this tranformation process by first showing the pre-annotated sentence with its syntactical dependencies and afterwards the sentence with semantic dependencies and the adjusted syntactical ones. When looking at the area within the left red rectangle one can see that the syntactic aggregation point for the part that is describing the date when the rental agreement has been signed, is the word ïm". As defined by the German grammar, this word in its here used function represents a *modifier* annotated with *mo*. The same is true for the right rectangle, the part that states who the landlord. Now within our semantic model, these high level nodes in the dependency tree are represented by the actual semantic legal concept. So the left modifier dependency is transformed to a dependency labeled with "datumänd the right one respectively to one labeled with "vmieter". Thinking back to what was said in section 4.3.1.4 about the underlying concept of spaCy's implementation of its dynamic oracle for how to decide which label is used to annotate the next arc, better results can be expected when also the non-semantic arcs are labeled and the node representing the upper semantic legal dependant is chosen as the head element. As we are not interested in extracting these non-semantic parts, the actual label name is not as important as it is to use the same labels no matter which type

[37] `https://spacy.io/models/de`

of legal concept the head node specifies. For the same reason, spaCy's default POS component of its German language model is used as the base model to train the model's dependency parser component from scratch. By doing that, one can leverage the neural-network features of spaCy's dependency parser training method.

(a) Sentence with semantic dependencies

(b) Sentence with semantic dependencies

Abbildung 5.2: Transformation of syntactic to semantic dependencies

Source: Screenshot INCEpTION annotation tool

Regarding the actual technical procedure to train the dependency parser model, the spaCy built-in command-line interface (CLI) was used.[38] In order to provide the training data in the required JSON-format, spaCy's built-in CLI-based converter functionality was used.[39] After the CONLL-file that had been exported from the INCEpTION tool was converted to spaCy's training JSON-format, the model's dependency parser component was trained by using the CLI-based training functionality with its default settings. Hence, these specific settings are not presented here.

5.1.2.2 Extraction Rules

The file *sentence_analyzer.py* contains all the routines for parsing a sentence's dependency tree and extracting the semantic information. As the file is a few hundred lines long and therefor looking at excerpts can not be considered useful. Please, directly refer to the source code.

Generally, the suggested bottom-up approach is used[40], meaning the tree is parsed by directly iterating over the root's children elements. As we are using semantic dependency labels, the logic for extracting a certain type of information completely is the direct technical representation of the logic the semantic ontology is built upon.

[38] `https://spacy.io/api/cli#train`
[39] `https://spacy.io/api/cli#convert`
[40] `https://spacy.io/usage/linguistic-features#navigating`

As it will be presented in chapter 6, compared to the small training data set, the results can be considered as good. Nevertheless, the quality is not high enough to build dependency trees with a high enough correctness of the arc labels so the exception handling of the theoretically straight-forward logic would be possible. As this implementation intends to only be prototypical one and not one on production level, the functionality of the implemented sentence analyzer is limited to sentences with a high level of label correctness. Another constraint is the storing of the extraction results. Since an automatic-like extraction is not possible, the implementation of a sophisticated solution for storing documents together with the semantic information like *Elasticsearch*[41] has not been done.

5.2 Frontend

The frontend is implemented using the JavaScript library *React*.[42] To actually being able to show a graph-based representation of a judgment's legal facts, the additional library *react-d3-graph* is used. Refering to what was said regarding the quality of the final extracted information, a dynamic implementation that instantly fetches data from the backend's API seemed not to make any sense. While the API has been setup in the backend, for the purpose of this work the frontend uses manually prepared data to illustrate the result of the analysis of a judgment's legal facts section.

[41] `https://www.elastic.co/products/elasticsearch`
[42] `https://reactjs.org/`

6 Evaluation

The evaluation of this work is split into two parts - a quantitative evaluation and a qualitative evaluation.

6.1 Quantitative Evaluation

For the quantitative evaluation, two different trained dependency parser models are compared. The first model was trained without spaCy's machine-learning capabilities regarding the in section 4.3.1.4 described implementation of a dynamic oracle for determining how the dependency arc is labeled. The other model contained spaCy's default pre-trained POS-component that provides the POS-tags for the training of the dependency parser component. Both models are compared by their respective *Unlabeled Attachment Score* (UAS) and the *Labeled Attachment Score* (LAS). The training data set consisted of different *38* sentences distributed over all types of supported legal relations. The evaluation data set consists of *25* sentences. The results can be seen in Table 6.1.

	Without POS-component	**With POS-Component**
UAS	79.46	80.10
LAS	42.94	68.37

Tabelle 6.1: Results

Remarkably, the model with the POS-component integrated, reached a LAS score over 25 points higher than the one of the other model. The probably most important result of this work is thereby that by using neuronal-network capacities for determining the next arc label, one does not need to fully annotate the sentence with semantic labels as long as the labels for the relations to

the surrounding tokens are consistently used one with respect to their naming and second regarding their head-child direction.

6.2 Qualitative Evaluation

Concerning the qualitative evaluation, Mr. Schaper, a lawyer of Verlag Dr. Otto Schmidt KG manually drawed graphical representations consisting of the involved parties and the legal relations among them. While the LAS score of 68.37 % can be considered relative high compared to the small amount of training data, the value turned out to be too low to actually reliable extract the information in a degree with that a qualitative evaluation would have made sense.

7 Summary and Discussion

7.1 Summary

With the existing potential regarding the use of semantic information retrieval solutions that has been shown within the introduction, the following section presented a short overview of the ongoing research of using NLP-techniques to extract semantic information from legal documents. During the following system design chapter, existing frameworks for legal metamodels and ontologies were shown alongside with an explanation of the NLP-techniques used for the implementation within this work. After the description of the prototypical implementation of the developed concept, an evaluation concluded the work. All of these parts were shaped around the three research questions defined in chapter 3. For each of them a conclusion is drawn in the following section.

7.2 Conclusion

How an ontology for representing semantic information of court decisions can look like? Although there is a lot of research on developing legal metamodels and ontologies in general, there is less research on how to specifically model information within court decisions and even less for German court decisions. Nevertheless it was accomplished to develop an ontology with that it was possible to construct valid projective dependency trees to structure the information within a sentence.

How the key information of a court decision can automatically be extracted using NLP? While defining extraction rules when one uses constituency parsing takes a lot of effort since all possible syntactic combinations must be considered to ensure a certain type of information is found regardless of its

grammatical representation, this does not apply to dependency parsing in this extent. The implemented extraction rules are quite simple as they directly follow the ontology's structure.This can be considered an advantage compared to syntactic dependencies. The reason why the extraction results are not so high is the relative small amount of training data which did not allow higher LAS scores. As the defined legal relations can be composites of each other thereby are also so annotated, incorrect dependency arcs consequently invalid the extraction rules and thereby lowering the overal result. Compared to the amount of training data used in other NLP projects, it can be expected that once more annotated training data is available, the results will rise.

How a prototype for a semantic analysis of court decisions can be implemented? A basic version of a prototype could be implemented. Of course the main focus of the implementation layed on the NLP-part. Regarding the API and the frontend, it has to be noted that this is intentially kept at the bare minimum.

7.3 Limitations and Future Work

Conclusively, we can summarize that the developed approach seems to be promising to be further developed. However concerning the current state, there exist some limitations to be named.

1. No real support of annotating and extracting information from relative clauses
2. Due to the too low LAS score to test proper extraction rules, limited support of extracting hierarchically wrapped legal concepts
3. Necessity of manually defining which party is the plaintiff and which is the defendant since not all judgments explicitly name it
4. Limited preprocessing quality

In the future, the major task will certainly be to create more training data. With the current result in mind, the expectations to finally reach practice suitable values with sufficient training data can be considered as high. Having a

solid foundation of training data available, one will able to improve the extraction logic and thereby also enable a proper visual representation. Regarding the before mentioned limitations, one task should be researching on the topic *coreferencing* to enable the handling of relative clauses. Next to this, also the improvement of the preprocessing is necessary in the future since implementing proper rules is time intensive and the scope was limited due the fixed amount of time available. All in all, the results are promising and once the concepts are consequently further developed, the chances are high to be able to provide extensive solutions for extracting semantic information.

Literaturverzeichnis

[AL] Akbik, A.; Li, Y.: *K-SRL: Instance-based Learning for Semantic Role Labeling*. Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers. pages 599–608. The COLING 2016 Organizing Committee. 4.2.2.1

[Al02] Alexy, R.: *A Theory of Constitutional Rights*. Oxford University Press UK. 2002. 4.1.2.4

[AMF] Afzal, N.; Mitkov, R.; Farzindar, A.: *Unsupervised Relation Extraction Using Dependency Trees for Automatic Generation of Multiple-Choice Questions*. Advances in Artificial Intelligence. pages 32–43. Springer Berlin Heidelberg. 2.2.2

[At15] Athan, T.; Governatori, G.; Palmirani, M.; Paschke, A.; Wyner, A.: *LegalRuleML: Design Principles and Foundations*. pages 151–188. Springer International Publishing. Cham. 2015. 4.1.2.3

[BCDF14] Bretonnel Cohen, K.; Demner-Fushman, D.: *Biomedical Natural Language Processing*. John Benjamins. 2014. 2.2

[Bh] Bhatia, J.; Evans, M. C.; Wadkar, S.; Breaux, T. D.: *Automated Extraction of Regulated Information Types Using Hyponymy Relations*. In *2016 IEEE 24th International Requirements Engineering Conference Workshops (REW)*. pages 19–25. 2.2.1

[Bu98] Busse, D.: *Rechtssprache als problems der Bedeutungsbeschreibung. Semastische Aspekte einer instituonellen Fachsprache. Sprache und Literatur*. 29(1):24–47. 1998. 4.2.2.1

[BVW04] Breuker, J.; Valente, A.; Winkels, R.: *Legal Ontologies in Knowledge Engineering and Information Management. Artificial Intelligence and Law*. 12(4):241–277. 2004. 4.1.2

[Co69] Cocke, J.: *Programming languages and their compilers: Preliminary notes.* New York University. 1969. B0007F4UOA. 4.2.1

[Co03] Collins, M.: *Head-Driven Statistical Models for Natural Language Parsing. Computational Linguistics.* 29(4):589–637. 2003. 4.2.2.2

[DF12] Dell'Orletta F., Marchi S., M. S. P. B. V. G.: *The SPLeT–2012 Shared Task on Dependency Parsing of Legal Texts.* In *SPLeT 2012 – Fourth Workshop on Semantic Processing of Legal Texts (SPLeT 2012) – First Shared Task on Dependency Parsing of Legal Texts (Istanbul, 27 Maggio 2012).* pages 42–51. 2012. 2.2.2

[Ei] Eisner, J. M.: *Three New Probabilistic Models for Dependency Parsing: An Exploration.* COLING 1996 Volume 1: The 16th International Conference on Computational Linguistics. 4.2.2.2

[Ev] Evans, M. C.; Bhatia, J.; Wadkar, S.; Breaux, T. D.: *An Evaluation of Constituency-Based Hyponymy Extraction from Privacy Policies.* In *2017 IEEE 25th International Requirements Engineering Conference (RE).* pages 312–321. 2.2.1, 4.2.3

[GAG] Griffo, C.; Almeida, J. P. A.; Guizzardi, G.: *Conceptual Modeling of Legal Relations.* Conceptual Modeling. pages 169–183. Springer International Publishing. (document), 4.1.2.4, 4.5

[GAP] Ghanavati, S.; Amyot, D.; Peyton, L.: *A systematic review of goal-oriented requirements management frameworks for business process compliance.* In *2011 Fourth International Workshop on Requirements Engineering and Law.* pages 25–34. 4.1.1.1

[Gi] Gildea, D.: *Corpus Variation and Parser Performance.* Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing. 2.2.2

[HGJ] Honnibal, M.; Goldberg, Y.; Johnson, M.: *A Non-Monotonic Arc-Eager Transition System for Dependency Parsing.* Proceedings of the Seventeenth Conference on Computational Natural Language Learning. pages 163–172. Association for Computational Linguistics. 4.2.2.3

[HJ] Honnibal, M.; Johnson, M.: *An Improved Non-monotonic Transition System for Dependency Parsing.* Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing. pages 1373–1378. Association for Computational Linguistics. 4.2.2.3

[HM15] Hirschberg, J.; Manning, C. D.: *Advances in natural language processing. Science.* 349(6245):261–266. 2015. 1.1

[Ho] Hoekstra, R.; Breuker, J.; Bello, M. D.; Boer, A.: *The LKIF Core Ontology of Basic Legal Concepts.* In *LOAIT.* (document), 4.1.2.2, 4.3, 4.4

[Ho17] Hohfeld, W. N.: *Fundamental Legal Conceptions as Applied in Judicial Reasoning. The Yale Law Journal.* 26(8):710–770. 1917. 4.1.1.3

[HSN04] Hansen-Schirra, S.; Neumann, S.: *Linguistische Verständlichmachung in der juristischen Realität. Lerch, Kent D.(Hg.): Recht Verstehen. Verständlichkeit, Missverständlichkeit und Unverständlichkeit von Recht. Berlin/New York: de Gruyter.* pages 167–184. 2004. 1.1, 4.2.2.1

[JLM] Jelinek, F.; Lafferty, J. D.; Mercer, R. L.: *Basic Methods of Probabilistic Context Free Grammars.* Speech Recognition and Understanding. pages 345–360. Springer Berlin Heidelberg. 2.2.1

[JM09] Jurafsky, D.; Martin, J. H.: *Speech and Language Processing (2nd Edition).* Prentice-Hall, Inc. 2009. 0131873210. (document), 2.2, 2.2.1, 2.2.2, 4.1.1.2, 4.1.3, 4.6

[KG16] Kiperwasser, E.; Goldberg, Y.: *Simple and Accurate Dependency Parsing Using Bidirectional LSTM Feature Representations. Transactions of the Association for Computational Linguistics.* 4:313–327. 2016. 4.3.1.4

[Ki] Kiyavitskaya, N.; Zeni, N.; Mich, L.; Cordy, J. R.; Mylopoulos, J.: *Text Mining Through Semi Automatic Semantic Annotation.* Practical Aspects of Knowledge Management. pages 143–154. Springer Berlin Heidelberg. 4.1.1.1

[Kl] Klie, J.-C.; Bugert, M.; Boullosa, B.; Castilho, R. E. d.; Gurevych, I.: *The INCEpTION Platform: Machine-Assisted and Knowledge-Oriented Interactive Annotation.* In *COLING*. 4.3.1.3

[LZY] Li, J.; Zhang, L.; Yu, Y.: *Learning to Generate Semantic Annotation for Domain Specific Sentences.* In *Semannot@K-CAP 2001*. 4.2.2.1

[NFG14] Nivre, J.; Fernández-González, D.: *Arc-Eager Parsing with the Tree Constraint. Computational Linguistics.* 40:259–267. 2014. 4.2.2.3

[Nia] Nivre, J.: *An Efficient Algorithm for Projective Dependency Parsing.* Proceedings of the Eighth International Conference on Parsing Technologies. pages 149–160. 4.2.2.3, 4.2.2.3, 4.2.3

[Nib] Nivre, J.; de Marneffe, M.-C.; Ginter, F.; Goldberg, Y.; Hajič, J.; Manning, C. D.; McDonald, R. et al.: *Universal Dependencies v1: A Multilingual Treebank Collection.* Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016). pages 1659–1666. European Language Resources Association (ELRA). 2.2.2

[Ni05] Nivre, J.: *Dependency grammar and dependency parsing.* Report. 2005. 4.2.2.1

[Ni08] Nivre, J.: *Algorithms for Deterministic Incremental Dependency Parsing. Computational Linguistics.* 34(4):513–553. 2008. 4.2.3

[PGK05] Palmer, M.; Gildea, D.; Kingsbury, P.: *The Proposition Bank: An Annotated Corpus of Semantic Roles. Computational Linguistics.* 31(1):71–106. 2005. 4.2.2.1

[RN203] : 2003. 2.2

[RN504] : 2004. (document), 4.2.2.3, 4.8

[RN606] : 2006. 4.3.1.3

[Sa03] Sampson, G.: *Thoughts on Two Decades of Drawing Trees.* pages 23–41. Springer Netherlands. Dordrecht. 2003. 4.2.1

[Se03] Seidewitz, E.: *What models mean. IEEE Software.* 20(5):26–32. 2003. 4.1

[Sh17] Shahab, E.: *A Short Survey of Biomedical Relation Extraction Techniques.* 2017. 2.2.2

[Si] Siena, A.; Mylopoulos, J.; Perini, A.; Susi, A.: *Designing Law-Compliant Software Requirements.* Conceptual Modeling - ER 2009. pages 472–486. Springer Berlin Heidelberg. (document), 4.1.1.2, 4.2

[Sl] Sleimi, A.; Sannier, N.; Sabetzadeh, M.; Briand, L.; Dann, J.: *Automated Extraction of Semantic Legal Metadata using Natural Language Processing.* In *2018 IEEE 26th International Requirements Engineering Conference (RE).* pages 124–135. 2.1, 2.2.2, 4.2.3

[VGN] Vijayarani, S.; Gunasekaran, T.; Nithya, S.: *Preprocessing Techniques for Text Mining-An Overview Dr.* 5.1

[Wy08] Wyner, A.: *An ontology in OWL for legal case-based reasoning. Artificial Intelligence and Law.* 16:361–387. 2008. 2.1, 4.1, 4.1.2.1

[Wy10] Wyner, A.; Mochales-Palau, R.; Moens, M.-F.; Milward, D.: *Approaches to Text Mining Arguments from Legal Cases.* pages 60–79. Springer Berlin Heidelberg. Berlin, Heidelberg. 2010. (document), 2.2.1, 4.2.1, 4.7, 4.2.3

[Xu11] Xu, H.; AbdelRahman, S.; Lu, Y.; Denny, J. C.; Doan, S.: *Applying semantic-based probabilistic context-free grammar to medical language processing – A preliminary study on parsing medication sentences. Journal of Biomedical Informatics.* 44(6):1068–1075. 2011. 4.2.1

[Yo67] Younger, D. H.: *Recognition and parsing of context-free languages in time n3. Information and Control.* 10(2):189–208. 1967. 4.2.1

[ZCL] Zhang, Q.; Chen, M.; Liu, L.: *A Review on Entity Relation Extraction.* In *2017 Second International Conference on Mechanical, Control and Computer Engineering (ICMCCE).* pages 178–183. 2.2.2

[Ze] Zeni, N.; Seid, E. A.; Engiel, P.; Ingolfo, S.; Mylopoulos, J.: *Building Large Models of Law with NómosT.* Conceptual Modeling. pages 233–247. Springer International Publishing. 4.1.1.2

[Ze15] Zeni, N.; Kiyavitskaya, N.; Mich, L.; Cordy, J. R.; Mylopoulos, J.: *GaiusT: supporting the extraction of rights and obligations for regulatory compliance. Requirements Engineering.* 20(1):1–22. 2015. (document), 4.1.1.1, 4.1, 4.1