FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

DOCTORAL THESIS

Vincent Kríž

Detecting semantic relations in texts and their integration with external data resources

Institute of Formal and Applied Linguistics

Supervisor of the doctoral thesis:	Mgr. Barbora Vidová Hladká, Ph.D.
Study programme:	Informatics (P1801)
Study branch:	4I3 (1103V012)

Prague 2019

Title:
Detecting semantic relations in texts and their integration with external data resources

Author:
Vincent Kríž

Department:
Institute of Formal and Applied Linguistics

Supervisor:
Mgr. Barbora Vidová Hladká, Ph.D.
Institute of Formal and Applied Linguistics

Abstract:
We present a strategy to automate the extraction of semantic relations from texts. Both machine learning and rule-based techniques are investigated and the impact of different linguistic knowledge is analyzed for the various approaches. To implement the extraction system RExtractor, several natural language processing tools have been improved: from sentence splitting and tokenization modules to dependency syntax parsers. Furthermore, we created the Czech Legal Text Treebank with several layers of linguistic annotation, which is used to train and test each stage of the proposed system. As a result of the performed work, new Semantic Web resources and tools are available for automatic processing of texts.

Keywords:
Information Extraction, Linked Data, Legal Texts, Dependency Treebank, Dependency Parsing

Acknowledgement

I thank Bára Hladká, the best supervisor in the world. She was always here for me and her patience is unbelievable.

I would like to thank my colleagues from the Institute of Formal and Applied Linguistics. Zdeňka Urešová is my superhero for her ability to annotate sentences in the Czech Legal Text Treebank. Many thanks for Martin Popel, Rudolf Rosa, Silvie Cinková and last but not least, Markéta Lopatková.

Thanks to my colleagues from the INTLIB project, especially Martin Nečaský, Tomáš Knap, Jakub Klímek and Honza Dědek.

I would like to thank my parents – I have let so many years pass without thanking you both, but you haven't let a single second pass without loving me unconditionally. Let me try catching up by telling you how wonderful you are. Thanks.

Many thanks to my friends Karel Petr Dušek, Honza Mrázek, Honza Kašpar and Petr Hrbek.

Finally, I thank Jan Turek for his endless love.

The Supreme Personality of Godhead said:
Fearlessness; purification of one's existence;
cultivation of spiritual knowledge; charity; self-control;
performance of sacrifice; study of the Vedas; austerity;
simplicity; nonviolence; truthfulness; freedom from anger;
renunciation; tranquillity; aversion to faultfinding;
compassion for all living entities; freedom from covetousness; gentleness;
modesty; steady determination; vigor; forgiveness; fortitude; cleanliness;
and freedom from envy and from the passion for honor
— these transcendental qualities, O son of Bharata,
belong to godly men endowed with divine nature.

Bhagavad-gītā As It Is, 16.1-3

Contents

1. Introduction

The rapid advancement in the digital age increases interest in communication between human and machine. Natural Language Processing (NLP) is a field of science which helps humans communicate with computers in their own language, in its spoken and written form. It spans over linguistics, computer science, information engineering, machine learning and statistics. On the way to human-machine communication, NLP solves language-related tasks starting from the morphological, syntactic and semantic analysis of unstructured text, sentiment analysis, information extraction to complex systems like machine translation. Humans express themselves differently in different situations. Analogously, NLP approaches are typically language and domain specific.

The staggering amount of unstructured data that is generated every day, emerges not only the human-machine communication but it results also in a growing need for effective and efficient techniques that allow data to be shared and reused across application, enterprise, and community boundaries.[1] To achieve this, Berners-Lee et al. [2001] expressed a need of shift in thinking from publishing data in human-readable documents to machine-readable documents and proposed the idea of Semantic Web. It is a vision where the existing World Wide Web provides machine-interpretable data for each published document. As a result, computers are able to make meaningful interpretation similar to the way humans process information to achieve their goals. For the last 20 years, the Semantic Web community developed several knowledge representation approaches that express meaning in formal computer models, from controlled vocabularies, taxonomies, schemas and ontologies to knowledge graphs.

Although the idea of Semantic Web of machine-interpretable data is useful and interesting, the main problem is what to do with all documents already published which do not have such structured data. As there is a huge amount of unstructured documents, it would be impossible to add machine-interpretable data manually. This resulted in Information Extraction – a new field of science in which Semantic Web and NLP cooperate and access the World Wide Web for machines.

Information Extraction focuses on extracting structured information automatically from unstructured or semi-structured documents. A typical Information Extraction strategy consists of (i) entity recognition and (ii) relation extraction. Entity recognition involves identifying (textual) mentions referring to elements in a given unstructured or semi-structured input source. Relation extraction involves identification of semantic relations between entities.

For illustration, let's assume the sentence *František Křižík was born in Plánice.* In the first step, particular Information Extraction system identifies and resolves en-

[1] According to the International Data Corporation, the Global Datasphere is experiencing tremendous growth. Reinsel et al. [2018] predict that the Global Datasphere will grow from 33 zettabytes (33 trillion gigabytes) in 2018 to 175 zettabytes by 2025.

tities, such as *František Křižík*[2] refers to the famous Czech inventor and *Plánice*[3] is the name of the picturesque South Bohemian town. In the second step, the system identifies the semantic relation *birthplace*[4] between extracted entities. On the output, we could expect machine-readable structured data, most typically expressed as triples, e.g. [*František Křižík*, *birthplace*, *Plánice*]. All elements in the triple are linked with the particular concepts in a knowledge base.

In the context of Semantic Web, the word *semantic* indicates *machine-processable* or *what a machine is able to do with the data.* Semantic data present a machine-readable meaning of information. In the context of NLP, the word *semantic* indicates sub-field that detects relationships between words, such as word sense disambiguation, topic-focus articulation, coreference and discourse. Typically, they exploit an existing theoretical framework, such as the Functional Generative Description proposed by Sgall [1967]. In this thesis we use the word *semantic* and the phrase *semantic relations* from the perspective of Semantic Web.

In this thesis we present a strategy to automate the extraction of semantic relations from texts. Both machine learning and rule-based techniques are investigated and the impact of different linguistic knowledge is analyzed for the various approaches. To implement the extraction system RExtractor, several natural language processing tools have been improved: from sentence splitting and tokenization modules to dependency syntax parsers. Furthermore, we created the Czech Legal Text Treebank with several layers of linguistic annotation, which is used to train and test each stage of the proposed system. As a result of the performed work, new Semantic Web resources and tools are available for automatic processing of texts.

Regarding domains and languages, the experiments are focused on the extraction of knowledge from Czech legal documents, namely acts and court decisions produced by the legislation of the Czech Republic. We consider the legal domain to be very interesting from the point of NLP as legal texts contain several peculiar morpho-syntactic phenomena, such as passive voice structures, impersonal constructions, non-finite and verbless clauses, all used in long and very complex sentences.

Czech legal documents are published by the authorities at various places on the Web not inter-linked in any way and in different formats. As a result, it is hard for users to find legislative documents they need and to search for related content. In this work we propose how legislative documents in the Czech Republic can be published using Semantic Web technologies and demonstrate advantages of such a style of publication. The extracted structured data allow to develop applications that improve users' comfort when searching in collections of legal texts.

The systems and data presented in this thesis were created as a part of the Intelligent Library project (INTLIB, [Nečaský et al., 2013, Holubová et al., 2014]).

[2]`https://cs.dbpedia.org/page/František_Křižík`
[3]`https://cs.dbpedia.org/page/Plánice`
[4]`https://cs.dbpedia.org/property/místoNarození`

Chapter Guide

Chapter 2 provides theoretical fundamentals and surveys previous work in the fields of Information Extraction, Semantic Web and syntactic parsing. We also present the current state of the Czech legal informatics and mention the official government projects, commercial systems and the academic research.

Chapter 3 presents the main language resource that we created and use in this work, the Czech Legal Text Treebank. It contains manually annotated sentences from the Collection of Laws of the Czech Republic. We published its two versions. Both CLTT 1.0 [Kríž et al., 2015, Kríž et al., 2016] and CLTT 2.0 [Kríž and Hladká, 2017, Kríž and Hladká, 2018] were presented at the International Conference on Language Resources and Evaluation (LREC).

Chapter 4 presents the RExtractor system that extracts a knowledge base of entities and semantic relations from unstructured texts. Both the general architecture and the application to the Czech legal documents are described. The work was published originally by Kríž et al. [2014b] and presented at the Mexican International Conference on Artificial Intelligence (MICAI). The system was presented at the system demonstrations session at the 2015 Conference of the North American Chapter of the Association for Computational Linguistics – Human Language Technologies (NAACL) [Kríž and Hladká, 2015].

Chapter 5 presents a new method for the dependency parsing of complex sentences. This method assumes segmentation of input sentences into clauses and does not require to re-train a parser of one's choice. The experiments with parsing of complex sentences were published in [Kríž and Hladká, 2016] and presented at the Annual Meeting of the Association for Computational Linguistics (ACL).

Chapter 6 describes the task of detection and classification of references in Czech court decisions. We report our experiments with supervised machine learning methods as well as the Czech Court Decisions Dataset of manually annotated court decisions. This work was originally published by Kríž et al. [2014a] and presented at the Mexican International Conference on Artificial Intelligence (MICAI). The Czech Court Decisions Dataset is available on-line as well [Kríž and Hladká, 2014].

Chapter 7 presents the integration of the data extracted by RExtractor and JTagger systems into the Czech Legal Linked Open Data Cloud. This work was originally published by Nečaský et al. [2013] and Kríž et al. [2014b].

Contributions

This thesis presents two original datasets: (i) the Czech Legal Text Treebank and (ii) the Czech Court Decisions Dataset. Regarding the Czech Legal Text Treebank, the author of the thesis proposed, implemented and evaluated a tailored preprocessing strategy that significantly reduces the complexity of the subsequent manual syntactic annotation. In addition, the author proposed several modifications of the standard annotation manual and implemented a new extension of the annotation tool. Next, the author created two new layers of annotation, namely the layer of named entities and the layer of semantic relations. Regarding the Czech Court Decisions Dataset, the author of this thesis created the annotation manual and provided the technical support, tools and data quality inspections during the manual annotation.

This thesis describes two systems proposed and implemented by the author: (i) the RExtractor system and (ii) the JTagger system. Regarding RExtractor, the author proposed a general system architecture as well as the custom modules for the processing of the Czech legal documents, including the idea of re-tokenization and the complex sentence multiplication. Both systems have been applied on a large set of legal documents.

This thesis reports experiments on the task of extraction of structured data from unstructured text. The author proposed and evaluated both machine learning and rule-based techniques. Also, he analyzed the impact of different linguistic knowledge for the various approaches.

The author of this thesis proposed a novel approach to parsing of long sentences. Based on the intrinsic evaluation on five different datasets, the method improves the performance of the parsing significantly. The method was also evaluated extrinsically by the RExtractor system which lead to the performance improvement on the task of information extraction.

The structured data extracted from the legal documents have been published in the Czech Legal Linked Open Data Cloud. As a result, the work presented in this thesis helps to solve the current problems in the Czech legislation where documents are published by different authorities at various places, not inter-linked in any way and in different formats. The extracted structured data placed in the cloud are ready for development of applications that improve users' comfort when searching in collections of legal texts.

2. Related Work

In this chapter we provide theoretical fundamentals and related work for the fields of Information Extraction (Section 2.1) and Semantic Web (Section 2.2). The relevant systems that extract knowledge from the legal texts are listed in Section 2.3. In Section 2.4 we present the current state of the Czech legal informatics and mention the official government projects, commercial systems and the academic research. Finally, Section 2.5 provides related work for the topic of parsing of long sentences.

2.1 Information Extraction

The extraction of structured data from noisy, unstructured sources is a fundamental task, that has engaged a large community of researchers over decades. With roots in the Natural Language Processing, the topic of structure extraction now engages many different communities in machine learning, information retrieval, database, web, and document analysis. Information Extraction (IE) is useful in a diverse set of applications, from news tracking, customer care, data cleaning, citation databases and comparison shopping to structured web searches. The field was strongly influenced by two competitions, the Message Understanding Conference (MUC, [Chinchor, 1998]) and Automatic Content Extraction (ACE) Program.[1] The rapid advance of the Web and the Semantic Web initiative emerged the need for developing IE systems that help people to process the enormous amount of data that is available on-line. This has lead to a cooperation that can be seen from two perspectives: the Semantic Web resources are used to improve Information Extraction, and Information Extraction helps to populate the Semantic Web resources.

Two most typical Semantic Web resources are *ontologies* and *knowledge bases*. An ontology is a formal description of knowledge. It is a set of concepts and relationships between them within a particular domain. A knowledge base is a machine-readable repository for information. In the field of Semantic Web, knowledge bases are typically huge databases of entities and relations between them, e.g. DBpedia.[2]

Two types of structured data are typically extracted from an unstructured source: *entities* and *semantic relations* between entities.

[1] https://tac.nist.gov/
[2] https://wiki.dbpedia.org/

Entity Detection

Entities are noun phrases and consist of a few tokens in an unstructured text. The most popular form of entities are *named entities* like names of persons, locations, and companies. Named entity recognition was introduced in the MUC-6 [Sundheim, 1995] and consisted of three named entities types (names of persons, locations and organizations). Nowadays the term *entity* also includes disease names, protein names, paper titles or journal names, etc. The TAC competition [Ji et al., 2017] for entity detection from texts lists more than 100 different entity types.

The entity detection task is composed of two main steps: (i) *recognition*, where relevant entity mentions in an input text are detected and (ii) *disambiguation* or *resolution*, where entity mentions are mapped to the particular entries in the knowledge base. These steps are (often) loosely coupled, as disambiguation requires the context of the entity mention from the input text. Some tools also detect entities that are not yet found in the knowledge base as proposals for the knowledge base population.

There are two main approaches how the entity detection can be performed. Systems can use an existing named entity recognition tool, e.g [Finkel et al., 2005] or [Straková et al., 2014]. Such tools extract entities for a limited number of types, such as persons, organizations, places, etc. They use statistical machine learning models for sequence labeling, i.e. entities are detected based on the probabilities of context words in the training data. Once the entity mentions are extracted from the text, the next phase involves disambiguation of these mentions by assigning them the entries from the knowledge base [Alani et al., 2003].

On the other hand, there are systems that use entries from an existing knowledge base as a dictionary to guide the extraction. The used knowledge base may contain entities of hundreds of types. The dictionary-based entity extraction is a predominant method in the recent literature [Hakimov et al., 2016, Nguyen et al., 2016, Zwicklbauer et al., 2016]. However, it still presents a challenge, especially for large dictionaries and/or large input collections of texts. In such situations, systems exploit approximate string-matching algorithms and index-based database approaches, see for example [Södergren, 2016, Nguyen et al., 2014, Lipczak et al., 2014].

Relation Extraction

Relation Extraction is a process of finding relationships between entities in a text. Semantic relations are defined over two or more entities related in a predefined way. For example, the relation *is_employee_of* presents a relationship between a person and an organization, the relation *is_acquired_by* presents a relationship between pairs of companies and the relation *is_price_of* links a product name with the currency amount. The extraction of relationships differs from the extraction of entities in one significant way – they express the associations between separate text snippets representing the entities.

In traditional relation extraction systems, extracted relations are used to enhance a knowledge base. Most recent systems adopt an idea of *distant supervision* [Mintz et al., 2009] – they use relations from an existing knowledge base to create general patterns, which are subsequently used to extract further relations. This method is used by the predominant number of tools, which apply a recursive extraction of relations from web, where extracted relations are used to guide the process of extracting further relations.

The very first methods for the relation extraction involved manually designed regular expressions [Nakashole et al., 2011]. With the advances in the field of NLP, systems have applied more and more sophisticated tools and theories of language understanding. For example, a popular method is to exploit a dependency parser – having the knowledge base, the shortest path or spanning tree of the dependency tree could be used to detect patterns applicable on new unseen sentences [Alani et al., 2003, Nguyen and Moschitti, 2011, Nakashole et al., 2013, Rospocher et al., 2016]. The Discourse Representation Theory exploits a first-order-logic style representation of the claims made in language [Kamp, 1981]. The relation extraction is then provided over the formal encoding of the claims made in a discourse spanning multiple sentences [Freitas et al., 2012]. With the development of new lexical language resources, there is a number of systems that use resources such as WordNet [Miller, 1995], FrameNet [Baker et al., 1998], VerbNet [Schuler, 2005] and PropBank [Palmer et al., 2005] to extract relations [Augenstein et al., 2012, Fossati et al., 2018]. Most recent systems exploit various new features based on neural networks, such as word embeddings or convolution and recurrent architectures [Weston et al., 2013, Schlichtkrull et al., 2018, He et al., 2019].

2.2 Data Integration and Linked Data

Sir Tim Berners-Lee invented the World Wide Web in 1989. In 2001, he expressed a need of shift in thinking from publishing data in human-readable documents to machine-readable documents and proposed the idea of Semantic Web [Berners-Lee et al., 2001]. The Semantic Web community developed several knowledge representation approaches that express meaning in formal computer models, from controlled vocabularies, taxonomies, schemas and ontologies to knowledge graphs. The most important concept in the Semantic Web community was presented in 2006 when Berners-Lee [2006] coined the term *Linked Data*.

Linked Data is a method of publishing structured data so that it can be interlinked and thus become more useful in semantic queries [Bizer et al., 2009]. This method exploits standard Web technologies such as Hypertext Transfer Protocol (HTTP), Recourse Description Framework (RDF) and Uniform Resource Identifiers (URIs) in a way that information can be read automatically by computers. It is a set of principles that enables to link related data. The links are recorded in a machine-readable form and published on the Web as a part of the data itself. The web of data is constructed analogously to the web of hypertext (i.e. the web as people know). In the web of hypertext, links present relationships in documents written in HTML. For the web of data, links present relationships between arbitrary things described by RDF. The URIs identify any kind of object or concept. But for HTML or RDF, the same expectations apply to make the web grow [Berners-Lee, 2006]:

1. Use URIs to denote things.

2. Use HTTP URIs so that things can be referred to and looked up by people and machines.

3. Provide useful information about the things when its URI is looked up (use standards such as RDF and SPARQL).

4. Include links to other related things (using their URIs) when published on the Web.

With the Linked Data methodology, one more initiative became significant. The initiative *Linked Open Data* suggests users to publish their data under an open license, which does not impede its reuse for free. As a result, the data can be freely used, reused and redistributed by anyone.

In 2009, Tim Berners-Lee developed a star rating system, in order to encourage people — especially government data owners — along the road to good Linked Open Data:

- **Rating** $\star$
  Data are available on the Web, in whatever format, but with an open license, to be Open Data.

- **Rating** $\star\star$
  Data are available in a machine-readable format (e.g. Excel instead of image scan of a table).

- **Rating** $\star\star\star$
  Data are published in a non-proprietary format (e.g. CSV instead of Excel).

- **Rating** $\star\star\star\star$
  Data are published using open standards formats from W3C (RDF and SPARQL), so they can be looked up by people and machines.

- **Rating** $\star\star\star\star\star$
  Data are linked to other people's data to provide context.

The Resource Description Framework (RDF) is a data model designed by the World Wide Web Consortium (W3C). It is used as a general method for conceptual description and information modeling in web resources, using a variety of syntax notations and data serialization formats. RDF was adopted as a W3C recommendation in 1999. The RDF 1.0 specification was published in 2004, the RDF 1.1 specification in 2014.

The RDF data model is similar to classical conceptual modeling approaches, such as entity–relationship or class diagrams [Halpin and Morgan, 2008]. It is based on the idea of making statements about resources using expressions of the form [*subject*, *predicate*, *object*], known as triples.

- **Subject** denotes a resource, it is an URI identifying the described thing.

- **Predicate** indicates what kind of relation exists between subject and object. For example, it is the name of the thing from the subject or date of birth, or an employer or someone a person knows. The predicate is also identified by a URI. These predicate URIs come from *vocabularies* – collections of URIs that can be used to represent information about a certain domain.

- **Object** can either be a simple literal value, like a string, number, or date; or a URI of another resource that is related to the subject.

A collection of RDF statements intrinsically represents a labeled, directed graph called *knowledge graph* [Singhal, 2012]. It has become prevalent in both of industry and academic circles these years, to be one of the most efficient and effective knowledge integration approaches.

In Appendix A we provide a simple illustration how to convert existing data according to the Linked Open Data principles with the ⋆⋆⋆⋆⋆ rating.

2.3 Information Extraction for Legal Domain

In the past decade, several approaches to entity recognition in legal texts were reported. Most of them address the task of references detection, especially in court decisions. Court decisions are texts with no unified style of citations. Even more, there are different opinions what to cite. Some judges cite other court decisions only, some of them cite various types of literature as well, some of them cite *everything* (blogs, internet sources, Bible, novels, etc.). In literature, we can distinguish two basic types of approaches to the entity detection – lookup and rule-based methods.

The lookup methods require a list of entities (sometime called a dictionary or *gazetteer*) and then simply tag all mentions of entities in texts [Dozier et al., 2010, de Maat et al., 2006]. However, when an entity is missing in the list, it will not be recognized. Most typically, new law names may be defined – these names will be missed unless they are added to the list.

By looking at development data, one can define a rule-based system with a set of rules that recognize the majority of entities in the data [Palmirani et al., 2003, Bacci et al., 2012]. Rule-based systems typically use various NLP tools to identify patterns, starting from sentence segmentation, tokenization and POS tagging [Bruckschen et al., 2010], syntactic information [Quaresma and Gonçalves, 2010] or different text similarity techniques [Panagis et al., 2017]. Other systems provide also a deeper analysis of the detected references. Panagis and Sadl [2015] explain the legal relevance of citations and cited cases, as well as their normative force. In the European Union context, the community focuses also on the systems that work for multiple languages [Agnoloni et al., 2017]. This presents an issue especially because of the linguistic diversity and specific peculiarities of national legal citation practices. In general, development of rule-based systems requires a large amount of effort from experienced rule writers. Maintenance of such rule sets can be tricky because rules often contain intricate interdependencies that are easy to forget and make modification risky.

Aside from the lookup and rule-based methods, there are almost no systems based on statistical models. One can see two possible reasons for such situation – development of statistical models requires manually annotated training data and the legal domain community has deep legal knowledge rather than machine learning skills.

Relation Extraction Systems

There are several initiatives that try to extract structured data from legal texts and capture it in a machine-readable format using Semantic Web and the Linked Data principles. Rubino et al. [2006] propose the LKIF Ontology for legal concepts. It is a detailed ontology of various legal concepts, including roles (epistemic roles, functions, person roles, etc.) and actions (processes which are performed by some actors in roles).

The CEN MetaLex project provides standards for representing acts as Linked Open Data [Boer, 2009]. MetaLex was successfully implemented by Dutch [Hoekstra, 2011] and British[3] legislation.

The INTLIB project follows the rules given by CEN MetaLex and applies them to Czech legal texts [Nečaský et al., 2013, Holubová et al., 2014]. In addition, the CEN MetaLex publication format was extended for a new kind of relationships not only between acts (as supported by MetaLex) but also between other sources of law (e.g., court decisions). The INTLIB project initiative is described in detail in Chapter 7.

The EUCases project presents a Pan-European platform for transforming multilingual legal data into Linked Data [Boella et al., 2015]. It includes two end-user applications: (i) the ConsumerCases service provides access to a multilingual collection of national court decisions and (ii) the EULinksChecker assists legal professionals while editing or browsing documents by identifying and establishing connections with regulations and legal ontologies.

[3]http://legislation.gov.uk

In 2017, the European Legislation Identifier project (ELI) was introduced.[4] It is a system to make legislation available on-line in a standardized format, so that it can be accessed, exchanged and reused across borders. ELI is based on a voluntary agreement between the EU countries. It proposes technical specifications for (i) identifying legal information (using URIs), (ii) metadata and (iii) a specific language for exchanging legislation in machine-readable formats.

Probably the most recent system is the GDPR Linked Data Resource [Pandit et al., 2018]. It links specific articles related to GDPR and uses ELI for exposing the GDPR as Linked Data.

Besides the work that propose new ontologies, there are also several works which explore specific legal concepts in detail. The LegalRuleML project [Nazarenko et al., 2016] aims to produce an interchange language for the legal domain using logic and other formal techniques [Bench-Capon et al., 2012]. Agnoloni et al. [2010] provides a study about the obligation legal concept and Gandon et al. [2017] propose the ontological representation of normative requirements.

2.4 Czech Legal Informatics

In this section we present a current state of the Legal Informatics in the Czech Republic. We start with the official government initiative, then we list the relevant commercial systems and provide a survey of the academic research over legal texts.

eLegislation and eLegislature

In the Czech Republic, there are several systems that provide access to different legal documents (acts, decrees, court decisions and other) in an electronic form. Some of them even claim to provide the consolidated[5] versions of acts, nevertheless none of them is an official version to be approved by the Head of the Parliament (who is responsible for it). In fact, even the members of the Czech Parliament work with these systems, i.e. with the unofficial data.

The solution of this problem is being provided in two closely related projects proposed by the Czech Government – eLegislation (*eSbírka*) and eLegislature (*eLegislativa*). The eLegislation system should publish official electronic versions of legal acts, including (current and past) consolidated texts. The eLegislature system is aimed to be a tool for law-making and negotiation. It should allow to write changes of acts directly into their consolidated text.

An official[6] deadline for the completion of the project is scheduled for 31 December 2019. The relevant legislation will become effective on 1 January 2020. However,

[4]https://eur-lex.europa.eu/eli-register/

[5]Consolidation consists of the integration in a legal act of its successive amendments and corrigenda. It provides more transparency and easier access to law.

[6]https://www.mvcr.cz/clanek/esbirka-a-elegislativa.aspx

the most recent news inform that the system will not be ready before 2021.[7]

Existing Systems

In this section we provide a brief overview of existing systems that enable to work with the Czech legislation.

ASPI[8] from the Wolters Kluwer, Czech Republic enables to browse and query electronic versions of legislation and related documents. In addition, it provides access to related basic literature where various acts are explained, commented and discussed. It supports full-text search supporting also all grammatical forms of Czech or Slovak respectively. The search can cover all texts, or selected parts such as titles, content, appendices, notes, or tables of contents. The system enables filtering the documents according to metadata.

Beck-online[9] is an on-line application from C. H. Beck publishing house. It enables users to browse and query electronic versions of various legal documents, such as documents from the Collection of Laws of the Czech Republic, court decisions and comments to these documents. Documents are manually analyzed by domain experts, so that documents are inter-linked to each other and labeled by keywords which can be subsequently queried by users in the metadata search.

CODEXIS[10] is an on-line software designed to work with information from all areas of law. The application includes both the legislation of the Czech Republic and the legislation of the European Union, documents from the decision-making activities of Czech and European courts, professional literature, articles, commentaries and others useful sources of information. Individual legislation is linked to implementing regulations, interpretations, case-law and other documents. The system allows users to search documents using metadata and keywords.

Public Administration Portal[11] involves various information for citizens, entrepreneurs and businessmen, foreigners living in the Czech Republic, and public authorities. The functionalities involve also a module for simple full-text search in legislation. It enables searching in texts of acts, their titles or according to their number.

Research in the Czech Legal Informatics

In the Czech Republic, first research activities in computer processing of legal documents have been done by Knapp [1963]. Currently, areas such as legal systems development, legal language, quantitative analysis of legal documents or various

[7]https://ekonomicky-denik.cz/esbirka-a-elegislativa-nejdrive-v-roce-2021/
[8]http://www.systemaspi.cz/
[9]http://beck-online.cz/
[10]http://codexis.cz/
[11]https://portal.gov.cz/

exploiting of AI in legal domain represent different subfields of *Legal Informatics* [Cvrček, 2010].

During the 1970s and 1980s, the first Czech thesaurus was created by V. Knapp and J. Cejpek [Knapp and Cejpek, 1979]. From '90s, very first legal systems have been designed. They allowed to keep full texts of legal documents which lead to very first quantitative analysis works [Cvrček et al., 1999] and subsequently to analysis of relations between various legal terms.

From 2004 to 2006, Czech Republic was a member of the international project LOIS[12]. The goal of LOIS was to localize a legally oriented WordNet into a number of European languages.

Since 2007, experience and ideas from the LOIS project were exploited and extended in the project PES (Právnický elektronický slovník, *Legal Electronic Dictionary*, in English, [Cvrček et al., 2012]). Currently, the PES project includes not only the legal term dictionary, but it is a collection of various dictionaries, legal ontologies, corpora and linguistic utilities. In addition, the data and tools are updated regularly.

The processing of Czech legal texts has been overviewed during the work on the Dictionary of law terms [Pala et al., 2010]. The authors used partial parsing to extract noun groups as the main candidates for legal terms, and they explored the valency frames of verbs to link together the established law terms [Pala and Mráková, 2010].

Most recent research activities in the field are also presented in the annual conference Czech Law and Information Technology (*České právo a informační technologie*). Probably the most relevant research activity is work of Harašta and Šavelka [2017] that deals with references in Czech legal documents.

2.5 Syntactic Parsing of Long Sentences

One of the most significant features of the legal language is a high frequency of long and complex sentences. In Chapter 5 we provide the evidence of difficulty in long sentences dependency parsing – as the sentence length increases, the unlabeled attachment score decreases. In this thesis we present a new method for the complex sentences dependency parsing. This method assumes segmentation of input sentences into clauses and does not require to re-train a parser of one's choice.

In literature there are several approaches which deal with the idea of dividing a parsing process into several parts. The idea of cascaded parsing exploits a cascade of specialized parsers instead of having one very complex general parser [Abney, 1996, Ciravegna and Lavelli, 1999]. The identification of chunks, syntactically related non-overlapping groups of words [Tjong Kim Sang and Buchholz, 2000],

[12]http://www.loisproject.org/

was used mainly in shallow parsing strategies [Federici et al., 1996]. Clausal parsing was designed to parse Hindi texts [Husain et al., 2011].

However, there is no work on exploiting chunks for full-scale parsing. A very interesting approach to dividing the parsing process into several parts has been introduced in the XDG theory [Debusmann et al., 2005]. Most recent approaches to dependency parsing focus almost exclusively on improving full-scale parsing algorithms using mostly neural networks [Pei et al., 2015, Weiss et al., 2015, Zhu et al., 2015].

In this thesis, we address the issue of parsing sentences that are segmented into clauses. The ideas and concepts of segmentation of Czech sentences are presented by Kuboň [2001], Kuboň et al. [2007], and Lopatková and Holan [2009]. They present the concept of segments and show that it is possible to draw the segmentation charts which reflect the mutual position of segments in a complex sentence without applying syntactic parsing of the whole sentence first. The method is based on the identification of separators (segment boundaries) and their classification.

Lopatková et al. [2012] show how clauses forming complex sentences can be identified based on the sentence segment annotation. In addition, they present the project aiming at building a collection of Czech sentences enriched with manually annotated clauses and their relationships. Krůza and Kuboň [2014] use this collection to develop an automatic procedure for recognizing clauses and their mutual relationship. Another automatic procedure for clause identification over dependency trees is introduced by Bejček et al. [2013] and was used for the clause annotation of the Prague Dependency Treebank 2.5.

3. Czech Legal Text Treebank

In this chapter, we present the main language resource that we created and subsequently used in our work. The Czech Legal Text Treebank (CLTT) is a member of the family of the Prague dependency treebanks. It contains 1,121 sentences manually annotated on several layers. It contains Czech texts exclusively from the legal domain, namely the documents from the Collection of Laws of the Czech Republic. We published its two versions — CLTT 1.0 [Kríž et al., 2015, Kríž et al., 2016] and CLTT 2.0 [Kríž and Hladká, 2017, Kríž and Hladká, 2018].

3.1 Introduction

Czech language is a prominent language when considering the existing language resources. However, we still identified several reasons why we decided to create another treebank.

(i) The RExtractor system presented in Chapter 4 implements an extraction pipeline which processes input texts by linguistically-aware tools and extracts entities and relations using queries over dependency trees. The purpose of creating CLTT is to provide gold-standard data for the RExtractor performance evaluation. The CLTT dataset allowed us to evaluate each part of the extraction pipeline from tokenization to detecting semantic relations.

(ii) Because of lack of any Czech gold legal-domain data, we have used the parser trained on data from the Prague Dependency Treebank, i.e., on newspaper texts. We decided to create a gold data set from the legal domain, in order to get at least a rough idea about the performance of the parser on a domain that is different from the domain of the parser's original training data.

(iii) One of the most distinctive feature of the legal texts is a high frequency of long and complex sentences. In Chapter 5 we show that as the sentence length increases, the parser's performance decreases. Therefore we propose a novel method for the complex sentences dependency parsing and exploit the CLTT data as a great source of manually annotated complex sentences.

Annotation principles applied in the CLTT fit the framework originally formulated in the Prague Dependency Treebank (PDT) project. A theoretical background of the PDT project has its roots in the Prague School of Functional and Structural Linguistics founded in 1926 by linguists Vilém Mathesius, Roman Jakobson and Bohumil Trnka. The Prague School influenced also professor Petr Sgall and his colleagues. In 1960s they proposed a new linguistic paradigm of explicit description of language — the Functional Generative Description (FGD, Sgall [1967]). The FGD theory is characterized by (i) an inclusion of an underlying syntactic layer (*tectogrammatics*) into the linguistic description, (ii) a usage of dependency syntax and (iii) a specification of a formal account of the information structure (topic-focus articulation) of the sentence and its integration into the description.

The annotations in PDT are organized into different *layers* of annotation, analogously to the stratification language description proposed in FGD. The key ones are the following layers of annotations:

- **Word Layer** (`w-layer`)
  The text is segmented into documents and paragraphs and individual tokens are recognized and associated with unique identifiers.

- **Morphological layer** (`m-layer`)
  The sequence of tokens of the word layer is segmented into sentences. Sentence annotation consists of attaching several attributes to the tokens of the w-layer, such as morphological lemma and tag.

- **Analytical layer** (`a-layer`)
  A sentence at the analytical layer is represented as a dependency tree. The dependency relation is captured by an edge between the two nodes. Each node has assigned a set of attributes, e.g. the attribute `id` contains a unique identifier of the node. The attribute `ord` contains the token position in the original sentence. A type of a dependency relation is represented by the analytical function attribute `afun`. The attributes `is_member` and `is_parenthesis_root` mark proper coordination, apposition and parenthesis interpretation.

- **Tectogrammatical layer** (`t-layer`)
  A sentence is represented as a tree that reflects the underlying (deep) structure of the sentence. The nodes stand for auto-semantic words only and are labeled by *grammatemes*. Grammatemes provide information about the node that cannot be derived from the structure.

The first version of PDT was published in 2001 [Hajič et al., 2001] and contained a morphology and surface syntax annotations. In almost two decades, a broad team of contributors added new annotation layers, especially the tectogramatical layer with complex underlying syntactic and semantic annotation. PDT 1.0 was followed by version 2.0 [Hajič et al., 2006], version 2.5 [Bejček et al., 2012], version 3.0 [Bejček et al., 2013] and finally by the most recent version 3.5 [Hajič et al., 2018].

The sentences in CLTT are taken from two documents from the Collection of Laws of the Czech Republic, namely (i) The Accounting Act, 563/1991 Coll. and (ii) The Decree on Double-entry Accounting for Undertakers, 500/2002 Coll. The manual annotations in CLTT are organized into layers analogously to PDT. However only some of them are available, namely the w-layer, m-layer and a-layer. The tectogrammatical layer is not available for CLTT. On the other hand, two new layers not available in PDT are used – the layer of named entities (e-layer) and the layer of semantic relations (r-layer).

3.2 Legal Texts Segmentation and Tokenization

With respect to the complexity of legal text sentences, we formulated an original preprocessing strategy to make the process of subsequent manual syntactic annotation as simple and painless as possible. The CLTT Preprocessing Strategy described in detail in this section consists of the following steps:

1. Standard segmentation and additional Complex sentence segmentation
2. Standard tokenization and subsequent Re-tokenization
3. Automatic parsing

The first two steps of the strategy target to eliminate long and complex sentences. Such simplification is beneficial for both human annotators and automatic parsers. Human annotators (1) correct a lower number of tokens; and (2) work with the smaller trees on the screen of the tree editor. As a result, the whole sentence is more clear and easier to check. Automatic parsers achieve significantly better performance on shorter sentences. In the last step of the strategy, we automatically parsed the sentences in CLTT. We believe that correcting of dependency trees presents an easier task for human annotators than manual annotation from scratch. An animated illustration of the strategy is available on-line.[1]

We implemented the CLTT Preprocessing Strategy using the Treex framework [Popel and Žabokrtský, 2010]. In Treex, input texts can be processed by a rich set of NLP tools (represented by *blocks*) organized into pipelines. In this section, we exploit the following existing Treex blocks:

- `W2A::CS::Segment` for standard sentence segmentation,
- `W2A::CS::Tokenize` for standard tokenization,
- `W2A::CS::TagMorphoDiTa` for morphological analysis and tagging. The block exploits the MorphoDiTa software [Straková et al., 2014].
- `W2A::CS::ParseMSTAdapted` for automatic parsing. The block exploits the MST parser [McDonald et al., 2005] adapted for Czech by Novák and Žabokrtský [2007].

Both complex sentence segmentation and re-tokenization are described in details in the following subsections. In the last subsection, we present a comparison of the original sentences with those simplified by the proposed preprocessing strategy.

3.2.1 Complex Sentence Segmentation

Almost all natural language processing tasks start with sentence segmentation and tokenization of input texts. Tokenization is a process of separating a text into meaningful units, e.g. tokens. Sentence segmentation is a process of separating a text into sentences, i.e. identifying sentence boundaries. We call the existing segmentation and tokenization procedures as *standard* as their aim is to work with reliable accuracy on all input texts.

[1] https://prezi.com/ecm-eeld1yfx/cltt-preprocessing-strategy/

However, in legal texts, whole sentences produced by the standard segmentation are still too complex and long. Therefore we propose an additional complex sentence segmentation procedure. The procedure splits long and complex sentences into *segments*. A *complex sentence* is a sentence containing at least two segments. A *segment* is a part of a sentence between two enumeration markers. As a result, manual annotation becomes more annotator friendly.

Example 3.1 presents a complex sentence with three subsections enumerated by letters (*a*, *b* and *c*). The segmentation procedure splits the sentence into four segments, as presented in Table 3.1.

Example 3.1

> The General Directorate of Customs
> a) is an administrative body exercising authority to customs offices,
> b) administers customs duty in compliance with the EU regulation,
> c) presents a customs authority with the competences of a police authority.

Segmented complex sentence	Sentence identifiers
The General Directorate of Customs	`document`*1*`-sentence`*1*`-section`*0*
a) is an administrative body exercising authority to customs offices,	`document`*1*`-sentence`*1*`-section`*1*
b) administers customs duty in compliance with the EU regulation,	`document`*1*`-sentence`*1*`-section`*2*
c) presents a customs authority with the competences of a police authority.	`document`*1*`-sentence`*1*`-section`*3*

Table 3.1: An example of a complex sentence segmented into 4 segments with particular identifiers as used in CLTT.

Technically, the Complex Sentence Segmentation procedure is implemented as a Treex Block `CLTT::Segment`. The segmentation algorithm uses rules for identifying different styles of enumeration. If a new segment is detected, the algorithm assigns a unique identifier for it. The identifiers express a hierarchical structure that helps human annotators to be oriented in complex sentences.

`document`*document_id*–`sentence`*sentence_id*–
[`section`*section_id*–[`subsection`*subsection_id*]]

Figure 3.1: The sentence identifiers schema used in CLTT.

The CLTT sentence identifier schema is presented in Figure 3.1 and it consists of the following elements:

- **Document identification** – `document`*document_id*
  CLTT is distributed in several files. Each sentence identifier starts with the

document_id to determine a PML file where the sentence is stored.

- **Sentence identification** – `sentence`*sentence_id*
  This identifier provides a unique sentence identification in a particular PML file.

- **Section identifier** – `section`*section_id*
  For complex sentences, this identifier determines the first enumeration layer in the sentence. We assign the `section0` identifier for the segment that introduces the enumeration.

- **Subsection identifier** – `subsection`*subsection_id*
  For complex sentences, this identifier determines the second enumeration layer in the sentence, i.e. an enumeration nested in the enumerated item. We assign the `subsection0` identifier for the segment that introduces the nested enumeration.

In fact, two numbering layers (i.e. section and subsection identifiers) were enough to cover all complex sentences in CLTT. However, this strategy could be easily extended to any number of nested enumerations.

3.2.2 Re-tokenization

To illustrate the disadvantages of the standard tokenization procedure, we assume the sample sentence from Example 3.2.

Example 3.2

> Účetní jednotky tvoří opravné položky podle ustanovení § 16, § 26, § 31, § 55 a § 57 a neoceňují majetek podle § 27, § 14, § 39, § 51—55, § 58 a § 60.
>
> *In English*:
> Accounting units create fixed items according to § 16, § 26, § 31, § 55 and § 57 and not apply § 27, § 14, § 39, § 51—55, § 58 and § 60.

Figure 3.2 presents its dependency tree created automatically over the tokens detected by the standard tokenization procedure. One can see that all inter-document references are split into the standalone tokens. As a result, the standard tokenization applied on the legal texts causes *overgeneration* of tokens. This has a bad influence on both automatic parser and annotators who check the dependency trees. Because of these reasons, we included *re-tokenization* to our preprocessing strategy. Figure 3.3 presents the dependency tree of the sample sentence from Example 3.2 parsed automatically over the re-tokenized tokens.

To improve the standard tokenization procedure, we started to consider references and enumeration markers as special entities which are not tokenized into

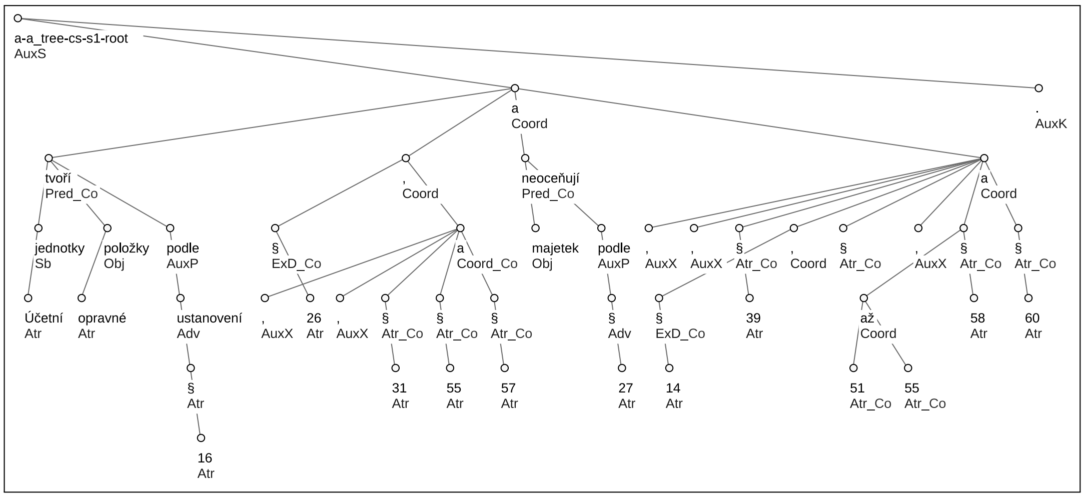


Figure 3.2: The dependency tree of the sample sentence parsed automatically. The sentence was tokenized by the standard Treex procedure.

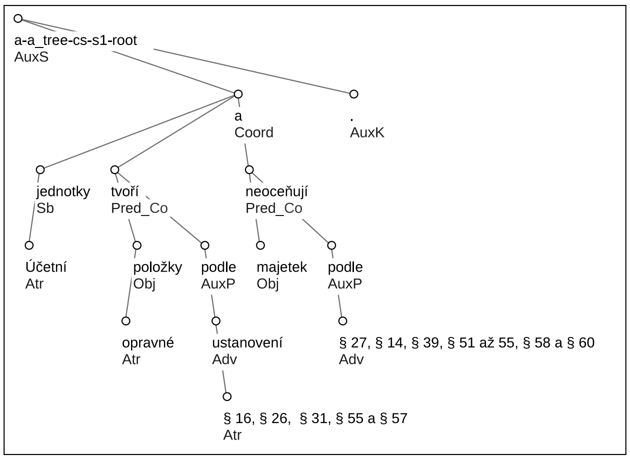


Figure 3.3: The dependency tree of the sample sentence parsed automatically. The sentence was tokenized by the standard tokenization procedure and subsequently re-tokenized using our automatic re-tokenization procedure.

more detailed tokens. The re-tokenization procedure is described in details in Appendix B.1.

3.2.3 CLTT Preprocessing Strategy Statistics

Using the complex sentence segmentation procedure, 92 out of 1,121 sentences in CLTT were identified as complex ones. These sentences were segmented into 507 segments. The average sentence length decreased from 35.9 to 26.2 tokens per sentence. Using the re-tokenization procedure, the number of tokens in CLTT decreased from 40,267 to 34,410. The average segment length decreased from 26.2 to 22.4 tokens per segment. Figure 3.4 presents how the sentence length distribution in CLTT was changed after the complex sentence segmentation and re-tokenization.

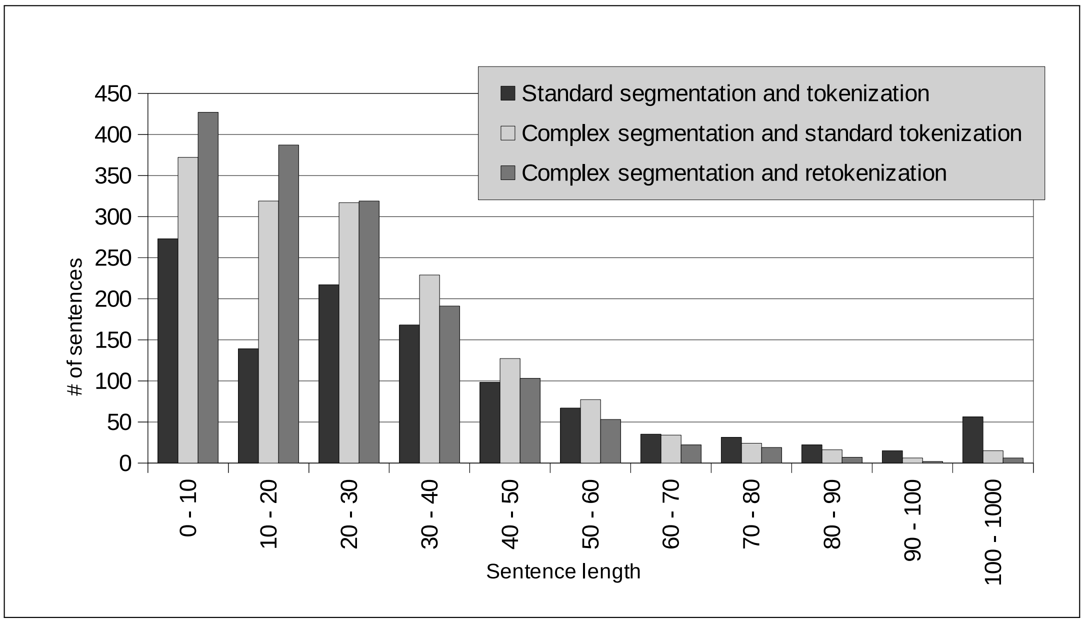


Figure 3.4: A comparison of sentence length distributions – the original segmentation (in blue), the complex sentence segmentation (in yellow), and both complex sentence segmentation and re-tokenization (in red).

3.3 Syntactic Annotation

The manual syntactic annotation of CLTT spans the first three layers used in PDT — word, morphological and analytical layer. Its manual annotation is in line with the PDT Annotation Guidelines [Hajič et al., 1999].

The human annotators checked and corrected the output of the automatic parser. As we described in Section 3.2, the complex sentences were split into individual segments and the annotators checked each segment individually – both the tree structure and the analytical function assignment. To be able to build final complex sentence dependency trees from partial segments, we enhance the manual annotation work-flow and introduced *inter-segment links* that allow to create dependencies between nodes from different segments.

The idea of the inter-segment linking is described and illustrated in Section 3.3.1. In Section 3.3.2 we present the annotation tool TrEd with our custom extension that implements the inter-segment linking. It also tracks all corrections made by human annotators. As a result, Section 3.3.3 presents several interesting statistics and lists the most frequent errors of the automatic parser applied on the legal texts.

3.3.1 Inter-segment Linking

Let us assume the sample complex sentence from Example 5.1 and focus on its annotation:

> The General Directorate of Customs
> a) is an administrative body exercising authority to customs offices,
> b) administers customs duty in compliance with the EU regulation,
> c) presents a customs authority with the competences of a police authority.

In the first step of the CLTT Preprocessing Strategy, the sample complex sentence is segmented into four segments, see Table 3.1. Subsequently, each segment is tokenized, re-tokenized (in this particular example, enumeration markers are re-tokenized, e.g. tokens "`a`" and "`)`" are joined into the "`a)`") and parsed by the automatic parser.

During the manual annotation, segments' dependency trees are firstly corrected individually. After that, an annotator builds the final complex sentence dependency tree using the inter-segment links. To provide links in line with the PDT annotation guidelines the annotator should respect the following instructions:

- The complex sentence presents a coordination of three different predicates (*is*, *administers* and *presents*). In PDT, the head of the coordination is the comma from the `section2` segment.

- To annotate coordination, two formal steps must be done: (i) the particular member of the coordination has to depend on the coordination head; and (ii) the root node of the coordinated subtree has to have a special attribute `is_member` set to *True*.

- The second coordination expression (a comma from the segment `section1`) has to depend on the coordination head, with the `is_member` attribute set to *False*.

- The first segment (`section0`) contains a subject of all three predicates. In PDT, the particular subject has to depend on the coordination head as well, however, the `is_member` attribute has to be set to *False*.

- The full-stop from the last segment (section3) has to depend on the technical root of the complex sentence. In CLTT, the complex sentence technical root is the technical root from the segment with the coordination head. (i.e the segment `section2`).

As the result, the annotator created the inter-segment links as they are schematically presented in Figure 3.5.

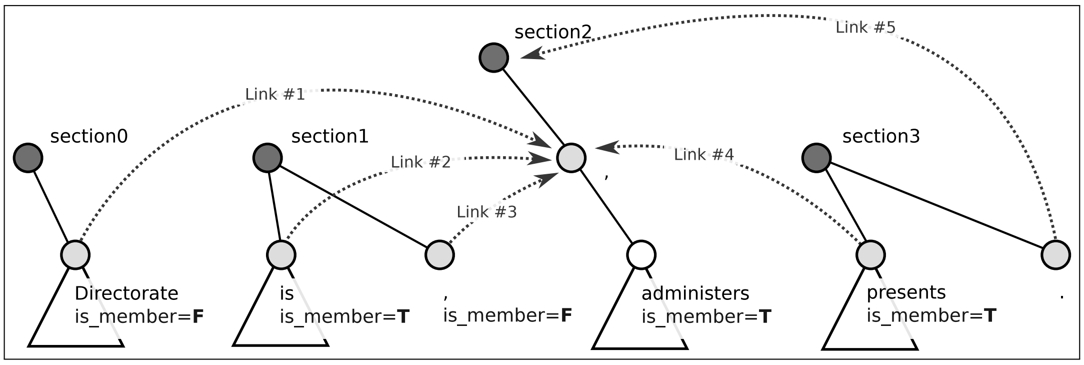


Figure 3.5: A schema of the correct inter-segment linking for the sample complex sentence from Example 5.1.

3.3.2 Annotation Tool

To provide both manual correction of dependency trees and inter-segment linking we exploit the tree editor TrEd.[2] TrEd is a fully customizable and programmable graphical editor and viewer for tree-like structures. It was used as the main annotation tool for syntactical and tectogrammatical annotations in PDT.

We implemented a TrEd's extension that introduced several features. The most important one supports inter-segment linking. The CLTT TrEd extension is called *Czech Legal Text Treebank 2.0* and it is available in the TrEd's extension repository.[3] In addition, the CLTT extension comes with several macros and shortcuts that made the annotation process as simple as possible. We describe its important features in details in Appendix B.2.

3.3.3 Annotation Statistics

The CLTT TrEd extension enables tracking of dependency corrections that an annotator is doing. In Table 3.2 we provide the total number of corrected dependencies and analytical functions. We also provide the data for simple and complex sentences separately.

	Simple sent.		Complex sent.		Total
Sentences	1,029	91.79 %	92	8.21 %	1,121
Segments	1,029	66.99 %	507	33.01 %	1,536
Nodes	24,290	70.59 %	10,120	29.41 %	34,410
Dependency corrections	4,608	18.97 %	2,448	24.19 %	7,056
Afun corrections	3,051	12.56 %	2,369	23.41 %	5,420

Table 3.2: Difference in error rates in the segmented vs. non-segmented sentences

[2] https://ufal.mff.cuni.cz/tred/
[3] http://ufal.mff.cuni.cz/tred/extensions/

One can see that there is a substantial difference (over 50% higher, relatively) between the percentage of errors in the segmented sentences vs. function assignment errors in the non-segmented ones. We can see two possible reasons for these results:

- Segments are not typical sentences. In most cases they represent a coordination member. Therefore one or more important analytical functions may be missing, e.g. subject, predicate or object. This may cause problems for parser trained on *standard* sentences.

- During the manual correction process, the annotator assigned analytical functions as they should be in original, complex sentence. On the other hand, automatic parser assigns functions locally, without the knowledge of the whole sentence. Table 3.3 presents the functions that have to be corrected most frequently.

Afun	# of corrections	% of corrections
Atr	1,149	21.20 %
Obj	896	16.53 %
Adv	889	16.40 %
graphical	552	10.18 %
Sb	457	8.43 %
other	1,477	27.25 %

Table 3.3: Manual corrections of analytical functions.

We can see that the overall dependency error rate is visibly higher than has been reported for the newspaper and magazine documents, as reported by Novák and Žabokrtský [2007]. However, the parser accuracy (dependency-wise) does not differ as much between the segmented and non-segmented sentences, which is the positive consequence of complex sentence segmentation, leading to similar and reasonable average segment sizes in around 21-25 words per segment.

3.4 Entities Annotation

In this section we describe the manually annotated layer of accounting entities that we call e-layer in CLTT. Historically, the annotation of entities presents the first layer of annotation that has been done in the CLTT documents. The manual annotation has been done in the INTLIB project. Having the annotated documents, we decided to enhance the existing annotations and created CLTT.

The accounting experts from the INTLIB team manually annotated relevant accounting terms in the source documents. In addition, each annotated entity was classified into one of 25 categories presenting general accounting concepts, see Table 3.4.

During the manual annotation, the annotators followed a simple set of rules:

account	general subject	obligation
accounting concept	general term	period
accounting report	incomes	regulation
activity	institution	revenues
agreement	legal person	right
assets	liabilities	state
costs	method	taxes
document	moment	
expenses	natural person	

Table 3.4: A list of categories in the Accounting Dictionary.

- An annotated entity should be as specific as possible. If such annotation contains a more general entity, it should not be annotated. For example, an annotator should annotate the text *účetní jednotky v úpadku* (*accounting units in decay*) as an entity, not *účetní jednotky*, although it is a valid and well-known accounting term.

- An annotated entity should consist of a continuous sequence of words. It was not allowed to annotate several fragments into one entity. For example, in the text *výkaz (rozvaha) zisku a ztráty* (*profit and loss statement (balance sheet)*) an annotator should avoid to annotate fragmented entity *výkaz zisku a ztráty* even though it is a valid and well-known accounting term.

- Coordinations can be annotated only if a particular coordination is typical and frequent in the accounting domain. For example, an annotator should annotate the text *hmotný a nehmotný majetek* (*tangible and intangible assets*), but not *společníci a členi družstva* (*shareholders and cooperative members*).

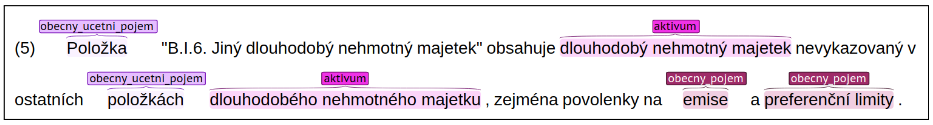
- If there is no relevant category to classify the particular entity, one should use the *general term* category.

- No overlapping entities are allowed.

The texts were annotated by two annotators. The annotators worked together and had to consult problems or tricky annotations and so we consider the annotation to be adjudicated and consistent.

For the manual annotation, we used the Brat annotation tool [Stenetorp et al., 2012]. Figure 3.6 displays the annotated entities in Brat and Figure 3.7 displays a sample sentence in the final CLTT e-layer, i.e. its dependency tree with the annotated entities. For the technical details, see Appendix B.3.

As a product of the annotation, we selected the unique accounting entities and created the *Accounting Dictionary*. The dictionary entry structure is described in Appendix B.4. The dictionary was published as a part of the CLTT 2.0 release.


obecny_ucetni_pojem Položka; aktivum dlouhodobý nehmotný majetek; obecny_ucetni_pojem položkách; aktivum dlouhodobého nehmotného majetku; obecny_pojem emise; obecny_pojem preferenční limity

(5) Položka "B.I.6. Jiný dlouhodobý nehmotný majetek" obsahuje dlouhodobý nehmotný majetek nevykazovaný v ostatních položkách dlouhodobého nehmotného majetku , zejména povolenky na emise a preferenční limity .

Figure 3.6: Accounting entities annotation in the Brat annotation tool.

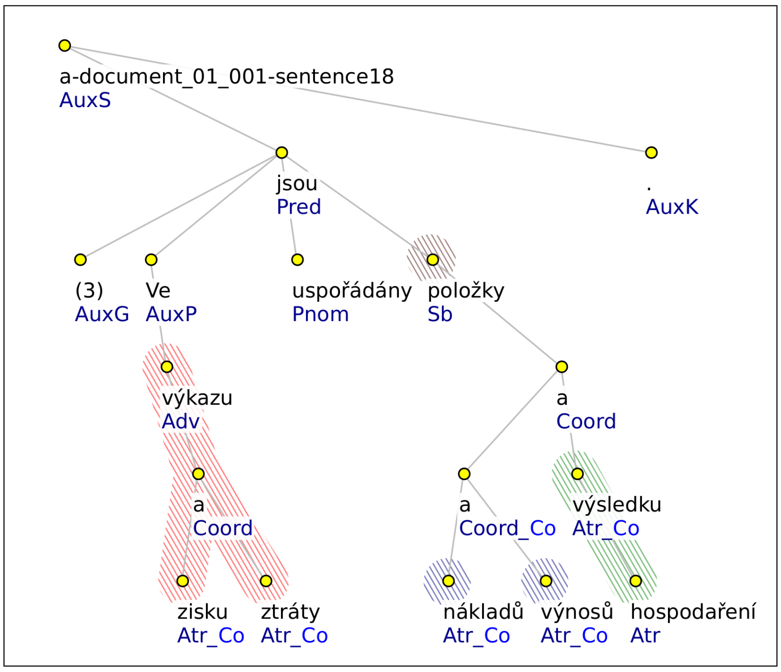


Figure 3.7: CLTT dependency tree with highlighted entities.

There are two main use cases how to work with the Accounting Entities in CLTT:

1. Users can use the plain text files and standalone JSON files where each entity is described, see Appendix B.5 for the technical details.

2. Users can use the TrEd editor and the modified PML files enriched with the entities annotation. It allows to create tree queries in which users can request nodes that belong to a particular entity or entity category. Please, refer to Appendix B.6 for the technical details.

In total, the annotators identified 8,554 entities in the CLTT sentences. Figure 3.8 presents the relative frequency distribution of sentences with the particular number of entities. In total, there are 152 (13.56%) sentences with no entity. In average, each sentence contains almost 8 entities. Table 3.5 presents top 10 most frequent entities.

Out of 8,554 annotated entities, 1,630 are unique. All of them were included as entries into the Accounting Dictionary. Figure 3.9 presents the relative frequency of entity categories as they appear in the dictionary as well as in the CLTT sentences.

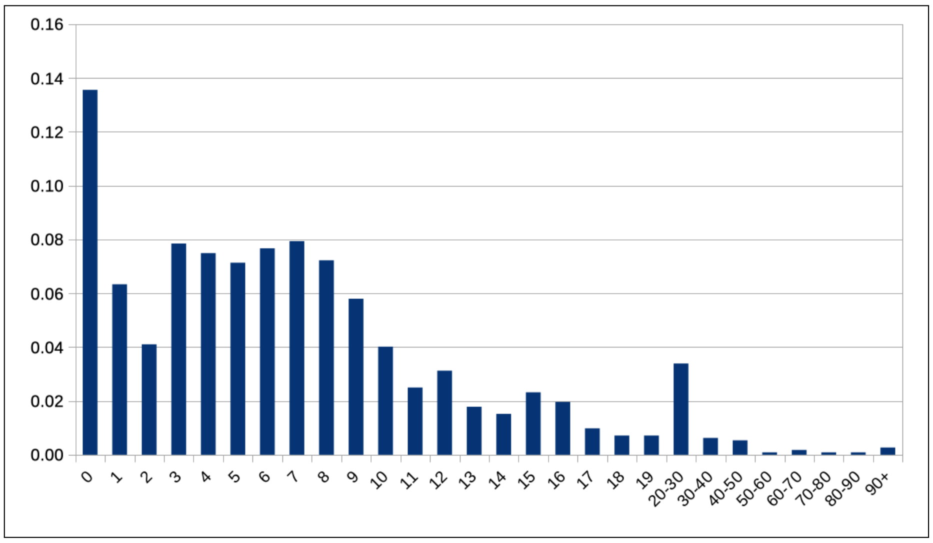


Figure 3.8: Relative frequency distribution of sentences with the particular number of entities. For example, there are almost 14% of sentences with no entity.

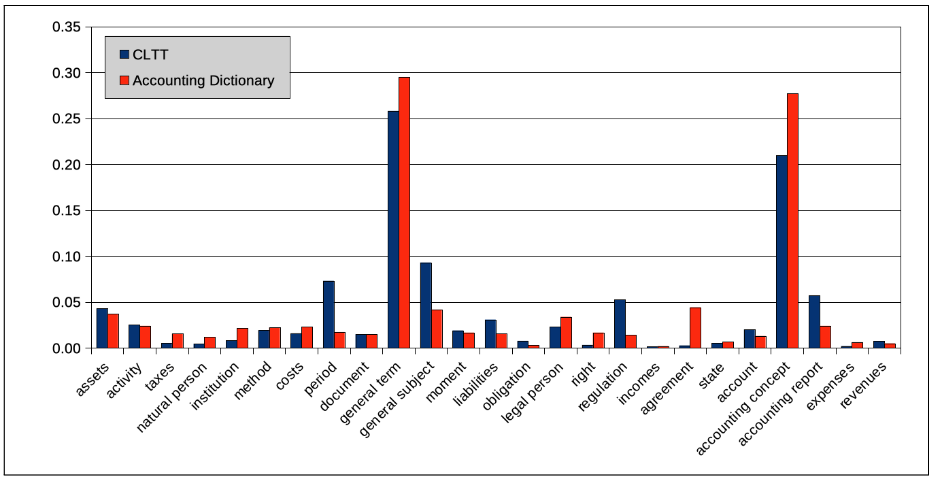


Figure 3.9: Relative distribution of different Accounting Dictionary Categories as they appear in CLTT (in blue) and Accounting Dictionary (in red).

Entity	Category	Frequency	
		absolute	relative
Accounting unit	General term	767	0.09
Day	Period	330	0.04
Item	General term	241	0.03
Financial statement	Accounting report	160	0.02
Property	Assets	155	0.02
Content	General term	148	0.02
Account	Account	139	0.02
Accounting period	Period	130	0.02
Act	Regulation	125	0.01
Liability	Liability	119	0.01

Table 3.5: Top 10 most frequent Accounting Dictionary entries.

3.5 Semantic Relations Annotation

In this section we describe the layer of manually annotated semantic relations that we call r-layer in CLTT. The annotation task is defined as annotating triples (*subject*, *predicate*, *object*), where *subjects* and *objects* are entities from the e-layer and *predicate* is an expression that links subjects and objects. Most typically, it is a verb.

The names of triple members (*subject*, *predicate* and *object*) come from the field of Information Extraction. Although this naming convention is motivated linguistically, it does not express any of linguistic phenomena.

We manually annotated three different types of relations in CLTT:

- **Definitions**
  link entities with their definitions.

- **Rights**
  link subjects with their rights, i.e. that someone has the right to do something.

- **Obligations**
  link subjects with their obligations, i.e. that someone is obligated to do something.

To annotate a definition, an annotator links an entity with its definition. Definitions in legal texts most typically follow a simple pattern, as presented in Figure 3.10. Therefore, we decided not to annotate a whole definition (i.e a whole sequence of words) but only a *key entity*, i.e. the first entity that appears in the definition.

For illustration, Example 3.3 presents the sentence with a definition. The annotator exploits existing entities from the e-layer and linked them with a predicate into the triple (*doba použitelnosti*, *rozumět*, *doba*). Figure 3.11 displays the sentence from Example 3.3 annotated using the Brat annotation tool. Table 3.6 presents some other samples of triples annotated in the CLTT data.

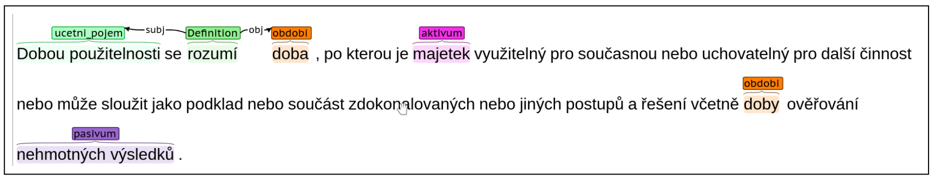
[*defined entity*] is a [*key entity*] which ...

Figure 3.10: A scheme of a typical definition in legal texts.

Example 3.3

Dobou použitelnosti se rozumí doba, po kterou je majetek využitelný pro současnou nebo uchovatelný pro další činnost nebo může sloužit jako podklad nebo součást zdokonalovaných nebo jiných postupů a řešení včetně doby ověřování nehmotných výsledků.

In English:
Shelf life means the period of time that an asset can be used for current or storable purposes or may serve as a basis or part of improved or other procedures and solutions, including the time of verification of intangible results.



Dobou použitelnosti se rozumí doba , po kterou je majetek využitelný pro současnou nebo uchovatelný pro další činnost nebo může sloužit jako podklad nebo součást zdokonalovaných nebo jiných postupů a řešení včetně doby ověřování nehmotných výsledků .

Figure 3.11: A CLTT sentence with the annotated entities and with the semantic relation (definition) between them. The manual annotations were done using the Brat annotation tool.

	Subject	Predicate	Object
1	dobou použitelnosti	(se) rozumí	doba
2	pořizovací cenou	(se) rozumí	cena
3	reprodukční pořizovací cenou	(se) rozumí	cena
4	rezervami	(se) rozumí	technické rezervy

Table 3.6: Samples of definitions relations in CLTT.

Annotated rights and obligations are triples, in which the *subject* position presents a carrier of a right (or an obligation), the *predicate* position presents the relation type and the *object* position represents a right (or obligation) itself. Both the carrier and the right has to be firstly annotated as entities on the e-layer.

For illustration, Example 3.4, Sentence (1) contain an obligation relation. The annotator links the subject entity *ministerstvo* with the object entity *vyhláška* using the predicate *vydat* which leads to the triple (*ministerstvo*, *vydat*, *vyhláška*). Sentence (2) from Example 5.4 leads to the same triple (*ministerstvo*, *vydat*, *vyhláška*), however, this particular triple will be annotated as the right relation.

Example 3.4

(1) Ministerstvo vydá vyhlášku k provedení § 4 odst. 8.
(2) Ministerstvo může vydat vyhlášku k provedení § 36 odst. 1.

In English:
(1) The Ministry shall issue a decree to implement Section 4 (8).
(2) The Ministry may issue a decree to implement Section 36 (1).

There are two main use cases how to work with the semantic relations in CLTT:

1. Users can use the plain text files and standalone JSON files where the relations are stored. Please, refer to Appendix B.8 for the technical details.

	Rel. type	Subject	Predicate	Object
1	Obligation	Ministerstvo	vydat	vyhláška
2	Right	Ministerstvo	vydat	vyhláška

Table 3.7: Semantic relations annotated in the sentences from Example 5.4.

2. Users can use the TrEd editor and modified PML files enriched with the relations annotation. This allows users to create special tree queries in which particular semantic relation types and arguments could be requested over nodes. Please, refer to Appendix B.9 for the technical details.

In total, there are 488 relations identified manually in CLTT. Table 3.8 presents the frequency distribution of three different relation types. In Appendix B.10 we provide the most frequent entities and entity types for the particular triple member.

	Frequency	
Relation type	absolute	relative
Definition	82	0.1680
Obligation	347	0.7111
Right	59	0.1209

Table 3.8: A distribution of three types of semantic relations annotated in CLTT.

3.6 Syntactic Phenomena in CLTT and PDT

Having the CLTT data, we are able to observe several syntactic phenomena that occur frequently. In addition, thanks to the annotation style we are able to provide a comparison of the selected phenomena as appear in the styles.

According to the theory of functional styles (as developed for Czech by the Prague School, primarily in Havránek's work [Havránek, 1932] and elaborated by many Czech scholars up to today, e.g., [Kořenský, 1989, Jelínek, 1995, Minářová et al., 2003]), the *function of the utterance* in communication is emphasized. This functional approach is based on goal-oriented language means and distinguishes several functional styles such as professional style, poetic style, colloquial style, etc. We are aware of the fact that the classification of the individual functional styles is a very complicated problem as mentioned e.g., in [Tiersma, 1999] or in [Gibbons, 2008].

However, having in mind general characteristics of the individual styles and the theoretical concepts of Czech functionally-oriented linguistics we tend to classify legal texts as texts belonging to the administrative-legal style (according to Jelínek [1996]) which is now earmarked as a unique functional style, standing next to other styles, such as professional, journalistic, literary or scientific. However, due to their specific function legal texts in many ways overlap with the professional style. Legal texts include very specific features related not only to vocabulary and syntax but also to various conventions and punctuation use. For example, impersonal style of legal texts understandably excludes the use of question marks and exclamation marks. On the other hand, we observe an extremely high usage of semicolon for purposes like enumeration, itemization and various types of listings.

The most important feature of legal texts is that they have a very specific syntactic structure with many peculiarities, such as:

- passive voice structures,
- impersonal constructions,
- non-finite and verbless clauses, and
- conjunctive groups.

Simple sentences are very rare. Typically, sentences are long and very complex. Punctuation plays a crucial role because legal texts usually include very complicated syntactic patterns or long lists separated by semicolons.

The complexity of legal sentences is obvious even from such a simple measure like the average sentence length applied to the selected Czech corpora, see Table 3.9.

Corpus	# of words	# of sentences	ASL
PDT	670,545	38,482	17.4
CAC	493,306	24,709	20.0
CLTT	34,410	1,121	30.7

Table 3.9: Average sentence length (ASL) in the Czech Academic Corpus (CAC, [Hladká et al., 2008]), PDT and CLTT.

Despite the fact the legal texts should be clear, comprehensible and explicit we found them sometimes difficult to understand and annotate, because of high usage of syntactic condensation and unusual language patterns, significant tendency to prefer abstract expressions, nominalizations, chains of genitive expressions etc.

The PDT is a corpus of journalistic style and contains the genres annotation as well, see Table 3.10. The genres classification was originally created for the Prague Discourse Treebank 1.0 [Poláková et al., 2013] aiming to observe how the discourse relations function in different types (in the genre sense) of language [Poláková et al., 2014].

Figure 3.12 shows that the legal texts are about 4.5 times *richer* in using a reflexive passive constructions while the use of periphrastic passive slightly prevails averaged across all genres in PDT.

Our comparison did not confirm our assumption of a frequent use of the construction with deverbative nouns ending on *–ní*, *-tí* with genitive – see Figure 3.12. Such a construction does not appear neither in CLTT nor in PDT texts very often.

Genre	Description	# of sentences
advice	advice column, interpretation, instructions	1,511
caption	descriptions of pictures, graphs, tables	507
collection	collection of various texts in one document	2,183
comment	commentary on an actual topic (short)	3,203
description	description of a product, company, services	5,942
essay	larger report or comment (longer)	6,793
invitation	to concerts, exhibitions, etc.	820
letter	letters (from readers)	434
news	current news report	13,603
other	genre is uncertain - especially in isolated sentences	1,164
overview	list of currency rates etc.	663
interview	interview with a person, multiple topics	1,471
plot	description of a plot (film, TV program)	100
program	(cultural) program of TV, radio, exhibitions	477
review	critical review (books, films, exhibitions, concerts)	2,332
sport	sports news, results	5,123
survey	survey and its results	383
topic	topical interview, *actual conversation*	2,608
weather	weather forecast	113

Table 3.10: Genre categories annotated in PDT.

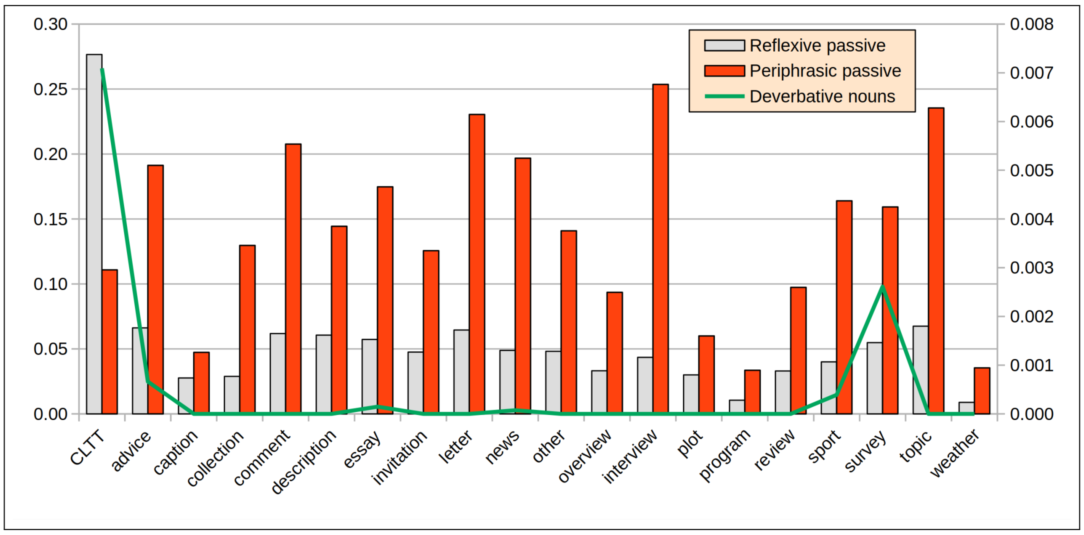


Figure 3.12: The relative frequency of (1) reflexive passive, (2) periphrastic passive and (3) deverbative nouns (with values on the second axis) in CLTT and PDT.

Figure 3.13 documents the expected dominance of chaining constructions with four, three and two genitives, respectively, in the legal texts. The biggest difference (percentage ratio) is observed in constructions with two genitives; the legal texts use noun phrases with (at least) two genitives about 4.4 times more often than the PDT texts on average, with the ratio ranging from 3.06 (news) to more than 10 (interviews – not surprisingly, people do not use these genitive chains much when speaking).[4]

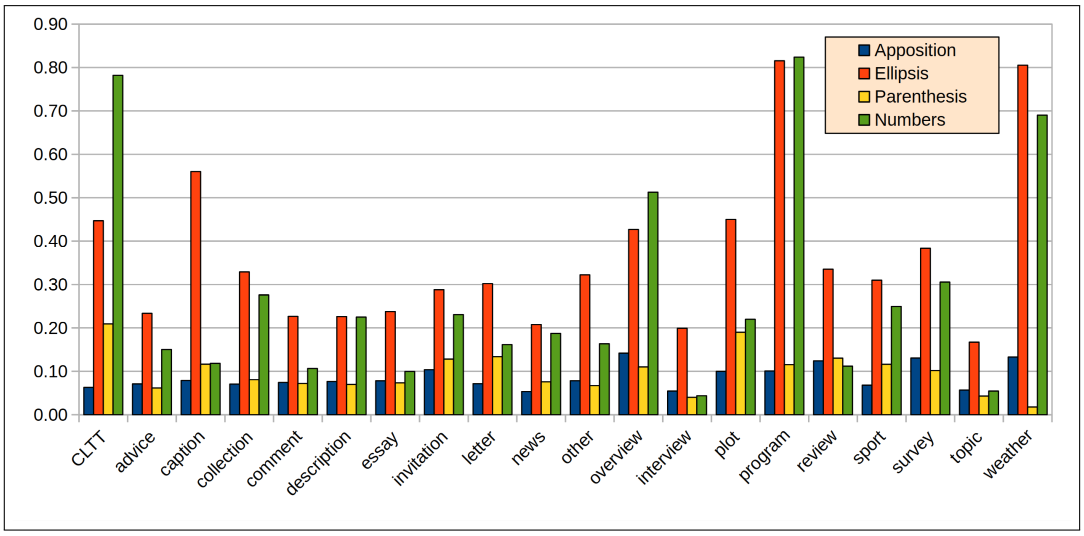


Figure 3.13: The relative frequency of (1) chains of four genitive expressions, (2) chains of three genitive expressions (both with values on the first axis), (3) chains of two genitive expressions (with values on the second axis) in CLTT and PDT.

Surprisingly enough, we observe that the constructions with apposition occur slightly more often (1.14 times) in the PDT texts – see Figure 3.14. On the other hand, the legal texts contain more constructions with ellipses; they are about 1.8 times more frequent in CLTT than in PDT, which in fact goes against the explicitness requirement assumed in the legal texts. Parenthetical constructions appear about 2.6 times more often in the legal texts, and constructions with numbers, which occur about 4.3 times more often in the legal texts, confirm their expected complex structure.

More detailed statistics are available in Appendix B.11.

3.7 Distribution notes

The work on the CLTT manual annotations was provided in several phases:

- The manual entity annotation was provided in August 2012.
- The manual syntactic annotation was provided in August 2013.
- The annotation of relation was provided in April 2014.

All three different annotations were revisited completely in August 2017.

Currently, two versions of CLTT are available:

- **Czech Legal Text Treebank 1.0**

[4]Disregarding broadcast programs and weather.

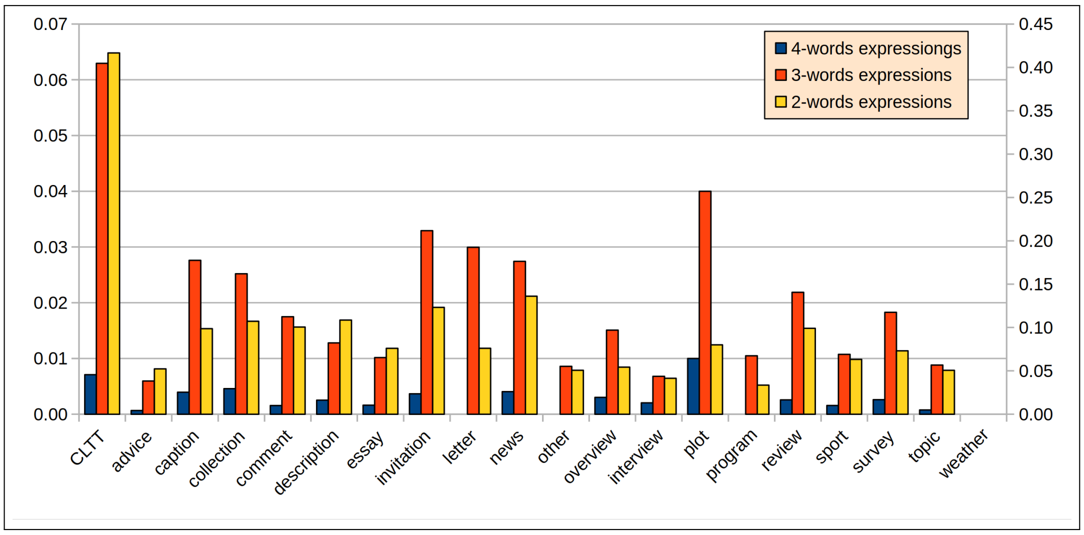


Figure 3.14: The relative frequency of (1) appositions, (2) ellipses, (3) parenthesis and (4) numbers in CLTT and PDT.

  – syntactic annotation (a-layer)

- **Czech Legal Text Treebank 2.0**
  – corrections in the syntactic annotation
  – entity annotation (e-layer)
  – relations annotation (r-layer)

There are various ways to access the Czech Legal Text Treebank. First, it can be downloaded from the LINDAT/CLARIN repository:

http://hdl.handle.net/11234/1-1516

The data of CLTT are distributed in the Prague Markup Language (PML, Pajas and Štěpánek [2006]) format. Each data file relates to one annotated treebank part – the base of its name is the identifier of the part (and it indicates the source of the document). The extension of the file expresses the layer of annotation of the document (`.w` denotes w-layer, `.m` denotes m-layer and `.a` denotes a-layer).

In addition, there are various tools for browsing and querying the treebank either off-line or on-line:

- the TrEd graphical editor
- the KonText KWIC search tool
- PML TreeQuery

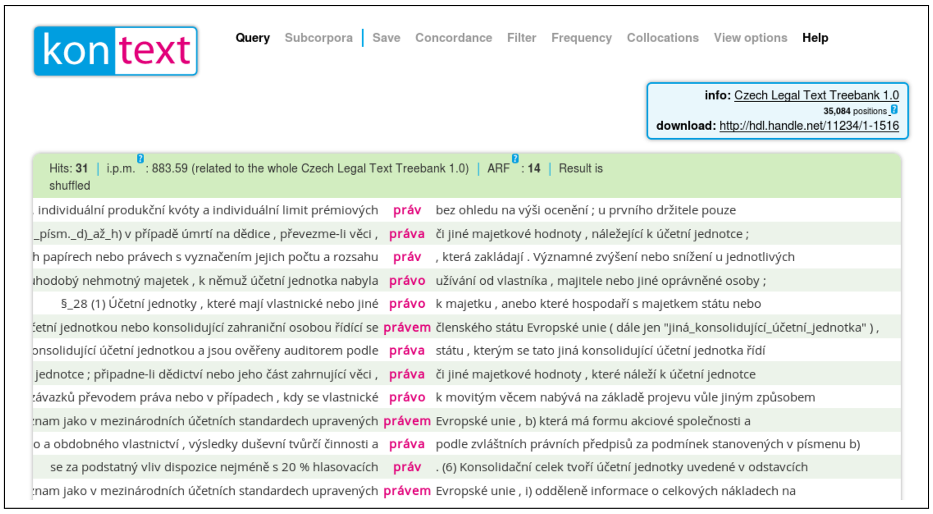
The users can view the treebank in the TrEd editor[5] that we used for the manual annotation. To browse the CLTT, users need to install the TrEd extension

[5]http://ufal.mff.cuni.cz/tred/

INTLIB Annotation. This extension can be installed directly in TrEd using *Setup → Manage Extensions → Get New Extensions*.

KonText[6] is a web application for querying corpora on-line within the LINDAT/CLARIN project. The users can evaluate simple and complex queries, display their results as concordance lines, compute frequency distribution, calculate association measures for collocations and do further work with the data.

Tree Query[7] is a powerful open-source search tool for all kinds of linguistically annotated treebanks available on-line within the LINDAT/CLARIN project. The users can evaluate complex tree queries and display their results graphically highlighted in the dependency trees.

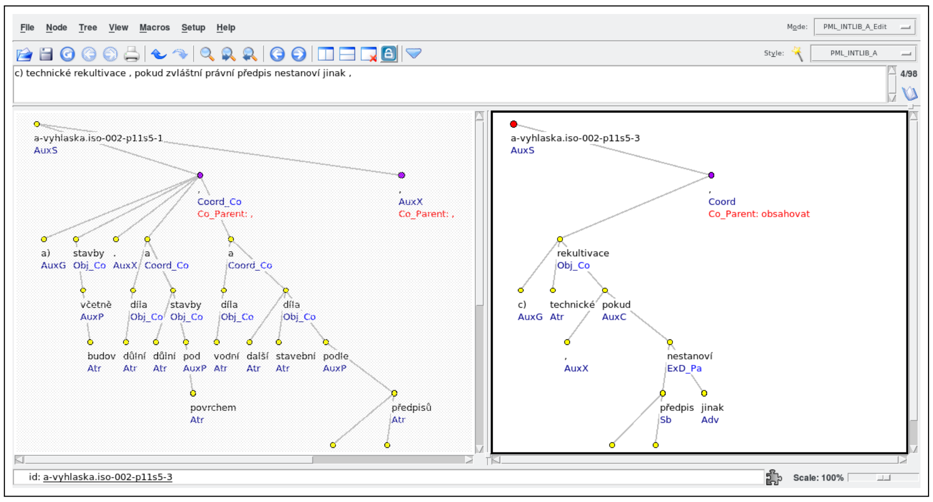


info: Czech Legal Text Treebank 1.0
35,084 positions
download: http://hdl.handle.net/11234/1-1516

Hits: **31** | i.p.m. : 883.59 (related to the whole Czech Legal Text Treebank 1.0) | ARF : **14** | Result is shuffled

individuální produkční kvóty a individuální limit prémiových	práv	bez ohledu na výši ocenění ; u prvního držitele pouze
_písm._d)_až_h) v případě úmrtí na dědice , převezme-li věci ,	práva	či jiné majetkové hodnoty , náležející k účetní jednotce ;
h papírech nebo právech s vyznačením jejich počtu a rozsahu	práv	, která zakládají . Významné zvýšení nebo snížení u jednotlivých
ıhodobý nehmotný majetek , k němuž účetní jednotka nabyla	právo	užívání od vlastníka , majitele nebo jiné oprávněné osoby ;
§_28 (1) Účetní jednotky , které mají vlastnické nebo jiné	právo	k majetku , anebo které hospodaří s majetkem státu nebo
ćetní jednotkou nebo konsolidující zahraniční osobou řídící se	právem	členského státu Evropské unie (dále jen "jiná_konsolidující_účetní_jednotka") ,
onsolidující účetní jednotkou a jsou ověřeny auditorem podle	práva	státu , kterým se tato jiná konsolidující účetní jednotka řídí
jednotce ; případne-li dědictví nebo jeho část zahrnující věci ,	práva	či jiné majetkové hodnoty , které náleží k účetní jednotce
závazků převodem práva nebo v případech , kdy se vlastnické	právo	k movitým věcem nabývá na základě projevu vůle jiným způsobem
ːnam jako v mezinárodních účetních standardech upravených	právem	Evropské unie , b) která má formu akciové společnosti a
o a obdobného vlastnictví , výsledky duševní tvůrčí činnosti a	práva	podle zvláštních právních předpisů za podmínek stanovených v písmenu b)
se za podstatný vliv dispozice nejméně s 20 % hlasovacích	práv	. (6) Konsolidační celek tvoří účetní jednotky uvedené v odstavcích
ːnam jako v mezinárodních účetních standardech upravených	právem	Evropské unie , i) odděleně informace o celkových nákladech na

Figure 3.15: The occurrences of the word *právo* in CLTT presented as concordances in the KonText on-line service.

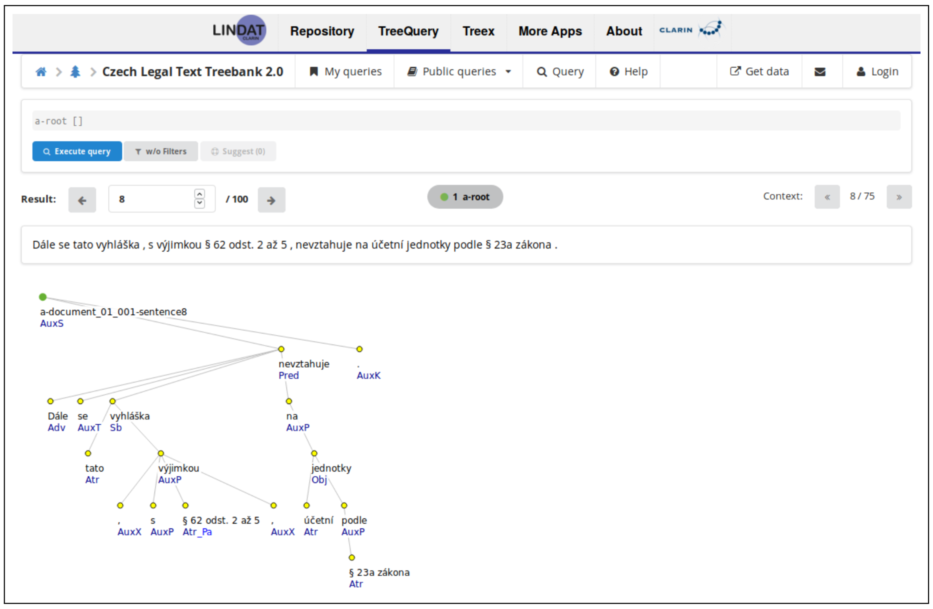


c) technické rekultivace , pokud zvláštní právní předpis nestanoví jinak ,

id: a-vyhlaska.iso-002-p11s5-3

Figure 3.16: The annotation of two complex sentence segments in the TrEd editor.

[6] https://lindat.mff.cuni.cz/services/kontext
[7] https://lindat.mff.cuni.cz/services/pmltq



```
a-root []
```

Dále se tato vyhláška , s výjimkou § 62 odst. 2 až 5 , nevztahuje na účetní jednotky podle § 23a zákona .

Figure 3.17: Browsing CLTT in the on-line PML-TQ browser.

The Czech Legal Text Treebank was transformed also into Universal Dependencies framework (UD, Nivre et al. [2018]). It is a project for cross-linguistically consistent grammatical annotation and an open community effort with over 200 contributors producing more than 100 treebanks in over 70 languages.

4. Detecting Semantic Relations

In this chapter we present the RExtractor system that extracts a *knowledge base* from raw unstructured texts. The knowledge base is a set of entities and their relations represented in an ontological framework. The RExtractor system implements an extraction pipeline. The pipeline processes input texts by linguistically-aware tools and extracts entities and relations between them. The system is designed both domain and language independent.

This work was published originally by Kríž et al. [2014b]. The RExtractor system was presented at the system demonstrations session at the 2015 Conference of the North American Chapter of the Association for Computational Linguistics – Human Language Technologies [Kríž and Hladká, 2015]

4.1 RExtractor

In this section we present the RExtractor system architecture. We firstly define its basic concepts. Then we describe its components. Technical details are provided in Appendix C.

4.1.1 Basic Concepts

The RExtractor system processes an input document and extracts its knowledge base, i.e. a set of entities and semantic relations between them.

In ontologies, an entity is as an ontological class. Subsequently, its individual appearance in the input document is the class instance. In the RExtractor system, we adopt the perspective of Natural Language Processing – each individual mention in the input document is called an entity and it is represented by the following attributes:

- **Unique entity identifier**
  Each entity is associated with an identifier that is unique in the input document.

- **Entity form**
  An exact form (i.e. a textchunk) as it appears in the source document.

- **Link to the Database of Entities**
  The Database of Entities (DBE) presents a storage of the ontological classes. If two entities share the same link to the Database, they present different instances of one specific ontological class.

Analogously to the entities, there are two different types of point of view on semantic relations. In ontologies, a semantic relation is a binary relation between two entities. Such relation is typically expressed in one or more source document sentences by the individual entity instances. In RExtractor, a semantic relation is a triple [subject, predicate, object], where the subject and object are entities and the predicate is a textchunk that expresses the type of the relation.

For illustration, assume the sentence in Example 5.1: Section A in Table 4.1 presents the semantic relation from the ontological point of view. The textchunks from the input sentence are considered to be instances of the abstract ontological classes. Section B presents the same semantic relation as it is provided by the RExtractor system. The textchunks from the input sentence are organized into a triple, where subject and object are entities, i.e. they have a unique identifier and they are associated with the particular entry in DBE.

Example 5.1

Účetní jednotky tvoří opravné položky.

In English:
Accounting units create fixed items.

	Textchunks	*účetní jednotky*	*tvoří*	*opravné položky*
A	Class	AccountingUnit	hasObligation	FixedItem
B	Triple element	subject	predicate	object
	Unique ID	`entity-1`	–	`entity-2`
	Entry in DBE	účetní jednotka	–	opravná položka

Table 4.1: Sample representation of the semantic relation using an ontological formalism (A) and the RExtractor concept (B).

Search applications build knowledge bases from single documents as their target task is to determine the source document where the information is encoded. For other applications (e.g. question-answering systems), individual knowledge bases are combined together so systems could use knowledge across the documents.

The knowledge base could be easily visualized by a knowledge graph where nodes represent entities and edges represent semantic relations.

4.1.2 System Architecture

The RExtractor system is an open source software written in Perl. The web front end is written using JavaScript's JQuery framework. The software is licensed under the *Attribution-NonCommercial-ShareAlike 4.0 International* license. Its source codes are available on Github:

https://github.com/VincTheSecond/rextractor

The RExtractor system requires Treex [Popel and Žabokrtský, 2010] and TrEd [Pajas and Štěpánek, 2009] to be installed on the host server. Both tools are available as open source, however the language models for taggers and parsers could be under different licenses.

RExtractor offers a simple document work flow, where input documents are processed by an extraction pipeline. The pipeline consists of five components, see Figure 4.1:

- **Conversion Component**
  It converts various input formats into the unified internal document representation.

- **Language Processing Component**
  It provides the NLP analysis of the document content, from the sentence segmentation to the dependency parsing.

- **Entity Detection Component**
  It identifies entities from the Database of Entities in the dependency trees.

- **Relation Extraction Component**
  It detects semantic relations in dependency trees with highlighted entities.

- **Export Component**
  It provides different conversions from the RExtractor internal document representation to a custom client format.

Each component offers several configuration options. A configuration for a particular extraction task is called *extraction strategy*. Different extraction strategies could be used on the single system installation. If so, users have to specify the extraction strategy that will be applied on input documents.

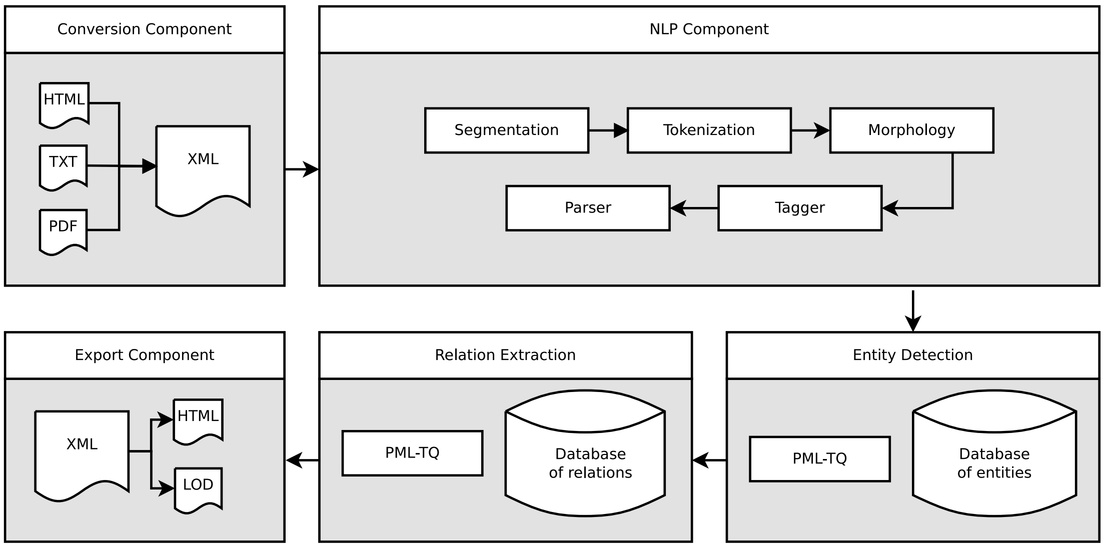


Figure 4.1: The RExtractor system architecture.

Technically, each component is implemented as a standalone server which waits for a document to be processed. As the document *flows* throw the extraction pipeline, it receives different states, see Appendix C.1.

As the processing of the submitted document takes several minutes, clients are supposed to submit a new document into the system queue and check for the document state subsequently. Once the document is processed completely, clients can download the results.

All documents submitted into the RExtractor system are stored in the *Document Collection*. Once the document is submitted to RExtractor, it is available in the Document Collection until the user removes it upon the request. Advanced users could access Document Collection directly, see Appendix C.2.

The RExtractor system offers two different APIs to interact with the system (see Appendix C.3). APIs allow users to submit new documents, obtain document states and download exported data. They are designed especially for external applications or users who want to obtain the RExtractor data for the subsequent processing.

4.1.3 System Components

Conversion Component

The Conversion Component converts various input formats into the RExtractor Internal Document Representation (see Appendix C.4). It regularly checks the Document Collection for new documents. Once such document exists, the component loads the particular extraction strategy associated with the document and applies the requested conversion method. Custom conversion methods could be implemented as Perl packages inherited from the `RExtractor::Conversion` abstract class.

NLP Component

The NLP Component provides various language analyses of input texts using the Treex framework. It regularly checks the Document Collection for documents processed by the Conversion Component. Once such document exists, the component loads the particular extraction strategy associated with the document and applies the NLP methods specified in the strategy. Custom NLP components could be implemented as a standalone Perl package inherited from the `RExtractor::NLP` abstract class. Typically, a custom method runs one or more Treex scenarios and it would transform Treex data to be ready for the next components.

Entity Detection Component

The Entity Detection Component uses queries over dependency trees to detect entities. The entities must be defined in advance in the Database of Entities (DBE, see Appendix C.5). RExtractor exploits the PML-TQ framework [Pajas and Štěpánek, 2009] which provides both a query language and a set of methods for matching queries over dependency trees. The PML-TQ is available in Treex. The exploitation of queries for the entity detection presents a new approach. It brings several advantages when compare it with the standard indexing system [Barsky et al., 2011].

For illustration, we assume the entity *běžný hmotný majetek* (*current tangible asset*) that should be identified in the input text *soupis běžného hmotného a nehmotného majetku* (*an inventory of the current tangible and intangible asset*). Figure 4.2 presents the tree query for the entity and the dependency tree of the input text.

We can see two main advantages of the presented approach:

- If the entity appears in a coordination in the source sentence, it presents a problem for other methods of text indexing. The same problem raises when the entity itself contains a coordination. In PML-TQ, coordinations are processed automatically without any additional work, i.e. the entity tree query in Figure 4.2 will match also the structure with the coordination.

- Tree queries uses lemmas of individual words in entities so all morphological forms of the entity are easily identified in texts. This is important especially for language with reach morphology. Figure 4.2 presents an entity that is automatically detected even over the different morphological word forms.

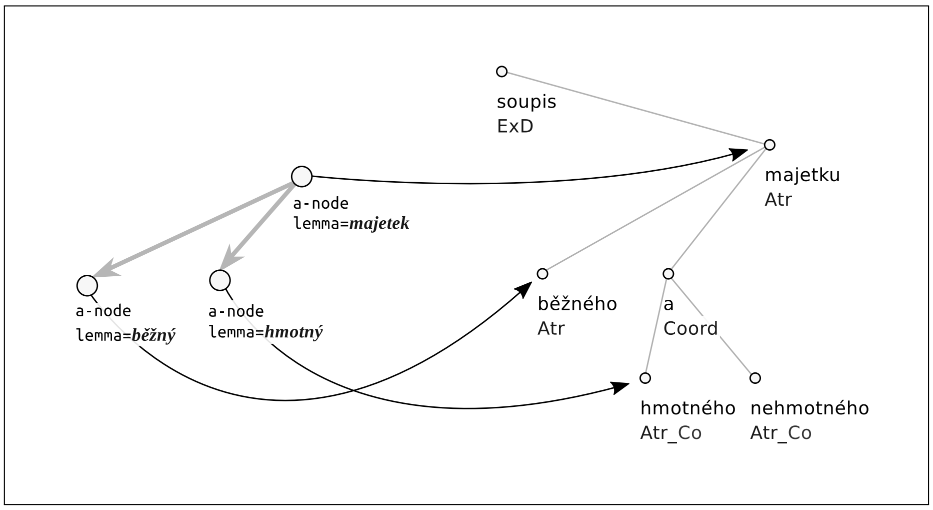


Figure 4.2: The tree query for the entity *běžný hmotný majetek* will be detected in the input text *soupis běžného hmotného a nehmotného majetku* automatically, even it contains a coordination and different morphological word forms.

As we present in Section 2.1, a predefined list of entities (typically a taxonomy or an ontology) is typical for domain specific information extraction systems. On the other hand, trained and tuned statistical models are typically used by named entity recognition systems (NER), where target categories (names of persons, organizations, locations, etc.) are open sets and could be determined from the context.

The Entity Detection Component regularly checks the Document Collection for documents processed by the NLP Component. Once such document exists, the Entity Detection Component loads the particular extraction strategy associated with the document and starts with the entity detection using a given DBE. One can consider the DBE to be a storage of ontological classes. Their individual instances are subsequently detected in the source documents. The single RExtractor system installation could work with several different DBEs. If so, the extraction strategy specifies the DBE for the particular document.

Relation Extraction Component

The Relation Extraction Component works analogously to the Entity Detection Component. The extraction strategy specifies the Database of Relations (DBR, see Appendix C.6). Each DBR entry contains a tree query which determines the final triple. The PML-TQ framework in Treex is used as the back end. The relation extraction queries could exploit all information provided by the previous components in the extraction pipeline.

The Relation Extraction Component regularly checks the Document Collection for documents processed by the Entity Detection Component. Once such document exists, the Relation Extraction Component loads the particular extraction strategy associated with the document and starts with the relation detection using the given DBR.

Export Component

The Export Component allows users to transform the extracted data to a custom format. The custom components could be implemented as a standalone Perl package inherited from the `RExtractor::Export` abstract class. Typically, a custom method extracts a needed data from the RExtractor Internal Document Representation and prepare a custom XML document.

4.2 RExtractor and Czech Legal Documents

We present the extraction strategy of the RExtractor system tailored for extracting a knowledge base from Czech legal texts.

4.2.1 Conversion Component

We implemented the custom conversion component, that converts source documents from the INTLIB project to the RExtractor Internal Document Representation.

A structure of complex sentences in legal texts allows us to split them into several simple sentences. We say that complex sentences are *multiplicated* into several sentences so that each enumerated subsection in the original sentence creates a new single sentence. Figure 4.3 presents a complex sentence with 5 enumerated subsections *a) – e)*. The conversion component multiplicates the sentence into 5 sentences, see Figure 4.4. In addition, the numbering expressions (*(1)*, *a)*, ...) are removed from the sentence.

(1) **The General Directorate of Customs**

a) is an administrative body exercising superior authority to customs offices,

b) administers the customs duty in compliance with the relevant EU regulation,

c) determines which cases under the remit of customs authorities are of nationwide or international importance,

d) is a customs authority with the competences of a police authority as defined in the penal code when dealing with cases of nationwide or international importance,

e) functions as a central analytical body analyzing risks.

Figure 4.3: The complex sentence with 5 enumerated subsections.

The General Directorate of Customs is an administrative body exercising superior authority to customs offices.

The General Directorate of Customs administers the customs duty in compliance with the relevant EU regulation.

The General Directorate of Customs determines which cases under the remit of customs authorities are of nationwide or international importance.

The General Directorate of Customs is a customs authority with the competences of a police authority as defined in the penal code when dealing with cases of nationwide or international importance.

The General Directorate of Customs functions as a central analytical body analyzing risks.

Figure 4.4: The complex sentence from Figure 4.3 *multiplicated* by the conversion component into 5 simple sentences.

4.2.2 NLP Component

This component processes texts using the following Treex procedures:

- `W2A::CS::Segment` for standard sentence segmentation,
- `W2A::CS::Tokenize` for standard tokenization,
- `W2A::CS::TagMorphoDiTa` for morphological analysis and tagging. The block exploits the MorphoDiTa software [Straková et al., 2014].

	Segmentation	Tokenization	# of sentences	# of tokens	ASL	UAS	Improvement
1	standard	standard	1,121	40,267	35.92	74.63	–
2	standard	re-token	1,121	34,410	22.40	78.99	5.85%
3	multi	re-token	1,438	36,596	25.45	81.76	9.55%

Table 4.2: Evaluation of the NLP Component. We report the average sentence length (ASL), the unlabeled attachment score (UAS) and the improvement over the baseline, i.e. the standard Treex pipeline for Czech texts.

- `W2A::CS::ParseMSTAdapted` for automatic parsing. The block exploits the MST parser [McDonald et al., 2005] adapted for Czech by Novák and Žabokrtský [2007].

In addition, we improve the processing pipeline using two tailored procedures:

- **Long sentences multiplication**
  Long sentences are split automatically by the Conversion Component to decrease their complexity.

- **Re-tokenization**
  We adopted the re-tokenization procedure used in CLTT (see Section 3.2) because it significantly improves the automatic parser performance.

To evaluate the proposed procedures, we used the data from CLTT. For three different experiment setups we investigated the automatic parser performance (see Table 4.2):

Experiment (1) presents the baseline which is the standard Treex pipeline for Czech texts without the multiplication and re-tokenization procedures. The automatic parser achieved really poor results that are almost 12% below the parser's reported performance. Novák and Žabokrtský [2007] report 84.69% UAS for the MST parser adapted for Czech trained on the PDT train dataset and evaluated on the PDT etest dataset.

Experiment (2) presents the parser performance when re-tokenization was applied. Using re-tokenization, the average sentence length (ASL) decreased from 35.92 to 22.40 tokens per sentence. This leads to the 5.85% improvement of the parser's performance.

The performance of the parser is even better in Experiment (3) where both multiplication and re-tokenization are applied. We achieved almost 10% improvement over the baseline.

4.2.3 Entity Detection Component

The entity detection component requires an instance of the Database of Entities. We used the CLTT Accounting Dictionary (see Section 3.4). The tree queries

	Parsing	TP	FN	FP	Precision	Recall	F1
1	gold	8,895	748	156	98.28	92.24	95.16
2	automatic	8,818	825	365	96.03	91.44	93.67

Table 4.3: Evaluation of the Entity Detection Component. We report the number of true positives (TP), false negatives (FN) and false positives (FP) as well as the Precision, Recall and F1-measure.

were created by an automatic procedure which transforms an entry's dependency tree to a query using a set of rules. This query is subsequently used by the component to detect the entity over a given dependency tree.

To evaluate the entity detection component, we performed two experiments using CLTT, see Table 4.3:

Experiment (1) presents the performance of the entity detection using the gold standard dependency trees. As the PML tree queries available in the CLTT Accounting Dictionary was created by the automatic procedure, we can see this experiment as an evaluation of this procedure as well.

Experiment (2) presents the performance of the entity detection using the dependency trees created by the NLP Component in its best setup (i.e. sentences were re-tokenized and multiplicated before the parsing).

4.2.4 Relation Extraction Component

In this section, we present two different approaches to the relation extraction. The first approach detects relations using manually designed tree queries. The work was presented and evaluated by Kríž et al. [2014b]. The second approach exploits supervised machine learning methods.

Rule-based Relation Extraction

We used the Accounting Act (*563/1991 Coll.*) from CCLT as the data for a manual query development. We obtained 5 queries for Definitions, 4 queries for Rights and 2 queries for Obligations. Some of the queries are presented in Table 4.4.

The process of formulating a query consists of the following steps:

1. A tree query expert browses through the dependency trees and tries to observe typical syntactic constructions for a given type of relation.
2. The expert designs the query for matching the most frequent construction.
3. The query is immediately evaluated and the expert can browse the matched sentences.

We carried out the evaluation on the Decree on Double-entry Accounting for Undertakers (*500/2002 Coll.*) from CCLT where we manually checked the detected relations. We obtained the results presented in Table 4.5. The row *Gold-standard* lists the number of manually detected relations.

We determined three types of errors for the incorrectly detected relations: (i) incorrect dependency tree, (ii) missing or incorrect query, (iii) missing or incorrect entity. The results are summarized in Table 4.6.

Query	Subject	Predicate	Object
D_4	CASE = 7	LEMMA = rozumět_se	POS = noun, CASE = 1
R_2	AFUN = Sb	LEMMA = odpovídat	LEMMA = za
O_1	ENTITY = true	LEMMA = moci	AFUN = Obj, POS = verb

Table 4.4: Simplified versions of the most successful queries. In all presented queries, both subject and object depend on predicate.

	D	O	R	Total
# of queries	5	4	2	11
Gold standard	97	308	62	467
Extracted	70	255	41	366
True positives	53	206	36	295
False negatives	44	102	26	172
False positives	17	49	5	71
Precision (%)	0.757	0.808	0.878	0.806
Recall (%)	0.546	0.669	0.581	0.632

Table 4.5: Evaluation of the Relation Extraction Component with the manually designed tree queries.

Error	# of errors	Ratio
Parser	145	59.7 %
Query	93	38.3 %
Entity	5	2.1 %

Table 4.6: Error analysis of incorrectly detected relations.

Supervised Machine Learning Relation Extraction

In the previous approach, we defined the *rule-based* system with a set of rules designed by observing a training dataset. Development of such rule-based systems requires a large amount of effort from experienced rule writers. Maintenance of such rule sets can be tricky because the rules often intricate interdependencies that make modification risky. Even more, the experiments with the manually designed tree queries are not reproducible and do not allow cross validation because of human bias from the previous experiments. Statistical models offer an alternative to the rules. One must develop features that correspond to cues, pick an appropriate statistical method, and train a model using training data.

Having the manually annotated entities and relations in CLTT, we extracted a dataset for supervised machine learning experiments. The dataset instances are generated by an automatic procedure. One dataset instance presents a candidate triple [*subject*, *predicate*, *object*] labeled by one of the output labels (*Definition*, *Right*, *Obligation*, *None*).

From each dependency tree with at least two entities, candidate instances present all possible pairs of entities as subjects and objects and all possible tree nodes on the path in the dependency tree between entities are taken as predicates. More formally:

- Let e_1 and e_2 be two different entities in the dependency tree.
- Let P be a set of tree nodes on the path from e_1 to e_2.
- For each tree node $n \in P$ create a candidate triple $[e_1, n, e_2]$.

Each candidate triple is represented using the following features:

- Number of tree nodes on the path between *subject* and *object*
- Number of entities on the path between *subject* and *object*
- Features for the *subject* entity:
  - Morphological (e.g. form, lemma, part of speech, case, gender)
  - Syntactic (analytic function)
  - Entity identifier, entity type
- Features for the *predicate*:
  - Morphological (e.g. form, lemma, part of speech, tense, gender)
  - Syntactic (analytic function, verb reflexivity)
- Features for the *object* entity:
  - Morphological (e.g. form, lemma, part of speech, case, gender)
  - Syntactic (analytic function)
  - Entity identifier, entity type

In total, there are 488 gold-standard relations in CLTT. The automatic procedure described above extracted 480 of relations as the positive instances and more than 400,000 negative instances.

We experimented with a range of different supervised machine learning algorithms. We achieved the best performance using Random Forests. Table 4.7 presents the final results and the discussion follows:

Experiment (1) presents the relation extraction performance when working with the gold-standard dependency trees as well as the gold-standard entities.

Experiment (2) presents the relation extraction performance when working with the gold-standard dependency trees and the automatically detected entities. Using the Entity Detection Component decreases the relation detection performance about 3.13%.

Finally, Experiment (3) presents the performance of the whole Czech Legal Documents extraction strategy. It uses the dependency trees parsed automatically by the NLP Component in its best setup, i.e. with re-retokenization and mul-

	Parsing method	Entities detection	TP	FN	FP	Precision	Recall	F1
1	gold	gold	441	20	19	95.87%	95.66%	95.77%
2	gold	mst	416	40	24	94.55%	91.23%	92.86%
3	mst	mst	275	125	80	77.46%	68.75%	72.85%

Table 4.7: Evaluation of the Relation Extraction Component using supervised machine learning approach. We report the number of true positives (TP), false negatives (FN) and false positives (FP) as well as the Precision, Recall and F1-measure.

tiplication of the complex sentences. The entities are detected automatically by the Entity Detection Component.

Comparing Experiment (1) and Experiment (3), the automatic dependency parsing has the most significant influence on the system performance.

In the next chapter, we present several experiments to improve parsing of Czech legal texts.

4.3 On-line Demo

To demonstrate the RExtractor system, we prepared its on-line demo. It allows to submit Czech or English legal texts and browse the extracted data. It is available on

https://quest.ms.mff.cuni.cz/rextractor/

5. Improving Knowledge Extraction from Long Sentences

The RExtractor system presented in Chapter 4 focused on extracting semantic relations from unstructured texts. It processes texts by linguistically-aware tools and extracts entities and relations.

One of the most significant feature of the legal language is a high frequency of long and complex sentences. Figure 5.1 provides another evidence of difficulty with the dependency parsing of long sentences — as the sentence length increases, the unlabeled attachment score decreases. The numbers are provided for five datasets from three Czech dependency treebanks (see Table 5.1).

In this chapter, we increase the RExtractor performance by a new method for the dependency parsing of complex sentences. This method segments input sentences into clauses and does not require to re-train a parser of one's choice.

We represent a sentence clause structure using clause charts that provide a layer of embedding for each clause in the sentence. Then we formulate a parsing strategy as a two-stage process where (i) coordinated and subordinated clauses are parsed separately with respect to the sentence clause chart and (ii) their dependency trees become subtrees of the final tree of the sentence.

We achieved a 0.78% improvement of the unlabeled attachment score averaged over five datasets and finally, we achieved 3.40% improvement of RExtractor on the Czech Legal Text Treebank dataset.

The experiments with parsing of complex sentences were published in [Kríž and Hladká, 2016].

5.1 Data and Tools

We experiment with three manually annotated dependency treebanks:

- Prague Dependency Treebank (PDT) 3.0,
- Czech Academic Corpus (CAC) 2.0,[1]
- Czech Legal Text Treebank (CLTT) 2.0.

As the baseline dependency parser, we use the MST dependency parser [McDonald et al., 2005] adapted for Czech by Novák and Žabokrtský [2007]. It was

[1]The Czech Academic Corpus (CAC, [Hladká et al., 2008]) is a treebank fully compatible with PDT and contains manual annotations of m-layer and a-layer. The original name of the corpus was Practical corpus (*Korpus věcného stylu*). It was created from 1971 till 1985 by a team from the Institute of the Czech Language. The corpus has been transferred into the PDT annotation scheme in 2007.

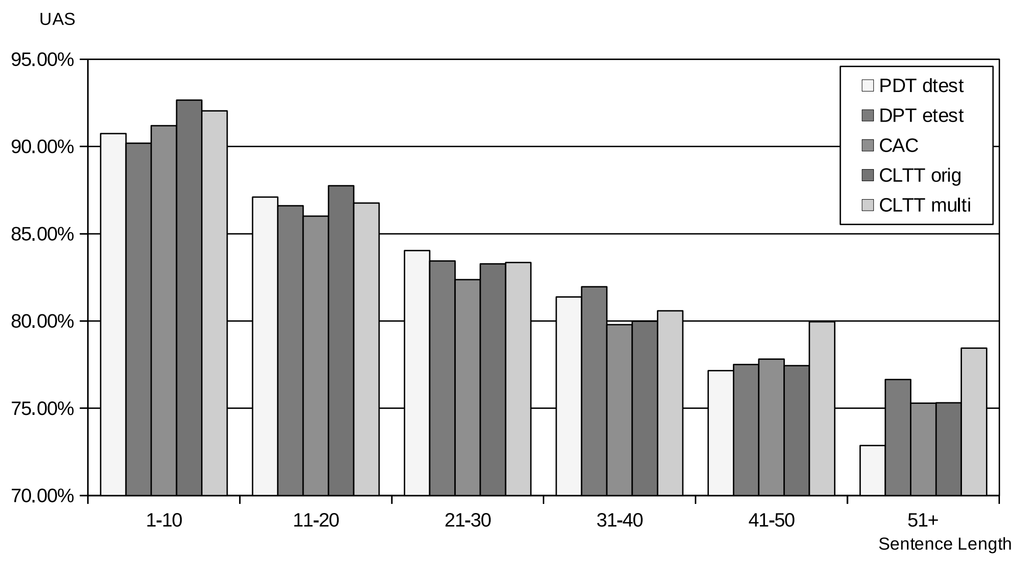


Figure 5.1: The longer the sentence the lower the unlabeled attachment score.

Treebank	Dataset	Sentences	Tokens	UAS
PDT 3.0	train	29,768	518,648	0.9341
	dtest	4,042	70,974	0.8450
	etest	4,672	80,923	0.8432
CAC 2.0	–	24,709	493,306	0.8268
CLTT 2.0	orig	1,121	34,410	0.7924
	multi	1,438	36,596	0.8176

Table 5.1: Part of the treebank (*Dataset*), the number of sentences, the number of tokens and the unlabeled attachment score (*UAS*) of the baseline parser.

trained on the PDT 3.0 train set. Table 5.1 presents basic characteristics of the treebanks and the Adapted MST parser performance on them.

The PDT 3.0 contains three disjunct datasets: *train*, *dtest* and *etest*. The baseline parser was trained on the *PDT train* dataset. Therefore, in the following experiments, we do not report scores achieved on this dataset.

For CLTT 2.0, we evaluate the experiments on two datasets. *CLTT orig* presents the original sentences as they are published in the treebank. The *CLTT multi* dataset contains sentences processed by the automatic splitting procedure, i.e. complex sentences are multiplicated into several simple sentences, see Section 4.2. Both CLTT datasets are re-tokenized (see Section 3.2.2).

We enriched all datasets with the annotation of clause segmentation done automatically using the rule based procedure [Bejček et al., 2013], see Figure 5.2 for illustration.

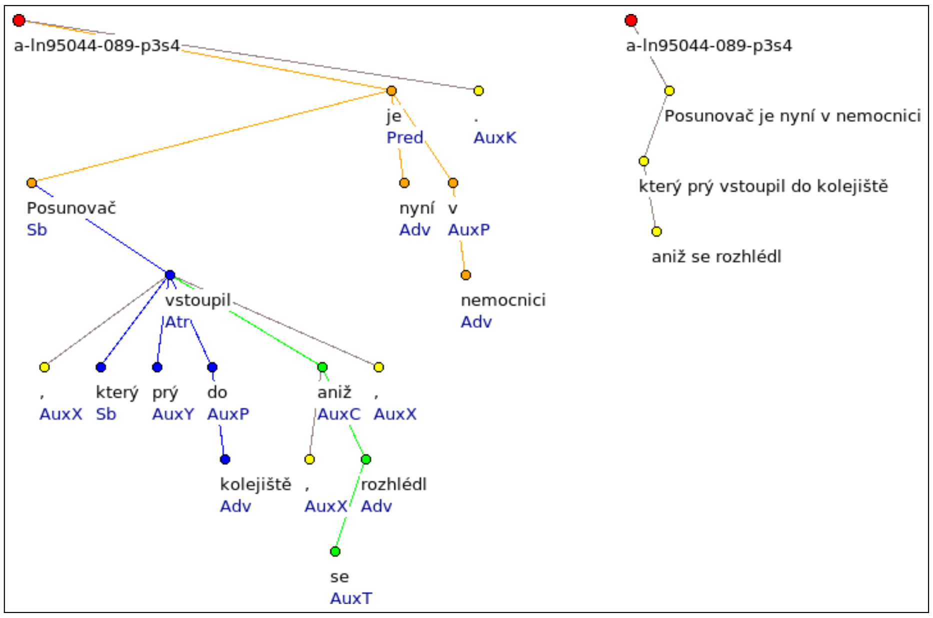


Figure 5.2: Sentence *Posunovač, který vstoupil do kolejiště, aniž se rozhlédl, je nyní v nemocnici* represented by two dependency trees: full (and colored) on the left side and with collapsed clauses on the right side. This illustration was published originally in [Bejček et al., 2013].

5.2 Clause Charts

A *clause chart* is defined to visualize relationships between clauses within the sentence and captures the layer of embedding of each individual clause. It is an $m \times n$ table where n is the number of clauses in the sentence and m is the number of layers. A cell (i, j) stands for relationship between the j-th clause and the i-th layer of embedding. Its value is initialized to the value of 0 corresponding to no relationship.

We defined four rules for changing the cell value from 0 to 1, i.e., for assigning a layer of embedding to each clause in the sentence:

1. All main clauses belong to the basic layer 0.
2. The clauses that depend on the clauses at the k-th layer belong to the $(k + 1)$-th layer.
3. The coordinated clauses and the clauses in apposition belong to the same layer.
4. The clauses in parentheses belong to the $(k + 1)$-th layer with respect to the k-th layer of their adjacent clauses.

Our definition is analogous to a segmentation chart defined by Lopatková and Holan [2009]. However, we handle the following situations differently:

- subordinating conjunctions at the beginning of each clause are considered as boundaries and are excluded from the clause; and
- clauses split into two parts by an embedded subordinated clause are con-

sidered as two different clauses.

5.2.1 Generating Clause Charts

We designed a procedure that generates a clause chart from a dependency tree with the clause annotation. Particularly, it generates a clause tree first and then a clause chart.

We assume a dependency tree where each non-boundary node has a special attribute bearing the identification of the clause it belongs to. The nodes with the same clause number belong to the same clause and thus generating a clause chart is uniquely determined by the clause identification. A layer of embedding of the clause is defined as its depth in a sentence clause tree where its nodes contain tokens with the same clause identification.

Figure 5.3 displays both the clause tree and the clause chart of the sample sentence presented by Kuboň et al. [2007]:

> ***While*** *failure is usually an orphan****,*** *the success tends to have many fathers****,*** *claiming eagerly* ***that*** *particularly they were present at its conception****.***

This sentence consists of four clauses delimited by the boundaries printed in bold, namely *while*, *that*, and two commas. In general, clause boundaries are either a single token or a sequence of tokens. Clause boundaries are not components of the clause tree. They are displayed there for understanding a linear representation of a clause chart, see `B1B0B1B2` where `B` stands for a clause boundary and the numbers are the layers of clause embedding.

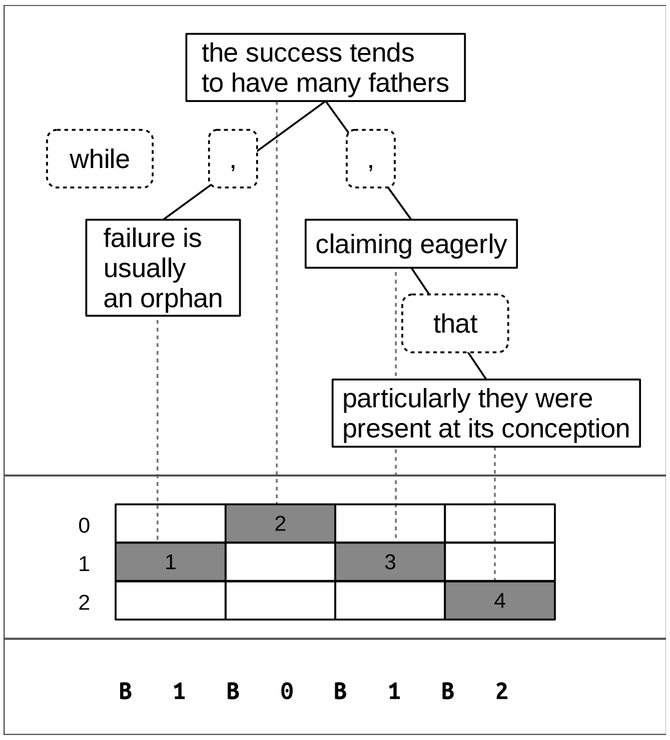
Since the last boundary is mainly technical and it does not have any influence on the proposed methods, we exclude it from the clause chart.

5.2.2 Exploring Clause Charts

We explored the datasets to study different types of clause charts. Table 5.2 and Table 5.3 provide statistics for the five most frequent clause charts that occur in the datasets. For example, 17.28% of the sentences in the *PDT dtest* contain a main clause and a subordinated clause encoded by the `0B1` pattern.

We also report the MST parser performance on the sentences having the given clause charts. For example, MST achieved UAS of 81.72% on the `0B1B0` sentences in the *PDT dtest*.

The PDT and CAC texts come from newspapers. Thus there is no surprise that the most frequent sentences in the treebanks are simple one clause sentences (`0`). They present more than 30% of the data. The second most frequent sentence structure consists of one main clause and one subordinated clause (`0B1`). It is



the success tends to have many fathers
while
,
,
failure is usually an orphan
claiming eagerly
that
particularly they were present at its conception
0
1
2
2
1
3
4
B 1 B 0 B 1 B 2

Figure 5.3: Clause tree (above), clause chart and its linear representation (below).

quite surprising that the parser processes these sentences better than one clause sentences. We observe decrease in the parser performance on coordination of two main clauses (i.e., on the `0B0` sentences).

For curiosity's sake, the most complex sentence in the treebanks consists of 36 clauses and comes from CAC. The `0B1B2B3B4B5B6` clause chart is a chart with the most deeply embedded layer and comes from PDT. The most complex sentence in CLTT consists of 29 clauses. The most deeply embedded layer in CLTT appears on the fourth layer.

The PDT and CAC datasets share the same top 5 most frequent clause charts. This is in contrast with the CLTT datasets. Both CLTT datasets have a specific top 5 most frequent charts. In the *CLTT multi*, `0B0` has only 3.17% of relative frequency. We can see this analysis as another example of quantitative differences between newspaper and legal texts.

5.3 Methods and Experiments

In this section, we present a method for improving dependency parsing of long sentences. In particular, we formulate an algorithm for parsing the coordinated clauses `0B0` and governing and dependent clauses `0B1`.

Clause chart	PDT dtest		PDT etest		CAC	
	RF	UAS	RF	UAS	RF	UAS
0	0.3233	0.8568	0.3158	0.8537	0.3293	0.8409
0B1	0.1728	0.8824	0.1751	0.8798	0.1628	0.8568
0B0	0.0850	0.8228	0.0798	0.8343	0.0907	0.8096
0B1B0	0.0570	0.8172	0.0463	0.8198	0.0474	0.7969
0B1B2	0.0320	0.9001	0.0402	0.8813	0.0358	0.8731

Table 5.2: Relative frequency (*RF*) of the five most frequent clause charts and the unlabeled attachment score (*UAS*) of the MST evaluated on the particular datasets.

CLTT orig			CLTT multi		
Clause chart	RF	UAS	Clause chart	RF	UAS
0	0.2626	0.8162	0	0.2996	0.8255
0B1	0.1056	0.8406	0B1	0.1338	0.8521
0B1B0	0.0699	0.8121	0B1B0	0.0912	0.8168
0B0	0.0416	0.7252	0B1B2	0.0447	0.8649
0B1B0B1	0.0316	0.8002	0B1B0B1	0.0370	0.8065

Table 5.3: Relative frequency (*RF*) of the five most frequent clause charts and the unlabeled attachment score (*UAS*) of the MST evaluated on the particular datasets.

5.3.1 Parsing Coordinated Clauses

Given the clause chart representation, we can recognize coordinated clauses in sentences in a straightforward way. Thus, we consider neighboring coordinated clauses C_1, C_2, ..., C_n on the same layer ($n > 1$) and we propose the following parsing strategy that we call *clause chart parsing* (CCP):

1. Apply the baseline parser to C_1, C_2, ..., C_n individually to get dependency trees T_1, T_2, ..., T_n with the r_1, r_2, ..., r_n root nodes, respectively.
2. Create a sequence $S = r_1\ B_{1,2}\ r_2\ B_{2,3}\ \ldots r_n$ where $B_{i,i+1}$ is a boundary between C_i and C_{i+1}.
3. Apply the baseline parser the sequence S to get a dependency tree T_S.
4. Build a final dependency tree so that the trees T_1, ..., T_n become subtree of T_S.

For illustration, we assume the sentence *John loves Mary and Linda hates Peter*. The sentence consists of two coordinated clauses C_1 = {*John loves Mary*}, C_2 = {*Linda hates Peter*} and one clause boundary $B_{1,2}$ = {*and*}. Therefore, the clause chart of the sentence is `0B0`. In Step 1, C_1 and C_2 are parsed to get T_1 and T_2 with the root nodes r_1 = *loves* and r_2 = *hates*, resp. In Step 2, the sequence S = *loves and hates* is created. In Step 3, S is parsed to get T_S and, finally, in Step 4, T_1 and T_2 become subtrees of T_S.

We evaluated the proposed parsing strategy only on the sentences having the `0B0` clause chart, i.e., on the subsets of the treebank datasets. Table 5.4 presents the unlabeled attachment score achieved for (i) the baseline parsing method, i.e., parsing of complete sentences using MST (*Baseline*) and (ii) the full-scale parsing using the *CCP* method. Using it we achieved an average 1.71% improvement in UAS.

Dataset	# of Sentences	Baseline	CCP
PDT dtest	319	0.8228	0.8480
PDT etest	352	0.8343	0.8467
CAC	2,272	0.8096	0.8234
CLTT orig	45	0.7252	0.7350
CLTT multi	41	0.7774	0.7817

Table 5.4: Parsing evaluation on the `OB0` sentences of two different parsing strategies: full-scale parsing using the baseline parser (*Baseline*) and full-scale parsing using CCP (*CCP*).

5.3.2 Parsing governing and dependent clauses

Table 5.5 presents the unlabeled attachment score achieved by the baseline parser for full-scale parsing and parsing individual clauses on `OB1` sentences. We observe almost no improvement when parsing individual clauses. Also, we observe that the parser performance on the `OB1` sentences is significantly higher than the parser performance on the whole datasets, compare the *FS* column in Table 5.5 and the *UAS* column in Table 5.1.

Dataset	# of Sentences	FS	Clauses
PDT dtest	604	0.8824	0.8823
PDT etest	704	0.8798	0.8864
CAC	3,669	0.8568	0.8576
CLTT orig	123	0.8406	0.8403
CLTT multi	188	0.8521	0.8526

Table 5.5: Parsing performance of the baseline parser on the `OB1` sentences when parsing whole sentences (FS) and individual clauses (Clauses).

Given this observation, we proposed the following strategy for parsing subordinated clauses and we updated the *CCP* method as follows:

1. Find the longest sequence of neighboring subordinated clauses C_1, C_2, ..., C_n so that $layer(C_{i+1}) = layer(C_i) + 1$ where *layer* stands for a layer of embedding in a clause chart.
2. Create a sequence $S = C_1\ B_{1,2}\ C_2\ B_{2,3}\ \ldots C_n$ where $B_{i,i+1}$ is a boundary between C_i and C_{i+1}.
3. Apply the baseline parser to sequence S to get a dependency tree T_S.

Using the CCP method for parsing the `OB0` and `OB1` sentences, we can parse the `OB1B0` sentences so that we apply the method for subordinated clauses first and subsequently for coordinated clauses. Table 5.6 presents the comparison of the baseline parser and the CPP method on the full-scale parsing task for `0B1B0` sentences.

Dataset	# of Sentences	Baseline	CCP
PDT dtest	166	0.8172	0.8298
PDT etest	160	0.8198	0.8422
CAC	885	0.7968	0.8084
CLTT orig	65	0.8121	0.8230
CLTT multi	91	0.8168	0.8210

Table 5.6: Parsing evaluation on the `0B1B0` sentences of two different parsing strategies: full-scale parsing (*Baseline*) using the baseline parser and full-scale parsing using CCP (*CCP*).

5.3.3 CCP as Full-scale Parsing

We learned from the experiments that

1. it is efficient to parse coordinated clauses individually and connect their trees subsequently;
2. it is efficient to parse a sequence of governing and dependent clauses at once.

Therefore we proposed and evaluated a final algorithm for dependency parsing that exploits sentence clause charts and a given dependency parser. The algorithm works in iterations. In each iteration, at least one layer of embedding in the clause chart is eliminated using the CCP strategy for `0B0` and `0B1` clauses.

Table 5.7 and Table 5.8 present the final comparison on the full-scale parsing task for the baseline parser and the CCP strategy. We achieved an average 0.78% improvement in UAS when parsing all the sentences in the treebanks.

Dataset	# of Sentences	Baseline	CCP
PDT dtest	2,044	0.8393	0.8472
PDT etest	2,339	0.8384	0.8464
CAC	12,756	0.8199	0.8342
CLTT orig	826	0.7839	0.7955
CLTT multi	1,007	0.8147	0.8201

Table 5.7: Parsing evaluation on the sentences containing at least two clauses.

Dataset	# of Sentences	Baseline	CCP
PDT dtest	4,042	0.8450	0.8503
PDT etest	4,672	0.8432	0.8487
CAC	24,709	0.8268	0.8364
CLTT orig	1,121	0.7924	0.8002
CLTT multi	1,438	0.8179	0.8218

Table 5.8: Final comparison of the full-scale parsing and CCP.

5.4 Improving relation extraction using Clause Chart Parsing

In this section we present an eccentric evaluation of the Clause Chart Parsing (CCP) method. We use the RExtractor system (see Chapter 5) and its Czech Legal Documents Extraction Pipeline as a baseline and compare it with an improved pipeline that exploits CCP. We report 3.40% improvement on the task of the semantic relation extraction.

Improved NLP Component

The improved Czech Legal NLP Component uses the CCP method that exploits the Adapted MST parser [Novák and Žabokrtský, 2007] as the baseline parser. We report 10.12% improvement against the baseline (Experiment 1 in Table 5.9). The CCP method itself (Experiment 4) increased the improvement by 5.89% against the best setup from Chapter 5 (Experiment 3).

	Segmentation	Tokenization	Parsing	UAS	Improvement
1	Standard	Standard	Adapted MST	74.63	–
2	Standard	Re-tokenization	Adapted MST	78.99	5.85%
3	Multi	Re-tokenization	Adapted MST	81.76	9.55%
4	Multi	Re-tokenization	CCP	82.18	10.12%

Table 5.9: Evaluation of the RExtractor's Czech Legal NLP Component for different tokenization, segmentation and parsing strategies. For each experiment we report the unlabeled attachment score (UAS) and the improvement over the baseline (i.e. a standard Treex pipeline for Czech texts).

Improved Entity Detection Component

The improved RExtractor's Entity Detection Component uses the same instance of the Database of Entities (i.e. the CLTT Accounting Dictionary, see Section 3.4). The only difference is that entities are detected over the dependency trees created automatically using the CCP method. Table 5.10 presents the evaluation of the entity detection.

The CCP method brings 0.93% improvement over the baseline RExtractor's Entity Detection Component (Experiment 2). In fact, this presents 99.35% of the component performance that detects entities over the gold standard dependency trees.

	Parsing	TP	FN	FP	Precision	Recall	F1
1	Gold	8,895	748	156	98.28	92.24	95.16
2	Adapted MST	8,818	825	365	96.03	91.44	93.67
3	CCP	8,973	751	284	96.93	92.28	94.55

Table 5.10: Evaluation of the Entity Detection Component for (1) the gold standard dependency trees, (2) dependency trees parsed by the Adapted MST and (3) dependency trees parsed by the CCP approach. For each experiment, we report the number of true positives (TP), false negatives (FN) and false positives (FP) as well as the Precision, Recall and F1-measure.

Improved Relation Extraction Component

We extracted a dataset for supervised machine learning experiments using the same automatic procedure as described in Section 4.2, but running over the dependency trees parsed automatically using the CCP strategy.

Table 5.11 presents the final results. The CCP method (Experiment 4) brings 3.40% improvement above the baseline (Experiment 3) and presents 78.66% of the possible performance having the gold standard dependency trees.

	Parsing method	Entities detection	TP	FN	FP	Prec.	Recall	F1
1	Gold	Gold	441	20	19	95.87%	95.66%	95.77%
2	Gold	Adapted MST	416	40	24	94.55%	91.23%	92.86%
3	Adapted MST	Adapted MST	275	125	80	77.46%	68.75%	72.85%
4	CCP	CCP	290	114	76	79.23%	71.78%	75.32%

Table 5.11: Evaluation of the RExtractor's Czech Legal Relation Extraction Component using supervised machine learning approach running on (1) gold-standard dependency trees and gold-standard entities, (2) gold-standard dependency trees and automatically detected entities and finally on (3) dependency trees and entities, both detected automatically using Adapted MST and finally (4) dependency trees and entities detected automatically using the CCP method. For each experiment, we report the number of true positives (TP), false negatives (FN) and false positives (FP) as well as the Precision, Recall and F1-measure.

6. References Recognition in Czech Court Decisions

In this chapter we describe the task of detection and classification of references in Czech court decisions. We handle these references like entities in the task of Named Entity Recognition. We approach the task using supervised machine learning methods. For this reason, we created the Czech Court Decisions Dataset of manually annotated court decisions. We report F-measure over 90% achieved by our system JTagger.

This work was originally published by Kríž et al. [2014a]. The Czech Court Decisions Dataset is available on-line [Kríž and Hladká, 2014].

6.1 Czech Court Decisions Dataset

We present the Czech Court Decisions Dataset (CCDD) that is a dataset of 300 court decisions published by The Supreme Court of the Czech Republic (SC) and the Constitutional Court of the Czech Republic (CC). In these decisions selected entities were manually detected and classified.

CCDD contains 150 court decisions published by the Supreme Court of the Czech Republic in 2012. We selected them randomly with respect to their distribution over the senates. Next, CCDD contains 150 court decisions published by the Constitutional Court of the Czech Republic in 2004 - 2012.

The following entities were annotated in CCDD:

- references to court decisions
- references to acts
- applicabilities of acts
- institutions

In addition, court decision references were linked with the institutions that issued them. For manual annotation we used the web-based annotation tool Brat [Stenetorp et al., 2012]. The annotators marked entity occurrences and labeled them with an appropriate tag. Then they marked relations between court decisions and institutions, see Figure 6.1.

We did a single annotation of 300 court decision. However, to measure the inter-annotator agreement we selected randomly 15 documents from the dataset and annotated them by three annotators. In average, the annotators marked 551 institutions, 258 court decision references, 402 act references, and 42 applicabilities. We used the Fleiss' kappa to calculate the agreement [Fleiss, 1971]. We report $\kappa = 0.85$ that we interpret as almost perfect agreement.

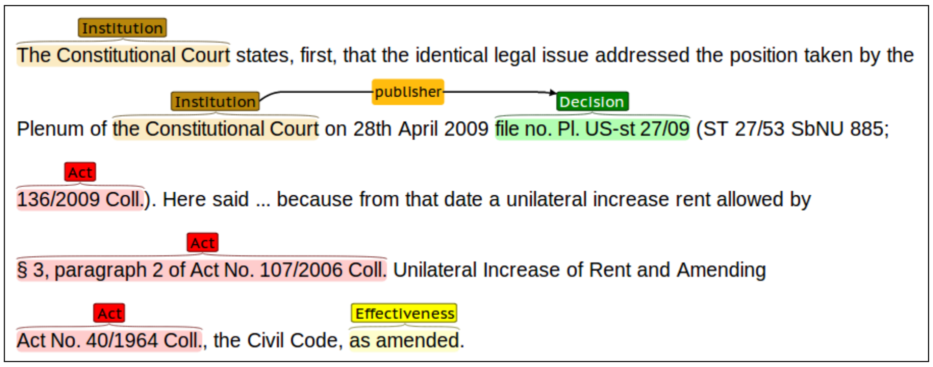

The Constitutional Court states, first, that the identical legal issue addressed the position taken by the Plenum of the Constitutional Court on 28th April 2009 file no. Pl. US-st 27/09 (ST 27/53 SbNU 885; 136/2009 Coll.). Here said ... because from that date a unilateral increase rent allowed by § 3, paragraph 2 of Act No. 107/2006 Coll. Unilateral Increase of Rent and Amending Act No. 40/1964 Coll., the Civil Code, as amended.

Figure 6.1: Annotation of court decisions in Brat.

	SC			CC		
Entity type	# of E	# of T	AEL	# of E	# of T	AEL
Institution	4,891	13,714	2.8	6,318	15,798	2.5
Decision references	1,449	6,967	4.8	1,644	8,146	5.0
Act references	4,387	33,628	7.7	2,597	18,774	7.2
Applicability	247	1,179	4.8	233	938	4.0

Table 6.1: Entity and token distribution in the dataset and average entity lengths (*AEL*) in tokens.

Table 6.1 presents statistics on the 300 annotated documents. The minimum reference length is five tokens. According to the entity lengths, the act references are the most complex entities while the institution references are the simplest ones. To run the cross-validation evaluation, we split dataset into 10 folds.

6.2 Methods and Experiments

We compared performance of two entity detection approaches: (i) the Hidden Markov model algorithm (HMM) as a baseline and (ii) the Perceptron Algorithm with Uneven Margins (PAUM).

Hidden Markov model (HMM)

Hidden Markov Models present historically a very first statistical model applied in the field of NLP [Merialdo, 1994]. The NLP community has effectively employed HMM models for many kinds of efforts starting with POS tagging, NER including [Bikel et al., 1997].

In our model, the output alphabet consists of all possible words occurring in the training data. The states contain labels assigned to the words. The goal is to compute the most likely sequence of tags that has generated the input text, so HMM *annotates* each token.

Perceptron Algorithm with Uneven Margins (PAUM)

The PAUM algorithm [Li et al., 2002] represents a slight modification of the classical Perceptron algorithm [Kim et al., 2005] used in neural networks and extended by SVM [Cortes and Vapnik, 1995]. It provides comparable performance to SVM, with much reduced training times.

In our experiments, PAUM was used in the chunk learning mode with the features listed in Table 6.2. We designed four models. First two models use only word forms. The last two models use lemmatization and POS tagging.

PAUM model	Features
PM small	trigrams of word forms: (w_{i-2}, w_{i-1}, w_i)
PM	5-grams of word forms: $(w_{i-2}, w_{i-1}, w_i, w_{i+1}, w_{i+2})$
PM pos	5-grams of lemmas and part of speech tags: $(l_{i-2}, l_{i-1}, l_i, l_{i+1}, l_{i+2})$, $(t_{i-2}, t_{i-1}, t_i, t_{i+1}, t_{i+2})$
PM pos ext	It extends `PM pos` with an orthography feature and it distinguishes first and last tokens in a sentence. The following orthographic categories are used: *upper initial* (e.g. Czech), *lowercase* (e.g. language), *all caps* (e.g. PAUM) and *mixedCaps* (e.g. JTagger).

Table 6.2: Features used by PAUM

Evaluation

We evaluated the performance of the proposed models using standard evaluation measures. When evaluation is being done on potentially multi-token entities, partially correct (or overlapping) matches can occur. The evaluation can be calculated in two ways, depending on what units are compared. One can use either individual (multi-token) entities or tokens from which the entities are composed of.

- **Token evaluation** uses tokens. It is easier to calculate since there is no need to construct any pairing between discovered and gold-standard entities. It also reflects the proportion of partially correct matches, but it does not reflect situations when e.g. two directly following entities are mistaken for one long entity. As a result, we do not know how many entities are entirely correct and how many only overlap.

- **Entity evaluation** is analogous to the token evaluation, except the number of true negatives can not be taken into account as it does not make sense in this case. *Strict* and *lenient* variants of performance measures allow dealing with partially correct matches in different ways:
  - Strict measures consider all partially correct matches as incorrect (spurious, false positive).

– Lenient measures consider all partially correct matches as correct (true positive).

We performed the experiments using 10-fold cross-validation. Statistical significance was computed using the corrected resampled (two tailed) t-test [Nadeau and Bengio, 2003].

Tables 6.3—6.8 present the cross-validation results, both token- and entity- based F-measure for CC and SC decisions separately.

Entity	HMM	PM pos ext	PM pos	PM	PM small
Act reference	0.75±0.02	0.91±0.02	0.91±0.03	0.89±0.03	0.88±0.03
Decision reference	0.82±0.08	0.97±0.02	0.96±0.02	0.95±0.03	0.94±0.02
Applicability	0.89±0.04	0.90±0.05	0.89±0.05	0.88±0.08	0.82±0.10
Institution	0.92±0.03	0.96±0.02	0.96±0.02	0.95±0.02	0.96±0.02

Table 6.3: Models performance on the SC dataset (average F-measures and confidence intervals over 10 cross-validation folds for each entity type) evaluated using strict entities strategy.

Entity	HMM	PM pos ext	PM pos	PM	PM small
Act reference	0.63±0.05	0.87±0.02	0.86±0.02	0.84±0.03	0.78±0.03
Decision reference	0.83±0.05	0.95±0.03	0.95±0.03	0.93±0.03	0.92±0.03
Applicability	0.96±0.03	0.96±0.03	0.96±0.03	0.96±0.03	0.96±0.03
Institution	0.91±0.02	0.93±0.02	0.93±0.02	0.92±0.01	0.92±0.01

Table 6.4: Models performance on the CC dataset (average F-measures and confidence intervals over 10 cross-validation folds for each entity type) evaluated using strict entities strategy.

Entity	HMM	PM pos ext	PM pos	PM	PM small
Act reference	0.93±0.02	0.96±0.01	0.96±0.01	0.95±0.01	0.95±0.02
Decision reference	0.91±0.03	0.98±0.01	0.97±0.02	0.96±0.02	0.95±0.02
Applicability	0.94±0.04	0.91±0.05	0.90±0.05	0.90±0.06	0.83±0.10
Institution	0.97±0.01	0.98±0.00	0.98±0.01	0.97±0.01	0.97±0.01

Table 6.5: Models performance on the SC dataset (average F-measures and confidence intervals over 10 cross-validation folds for each entity type) evaluated using lenient entities strategy.

Error analysis

We manually checked the output of both HMM and PAUM algorithms and we identified the following rather frequent error types:

- References labeled with two separate tags instead of one tag. For example, in the reference *file no. 7 To 346/2011*, token *To* is not recognized as a

Entity	HMM	PM pos ext	PM pos	PM	PM small
Act reference	0.89±0.02	0.94±0.01	0.94±0.01	0.94±0.01	0.93±0.02
Decision reference	0.93±0.03	0.97±0.02	0.97±0.02	0.96±0.02	0.95±0.03
Applicability	0.96±0.03	0.96±0.03	0.96±0.03	0.96±0.03	0.96±0.03
Institution	0.97±0.01	0.98±0.01	0.98±0.01	0.97±0.01	0.97±0.01

Table 6.6: Models performance on the CC dataset (average F-measures and confidence intervals over 10 cross-validation folds for each entity type) evaluated using lenient entities strategy.

Entity	HMM	PM pos ext	PM pos	PM	PM small
Act reference	0.96±0.01	0.96±0.01	0.96±0.01	0.96±0.02	0.95±0.02
Decision reference	0.95±0.02	0.98±0.01	0.98±0.02	0.97±0.02	0.96±0.02
Applicability	0.94±0.03	0.89±0.06	0.88±0.06	0.88±0.06	0.79±0.12
Institution	0.96±0.01	0.97±0.01	0.97±0.01	0.97±0.01	0.96±0.02

Table 6.7: Models performance on the SC dataset (average F-measures and confidence intervals over 10 cross-validation folds for each entity type) evaluated using token strategy.

part of document reference. Such error may be caused by a lack of train data – models did not learn all possible register marks. Two possible things may help: (i) we should add more annotated data, or (ii) we can add a new boolean feature `is_register_mark` which will be *true* when token matches an item in the list of possible register marks.

- An institution's name ends with a number, like *District Court for Prague 4*, and the last token *4* is not recognized as a part of the reference entity. Analogously to the previous error type, we can add more training data so models learn that *Prague 4* is a district name. Another approach could be to define a new boolean orthography feature `is_number` which will be true for numerical tokens. Models than can learn that court names may end with a number.

- Names of foreign courts, e.g. *Land Court in Norimberg, Germany.* Actually, we can not see any trick how to deal with the foreign courts names other than adding more training data, as foreign court names is an open set.

Links Detection

After the automatic entity detection, a subsequent component detected links between the court decision references and their publishers. According to the CCDD, 96.8%[1] of all links follow the simple strategy:

1. Take the first occurrence of the court decision reference in the input text.

[1]Averaged accuracy over 10 cross-validation folds.

Entity	HMM	PM pos ext	PM pos	PM	PM small
Act reference	0.94±0.01	0.94±0.01	0.93±0.01	0.93±0.02	0.89±0.02
Decision reference	0.95±0.02	0.96±0.02 ∘	0.96±0.01	0.96±0.02	0.94±0.02
Effectiveness	0.96±0.03	0.96±0.04	0.96±0.04	0.96±0.04	0.96±0.04
Institution	0.95±0.01	0.95±0.01	0.95±0.02	0.95±0.01	0.94±0.01

Table 6.8: Models performance on the CC dataset (average F-measures and confidence intervals over 10 cross-validation folds for each entity type) evaluated using token strategy.

2. Look up the nearest institution entity before the reference.

Having this simple rule-based strategy, we did not experiment with any machine learning approaches.

6.3 On-line Demo

The system for the reference recognition in Czech court decisions is distributed as a standalone software called *JTagger*. The on-line demo allows to process any court decision. It is available on

https://quest.ms.mff.cuni.cz/jtagger/

7. Czech Legal Linked Open Data Cloud

The RExtractor system presented in Chapter 4 extracts a knowledge base from legal documents. The JTagger system presented in Chapter 6 recognizes relations between documents by detecting references in court decisions. The structured data extracted from both systems were integrated into the Czech Legal Linked Open Data Cloud (CLLODC) that was created as a part of the Intelligent Library project (INTLIB). In this chapter we firstly describe the INTLIB project and subsequently present the ontologies for capturing a structure between the legal documents as well as for capturing their knowledge base.

7.1 The INTLIB Project

In many domains, large collections of unstructured documents form main sources of information. Their efficient browsing and querying present key aspects in many areas of human activities. The INTLIB project demonstrates how knowledge represented in the Linked Data framework can improve searching large collections of documents. It proposes a new approach to semantic search with two essential parts:

1. an understanding of document semantics; and
2. an integration of extracted knowledge with other machine-readable sources

The semantic search as proposed by INTLIB presents a symbiotic relationship between the fields of Information Extraction and Semantic Web. Figure 7.1 displays the INTLIB document pipeline – components of *Gathering* and *Extracting data* belong to Information Extraction and the components of *Data representation* and *Data linking* belong to Semantic Web. All of them are characterized by general features that are typically domain and language independent. However, their design must take into account the specification of applications that will work with data under consideration.

The INTLIB project shows how legal documents in the Czech Republic can be published as Linked Open Data and demonstrates the advantages of such style

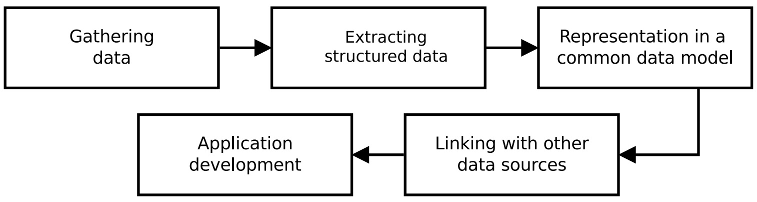

Gathering data
Extracting structured data
Representation in a common data model
Linking with other data sources
Application development

Figure 7.1: A scheme of data extraction, its representation and exploitation.

of publication. As a result, users' comfort increases when searching in collections of the legal texts. Two different types of data were extracted and published in the Linked Data Cloud:

1. **The logical structure between legal documents** (see Section 7.2)
   It includes detecting references between documents and it exploits the system JTagger (Chapter 6),
2. **The semantic interpretation of the legal documents** (see Section 7.3)
   It is represented by relations between entities and it exploits the system RExtractor (Chapter 4, Chapter 5) that extracts definitions, right and obligations from legal texts.

The INTLIB initiative in the legal domain is motivated by the situation of the legal informatics in the Czech Republic (see Section 2.4). Different legal documents (called *sources of law*, for example acts, decrees, regulations and court decisions) are published by the authorities at various places on the Web in a distributed way in different formats (usually HTML or PDF). At some places only metadata about legal documents are published. At other places documents themselves are published. Those different places are not linked in any way. As a result, it is hard for citizens to find legal documents they need and to search for related content. The Czech Legal Linked Open Data Cloud integrates all those different data sources in the machine-readable format.

7.2 Legal Documents Network

Acts, decrees and other documents from the Collection of Laws of the Czech Republic are usually structured to sections which may contain further subsections. In addition, a document may contain references to other documents. A reference may target not only a whole document but also its particular section. Therefore, the structure encoded in documents and references between them form a complex network, for illustration see Figure 7.2. Moreover, related documents are often published by different public authorities.

It would be useful to browse this distributed network among different data sources and search for relationships between documents and their parts. Examples of common use cases are presented in the following list. For illustration, we use Figure 7.2, originally published by Nečaský et al. [2013]. It shows a part of the network comprising several related acts and court decisions.

1. A user is reading a particular section of an act (e.g., *Act Section* §*102* of *Act 99/1963*). He would like to see what court decisions have been made in the last decade related to this particular section (i.e., decisions *20 Cdo 1691/2005* and *20 Co 16/2006*).

2. A user is reading a particular section of an act (e.g., *Act Section* §*5* of *Act 482/1991*). He would like to find out what amendments correcting Section

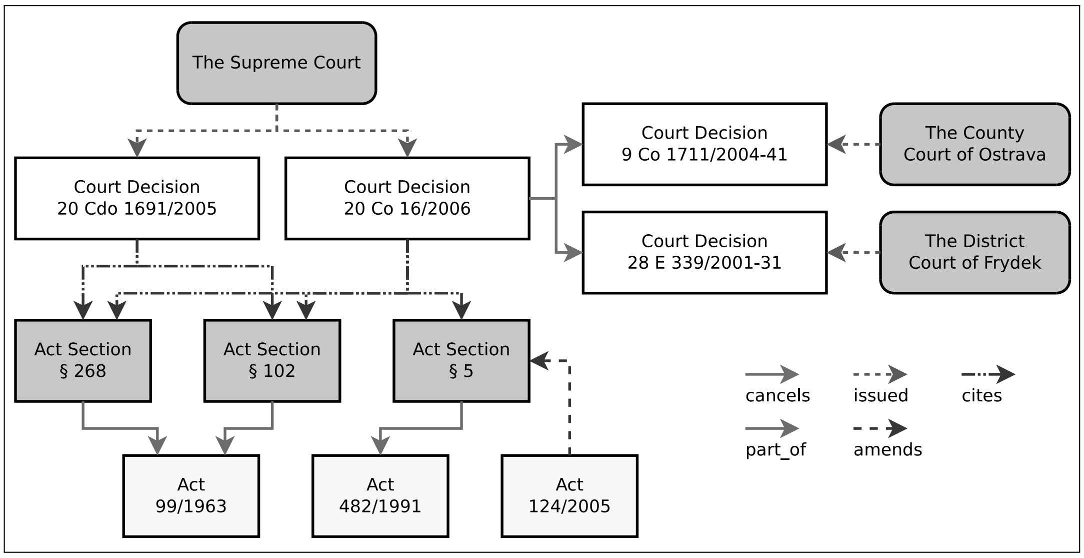

The Supreme Court
Court Decision 20 Cdo 1691/2005
Court Decision 20 Co 16/2006
Court Decision 9 Co 1711/2004-41
Court Decision 28 E 339/2001-31
The County Court of Ostrava
The District Court of Frydek
Act Section § 268
Act Section § 102
Act Section § 5
Act 99/1963
Act 482/1991
Act 124/2005
cancels
issued
cites
part_of
amends

Figure 7.2: Sample Links in Czech court decisions.

§5 belongs to came to force in 2005 (i.e., an amendment of *Act 482/1991* defined by *Act 124/2005*).

3. A user received a court decision (e.g., *20 Cdo 389/2004*). There are various references to other court decisions and also sections of acts and amendments encoded in the text. He would like to see the reading of each of the referenced decisions and sections.

All these use cases are problematic because documents are published as unstructured textual documents by various authorities at different places of the Web. Moreover, the sources are not interlinked at all. Their logical structure (sections and their subsections) and links between them are encoded in the text in a way which can only be interpreted by a human. Therefore, the user can only read the sources and has to search for relationships manually. This is very time consuming, cumbersome and the user can omit important relationships very easily.

Ontology for Legal Documents Network

To address the use cases described above, Nečaský et al. [2013] designed the LEX Ontology — an ontology for Czech Legal Linked Data. In the LEX Ontology:

- *things* are sources of law and their parts;
- useful information about the things are metadata about the sources or their parts, e.g. the creation date, the date from which the source is valid, an author/publisher of the source, etc.;
- links between the things are relationships between the sources, e.g. a section is a part of an act, an act amends another act, court decision cites a section

of an act, court decision cancels another court decision, etc.

The core LEX Ontology classes and properties are presented in Figure 7.3. Classes and properties already covered by the existing ontologies were reused. The resulting LEX Ontology is, therefore, a mixture of existing ontologies and the new components.

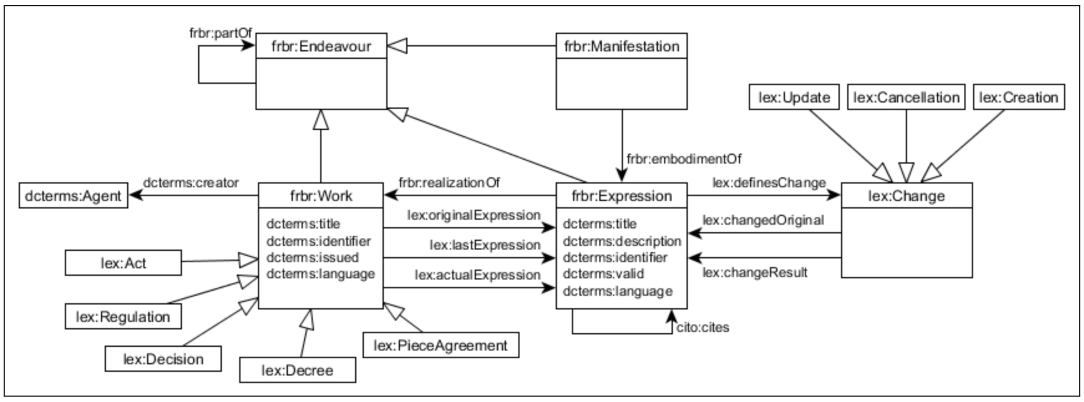


Figure 7.3: The LEX Ontology classes and properties.

For different kinds of legal documents (i.e. sources of law), the ontology defines the following classes:

- `lex:SourceOfLaw` as a superclass of all other classes;
- `lex:Act` for acts;
- `lex:Decree` for decrees;
- `lex:Regulation` for regulations; and
- `lex:Decision` for court decisions

Sources of law of most kinds (except of court decisions) exist in different versions. Some versions are outdated, at most one version is currently valid, and some versions are enacted but have not come to force yet. A particular version may have several physical embodiments, i.e. may be embodied in different physical documents (of different formats, e.g. HTML, PDF, XML, RDF, etc.). From this point of view, it is reasonable to represent a source of law as an abstract notion of intellectual creation which is independent of particular versions of the source. Moreover, each version of the source as well as its each physical embodiment should have representation on its own. To express all these situation, the LEX Ontology exploit the Functional Requirements for Bibliographic Records Ontology (FRBR, [Records and Cataloguing, 2013]). We reuse the following three FRBR classes:

- `frbr:Work` for abstract notions of an intellectual creation which are sources of law;
- `frbr:Expression` for particular versions of sources of law; and
- `frbr:Manifestation` for particular documents which are physical embodiments of particular versions of sources of law.

The usage of FRBR allows to distinguish a source of law itself, its particular versions and their physical embodiments. From the linked data point of view, it is therefore possible to link and query the source of law as an abstract entity which is independent of particular versions of the source. It is also possible to link and query its particular versions.

The LEX Ontology also uses two FRBR properties:

- `frbr:realizationOf` to link the particular source of law with its version; and
- `frbr:embodimentOf` to link the particular embodiment with the version.

For a given source of law we need to know its currently valid version, original version (i.e. the first version), and the last enacted version (which have not necessarily needed to come to force yet). As there are no corresponding properties provided by FRBR, the LEX Ontology introduces three new properties:

- `lex:originalExpression` to link the original (first) version to the respective source of law;
- `lex:actualExpression` to link the currently valid version to the respective source of law; and
- `lex:lastExpression` to link the last enacted version to the respective source of law.

A source of law can change another source of law. For example, an act can be an amendment of one or more another acts. The same is true for other kinds of sources of law. A regulation or decree can change another regulation or decree, respectively, and a court decision can cancel another court decision.

For representing changes, the LEX Ontology proposes a new class `lex:Change`. A change (an instance of `lex:Change`) is defined by some source of law in some version. Therefore, the instance is referred from the respective expression (instance of `frbr:Expression`) which represents that version (via property `lex:definesChange`). Each change says that a given version of a given source of law is changed and a new version of the same source of law is created. Therefore, the resource representing the change is linked to the expression which is changed via property `lex:changedOriginal` and to the expression which is the result of the change via property `lex:changeResult`.

The LEX Ontology is available on-line:

https://github.com/mff-uk/lex-ontology

Legislative Linked Open Data Cloud

In the INTLIB project, we transformed sources of law available in the Czech Republic to the Linked Data representation according to the LEX Ontology. There are several web sites where data about sources of law can be accessed:

- **The National Parliament web site**
  Here, metadata about each act, decree and regulation can be obtained in a form of HTML pages. It is also possible to get information that a given source of law changes another source of law (i.e. updates, creates or changes). However, this is only at the level of whole sources of law. The parliament does not provide information about parts of the sources of law.

- **The National Governmental Portal**
  Here, it is possible to access the original expressions of all acts, regulations and decrees. Also, the consolidated versions of some of these sources of law can be accessed here (approx. 20% of all of the published sources of law). However, the national portal only provides texts, there are no information about their published modifications.

- **The Supreme Court of the Czech Republic and The Constitutional Court of the Czech Republic**
  Both publish their decisions on their web sites. Other courts in the Czech Republic do not publish their decisions on their web sites.

We created a HTTP URI for each such resource and create RDF triples recording metadata for each source. For constructing URIs we chose the following pattern:

`http://linked.opendata.cz/resource/legislation/cz/`
sourceOfLawKind / *validFromYear* / *number*

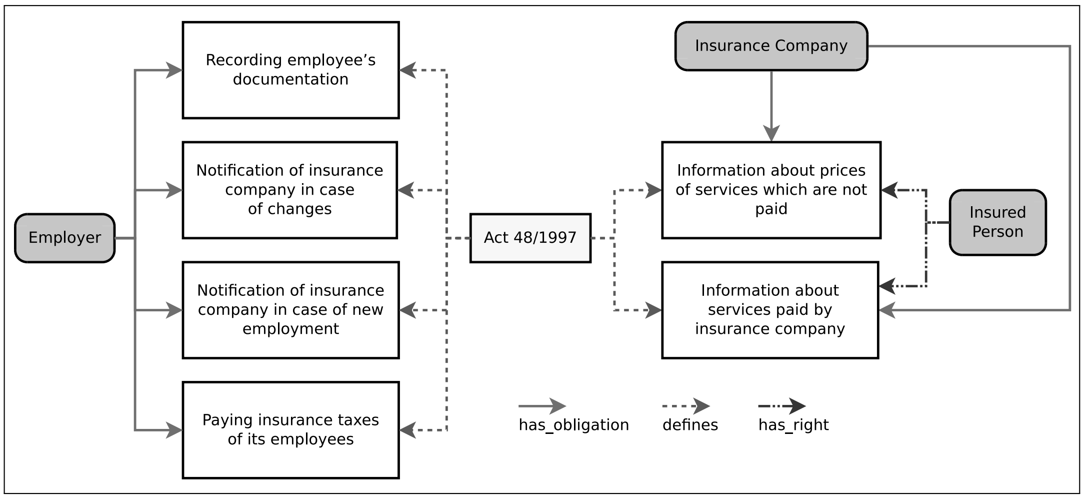
where *sourceOfLawKind* is *act*, *decree*, or *regulation*; *validFromYear* is the year in which the source of law became or will become valid; and *number* is the number of the source of law issued by the parliament. For example, the Public Procurement Act (*Zákon o veřejných zakázkách*), number 137/2006 Sb. is available on the following URI:

```
http://linked.opendata.cz/resource/legislation/
cz/act/2006/137-2006
```

7.3 Legal Documents Semantics

Besides the logical structure and links between documents, acts and other legal documents define rights and obligations of persons. Different documents define different rights and obligations for the same kind of person or for different persons which are, however, semantically related (e.g. one person is a special type of another person and it *inherits* its rights and obligations).

Therefore, the rights and obligations of persons defined by acts and other legal documents form a complex network, similar to the network of links among legal documents described in Section 7.2. In this case the network is defined by the



Recording employee's documentation
Notification of insurance company in case of changes
Notification of insurance company in case of new employment
Paying insurance taxes of its employees
Employer
Act 48/1997
Insurance Company
Information about prices of services which are not paid
Information about services paid by insurance company
Insured Person
has_obligation
defines
has_right

Figure 7.4: A sample of semantic concepts extracted from the Public Health Act.

semantic information encoded in the documents and we can therefore speak about a semantic network or a knowledge graph. Again, it would be useful for users to be able to browse and query such network. In [Kríž et al., 2014b] we list some common use cases and demonstrate them in Figure 7.4:

- A user wants to know what are the obligations of his employer regarding his health insurance. For example, according to the sample network depicted in Figure 7.4, the user can get information that his employer has an obligation to record employee's documentation, notify insurance company about changes in case of changes in employee's information, etc.

- A user wants to know what kind of information his health inssurance company has to provide him. For example, according to Figure 7.4, the user can see that he has the right to obtain information from his insurance company about services provided and paid by the company as well as the information about prices of services which are paid by him.

Ontology for Legislative Documents Semantics

In this section we describe the extension of the LEX Ontology for representing rights, obligations and definitions over the Legislative Documents.

The LEX Ontology Extension has two parts:

1. **Legal Concepts Ontology**
   The ontology enables to represent rights, obligations, persons and relationships between them. Such legislative semantic network will be independent on the source documents from which data are extracted.

2. **Linguistic Ontology**
   The second LEX Ontology Extension allows to map extracted information

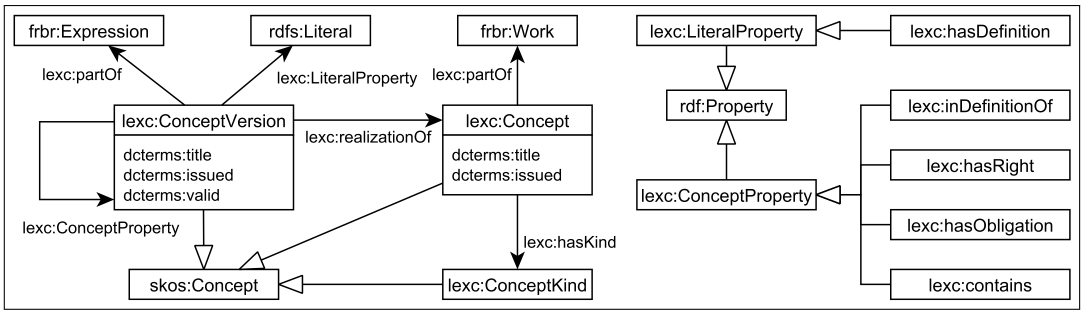


Figure 7.5: Legal Concepts Ontology.

to the original textual context.

The classes and predicates introduced by the Legal Concepts Ontology are presented in Figure 7.5. The core class of the Ontology is `Concept` whose instances represent *concepts* extracted from the source documents.

A concept defined by an act exists independently of particular version of the act. However, because the act exists in one or more versions, there are also respective versions of concepts defined by the act:

- There is a version in which the act defines a concept for the first time.
- The concept then exists in the following versions of the act until it is canceled. For each following version, there is a respective version of the concept.

Therefore, for each version of the act which mentions the concept, we also create an instance of the class `ConceptVersion`. Such instance represents a particular version of the concept defined by the respective version of the act.

Each concept is linked to the act and its sections. It enables to show users the list of concepts which appear in a chosen act or in any of its sections.

The extracted relations between concepts and their literal properties are represented with sub-properties of the abstract properties `ConceptProperty` and `LiteralProperty`, respectively. However, because each relationship and literal property is extracted from a particular version of an act, the domain of those properties is not the class `Concept` but the class `ConceptVersion`.

As Figure 7.5 shows, there are various sub-properties of the abstract properties and it is easy to add new properties. To represent definitions, rights and obligations, one can use:

- a literal property `hasDefinition`, which enables to link a concept to its literal definition;
- a concept property `hasRight`; and
- a concept property `hasObligation`.

Properties enable to link a concept to another concept which is the right or obligation of the concept, or is contained in the definition of the concept, or is a

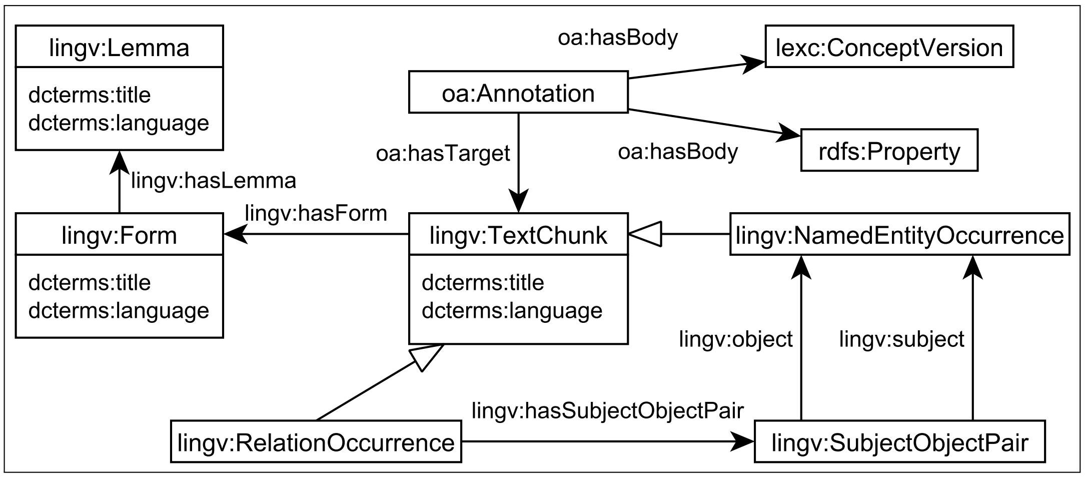


Figure 7.6: Linguistic LEX Ontology.

part of that concept, respectively.

The Legal Concepts Ontology enables to search for literal or concept properties. However, it is not possible to show users the original text of the act from which a property was extracted. Thus, we provide the Linguistic LEX Ontology Extension. The classes and predicates defined by the ontology are depicted in Figure 7.6.

The core class of the ontology is the `TextChunk` class. It represents a part of the original text (called text chunk) which is the occurrence of some entity (see the sub-class `NamedEntityOccurrence`) or the occurrence of a relationship specification (see the sub-class `RelationOccurrence`).

Each text chunk is annotated by its meaning which is

- a version of some concept (an instance of the class `ConceptVersion`);
- relationship between two concepts (a sub-property of `ConceptProperty`); or
- literal property (a sub-property of `LiteralProperty`).

For representing annotations in RDF we use the Open Annotation Ontology (we use prefix `oa:`).[1]

The Linguistic LEX Ontology Extension enables to display users text chunks from which concepts and relations between them are extracted. Because a text chunk is also a part of the original text, we are able to show users each text chunk in the context of original documents.

[1]http://www.openannotation.org/spec/core/

Legislative Linked Open Data Cloud

In the INTLIB project, we used RExtractor to extract entities, their definitions, rights and obligations from available original expressions of all acts, regulations and decrees. Also, we processed all available consolidated versions of the documents. We published the outputs in the Linked Data representation according to the Extended LEX Ontology.

8. Conclusion

In this thesis we present the strategy to automate the extraction of knowledge from documents. We implemented the extraction system RExtractor and improved several natural language processing tools, namely (i) the sentence segmentation procedure that multiplicates complex sentences, (ii) the re-tokenization procedure that significantly decreases the complexity of long sentences and (iii) the clause chart parsing method that improves the dependency parsing of complex sentences. Also, we implemented the system JTagger that detects references in court decisions.

The structured data extracted by both the systems were integrated into the Czech Legal Linked Open Data Cloud. This cloud encodes a complex network of legal documents. Moreover, for each document it contains its knowledge base consisting of entities and relations. The cloud is machine-readable thanks to the proposed and extended LEX Ontology.

The work presented in this thesis helps to solve the current problems in the Czech legislation where documents are published by different authorities at various places, not inter-linked in any way and in different formats. The extracted structured data placed in the cloud are ready for developers of applications that improve users' comfort when searching in collections of legal texts.

We created two original language sources: (i) the Czech Legal Text Treebank with several layers of linguistic annotation and (ii) the Czech Court Decisions Dataset with manually annotated entities.

Future Perspectives

We can see several future perspectives regarding the Czech Legal Text Treebank. Multiplicated complex sentences created and used for the evaluation of the RExtractor pipeline are not distributed in the current version of the treebank. They should be published in the next version of the treebank. So far, only the a-layer of the treebank has been ported into the Universal Dependences (UD). We plan to explore the possibilities of porting the e-layer and the r-layer into to UD as well. Currently, the main data format of the treebank is quite deprecated and readable only by the TrEd application. The UD should become the main CLTT data format. As the treebank contains quite small number of sentences, it would be beneficial to add more sentences, especially the complex ones.

Regarding RExtractor, the next version of the system should consider new technology trends that have appeared since the implementation. We propose to refactor the system in the Python programming language. Regarding new NLP tools and approaches, it would be nice to exploit the UD tools in the pipeline, from the NLP component to the tree queries mechanism in the Relation Extraction components. Finally, it would be very exciting to update the RExtractor system about the newest dependency parsers that would use word embeddings trained on legal texts.

Regarding the JTagger system, we can see its next development analogously to the RExtractor system. It should be reimplemented in Python and offer the REST API for easier system integration. The resolving step of the references detection is actually implemented in Java outside JTagger. It would be nice to merge these two steps into a single system.

In the presented experiments with the clause chart parsing method (CPP), we used the parser trained on complete sentences. However, the CCP strategy requires to parse individual clauses in some situations. We already tried to train a specialized parser for different situations, but the models did not achieve the reliable performance, as we significantly reduced the amount of the training data. However, we can still see two potential ways for next experiments: (i) applying the method to Universal Dependences could bring more training data and (ii) using new parsers could bring another improvement.

Having the Legal Linked Open Data Cloud, developers are able to implement new useful applications. Justinian.cz[1] presents such application based on the cloud data. Currently, the application as well as the cloud is no more updated after the end of the INTLIB project. We believe that it should be maintained and updated, especially in the current state of the Czech legal informatics, where the government projects are still not finished.

[1] http://justinian.cz

Bibliography

Steven Abney. Partial Parsing via Finite-State Cascades. *Natural Language Engineering*, 2(04):337–344, 1996.

Tommaso Agnoloni, Meritxell Fernández Barrera, Maria Teresa Sagri, Daniela Tiscorni, and Giulia Venturi. When a framenet-style knowledge description meets an ontological characterization of fundamental legal concepts. In Pompeu Casanovas, Ugo Pagallo, Giovanni Sartor, and Gianmaria Ajani, editors, *AI Approaches to the Complexity of Legal Systems. Complex Systems, the Semantic Web, Ontologies, Argumentation, and Dialogue*, pages 93–112, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg. ISBN 978-3-642-16524-5.

Tommaso Agnoloni, Lorenzo Bacci, Ginevra Peruginelli, Marc van Opijnen, Jos van den Oever, Monica Palmirani, Luca Cervone, Octavian Bujor, Arantxa Arsuaga Lecuona, Alberto Boada García, Luigi Di Caro, and Giovanni Siragusa. Linking european case law: BO-ECLI parser, an open framework for the automatic extraction of legal links. In Adam Z. Wyner and Giovanni Casini, editors, *Legal Knowledge and Information Systems - JURIX 2017: The Thirtieth Annual Conference, Luxembourg, 13-15 December 2017.*, volume 302 of *Frontiers in Artificial Intelligence and Applications*, pages 113–118. IOS Press, 2017. doi: 10.3233/978-1-61499-838-9-113. URL https://doi.org/10.3233/978-1-61499-838-9-113.

Harith Alani, Sanghee Kim, David Millard, Mark J. Weal, Wendy Hall, Paul H. Lewis, and Nigel R. Shadbolt. Automatic Ontology-Based Knowledge Extraction from Web Documents. *Intelligent Systems, IEEE*, 18:14–21, 01 2003. doi: 10.1109/MIS.2003.1179189.

Isabelle Augenstein, Sebastian Padó, and Sebastian Rudolph. Lodifier: Generating linked data from unstructured text. In Elena Simperl, Philipp Cimiano, Axel Polleres, Oscar Corcho, and Valentina Presutti, editors, *The Semantic Web: Research and Applications*, pages 210–224, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg. ISBN 978-3-642-30284-8.

L Bacci, E Francesconi, and MT Sagri. A rule-based parsing approach for detecting case law references in italian court decisions. In *Semantic Processing of Legal Texts (SPLeT-2012) Workshop Programme*, page 27, 2012.

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. The Berkeley FrameNet Project. In *Proceedings of the 17th International Conference on Computational Linguistics - Volume 1*, COLING '98, pages 86–90, Stroudsburg, PA, USA, 1998. Association for Computational Linguistics. doi: 10.3115/980451.980860. URL https://doi.org/10.3115/980451.980860.

M. Barsky, A. Thomo, and U. Stege. *Full-Text (Substring) Indexes in External Memory*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2011. ISBN 9781608457960. URL https://books.google.cz/books?id=MeVcAQAAQBAJ.

Eduard Bejček, Jarmila Panevová, Jan Popelka, Pavel Straňák, Magda Ševčíková, Jan Štěpánek, and Zdeněk Žabokrtský. Prague Dependency Treebank 2.5 – a revisited version of PDT 2.0. In Martin Kay and Christian Boitet, editors, *Proceedings of the 24th International Conference on Computational Linguistics (COLING 2012)*, pages 231–246, Mumbai, India, 2012. IIT Bombay, COLING 2012 Organizing Committee.

Eduard Bejček, Eva Hajičová, Jan Hajič, Pavlína Jínová, Václava Kettnerová, Veronika Kolářová, Marie Mikulová, Jiří Mírovský, Anna Nedoluzhko, Jarmila Panevová, Lucie Poláková, Magda Ševčíková, Jan Štěpánek, and Šárka Zikánová. Prague Dependency Treebank 3.0. Prague, Czech republic, 2013. Univerzita Karlova v Praze, MFF, ÚFAL, Univerzita Karlova v Praze, MFF, ÚFAL. URL http://ufal.mff.cuni.cz/pdt3.0.

Trevor Bench-Capon, MichałAraszkiewicz, Kevin Ashley, Katie Atkinson, Floris Bex, Filipe Borges, Daniele Bourcier, Paul Bourgine, Jack G. Conrad, Enrico Francesconi, Thomas F. Gordon, Guido Governatori, Jochen L. Leidner, David D. Lewis, Ronald P. Loui, L. Thorne Mccarty, Henry Prakken, Frank Schilder, Erich Schweighofer, Paul Thompson, Alex Tyrrell, Bart Verheij, Douglas N. Walton, and Adam Z. Wyner. A History of AI and Law in 50 Papers: 25 Years of the International Conference on AI and Law. *Artif. Intell. Law*, 20(3): 215–319, September 2012. ISSN 0924-8463. doi: 10.1007/s10506-012-9131-x. URL http://dx.doi.org/10.1007/s10506-012-9131-x.

T. Berners-Lee. Linked Data. *W3C Design Issues*, 7 2006. URL https://www.w3.org/DesignIssues/LinkedData.html.

Tim Berners-Lee, James Hendler, and Ora Lassila. The Semantic Web. *Scientific American*, 284(5):34–43, May 2001. URL http://www.sciam.com/article.cfm?articleID=00048144-10D2-1C70-84A9809EC588EF21.

Daniel M Bikel, Scott Miller, Richard Schwartz, and Ralph Weischedel. Nymble: a high-performance learning name-finder. In *Proceedings of the fifth conference on Applied natural language processing*, pages 194–201. Association for Computational Linguistics, 1997.

C. Bizer, T. Heath, and T. Berners-Lee. Linked data - the story so far. *Int. J. Semantic Web Inf. Syst.*, 5(3):1–22, 2009.

Guido Boella, Luigi Di Caro, Michele Graziadei, Loredana Cupi, Carlo Emilio Salaroglio, Llio Humphreys, Hristo Konstantinov, Kornel Marko, Livio Robaldo, Claudio Ruffini, Kiril Simov, Andrea Violato, and Veli Stroetmann. Linking Legal Open Data: Breaking the Accessibility and Language Barrier in European Legislation and Case Law. In *Proceedings of the 15th International Conference on Artificial Intelligence and Law*, ICAIL '15, pages 171–175, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3522-5. doi: 10.1145/2746090.2746106. URL http://doi.acm.org/10.1145/2746090.2746106.

Alexander Boer. MetaLex Naming Conventions and the Semantic Web. In *Proceedings of the 2009 Conference on Legal Knowledge and Information Systems:*

JURIX 2009: The Twenty-Second Annual Conference, pages 31–36, Amsterdam, The Netherlands, The Netherlands, 2009. IOS Press. ISBN 978-1-60750-082-7. URL http://dl.acm.org/citation.cfm?id=1671082.1671087.

Mírian Bruckschen, Caio Northfleet, DM Silva, Paulo Bridi, Roger Granada, Renata Vieira, Prasad Rao, and Tomas Sander. Named entity recognition in the legal domain for ontology population. In *Workshop Programme*, page 16, 2010.

Nancy A. Chinchor. Proceedings of the Seventh Message Understanding Conference (MUC-7) Named Entity Task Definition. In *Proceedings of the Seventh Message Understanding Conference (MUC-7)*, Fairfax, VA, April 1998. URL http://acl.ldc.upenn.edu/muc7/ne_task.html.

Fabio Ciravegna and Alberto Lavelli. Full Text Parsing Using Cascades of Rules: An Information Extraction Perspective. In *Proceedings of the Ninth Conference on European Chapter of the Association for Computational Linguistics*, EACL '99, pages 102–109, Stroudsburg, PA, USA, 1999. Association for Computational Linguistics. doi: 10.3115/977035.977050. URL http://dx.doi.org/10.3115/977035.977050.

Corinna Cortes and Vladimir Vapnik. Support-Vector Networks. *Mach. Learn.*, 20 (3):273–297, September 1995. ISSN 0885-6125. doi: 10.1023/A:1022627411411. URL http://dx.doi.org/10.1023/A:1022627411411.

František Cvrček. *Právní informatika.* Aleš Čeněk s.r.o., 2010. ISBN 978-80-87439-00-5.

František Cvrček, Jan Kořenský, and František Novák. *Juristická a lingvistická analýza právních textů: (právněinformatický přístup).* Academia, 1999. ISBN 80-200-0730-X.

František Cvrček, Karel Pala, and Pavel Rychlý. Legal electronic dictionary for Czech. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Mehmet Uğur Doğan, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey, may 2012. European Language Resources Association (ELRA). ISBN 978-2-9517408-7-7.

Emile de Maat, Radboud Winkels, and Tom M. van Engers. Automated detection of reference structures in law. In Tom M. van Engers, editor, *JURIX*, volume 152 of *Frontiers in Artificial Intelligence and Applications*, pages 41–50. IOS Press, 2006. ISBN 978-1-58603-698-0.

Ralph Debusmann, Denys Duchier, and Andreas Rossberg. Modular Grammar Design with Typed Parametric Principles. *Proceedings of FG-MOL 2005*, 2005.

Christopher Dozier, Ravikumar Kondadadi, Marc Light, Arun Vachher, Sriharsha Veeramachaneni, and Ramdev Wudali. Named entity recognition and resolution in legal text. In *Semantic Processing of Legal Texts*, pages 27–43. Springer, 2010.

Stefano Federici, Simonetta Montemagni, and Vito Pirrelli. Shallow Parsing and Text Chunking: a View on Underspecification in Syntax. *Cognitive Science Research Paper - University of Sussex CSRP*, pages 35–44, 1996.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 363–370, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics. doi: 10.3115/1219840.1219885. URL https://doi.org/10.3115/1219840.1219885.

Joseph L Fleiss. Measuring nominal scale agreement among many raters. *Psychological bulletin*, 76(5):378, 1971.

Marco Fossati, Emilio Dorigatti, and Claudio Giuliano. N-ary relation extraction for simultaneous t-box and a-box knowledge base augmentation. *Semantic Web*, 9(4):413–439, 2018.

André Freitas, Danilo S. Carvalho, João Carlos Pereira da Silva, Seán O'Riain, and Edward Curry. A semantic best-effort approach for extracting structured discourse graphs from wikipedia. In *WoLE@ISWC*, volume 906 of *CEUR Workshop Proceedings*, pages 70–81. CEUR-WS.org, 2012.

Fabien Gandon, Guido Governatori, and Serena Villata. Normative Requirements as Linked Data. In *JURIX 2017 - The 30th international conference on Legal Knowledge and Information Systems* , pages 1–10, Luxembourg, Luxembourg, December 2017. URL https://hal.archives-ouvertes.fr/hal-01643769.

John Gibbons. *Language and the Law*, pages 285–303. Blackwell Publishing Ltd, 2008.

Jan Hajič, Jarmila Panevová, Eva Buráňová, Zdeňka Urešová, and Alla Bémová. Anotace Pražského závislostního korpusu na analytické rovině: pokyny pro anotátory. Technical Report 28, ÚFAL MFF UK, Prague, Czech Republic, 1999.

Jan Hajič, Barbora Vidová Hladká, Jarmila Panevová, Eva Hajičová, Petr Sgall, and Petr Pajas. Prague Dependency Treebank 1.0 (Final Production Label). In *CDROM*. CAT: LDC2001T10., ISBN 1-58563-212-0, 2001.

Jan Hajič, Jarmila Panevová, Eva Hajičová, Petr Sgall, Petr Pajas, Jan Štěpánek, Jiří Havelka, Marie Mikulová, Zdeněk Žabokrtský, and Magda Ševčíková-Razímová. Prague Dependency Treebank 2.0, 2006.

Jan Hajič, Eduard Bejček, Alevtina Bémová, Eva Buráňová, Eva Hajičová, Jiří Havelka, Petr Homola, Jiří Kárník, Václava Kettnerová, Natalia Klyueva, Veronika Kolářová, Lucie Kučová, Markéta Lopatková, Marie Mikulová, Jiří Mírovský, Anna Nedoluzhko, Petr Pajas, Jarmila Panevová, Lucie Poláková, Magdaléna Rysová, Petr Sgall, Johanka Spoustová, Pavel Straňák, Pavlína Synková, Magda Ševčíková, Jan Štěpánek, Zdeňka Urešová, Barbora Vidová Hladká, Daniel Zeman, Šárka Zikánová, and Zdeněk Žabokrtský. Prague Dependency Treebank 3.5, 2018. URL http://hdl.handle.net/11234/

1-2621. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (
'UFAL), Faculty of Mathematics and Physics, Charles University.

Sherzod Hakimov, Hendrik ter Horst, Soufian Jebbara, Matthias Hartung, and Philipp Cimiano. Combining textual and graph-based features for named entity disambiguation using undirected probabilistic graphical models. In Eva Blomqvist, Paolo Ciancarini, Francesco Poggi, and Fabio Vitali, editors, *Knowledge Engineering and Knowledge Management*, pages 288–302, Cham, 2016. Springer International Publishing. ISBN 978-3-319-49004-5.

Terry Halpin and Tony Morgan. Front matter. In *Information Modeling and Relational Databases (Second Edition)*, The Morgan Kaufmann Series in Data Management Systems, page iii. Morgan Kaufmann, San Francisco, second edition edition, 2008. ISBN 978-0-12-373568-3. doi: https://doi.org/10.1016/B978-0-12-373568-3.50029-1. URL http://www.sciencedirect.com/science/article/pii/B9780123735683500291.

Jakub Harašta and Jaromír Šavelka. Toward Linking Heterogenous References in Czech Court Decisions to Content. In Giovanni Casini Adam Wyner, editor, *Frontiers in Artificial Intelligence and Applications , vol. 302 (Proceedings of JURIX 2017)*, pages 177–182, Amsterdam - Berlin - Washington, DC, 2017. IOS Press. ISBN 978-1-61499-838-9.

Bohuslav Havránek. *Spisovná čeština a jazyková kultura.* Praha: Melantrich, 1932.

Bin He, Yi Guan, and Rui Dai. Classifying medical relations in clinical text via convolutional neural networks. *Artificial intelligence in medicine*, 93:43–49, 2019.

Barbora Vidová Hladká, Jan Hajič, Jiří Hana, Jaroslava Hlaváčová, Jiří Mírovský, and Jan Raab. The Czech Academic Corpus 2.0 Guide. *The Prague Bulletin of Mathematical Linguistics*, (89):41–96, 2008. ISSN 0032-6585.

Rinke Hoekstra. The Metalex Document Server Legal Documents As Versioned Linked Data. In *Proceedings of the 10th International Conference on The Semantic Web - Volume Part II*, ISWC'11, pages 128–143, Berlin, Heidelberg, 2011. Springer-Verlag. ISBN 978-3-642-25092-7. URL http://dl.acm.org/citation.cfm?id=2063076.2063086.

Irena Holubová, Vincent Kríž, Martin Nečaský, and Barbora Hladká. INTLIB - an INTelligent LIBrary. In Karel Richta and Václav Snášel, editors, *Proceedings of the Dateso 2014 Annual International Workshop on DAtabases, TExts, Specifications and Objects*, pages 13–24, Praha, Czechia, 2014. Czech Technical University in Prague, Faculty of Information Technology, Czech Technical University in Prague, Faculty of Information Technology. ISBN 978-80-01-05482-6.

Samar Husain, Phani Gadde, Joakim Nivre, and Rajeev Sangal. Clausal parsing helps data-driven dependency parsing: Experiments with Hindi. In *Proceedings of the 5th International Joint Conference on Natural Language Processing*, page 1279–1287, Chiang Mai, Thailand, 2011.

Milan Jelínek. Kultura jazyka a odborný styl. In *Termina 94*, pages 7–29, Liberec: PFTU, 1995.

Milan Jelínek. Styl administrativně-právní. In *Jazyk a jeho užívání. Sborník k životnímu jubileu prof. O. Uličného*, pages 240–250, Praha, 1996.

Heng Ji, Xiaoman Pan, Boliang Zhang, Joel Nothman, James Mayfield, Paul McNamee, and Cash Costello. Overview of TAC-KBP2017 13 Languages Entity Discovery and Linking. In *Proceedings of the 2017 Text Analysis Conference, TAC 2017, Gaithersburg, Maryland, USA, November 13-14, 2017*. NIST, 2017. URL https://tac.nist.gov/publications/2017/additional.papers/TAC2017.KBP_Entity_Discovery_and_Linking_overview.proceedings.pdf.

Hans Kamp. A Theory of Truth and Semantic Representation. In P. Portner and B. H. Partee, editors, *Formal Semantics - the Essential Readings*, pages 189–222. Blackwell, 1981.

Kwangbaek Kim, Sungshin Kim, Younghoon Joo, and Am-Sok Oh. Enhanced fuzzy single layer perceptron. In Jun Wang, Xiaofeng Liao, and Zhang Yi, editors, *Advances in Neural Networks – ISNN 2005*, volume 3496 of *Lecture Notes in Computer Science*, pages 603–608. Springer Berlin Heidelberg, 2005. ISBN 978-3-540-25912-1. doi: 10.1007/11427391_96. URL http://dx.doi.org/10.1007/11427391_96.

Viktor Knapp. *O moznosti pouziti kybernetickych metod v pravu*. Nakladatelství Československé akademie věd, 1963.

Viktor Knapp and Jiří Cejpek. *Automatizované vyhladávanie informácií v právnych textoch*. Slovenská technická knižnica, 1979.

Jan Kořenský. Právní jazyk, právní komunikace a interpretace. In *Stát a právo 27*, pages 33–39, Praha, 1989.

Vincent Kríž and Barbora Hladká. Czech Court Decisions Dataset, 2014. URL http://hdl.handle.net/11234/1-2853. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Vincent Kríž and Barbora Hladká. RExtractor: a Robust Information Extractor. In Matt Gerber, Catherine Havasi, and Finley Lacatusu, editors, *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 21–25, Stroudsburg, PA, USA, 2015. Association for Computational Linguistics. ISBN 978-1-941643-49-5.

Vincent Kríž and Barbora Hladká. Improving Dependency Parsing Using Sentence Clause Charts. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics – Student Research Workshop*, pages 86–92, Stroudsburg, PA, USA, 2016. Association for Computational Linguistics, Association for Computational Linguistics. ISBN 978-1-945626-02-9.

Vincent Kríž and Barbora Hladká. Czech Legal Text Treebank 2.0, 2017. URL http://hdl.handle.net/11234/1-2498. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Vincent Kríž, Barbora Hladká, Martin Nečaský, and Jan Dědek. Statistical Recognition of References in Czech Court Decisions. In *13th Mexican International Conference on Artificial Intelligence, MICAI 2014, Tuxtla Gutiérrez, Mexico, November 16-22, 2014. Proceedings, Part I*, volume 8856 of *Lecture Notes in Computer Science*, pages 51–61, Switzerland, 2014a. Instituto Tecnológico de Tuxtla Gutiérrez, Springer International Publishing. ISBN 978-3-319-13646-2.

Vincent Kríž, Barbora Hladká, Martin Nečaský, and Tomáš Knap. Data Extraction Using NLP Techniques and Its Transformation to Linked Data. In *13th Mexican International Conference on Artificial Intelligence, MICAI 2014, Tuxtla Gutiérrez, Mexico, November 16-22, 2014. Proceedings, Part I*, volume 8856 of *Lecture Notes in Computer Science*, pages 113–124, Switzerland, 2014b. Instituto Tecnológico de Tuxtla Gutiérrez, Springer International Publishing. ISBN 978-3-319-13646-2.

Vincent Kríž, Barbora Hladká, and Zdeňka Urešová. Czech Legal Text Treebank, 2015. URL http://hdl.handle.net/11234/1-1516. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Vincent Kríž, Barbora Hladká, and Zdeňka Urešová. Czech Legal Text Treebank 1.0. In Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Marko Grobelnik, Bente Maegaard, Joseph Mariani, Asunción Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC 2016)*, pages 2387–2392, Paris, France, 2016. European Language Resources Association. ISBN 978-2-9517408-9-1.

Oldřich Krůza and Vladislav Kuboň. Automatic Recognition of Clauses. *International Journal of Computational Linguistics and Applications*, 5(1):125–138, 2014. ISSN 0976-0962.

Vincent Kríž and Barbora Hladká. Czech Legal Text Treebank 2.0. In Nicoletta Calzolari (Conference chair), Khalid Choukri, Christopher Cieri, Thierry Declerck, Sara Goggi, Koiti Hasida, Hitoshi Isahara, Bente Maegaard, Joseph Mariani, Hélène Mazo, Asuncion Moreno, Jan Odijk, Stelios Piperidis, and Takenobu Tokunaga, editors, *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Paris, France, may 2018. European Language Resources Association (ELRA). ISBN 979-10-95546-00-9.

Vladislav Kuboň. *Problems of Robust Parsing of Czech*. PhD thesis, Faculty of Mathematics and Physics, Charles University in Prague, Prague, Czech Republic, 2001.

Vladislav Kuboň, Markéta Lopatková, Martin Plátek, and Patrice Pognan. A Linguistically-Based Segmentation of Complex Sentences. In David Wilson and Geoffrey Sutcliffe, editors, *Proceedings of FLAIRS 2007 (20th International Florida Artificial Intelligence Research Society Conference)*, pages 368–373, Key West, FL, USA, 2007. AAAI Press. ISBN 978-1-57735-319-5.

Yaoyong Li, Hugo Zaragoza, Ralf Herbrich, John Shawe-Taylor, and Jaz S. Kandola. The Perceptron Algorithm with Uneven Margins. In *Proceedings of the Nineteenth International Conference on Machine Learning*, ICML '02, pages 379–386, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc. ISBN 1-55860-873-7. URL http://dl.acm.org/citation.cfm?id=645531.655993.

Marek Lipczak, Arash Koushkestani, and Evangelos Milios. Tulip: Lightweight Entity Recognition and Disambiguation Using Wikipedia-based Topic Centroids. In *Proceedings of the First International Workshop on Entity Recognition &#38; Disambiguation*, ERD '14, pages 31–36, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-3023-7. doi: 10.1145/2633211.2634351. URL http://doi.acm.org/10.1145/2633211.2634351.

Markéta Lopatková and Tomáš Holan. Segmentation Charts for Czech – Relations among Segments in Complex Sentences. In Adrian Dediu, Armand Ionescu, and Carlos Martín-Vide, editors, *Language and Automata Theory and Applications. Third International Conference, LATA 2009, Tarragona, Spain, April 2-8, 2009. Proceedings*, volume 5457 of *Lecture Notes in Computer Science*, pages 542–553, Berlin / Heidelberg, 2009. Universitat Rovira i Virgili, Springer. ISBN 978-3-642-00981-5.

Markéta Lopatková, Petr Homola, and Natalia Klyueva. Annotation of Sentence Structure: Capturing the Relationship between Clauses in Czech Sentences. *Language Resources and Evaluation*, 46(1):25–36, 2012. ISSN 1574-020X.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. Non-Projective Dependency Parsing using Spanning Tree Algorithms. In *Proceedings of Human Langauge Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 523–530, Vancouver, BC, Canada, 2005. Association for Computational Linguistics, Association for Computational Linguistics. ISBN 1-932432-55-8.

Bernard Merialdo. Tagging English text with a probabilistic model. *Comput. Linguist.*, 20(2):155–171, June 1994. ISSN 0891-2017. URL http://dl.acm.org/citation.cfm?id=972525.972526.

George A. Miller. WordNet: A Lexical Database for English. *Commun. ACM*, 38(11):39–41, November 1995. ISSN 0001-0782. doi: 10.1145/219717.219748. URL http://doi.acm.org/10.1145/219717.219748.

Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 1003–1011, Suntec, Singapore, August 2009. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/P09-1113.

Eva Minářová, Marie Krčmová, Jan Chloupek, and Marie Čechová. *Současná česká stylistika*. ISV nakladatelství, Praha, 1 edition, 2003. ISBN 80-86642-00-3.

Claude Nadeau and Yoshua Bengio. Inference for the generalization error. *Machine Learning*, 52(3):239–281, 2003. ISSN 0885-6125. doi: 10.1023/A:1024068626366. URL http://dx.doi.org/10.1023/A%3A1024068626366.

Ndapandula Nakashole, Martin Theobald, and Gerhard Weikum. Scalable Knowledge Harvesting with High Precision and High Recall. In *Proceedings of the Fourth ACM International Conference on Web Search and Data Mining*, WSDM '11, pages 227–236, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0493-1. doi: 10.1145/1935826.1935869. URL http://doi.acm.org/10.1145/1935826.1935869.

Ndapandula Nakashole, Gerhard Weikum, and Fabian Suchanek. Discovering Semantic Relations from the Web and Organizing Them with PATTY. *SIGMOD Rec.*, 42(2):29–34, July 2013. ISSN 0163-5808. doi: 10.1145/2503792.2503799. URL http://doi.acm.org/10.1145/2503792.2503799.

Adeline Nazarenko, Francois Levy, and Adam Wyner. Towards a methodology for formalizing legal texts in LegalRuleML. In *Legal Knowledge and Information Systems - JURIX 2016*, volume 294 of *Frontiers in Artificial Intelligence and Applications*, pages 149–154, Netherlands, 2016. IOS Press. doi: 10.3233/978-1-61499-726-9-149.

Martin Nečaský, Tomáš Knap, Jakub Klímek, Irena Holubová, and Barbora Hladká. Linked Open Data for Legislative Domain - Ontology and Experimental Data. In *Lecture Notes in Business Information Processing*, pages 172–183, Berlin / Heidelberg, 2013. Springer. ISBN 978-3-642-41686-6.

Dat Ba Nguyen, Johannes Hoffart, Martin Theobald, and Gerhard Weikum. AIDA-light: High-Throughput Named-Entity Disambiguation. In *Proceedings of the Workshop on Linked Data on the Web co-located with the 23rd International World Wide Web Conference (WWW 2014), Seoul, Korea, April 8, 2014.*, volume 1184 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2014. URL http://dbis.eprints.uni-ulm.de/1206/.

Dat Ba Nguyen, Martin Theobald, and Gerhard Weikum. J-nerd: Joint named entity recognition and disambiguation with rich linguistic features. *Transactions of the Association for Computational Linguistics*, 4:215–229, December 2016. doi: 10.1162/tacl_a_00094. URL https://www.aclweb.org/anthology/Q16-1016.

Truc Vien T. Nguyen and Alessandro Moschitti. End-to-end relation extraction using distant supervision from external semantic repositories. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 277–282, Portland, Oregon, USA, June

2011. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/P11-2048.

Joakim Nivre, Mitchell Abrams, Željko Agić, Lars Ahrenberg, Lene Antonsen, Katya Aplonova, Maria Jesus Aranzabe, Gashaw Arutie, Masayuki Asahara, Luma Ateyah, Mohammed Attia, Aitziber Atutxa, Liesbeth Augustinus, Elena Badmaeva, Miguel Ballesteros, Esha Banerjee, Sebastian Bank, Verginica Barbu Mititelu, Victoria Basmov, John Bauer, Sandra Bellato, Kepa Bengoetxea, Yevgeni Berzak, Irshad Ahmad Bhat, Riyaz Ahmad Bhat, Erica Biagetti, Eckhard Bick, Rogier Blokland, Victoria Bobicev, Carl Börstell, Cristina Bosco, Gosse Bouma, Sam Bowman, Adriane Boyd, Aljoscha Burchardt, Marie Candito, Bernard Caron, Gauthier Caron, Gülşen Cebiroğlu Eryiğit, Flavio Massimiliano Cecchini, Giuseppe G. A. Celano, Slavomír Čéplö, Savas Cetin, Fabricio Chalub, Jinho Choi, Yongseok Cho, Jayeol Chun, Silvie Cinková, Aurélie Collomb, Çağrı Çöltekin, Miriam Connor, Marine Courtin, Elizabeth Davidson, Marie-Catherine de Marneffe, Valeria de Paiva, Arantza Diaz de Ilarraza, Carly Dickerson, Peter Dirix, Kaja Dobrovoljc, Timothy Dozat, Kira Droganova, Puneet Dwivedi, Marhaba Eli, Ali Elkahky, Binyam Ephrem, Tomaž Erjavec, Aline Etienne, Richárd Farkas, Hector Fernandez Alcalde, Jennifer Foster, Cláudia Freitas, Katarína Gajdošová, Daniel Galbraith, Marcos Garcia, Moa Gärdenfors, Sebastian Garza, Kim Gerdes, Filip Ginter, Iakes Goenaga, Koldo Gojenola, Memduh Gökırmak, Yoav Goldberg, Xavier Gómez Guinovart, Berta Gonzáles Saavedra, Matias Grioni, Normunds Grūzītis, Bruno Guillaume, Céline Guillot-Barbance, Nizar Habash, Jan Hajič, Jan Hajič jr., Linh Hà Mỹ, Na-Rae Han, Kim Harris, Dag Haug, Barbora Hladká, Jaroslava Hlaváčová, Florinel Hociung, Petter Hohle, Jena Hwang, Radu Ion, Elena Irimia, Ọlájídé Ishola, Tomáš Jelínek, Anders Johannsen, Fredrik Jørgensen, Hüner Kaşıkara, Sylvain Kahane, Hiroshi Kanayama, Jenna Kanerva, Boris Katz, Tolga Kayadelen, Jessica Kenney, Václava Kettnerová, Jesse Kirchner, Kamil Kopacewicz, Natalia Kotsyba, Simon Krek, Sookyoung Kwak, Veronika Laippala, Lorenzo Lambertino, Lucia Lam, Tatiana Lando, Septina Dian Larasati, Alexei Lavrentiev, John Lee, Phng Lê H`ông, Alessandro Lenci, Saran Lertpradit, Herman Leung, Cheuk Ying Li, Josie Li, Keying Li, KyungTae Lim, Nikola Ljubešić, Olga Loginova, Olga Lyashevskaya, Teresa Lynn, Vivien Macketanz, Aibek Makazhanov, Michael Mandl, Christopher Manning, Ruli Manurung, Cătălina Mărănduc, David Mareček, Katrin Marheinecke, Héctor Martínez Alonso, André Martins, Jan Mašek, Yuji Matsumoto, Ryan McDonald, Gustavo Mendonça, Niko Miekka, Margarita Misirpashayeva, Anna Missilä, Cătălin Mititelu, Yusuke Miyao, Simonetta Montemagni, Amir More, Laura Moreno Romero, Keiko Sophie Mori, Shinsuke Mori, Bjartur Mortensen, Bohdan Moskalevskyi, Kadri Muischnek, Yugo Murawaki, Kaili Müürisep, Pinkey Nainwani, Juan Ignacio Navarro Horñiacek, Anna Nedoluzhko, Gunta Nešpore-Bērzkalne, Lng Nguy˜ên Thị, Huy`ên Nguy˜ên Thị Minh, Vitaly Nikolaev, Rattima Nitisaroj, Hanna Nurmi, Stina Ojala, Adédayọ Olúòkun, Mai Omura, Petya Osenova, Robert Östling, Lilja Øvrelid, Niko Partanen, Elena Pascual, Marco Passarotti, Agnieszka Patejuk, Guilherme Paulino-Passos, Siyao Peng, Cenel-Augusto Perez, Guy Perrier, Slav Petrov, Jussi Piitulainen, Emily Pitler, Barbara Plank, Thierry Poibeau, Martin Popel, Lauma Pretkalniņa, Sophie Prévost, Prokopis Prokopidis, Adam

Przepiórkowski, Tiina Puolakainen, Sampo Pyysalo, Andriela Rääbis, Alexandre Rademaker, Loganathan Ramasamy, Taraka Rama, Carlos Ramisch, Vinit Ravishankar, Livy Real, Siva Reddy, Georg Rehm, Michael Rießler, Larissa Rinaldi, Laura Rituma, Luisa Rocha, Mykhailo Romanenko, Rudolf Rosa, Davide Rovati, Valentin Roșca, Olga Rudina, Jack Rueter, Shoval Sadde, Benoît Sagot, Shadi Saleh, Tanja Samardžić, Stephanie Samson, Manuela Sanguinetti, Baiba Saulīte, Yanin Sawanakunanon, Nathan Schneider, Sebastian Schuster, Djamé Seddah, Wolfgang Seeker, Mojgan Seraji, Mo Shen, Atsuko Shimada, Muh Shohibussirri, Dmitry Sichinava, Natalia Silveira, Maria Simi, Radu Simionescu, Katalin Simkó, Mária Šimková, Kiril Simov, Aaron Smith, Isabela Soares-Bastos, Carolyn Spadine, Antonio Stella, Milan Straka, Jana Strnadová, Alane Suhr, Umut Sulubacak, Zsolt Szántó, Dima Taji, Yuta Takahashi, Takaaki Tanaka, Isabelle Tellier, Trond Trosterud, Anna Trukhina, Reut Tsarfaty, Francis Tyers, Sumire Uematsu, Zdeňka Urešová, Larraitz Uria, Hans Uszkoreit, Sowmya Vajjala, Daniel van Niekerk, Gertjan van Noord, Viktor Varga, Eric Villemonte de la Clergerie, Veronika Vincze, Lars Wallin, Jing Xian Wang, Jonathan North Washington, Seyi Williams, Mats Wirén, Tsegay Woldemariam, Tak-sum Wong, Chunxiao Yan, Marat M. Yavrumyan, Zhuoran Yu, Zdeněk Žabokrtský, Amir Zeldes, Daniel Zeman, Manying Zhang, and Hanzhi Zhu. Universal Dependencies 2.3, 2018. URL http://hdl.handle.net/11234/1-2895. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Václav Novák and Zdeněk Žabokrtský. Feature Engineering in Maximum Spanning Tree Dependency Parser. In Václav Matoušek and Pavel Mautner, editors, *Lecture Notes in Artificial Intelligence, Proceedings of the 10th International Conference on Text, Speech and Dialogue*, volume 4629 of *Lecture Notes in Computer Science*, pages 92–98, Berlin / Heidelberg, 2007. Springer. ISBN 978-3-540-74627-0.

Petr Pajas and Jan Štěpánek. XML-based representation of multi-layered annotation in the PDT 2.0. In Richard Erhard Hinrichs, Nancy Ide, Martha Palmer, and James Pustejovsky, editors, *Proceedings of the LREC Workshop on Merging and Layering Linguistic Information (LREC 2006)*, pages 40–47, Genova, Italy, 2006. ELRA, ELRA. ISBN 2-9517408-2-4.

Petr Pajas and Jan Štěpánek. System for Querying Syntactically Annotated Corpora. In Gary Lee and Sabine Schulte im Walde, editors, *Proceedings of the ACL-IJCNLP 2009 Software Demonstrations*, pages 33–36, Suntec, Singapore, 2009. Association for Computational Linguistics. ISBN 1-932432-61-2.

Karel Pala and Eva Mráková. Legal Terms and Word Sketches: a Case Study. In Horák A. Sojka P., editor, *Proceedings of Fourth Workshop on Recent Advances in Slavonic Natural Languages Processing, RASLAN 2010.*, pages 31–39, Brno, 2010. Tribun s.r.o. ISBN 978-80-7399-246-0.

Karel Pala, Pavel Rychlý, and Pavel Šmerk. Automatic Identification of Legal Terms in Czech Law Texts. In *Semantic Processing of Legal Texts*, pages 83–94, Berlin, 2010. Springer. ISBN 978-3-642-12836-3.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. The Proposition Bank: An Annotated Corpus of Semantic Roles. *Comput. Linguist.*, 31(1):71–106, March 2005. ISSN 0891-2017. doi: 10.1162/0891201053630264. URL http://dx.doi.org/10.1162/0891201053630264.

Monica Palmirani, Raffaella Brighi, and Matteo Massini. Automated extraction of normative references in legal texts. In *Proceedings of the 9th international conference on Artificial intelligence and law*, pages 105–106. ACM, 2003.

Yannis Panagis and Urska Sadl. The force of EU case law: A multi-dimensional study of case citations. In *Legal Knowledge and Information Systems - JURIX 2015: The Twenty-Eighth Annual Conference, Braga, Portugal, December 10-11, 2015*, volume 279 of *Frontiers in Artificial Intelligence and Applications*, pages 71–80. IOS Press, 2015. doi: 10.3233/978-1-61499-609-5-71. URL https://doi.org/10.3233/978-1-61499-609-5-71.

Yannis Panagis, Urska Sadl, and Fabien Tarissan. Giving every case its (legal) due - the contribution of citation networks and text similarity techniques to legal studies of european union law. In Adam Z. Wyner and Giovanni Casini, editors, *Legal Knowledge and Information Systems - JURIX 2017: The Thirtieth Annual Conference, Luxembourg, 13-15 December 2017.*, volume 302 of *Frontiers in Artificial Intelligence and Applications*, pages 59–68. IOS Press, 2017. doi: 10.3233/978-1-61499-838-9-59. URL https://doi.org/10.3233/978-1-61499-838-9-59.

Harshvardhan J. Pandit, Kaniz Fatema, Declan O'Sullivan, and Dave Lewis. Gdprtext - gdpr as a linked data resource. In Aldo Gangemi, Roberto Navigli, Maria-Esther Vidal, Pascal Hitzler, Raphaël Troncy, Laura Hollink, Anna Tordai, and Mehwish Alam, editors, *The Semantic Web*, pages 481–495, Cham, 2018. Springer International Publishing. ISBN 978-3-319-93417-4.

Wenzhe Pei, Tao Ge, and Baobao Chang. An Effective Neural Network Model for Graph-based Dependency Parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 313–322, Beijing, China, July 2015. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P15-1031.

Lucie Poláková, Jiří Mírovský, Anna Nedoluzhko, Pavlína Jínová, Šárka Zikánová, and Eva Hajičová. Introducing the Prague Discourse Treebank 1.0. In *Proceedings of the 6th International Joint Conference on Natural Language Processing*, pages 91–99, Nagoya, Japan, 2013. Asian Federation of Natural Language Processing, Asian Federation of Natural Language Processing. ISBN 978-4-9907348-0-0.

Lucie Poláková, Pavlína Jínová, and Jiří Mírovský. Genres in the Prague Discourse Treebank. In Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Hrafn Loftsson, Bente Maegaard, and Joseph Mariani, editors, *Proceedings of the 9th International Conference on Language Resources and Evaluation (LREC 2014)*, pages 1320–1326, Reykjavík, Iceland, 2014. European Language Resources Association. ISBN 978-2-9517408-8-4.

Martin Popel and Zdeněk Žabokrtský. TectoMT: Modular NLP framework. In Hrafn Loftsson, Eirikur Rögnvaldsson, and Sigrun Helgadottir, editors, *Lecture Notes in Artificial Intelligence, Proceedings of the 7th International Conference on Advances in Natural Language Processing (IceTAL 2010)*, volume 6233 of *Lecture Notes in Computer Science*, pages 293–304, Berlin / Heidelberg, 2010. Iceland Centre for Language Technology (ICLT), Springer. ISBN 978-3-642-14769-2.

Paulo Quaresma and Teresa Gonçalves. Using linguistic information and machine learning techniques to identify entities from juridical documents. In *Semantic Processing of Legal Texts*, pages 44–59. Springer, 2010.

I.S.G.F.R.B. Records and S.C.I.S. Cataloguing. *Functional Requirements for Bibliographic Records: Final Report*. UBCIM Publications. New Series. De Gruyter, 2013. ISBN 9783110962451. URL https://books.google.cz/books?id=YW8hAAAAQBAJ.

David Reinsel, John Gantz, and John Rydning. White paper: The Digitization of the World From Edge to Core. Technical Report US44413318, International Data Corporation, Framingham, Massachusetts, USA, November 2018. URL https://www.seagate.com/files/www-content/our-story/trends/files/idc-seagate-dataage-whitepaper.pdf.

Marco Rospocher, Marieke van Erp, Piek Vossen, Antske Fokkens, Itziar Aldabe, German Rigau, Aitor Soroa, Thomas Ploeger, and Tessel Bogaard. Building event-centric knowledge graphs from news. *J. Web Semant.*, 37-38:132–151, 2016.

Rossella Rubino, Antonino Rotolo, and Giovanni Sartor. An OWL Ontology of Fundamental Legal Concepts. In *Proceedings of the 2006 Conference on Legal Knowledge and Information Systems: JURIX 2006: The Nineteenth Annual Conference*, pages 101–110, Amsterdam, The Netherlands, The Netherlands, 2006. IOS Press. ISBN 1-58603-698-X. URL http://dl.acm.org/citation.cfm?id=1563577.1563589.

Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*, pages 593–607. Springer, 2018.

Karin Kipper Schuler. *Verbnet: A Broad-coverage, Comprehensive Verb Lexicon*. PhD thesis, Philadelphia, PA, USA, 2005. AAI3179808.

Petr Sgall. *Generativní popis jazyka a česká deklinace*. Prague:Academia, 1967.

Amit Singhal. Introducing the Knowledge Graph: Things, not Strings. *Google Official Blog*, 2012. URL http://googleblog.blogspot.com/2012/05/introducing-knowledge-graph-things-not.html.

Pontus Stenetorp, Sampo Pyysalo, Goran Topić, Tomoko Ohta, Sophia Ananiadou, and Jun'ichi Tsujii. BRAT: A Web-based Tool for NLP-assisted Text Annotation. In *Proceedings of the Demonstrations at the 13th Conference of the*

European Chapter of the Association for Computational Linguistics, EACL '12, pages 102–107, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics. URL http://dl.acm.org/citation.cfm?id=2380921.2380942.

Jana Straková, Milan Straka, and Jan Hajič. Open-source tools for morphology, lemmatization, POS tagging and named entity recognition. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 13–18, Stroudsburg, PA, USA, 2014. Johns Hopkins University, Baltimore, MD, USA, Association for Computational Linguistics. ISBN 978-1-941643-00-6.

Jana Straková, Milan Straka, and Jan Hajič. Open-Source Tools for Morphology, Lemmatization, POS Tagging and Named Entity Recognition. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 13–18, Baltimore, Maryland, June 2014. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P/P14/P14-5003.pdf.

Beth Sundheim. Overview of results of the MUC-6 evaluation. volume 1996, pages 13–31, 01 1995. doi: 10.3115/1119018.1119073.

Anton Södergren. HERD - Hajen Entity Recognition and Disambiguation, 2016. ISSN 1650-2884. Student Paper.

Peter Tiersma. *Legal Language*. University of Chicago Press, 1999. ISBN 9780226803029. URL https://books.google.cz/books?id=Sq8XXTo3A48C.

Erik F. Tjong Kim Sang and Sabine Buchholz. Introduction to the CoNLL-2000 Shared Task: Chunking. In *Proceedings of the 2nd Workshop on Learning Language in Logic and the 4th Conference on Computational Natural Language Learning - Volume 7*, CoNLL'00, pages 127–132, Stroudsburg, PA, USA, 2000. Association for Computational Linguistics. doi: 10.3115/1117601.1117631. URL http://dx.doi.org/10.3115/1117601.1117631.

David Weiss, Chris Alberti, Michael Collins, and Slav Petrov. Structured Training for Neural Network Transition-Based Parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 323–333, Beijing, China, July 2015. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P15-1032.

Jason Weston, Antoine Bordes, Oksana Yakhnenko, and Nicolas Usunier. Connecting language and knowledge bases with embedding models for relation extraction. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1366–1371, Seattle, Washington, USA, October 2013. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/D13-1136.

Chenxi Zhu, Xipeng Qiu, Xinchi Chen, and Xuanjing Huang. A Re-ranking Model for Dependency Parser with Recursive Convolutional Neural Network. In *Proceedings of the 53rd Annual Meeting of the Association for Computational*

Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 1159–1168, Beijing, China, July 2015. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P15-1112.

Stefan Zwicklbauer, Christin Seifert, and Michael Granitzer. Doser - a knowledge-base-agnostic framework for entity disambiguation using semantic embeddings. In Harald Sack, Eva Blomqvist, Mathieu d'Aquin, Chiara Ghidini, Simone Paolo Ponzetto, and Christoph Lange, editors, *The Semantic Web. Latest Advances and New Domains*, pages 182–198, Cham, 2016. Springer International Publishing. ISBN 978-3-319-34129-3.

List of Figures

List of Tables

List of publications

Vincent Kríž and Barbora Hladká. Czech Court Decisions Dataset, 2014. URL http://hdl.handle.net/11234/1-2853. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University

Vincent Kríž, Barbora Hladká, Martin Nečaský, and Jan Dědek. Statistical Recognition of References in Czech Court Decisions. In *13th Mexican International Conference on Artificial Intelligence, MICAI 2014, Tuxtla Gutiérrez, Mexico, November 16-22, 2014. Proceedings, Part I*, volume 8856 of *Lecture Notes in Computer Science*, pages 51–61, Switzerland, 2014a. Instituto Tecnológico de Tuxtla Gutiérrez, Springer International Publishing. ISBN 978-3-319-13646-2

Vincent Kríž, Barbora Hladká, Martin Nečaský, and Tomáš Knap. Data Extraction Using NLP Techniques and Its Transformation to Linked Data. In *13th Mexican International Conference on Artificial Intelligence, MICAI 2014, Tuxtla Gutiérrez, Mexico, November 16-22, 2014. Proceedings, Part I*, volume 8856 of *Lecture Notes in Computer Science*, pages 113–124, Switzerland, 2014b. Instituto Tecnológico de Tuxtla Gutiérrez, Springer International Publishing. ISBN 978-3-319-13646-2

Irena Holubová, Vincent Kríž, Martin Nečaský, and Barbora Hladká. INTLIB - an INTelligent LIBrary. In Karel Richta and Václav Snášel, editors, *Proceedings of the Dateso 2014 Annual International Workshop on DAtabases, TExts, Specifications and Objects*, pages 13–24, Praha, Czechia, 2014. Czech Technical University in Prague, Faculty of Information Technology, Czech Technical University in Prague, Faculty of Information Technology. ISBN 978-80-01-05482-6

Vincent Kríž and Barbora Hladká. RExtractor: a Robust Information Extractor. In Matt Gerber, Catherine Havasi, and Finley Lacatusu, editors, *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 21–25, Stroudsburg, PA, USA, 2015. Association for Computational Linguistics. ISBN 978-1-941643-49-5

Vincent Kríž, Barbora Hladká, and Zdeňka Urešová. Czech Legal Text Treebank, 2015. URL http://hdl.handle.net/11234/1-1516. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University

Vincent Kríž, Barbora Hladká, and Zdeňka Urešová. Czech Legal Text Treebank 1.0. In Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Marko Grobelnik, Bente Maegaard, Joseph Mariani, Asunción Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC 2016)*, pages 2387–2392, Paris, France, 2016. European Language Resources Association. ISBN 978-2-9517408-9-1

Vincent Kríž and Barbora Hladká. Improving Dependency Parsing Using Sentence Clause Charts. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics – Student Research Workshop*, pages 86–92, Stroudsburg, PA, USA, 2016. Association for Computational Linguistics, Association for Computational Linguistics. ISBN 978-1-945626-02-9

Vincent Kríž and Barbora Hladká. Czech Legal Text Treebank 2.0, 2017. URL http://hdl.handle.net/11234/1-2498. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University

Vincent Kríž and Barbora Hladká. Czech Legal Text Treebank 2.0. In Nicoletta Calzolari (Conference chair), Khalid Choukri, Christopher Cieri, Thierry Declerck, Sara Goggi, Koiti Hasida, Hitoshi Isahara, Bente Maegaard, Joseph Mariani, Hélène Mazo, Asuncion Moreno, Jan Odijk, Stelios Piperidis, and Takenobu Tokunaga, editors, *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Paris, France, may 2018. European Language Resources Association (ELRA). ISBN 979-10-95546-00-9

A. Linked Data Illustration

We present steps how to transform the input data to ⋆⋆⋆⋆⋆ data, see Section 2.2.

As an illustration, let Table A.1 and Table A.2 be sample ⋆⋆⋆ data, i.e. they are (i) available on the Web as open data; (ii) published in a machine-readable and (iii) non-proprietary format.

Name	ID	City
Okresní soud v Mostě	196315	Most
Okresní soud v Děčíně	10709	Děčín

Table A.1: Sample ⋆⋆⋆ data with a list of Czech courts.

Decision	ID	Publisher
9 As 338/2017 - 32	184585	Okresní soud v Mostě
3 Tz 27/2018	285869	Okresní soud v Děčíně

Table A.2: Sample ⋆⋆⋆ data with a list of published court decisions.

According to the first Linked Data principle, for each record in our data we should define a unique URI. As all records already contain a unique identifier (the ID column), we can immediately create URIs. In addition, the cities in Table A.1 are also ideal candidates for their own URIs:

```
http://examp.le/court/196315
http://examp.le/decision/184585
http://examp.le/city/Most
```

One can note that we already used HTTP URIs, which is in line with the second Linked Data principle. However, we should ensure, that our web server knowns defined URIs and returns valid responses when a URI is looked up. Third Linked Data principle recommends, how such relevant response should looks like.

For example, once the URI `http://examp.le/court/196315` is requested, the response could look like this:

```
<http://examp.le/court/196315> id "196315" .
<http://examp.le/court/196315> name "Okresní soud v Mostě" .
<http://examp.le/court/196315> city <http://examp.le/city/Most> .
```

It is possible, that there are other data sources where our *things* are also described. For example, the city of Most is also available on DBPedia.[1] According to the

[1] https://dbpedia.org

fourth Linked Data principle, we should link our things with other descriptions available on the Web and, in fact, link our data to the rest of the Web.

To do so, our web server may return the following response when requesting `http://examp.le/city/Most`:

```
<http://examp.le/city/Most>
   same_as <https://cs.dbpedia.org/page/Most_(město)> .
```

Data as presented above could be already processed by machines automatically, however, we did not defined the meaning of our predicates. Therefore, the last thing towards the ⋆⋆⋆⋆⋆ data, is to use known and well defined predicates from public *vocabularies*. For example, our `same_as` predicate will become

<http://www.w3.org/2002/07/owl#sameAs>.

B. Czech Legal Text Treebank

We present various technical aspects and details about the Czech Legal Text Treebank presented in Chapter 3.

B.1 Re-Tokenization Procedure

The automatic re-tokenization procedure used in the CLTT annotation strategy is implemented as Treex Block `CLTT::Retokenize`.

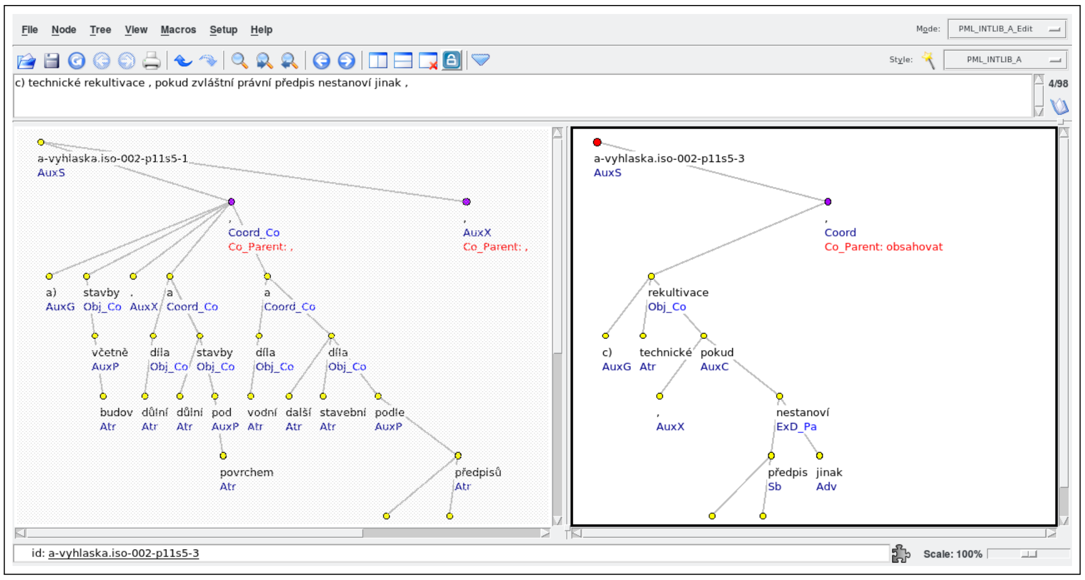
The re-tokenization algorithm works in iterations. In each iteration, at least two tokens are joined. If not, the re-tokenization is finished. Token joining is based on a set of rules, designed for the Czech legal documents. Once two tokens are joined, a special token's attribute `joined` is set to *True*. This attribute can be subsequently used in next iteration. The implemented rules are listed in Table B.1.

	Rule		**Example**
1	t_n	form = [a-z]	*a)*
	t_{n+1}	form =)	
2	t_n	form = Sb	*Sb.*
	t_{n+1}	form = .	
3	t_n	form = {čl, č, odst, písm}	*odst. 3a*
	t_{n+1}	form = .	
	t_{n+2}	numbering expression	
4	t_n	lemma = {odstavec, písmeno, §, bod}	*písmene 5*
	t_{n+1}	numbering expression	
5	t_n	joined expression	*§2 a 5*
	t_{n+1}	form = {a, nebo, až, ,}	
	t_{n+2}	numbering expression	
6	t_n	joined expression	*§2, písmeno a)*
	t_{n+1}	form = {a, nebo, až, ,}	
	t_{n+2}	joined expression	
7	t_n	joined expression	*písmeno a) odst. 5*
	t_{n+1}	joined expression	

Table B.1: List of rules used in Treex's `W2A::CS::Retokenize`.

B.2 CLTT TrEd Extension

The CLTT TrEd extension is called *Czech Legal Text Treebank 2.0* and it is available in the TrEd's extension repository:


c) technické rekultivace , pokud zvláštní právní předpis nestanoví jinak ,

id: a-vyhlaska.iso-002-p11s5-3

Figure B.1: A screenshot of the TrEd editor with two sub-windows, where two interlinked trees are displayed.

http://ufal.mff.cuni.cz/tred/extensions/

The CLTT extension comes with the following macros and shortcuts:

Automatic CLTT Files Identification

When a PML file is going to be opened in TrEd, the extension checks its PML Schema. For the CLTT PML Schema, the TrEd annotation mode is automatically changed to the CLTT Annotation Mode.

Inter-segment Linking

The extension offers several features to make the manual inter-segment linking easier and transparent. When an annotator identifies the head of the coordination, (s)he clicks on the particular node and presses the `SPACE` key to remember the node in the clipboard. Once (s)he identifies the node from which the inter-segment link should start, (s)he clicks on the particular node and press `CTRL + SPACE` to create the link. The created link can be deleted using `SHIFT + SPACE`.

To provide even more transparent environment for annotators, we implemented a special view on linked trees. Once the inter-segment link exists, an annotator can use `CTRL+M` to split an annotation window to two sub-windows – the first window displays the tree when the link starts and the second the tree when the link ends. See Figure B.1.

To make linked nodes easily visible, the node is displayed using violet color. In

Shortcut	Afun	Shortcut	Afun
q	Pred	p	AuxP
n	Pnom	c	AuxC
v	AuxV	o	AuxO
s	Sb	h	Atv
b	Obj	j	AtvV
a	Atr	z	AuxZ
d	Adv	y	AuxY
i	Coord	g	AuxG
t	AuxT	k	AuxK
r	AuxR	x	AuxX
u	Apos	e	ExD

Table B.2: A list of shortcuts for changing an analytical function of the selected node.

addition, its textual description contains the linked node's lemma.

Dependence Correction

The extension offers two procedures that make dependence correction more easy, especially in large trees. The standard way how to change dependencies in TrEd is based on the drag-and-drop principle. An annotator has to drag a node and move it to a new place in the tree. We implemented the shortcut `CTRL+INSERT` which cuts the selected node with its whole subtree and save it to the clipboard. Once the new parent node is selected, an annotator puts the saved subtree to a new place using `SHIFT+INSERT`.

Analytical Function Correction

We defined several shortcuts for simple assigning a new analytical function (`afun`) to a selected node. Table B.2 presents a list of available shortcuts.

Coordination Members

The extension defines the shortcut `1` to change the `is_member` attribute from *True* to *False* and vice versa.

Annotators Comment

Annotators can insert a comment to each node using the `ALT+C` shortcut.

B.3 Accounting Entities Annotation Tool

For the manual annotation, we used the Brat annotation tool [Stenetorp et al., 2012]. In the accounting entities annotation, we exploited brat's *text-bound annotations*. Such annotation is identified by a specific span of text and assigns its type.

In Brat, annotations are stored in files with the `.ann` suffix. The example of annotations of three named entities (`E1`, `E2`, `E3`) are shown in Figure B.2. All text-bound annotations have the same structure. The ID occurs first and is delimited from the rest of the line with the `TAB` character. The primary annotation is given as a `SPACE`-separated triple of type, start-offset, end-offset. The start-offset is the index of the first character of the annotated span in the text, i.e. the number of characters in the document preceding it. The end-offset is the index of the first character after the annotated span. Thus, the character at the end-offset position is not included in the annotated span. For reference, the text spanned by the annotation is included, separated by the `TAB` character.

`E1`	`Organization`	0	4	*Sony*
`E2`	`Organization`	33	41	*Ericsson*
`E3`	`Country`	75	81	*Sweden*

Figure B.2: Annotation of three named entities (E1, E2, E3) in the Brat annotation tool.

B.4 Accounting Dictionary Entry Structure

An entry in the Accounting Dictionary presents a unique entity that appears in the annotated documents. For each entry, the following attributes are available:

- **Entry Identifier** (`entity_id`)
  Unique identification of the entry in the dictionary.
- **Category** (`entity_type`)
  One of the 25 general accounting categories, see Table 3.4.
- **Normalized form** (`entity_form`)
  A base form of an accounting term. For example *účetní jednotka v úpadku*.
- **Lemmatized form** (`lemmatized_form`)
  An accounting term where all words are lemmatized. For example *účetní jednotka v úpadek*.
- **Dependency tree** (`dependency_tree`)
  Dependecny tree of an accounting term parsed automatically.

- **PML-TQ Query** (pmltq)
  A tree query that could be used in the PML-TQ application. It allows users to detect a particular accounting term in given dependency trees. The tree queries were created by an automatic procedure which transforms a entry's dependency tree to a query using a set of rules.

The Accounting Dictionary is distributed together with the CLTT treebank. Two data formats are available - JSON and XML file. Figure B.3 presents a sample dictionary entry in the JSON format and Figure B.4 presents the same entry in the XML format.

```
{
   "entity_id":"0834",
   "entity_form":"předkupní právo",
   "entity_type":"right",
   "lemmatized_form":"předkupní právo",
   "pml_tq":"a-node $n2 := [ m/lemma ~ \"právo\", echild ...",
   "dependency_tree":[
      {
         "parent":2,
         "lemma":"předkupní",
         "tag":"AANS4----1A----",
         "order":1,
         "form":"předkupní"
      },
      {
         "parent":0,
         "lemma":"právo",
         "tag":"NNNS4-----A----",
         "order":2,
         "form":"právo"
      }
   ]
}
```

Figure B.3: Sample dictionary entry in the JSON format.

```
<entity id="0834">
   <type>right</type>
   <original_form>předkupní právo</original_form>
   <lemmatized>předkupní právo</lemmatized>
   <dependency_tree>
       <word form="předkupní" lemma="předkupní" tag="AANS4----1A----"
             ord="1" parent="2"/>
       <word form="právo" lemma="právo" tag="NNNS4-----A----"
             ord="2" parent="0"/>
   </dependency_tree>
```

Figure B.4: Sample dictionary entry in the XML format.

B.5 e-Layer JSON Files

Standalone JSON files allow users to work with the CLTT accounting entities in plain text files,[1] i.e. without working with the PML files. This could be useful especially for users who are focused on named entities and do not need to process XML files.

One entry in the JSON file is described by the following attributes (see Figure B.5):

- **Entity Identifier** (`entity_id`)
  Each annotated entity has identifier unique in the CLTT. The entity identifier starts with the document ID, so each entity could be easily aligned with a particular plain text file.

- **Accounting Dictionary Entry** (`dictionary_id`)
  An accounting term can occur several times in data. All such entities have the same link to the the Accounting Dictionary.

- **List of dependency tree nodes** (`node_ids`)
  The entity (a sequence of tokens) is here mapped to correspond nodes in the dependency tree.

- **Character offsets** (`text_chunk_start_offset`, `text_chunk_end_offset`)
  Character offsets in the plain text documents where the entity appeared. The start offset expresses a number of characters from the beginning of the file to the entity's first character. Analogously, the end offset presents the number of characters from the beginning of the file to the last entity's character.

- **Entity form** (`text_chunk_form`)
  Original textual form of the entity as it appears in the CLTT plain text document.

```
{
    "dictionary_id": "1163",
    "entity_id": "document_01_001-entity0013",
    "text_chunk_start_offset": 265,
    "text_chunk_end_offset": 279,
    "text_chunk_form": "účetní závěrce"
    "node_ids": [
        "document_01_001-sentence4-section1-node26",
        "document_01_001-sentence4-section1-node27"
    ]
}
```

Figure B.5: A sample entity as appear in the e-layer JSON files.

[1] The CLTT plain text files contain one sentence per line, as it appear in the source document.

B.6 e-layer PML Files

To allow users to work with both dependency trees and accounting entities, we included the entity annotation into the PML files. This required a particular modification of the CLTT PML Schema. In addition, the CLTT TrEd extension allows users to view the entities highlighted in the dependency trees.

Once the tree node is a part of an entity, it receives the following additional attributes:

- **Unique Entity Identifier** (`cltt_entity_id`)
  The entity identifier described in the e-layer JSON files.

- **Accounting Dictionary Entry** (`cltt_entity_dictionary_id`)
  A reference to the Accounting Dictionary. It allows users to identify all mentions of a particular dictionary entry in the data.

- **Accounting Dictionary Category** (`cltt_entity_type`)
  A category of the Accounting Dictionary entry. Is allows users to identify all entities that belong to the particular category.

Figure B.6 presents the sample entity from Figure B.5 as it represented in the PML `m-file`.

```
<m id="m-document_01_001-sentence4-section1-node26">
    <cltt_entity_id>document_01_001-entity0013</cltt_entity_id>
    <cltt_entity_type>ucetni_vykaz</cltt_entity_type>
    <cltt_entity_dictionary_id>1163</cltt_entity_dictionary_id>
    <form>účetní</form>
</m>
<m id="m-document_01_001-sentence4-section1-node27">
    <cltt_entity_id>document_01_001-entity0013</cltt_entity_id>
    <cltt_entity_type>ucetni_vykaz</cltt_entity_type>
    <cltt_entity_dictionary_id>1163</cltt_entity_dictionary_id>
    <form>závěrce</form>
 </m>
```

Figure B.6: A sample entity in the PML `m-file`.

B.7 Semantic Relations Annotation Tool

For the manual annotation, we used the Brat annotation tool, namely its binary relations annotation. As semantic relations annotated in CLTT are triples, two binary relations are needed to represent one triple. Annotators have to follow this scenario:

1. An annotator annotates the predicate as a special entity. Three different

E1	Accounting term	0	19	*Dobou použitelnosti*
E2	Definition	23	28	*rozumí*
E3	Duration	29	33	*doba*
R1	Subject			Arg1:E2 Arg2:E1
R2	Object			Arg1:E2 Arg2:E3

Figure B.7: Annotation of the semantic relation from Table 3.6 and Figure 3.11.

entity types are possible (*definition*, *right* and *obligation*).

2. Subsequently, the annotator links the predicate entity with the existing subject entity, i.e. (s)he creates a binary relation *predicate* $\rightarrow$ *subject*.

3. Finally, (s)he links the predicate entity with the existing object entity, i.e. (s)he creates a binary relation *predicate* $\rightarrow$ *object*.

In Brat, the relation annotations are stored in the `.ann` file together with the entities. Figure B.7 presents one semantic relation: two accounting entities `E1` and `E3`, one *predicate* entity `E2` and two binary relations that links *predicate* with subject and object.

Using of binary relations for annotating triples is effective in sentences with coordination. It is possible that one *subject* and *predicate* are linked with several objects if there is a coordination of the object entities. For example, two different rights could be declared for one entity in the particular sentence.

B.8 r-layer JSON Files

Standalone JSON files allows users to work with the CLTT semantic relations in plain text files, i.e. without working with the PML files. This could be useful especially for users who are focused on semantic relations detection and not need to process XML files.

One JSON file entry is described by the following attributes (see Figure B.8):

- **Relation Identifier** (`relation_id`)
  Each semantic relation has an identifier unique in the CLTT. The relation identifier starts with the document ID, so each relation could be easily aligned with a particular plain text file.

- **Relation Type** (`relation_type`)
  One of *Definition*, *Obligation*, *Right*.

- **Subject** (`subject`)
  It describes the subject from the particular semantic relation. It is a struc-

```
{
    "relation_type": "Definition",
    "relation_id": "document_01_001-relation0015",
    "document_id": "document_01_001",
    "subject": {
        "dictionary_id": "0203",
        "entity_id": "document_01_001-entity0368",
        "entity_type": "obdobi",
        "node_ids": [
            "document_01_001-sentence56-node1",
            "document_01_001-sentence56-node2"
        ],
        "text_chunk_start_offset": 9027,
        "text_chunk_end_offset": 9046,
        "text_chunk_form": "Dobou použitelnosti",
    },
    "predicate": {
        "text_chunk_end_offset": 9056,
        "text_chunk_start_offset": 9050,
        "text_chunk_form": "rozumí",
        "node_ids": ["document_01_001-sentence56-node4"]
    },
    "object": {
        "dictionary_id": "0195",
        "entity_id": "document_01_001-entity0369",
        "entity_type": "obdobi",
        "node_ids": ["document_01_001-sentence56-node5"],
        "text_chunk_start_offset": 9057,
        "text_chunk_end_offset": 9061,
        "text_chunk_form": "doba"
    }
}
```

Figure B.8: Sample semantic relation as it appears in the r-layer JSON files.

ture with the same attributes as entities stored in e-layer JSON files.

- **Predicate** (`predicate`)
  It describes the predicate from the particular semantic relation.

- **Object** (`object`)
  It describes the object from the particular semantic relation. It is a structure with the same attributes as entities stored in e-layer JSON files.

B.9 r-layer PML Files

To allow users to work with both dependency trees and semantic relations, we included the semantic relations annotation into the PML files. This required a particular modification of the CLTT PML Schema. As a result, the CLTT TrEd extension allows users to view the relations in the dependency trees.

Once the tree node is a part of an relation, it receives the following additional attributes:

- **Unique Relation Identifier** (`cltt_relation_id`)

The relation identifier described in the r-layer JSON files.

- **Relation Type** (`cltt_relation_type`)
  One of *Definition*, *Obligation* or *Right*.

- **Relation Subpart** (`cltt_relation_subpart`)
  For the particular tree node, it specifies its position in the triple. For example, Figure B.9 presents the node that is a part of the relation's predicate.

- **Relation references**
  It contains references to the rest of the triple positions. For example, the node from Figure B.9 is a part of the relation's predicate. Therefore, the predicate reference is empty and the subject and object references links the node with subject and object nodes.

```
<m id="m-document_01_001-sentence56-node4">
   <cltt_relations>
      <cltt_relation>
         <cltt_relation_id>document_01_001-relation0015</cltt_relation_id>
         <cltt_relation_type>Definition</cltt_relation_type>
         <cltt_relation_subpart>predicate</cltt_relation_subpart>
         <cltt_relation_subject_reference>
            document_01_001-sentence56-node1
         </cltt_relation_subject_reference>
         <cltt_relation_predicate_reference></cltt_relation_predicate_reference>
         <cltt_relation_object_reference>
            document_01_001-sentence56-node5
         </cltt_relation_object_reference>
      </cltt_relation>
   </cltt_relations>
   <w.rf>w#w-document_01_001-sentence56-node4</w.rf>
   <form>rozumí</form>
   <lemma>rozumět</lemma>
   <tag>VB-S---3P-AA---</tag>
</m>
```

Figure B.9: Sample relation in the PML `m-file`.

B.10 R-layer Statistics

Table 3.8 and Figure B.11 present a relation types distribution. Figure B.12 presents how many relations occur in one sentence. Table B.3 and Table B.4 present the most frequent subjects. Analogously, Table B.5 and Table B.6 present the most frequent objects and finally, Table B.7 presents the most frequent predicates.

	Entity	Frequency	
		absolute	relative
1	účetní jednotka	306	0.6270
2	konsolidující účetní jednotka	040	0.0820
3	nástupnická účetní jednotka	012	0.0246
4	přejímající společník	011	0.0225
5	ministerstvo	7	0.0143
6	rezerva	6	0.0123
7	zanikající účetní jednotka	6	0.0123
8	účetní jednotka rozdělovaná odštěpením	5	0.0102
9	vybraná účetní jednotka	3	0.0061
10	tržní hodnota	3	0.0061

Table B.3: Top 10 most frequent subject entities.

	Accounting category	Frequency	
		absolute	relative
1	general subject	369	0.7561
2	general term	031	0.0635
3	legal person	026	0.0533
4	accounting concept	020	0.0410
5	institution	9	0.0184
6	liabilities	6	0.0123
7	activity	5	0.0102
8	document	5	0.0102
9	accounting report	4	0.0082
10	assets	3	0.0061

Table B.4: Top 10 most frequent subject categories.

	Entity	Frequency	
		absolute	relative
1	účetní záznam	24	0.492
2	účet	23	0.471
3	účetní závěrka	20	0.410
4	účetní kniha	14	0.287
5	závazek	14	0.287
6	majetek	13	0.266
7	informace	11	0.225
8	obchodní firma	08	0.164
9	sídlo	08	0.164
10	výroční zpráva	06	0.123

Table B.5: Top 10 most frequent object entities.

	Accounting category	**Frequency**	
		absolute	**relative**
1	accounting concept	150	0.3074
2	general term	102	0.2090
3	accounting report	055	0.1127
4	activity	025	0.0512
5	account	024	0.0492
6	liabilities	023	0.0471
7	assets	022	0.0451
8	document	017	0.0348
9	method	016	0.0328
10	regulation	011	0.0225

Table B.6: Top 10 most frequent object categories.

Definitions			**Obligations**			**Rights**		
Predicate	AF	RF	Predicate	AF	RF	Predicate	AF	RF
rozumí	56	0.6829	uvede	99	0.2853	vést	8	0.1356
považuje	13	0.1585	účtuje	20	0.0576	rozhodnout	7	0.1186
považují	9	0.1098	sestavují	16	0.0461	použít	6	0.1017
je	2	0.0244	účtují	16	0.0461	změnit	5	0.0847
znamená	2	0.0244	vést	15	0.0432	Uplatnit	4	0.0678
			použije	7	0.0202	provést	3	0.0508
			dodržovat	7	0.0202	účtovat	2	0.0339
			účtovat	6	0.0173	provádět	2	0.0339
			stanoví	6	0.0173	vydat	2	0.0339
			otevírají	5	0.0144	sestavit	2	0.0339

Table B.7: Top 10 most frequent predicates.

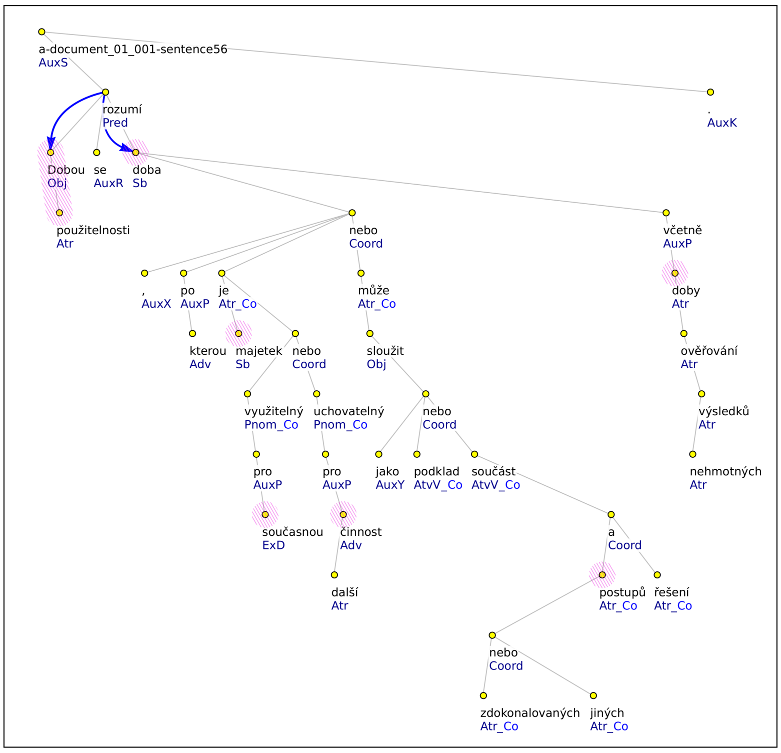


Figure B.10: CLTT dependency tree with the highlighted entities and semantic relations.

B.11 Syntactic Phenomena in CLTT and PDT

In this section we provide the relative frequencies of the language phenomena discussed in Section 3.6, namely: (1) reflexive passive, (2) periphrastic passive, (3) chains of four genitive expressions, (4) chains of three genitive expressions, (5) chains of two genitive expressions, (6) construction with deverbative noun ending on −*ní*, -*tí* with genitive, (7) apposition, (8) ellipsis, (9) parenthesis, (10) numbers. Table B.8 presents the comparison of the CLTT and PDT datasets. In Table B.9 and Table B.10 relative frequencies from PDT are split into genres.

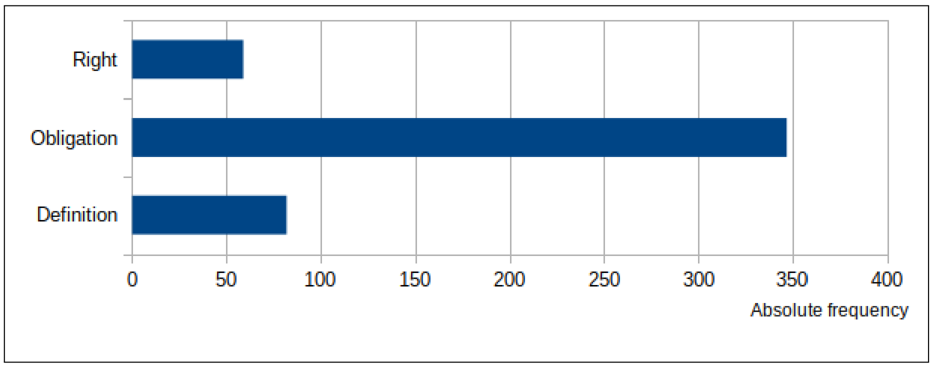


Figure B.11: Histogram of three types of the semantic relations annotated in CLTT.

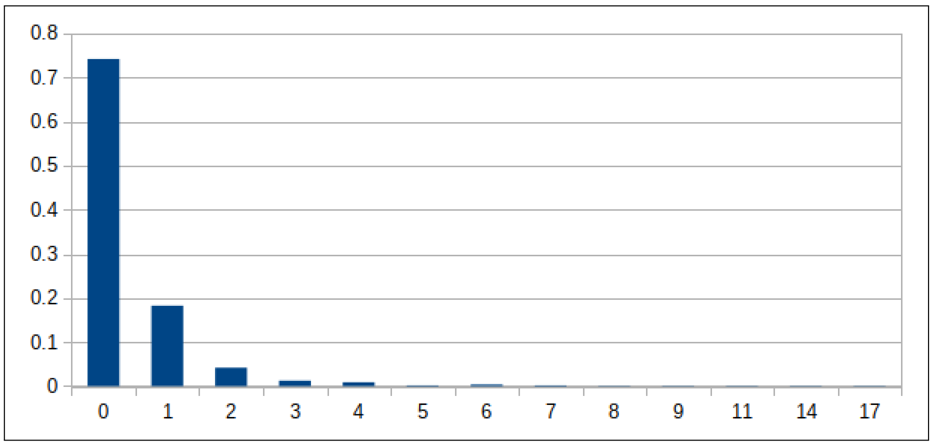


Figure B.12: Relative frequency of sentences with a given number of relations.

Corpus	**(1)**	**(2)**	**(3)**	**(4)**	**(5)**
CLTT	0.2766	0.1108	0.0071	0.0629	0.4167
PDT	0.0503	0.1730	0.0025	0.0175	0.0956

Corpus	**(6)**	**(7)**	**(8)**	**(9)**	**(10)**
CLTT	0.0071	0.0629	0.4468	0.2092	0.7819
PDT	0.0001	0.0716	0.2561	0.0809	0.1808

Table B.8: Relative frequencies of the discussed language phenomena.

Genre	**(1)**	**(2)**	**(3)**	**(4)**	**(5)**
CLTT	0.2766	0.1108	0.0071	0.0629	0.4167
advice	0.0662	0.1913	0.0007	0.0060	0.0523
caption	0.0276	0.0473	0.0039	0.0276	0.0986
collection	0.0289	0.1296	0.0046	0.0252	0.1072
comment	0.0618	0.2076	0.0016	0.0175	0.1005
description	0.0606	0.1444	0.0025	0.0128	0.1085
essay	0.0573	0.1747	0.0016	0.0102	0.0760
interview	0.0435	0.2536	0.0020	0.0068	0.0415
invitation	0.0476	0.1256	0.0037	0.0329	0.1232
letter	0.0645	0.2304	0.0000	0.0300	0.0760
news	0.0488	0.1968	0.0040	0.0274	0.1361
other	0.0481	0.0109	0.0000	0.0086	0.0507
overview	0.0332	0.0935	0.0030	0.0151	0.0543
plot	0.0300	0.0600	0.0100	0.0400	0.0800
program	0.0105	0.0335	0.0000	0.0105	0.0335
review	0.0330	0.0973	0.0026	0.0219	0.0991
sport	0.0400	0.1640	0.0016	0.0107	0.0632
survey	0.0548	0.1593	0.0026	0.0183	0.0731
topic	0.0675	0.2354	0.0008	0.0088	0.0506
weather	0.0088	0.0354	0.0000	0.0000	0.0000

Table B.9: Relative frequencies of the discussed language phenomena.

Genre	**(6)**	**(7)**	**(8)**	**(9)**	**(10)**
CLTT	0.0071	0.0629	0.4468	0.2092	0.7819
advice	0.0007	0.0708	0.2336	0.0615	0.1502
caption	0.0000	0.0789	0.5602	0.1164	0.1183
collection	0.0000	0.0705	0.3289	0.0806	0.2758
comment	0.0000	0.0743	0.2267	0.0721	0.1065
description	0.0000	0.0766	0.2260	0.0698	0.2248
essay	0.0001	0.0780	0.2376	0.0733	0.0997
interview	0.0000	0.0544	0.1992	0.0401	0.0435
invitation	0.0000	0.1037	0.2878	0.1280	0.2305
letter	0.0000	0.0714	0.3018	0.1336	0.1613
news	0.0001	0.0534	0.2077	0.0756	0.1874
other	0.0000	0.0782	0.3222	0.0670	0.1632
overview	0.0000	0.1418	0.4268	0.1101	0.5128
plot	0.0000	0.1000	0.4500	0.1900	0.2200
program	0.0000	0.1006	0.8155	0.1153	0.8239
review	0.0000	0.1239	0.3353	0.1304	0.1119
sport	0.0004	0.0681	0.3100	0.1161	0.2495
survey	0.0026	0.1305	0.3838	0.1018	0.3055
topic	0.0000	0.0567	0.1672	0.0429	0.0544
weather	0.0000	0.1327	0.8053	0.0177	0.6903

Table B.10: Relative frequencies of the discussed language phenomena.

C. RExtractor Technical Details

We present various technical aspects of the RExtractor system described in Chapter 4.

C.1 Document States

As the document *flows* throw the RExtractor pipeline, it receives different states that allow to determine the actual state of the processing:

- `200` **Submitted correctly.**
  Document was submitted into the system. It is waiting to be processed by the Conversion Component.

- `300` **Conversion started.**
  Document is being processed by the Conversion Component.

- `310` **Conversion failed.**
  An error occurred during the conversion.

- `320` **Conversion finished.**
  Document was correctly processed by the Conversion Component. It is waiting for processing by the NLP Component.

- `400` **NLP started.**
  Document is being processed by the NLP Component.

- `410` **NLP failed.**
  An error occurred during the NLP processing.

- `420` **NLP finished.**
  Document was correctly processed by the NLP Component. It is waiting for processing by the Entity Detection Component.

- `500` **Entity detection started.**
  Document is being processed by the Entity Detection Component.

- `510` **Entity detection failed.**
  An error occurred during the entity detection.

- `520` **Entity detection finished.**
  Document was correctly processed by the Entity Detection Component. It is waiting for processing by the Relation Extraction Component.

- `600` **Relation extraction started.**

Document is being processed by the Relation Extraction Component.

- `610` **Relation extraction failed.**
  An error occurred during the relation extraction processing.

- `620` **Relation extraction finished.**
  Document was correctly processed by the Relation Extraction Component. It is waiting for processing by the Export Component.

- `700` **Export started.**
  Document is being processed by the Export Component.

- `710` **Export failed.**
  An error occurred during the export.

- `720` **Export finished.**
  Document was exported correctly. Document was correctly processed by the RExtractor pipeline.

If an error occurs in the processing of a particular document (i.e. document's state is `*10`), the processing of a next document is paused until the system administrator fixes the problem.

C.2 Document Collection Structure

To make RExtractor as simple as possible, it does not use any databases. All documents are stored on the host server file system, namely in the `/data` directory. An advanced user could access data directly:

- `/data/logs`
  For each submitted document, RExtractor stores its log file here. The log contains details about the document processing. Users can access a particular log file using the filename pattern *document_id*`.log`.

- `/data/submitted`
  A repository of all submitted documents in their original form.

- `/data/converted`
  A repository of documents in the internal format (see Appendix C.4). Users can access the converted document using the filename pattern *document_id*`.xml`.

- `/data/treex`
  Once the document is analyzed by the NLP component, the particular Treex file is available here with the filename pattern *document_id*`.treex.gz`.

- `/data/exported`
  This directory contains exported files as specified in the used extraction strategy.

C.3 RExtractor API

The RExtractor system offers two different APIs:

1. the HTTP REST API
2. the Command Line Interface (CLI)

Both APIs allow to submit new documents, get document states and download exported data. The REST API is suitable for applications or users who want to incorporate the RExtractor data into more general pipelines. It is used by the RExtractor's web interface[1] as well. The CLI API is aimed for system administrators and advanced users. It allows batch operations and direct access to the Document Collection.

Both APIs return the same semi-structured plain text format. The response starts with the code (`[OK]` or `[ERROR]`) on the first line. The second line contains an error description or requested data when the operation was successful.

Server start

It starts the RExtractor's components and subsequently returns their health-check status. This command is vailable only from CLI.

REST	(not available)
CLI	`rextractor server-start`

Server stop

It terminates the RExtractor's components and subsequently returns their health-check status.

REST	(not available)
CLI	`rextractor server-stop`

[1] https://quest.ms.mff.cuni.cz/rextractor/

Server healthcheck

This command provides the health-check status of all of the running components. The system is considered to be healthy if all components are alive and are able to process documents.

REST	`GET /?command=server-state`
CLI	`rextractor server-state`

Sample output:

```
[OK]
Conversion server is ON.
NLP server is ON.
Entity server is ON.
Relation server is ON.
Export server is ON.
```

Document submission

The command allows to submit a new document to be processed by the RExtractor system. The submitted document identifier has to be unique in the RExtractor document collection.

REST	`POST /?command=document-submit`
CLI	`rextractor document-submit`

REST API POST data:

- `doc_id` – document identifier
- `doc_content` – document content (plain text)
- `doc_strategy` – extraction strategy for the document

CLI command arguments:

- `doc_strategy` – extraction strategy for the document
- `path_to_file` – filepath to the submitted document

Sample request:

```
./?command=document-submit
doc_id=test01
doc_content=Test sentence.
doc_strategy=strategy_01
```

Sample output:

```
[OK]
Submitted correctly.
```

Document state

It provides the current state for the requested document, see Section C.1.

REST	`GET /?command=document-state&doc_id=`*document_id*
CLI	`rextractor document-state` *document_id*

Sample output:

```
[OK]
720 Document exported successfully.
2014-10-16 15:35:57
```

Removing document

It removes the specified document from the Document Collection.

REST	`DELETE /?command=document-delete&doc_id=`*document_id*
CLI	`rextractor document-delete` *document_id*

Sample output:

```
[OK]
Deleted.
```

Document Download

It returns the exported file for the specified document.

REST	`GET /?command=export-description&doc_id=`*document_id*
CLI	(not available)

List of documents

It lists the documents in the Document Collection.

REST	`GET /?command=list-all`
CLI	`rextractor list-all`

Optional command arguments:

- `start` – specifies the offset of the first document to return
- `limit` – specifies the maximum number of documents in the response
- `order_by` – specifies the key attribute for the sorting. Available values:
    - *ctime* – the document submission time
    - *status* – the document processing state
    - *id* – document identifier
- `order_dir` – specifies the direction of the ordering, one of {*asc*, *desc*}

In the response, each document is printed on a separate line which contains three columns:

1. the document identifier
2. the submission timestamp
3. the current document status

Sample output:

```
[OK]
test01       2014-10-27 13:42:43       720 Document exported successfully.
test05       2014-10-22 14:56:43       720 Document exported successfully.
```

C.4 Internal Document Representation

Each submitted document is transformed to the internal document format. It is an XML file with three main parts:

- **Metadata** (`/document/metadata`)
  The Conversion Component can also extract additional metadata from the submitted document. If so, they are stored in this section.

- **Body** (`/document/body`)
  It contains plain texts extracted from the source document. It is a sequence of `text` elements. One element represents a piece of plain text. It could be a sentence, paragraph or whole text – there is neither restriction nor rule on the plain text itself given by the system and it depends on the used Conversion Component. Each text element will be processed individually by the subsequent components in the pipeline.

- **Description** (`/document/description`)
  Detected entities and relations are stored in this section. One can see this section as a space for saving different layers of annotation over the texts from the body section. Each component in the pipeline typically adds a new layer of annotation. Two main annotation layers, namely (i) the layers of detected entities and (ii) the layer of extracted semantic relations are described in the rest of this section.

C.4.1 Entity Annotation

The RExtractor system architecture allows to detect entities which consist of noncontinuous sequence of tokens, i.e. one entity is split into several text chunks. The architecture supports also the situation, where one text chunk is a part of several different entities, i.e. overlaying entities. As a result, the entity annotation consists of two annotation layers.

All chunks are listed in the `/document/description/chunks` element. Each chunk is defined by the `chunk` element. It contains the following attributes:

- `chunk_id`
  A unique identifier.

- `text_id`
  Link to the `text` element from which the chunk is extracted.

- `start`
  Start character offset of the chunk in the plain text from the `text` element.

- `end`
  End character offset of the chunk in the plain text from the `text` element.

- `nodes`
  A list of the Treex node identifiers of all of the tokens that appear in the chunk. Node identifiers are separated by a whitespace.

The entities are listed in the `/document/description/entities` element. One entity is described in the `entity` element and it contains the following attributes:

- `entity_id` – a unique entity identifier
- `dbe_id` – link to the Database of Entities
- `chunk_ids` – list of the chunk identifiers from which the entity consists of
- `nodes` – list of the Treex nodes that appear in the entity

C.4.2 Relation Annotation

To make the architecture of the RExtractor system easier, semantic relations predicates are internally considered as entities as well. The only difference is the absence of the DBE link in the entity definition.

All of the extracted relations are listed under the path

`/document/description/relations`

One particular relation is represented by the `relation` element with the following attributes:

- `relation_id` – a unique relation identifier
- `dbr_id` – link to the Database of Relations
- `(subject|object)_id` – identifier of the entity for the particular position
- `(subject|object)_concept` – identifier of the ontological class for the particular position

C.5 Database of Entities

The Database of Entities is a XML file. The root element `/entities` lists the DBE entries. Each entry is described using the `<entity>` element with the following attributes:

- **Entity identifier** (`entity_id`)
  A unique identifier of the entity in the DBE.

- **Original form** (`original_form`)
  An original entity form.

- **Lemmatized form** (`lemmatized`)
  It contains a lemmatized form of each token presented in the entity original form.

- **Dependency tree** (`dependency_tree`)
  Syntactical analysis of the entity original form. For each token in the original form, the following attributes are available:
  - word form
  - word lemma
  - morphological tag
  - order of the word in the original form
  - order of the word parent in the dependency tree

- **PML Tree Query** (`pml_tq`)
  A tree query designed for entity matching over dependency trees.

C.6 Database of Relations

The Database of Relations is a XML file. The root element `/queries` lists the DBR entries. Each query is described using the `<query>` element with the following parts:

- Relation metadata

- PML tree query
- Annotation instructions
- Semantic relation definition

Relation Metadata

Each DBR entry is identified by a unique identifier. In addition, a `description` attribute is available for a human-readable explanation and description of the relation and its query.

PML Tree Query

A tree query that matches the semantic relation in a dependency tree. The query matches a subject, predicate and an object of the semantic relation. If a dependency tree matches the particular query, PML-TQ returns a list of tree node identifiers. Their order and meaning is declared in the `annotation` section of the DBR entry.

Annotation Description

Aside the tree structure itself, tree queries contains also a specification of a list of returned nodes. The list length may vary and list elements may have different meaning for different queries. A definition of the list elements is stored in the `<annotation>` section. Each definition is defined by the `<annotate>` element by the following attributes:

- **Position** `/annotate@position`
  It specifies the position of the explained node identifier in the returned list.

- **Type** `/annotate@type`
  It allows to use the node in three different ways:
  1. `node`
     The specified node will be included into the output relation as it is.

  2. `entity`
     The specified node is a part of an entity and this entity will be included into the final semantic relation.

  3. `tree`
     The specified node is a root of a dependency subtree and this subtree will be included into the semantic relation.

Relation Description

It defines the semantic relation which will be returned by the system. The semantic relation is a triple [*subject*, *predicate*, *object*]. Each position in the triple is described by the `column` structure with the following attributes:

- **Annotation Identifier** `/column@chunk_id`
  It determines which tech chunk (from the annotation section) will be used for this semantic relation position.

- **Ontological Class** `/column@concept`
  It specifies the ontological class for the particular semantic relation position.