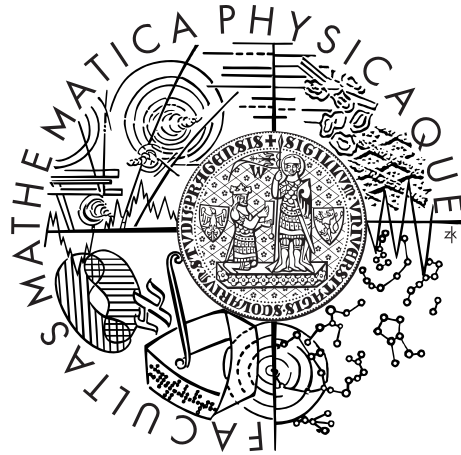


Charles University in Prague
Faculty of Mathematics and Physics

MASTER THESIS



Ivana Lukšová

Ontology Enrichment Based on Unstructured Text Data

Department of Software Engineering

Supervisor of the master thesis: Mgr. Martin Nečaský, Ph.D.

Study programme: Computer science

Specialization: Software systems

Prague 2013

I would like to thank my supervisor, Mgr. Martin Nečaský, Ph.D., for his patience and for his valuable expert advices. Moreover, I would like to thank Mgr. Barbora Vidová Hladká, Ph.D., for her consultations.

I declare that I carried out this master thesis independently, and only with the cited sources, literature and other professional sources.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular the fact that the Charles University in Prague has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 paragraph 1 of the Copyright Act.

In Prague, April 11, 2013

Ivana Lukšová

Název práce: Rozšiřování ontologie z nestrukturovaného textu

Autor: Ivana Lukšová

Katedra: Katedra softwarového inženýrství

Vedoucí diplomové práce: Mgr. Martin Nečaský, Ph.D., Katedra softwarového inženýrství

Abstrakt: Sémantická anotace - přiřazení sémantických informací k textu - je základní úloha v oblasti získávání znalostí. V posledních letech byly navrženy víceré platformy pro sémantickou anotaci, avšak automatické vytváření ontologií z textu je stále náročný problém. V této práci je představena nová metoda pro polo-automatické rozšiřování ontologie z nestrukturovaného textu, která by měla tento proces usnadnit. Nové elementy ontologie, jako koncepty a relace, jsou extrahovány z textu - zapojením metod zpracování přirozeného jazyka a strojového učení. Naše metoda dosahuje F-skóre až 71% pro extrakci konceptů a až 68% pro extrakci relací.

Klíčová slova: ontologie, strojové učení, získávání znalostí

Title: Ontology Enrichment Based on Unstructured Text Data

Author: Ivana Lukšová

Department: Department of Software Engineering

Supervisor: Mgr. Martin Nečaský, Ph.D., Department of Software Engineering

Abstract: Semantic annotation, attaching semantic information to text data, is a fundamental task in the knowledge extraction. Several ontology-based semantic annotation platforms have been proposed in recent years. However, the process of automated ontology engineering is still a challenging problem. In this paper, a new semi-automatic method for ontology enrichment based on unstructured text is presented to facilitate this process. NLP and machine learning methods are employed to extract new ontological elements, such as concepts and relations, from text. Our method achieves F-measure up to 71% for concepts extraction and up to 68% for relations extraction.

Keywords: ontology, machine learning, knowledge extraction

Contents

Introduction	4
1 Ontologies and ontology learning	6
1.1 Ontologies	6
1.1.1 Ontological elements	6
1.2 Representation of ontologies	8
1.3 Semantic Web	8
1.3.1 Role of ontologies in the Semantic Web	8
1.4 Semantic annotation	9
1.4.1 Annotation ontology	9
1.5 Natural language processing	11
1.6 Ontology learning from text	11
1.6.1 Extraction of terminology	11
1.6.2 Identification of synonym terms	11
1.6.3 Formation of concepts	12
1.6.4 Formation of a concept hierarchy	12
1.6.5 Relations learning	12
1.6.6 Other ontology learning tasks	12
1.7 Ontology induction and enrichment	12
1.8 Conclusion	12
2 Task of ontology enrichment	13
2.1 Assumptions and input	13
2.1.1 Text documents	13
2.1.2 Annotation ontology	15
2.2 Machine learning task	16
2.2.1 Formal definition of the classification task	17
2.3 Pre-annotation and the use of existing data sources	21
2.4 Prototype implementation	21
3 Related work	22
3.1 Methodologies	22
3.2 Overview of related work	22
3.2.1 Text-To-Onto	22
3.2.2 OntoLT	22
3.2.3 OTTO	23
3.2.4 C-Pankow	23
3.2.5 Using Decision Trees and Text Mining Techniques for Ex- tending Taxonomies	23
3.2.6 OnTeA	23
3.2.7 Learning syntactic patterns for automatic hypernym dis- covery	24
3.2.8 Semantic Taxonomy Induction from Heterogenous Evidence	24
3.2.9 CRCTOL	24
3.2.10 OntoUSP	24

3.2.11	SOFIE	25
3.3	Conclusion	25
4	Analysis of the problem	27
4.1	Concepts and relations assignment for a particular user	27
4.2	Language independence	27
4.3	Data set processing	28
4.3.1	Features extraction	28
4.4	The classifier learning process	29
4.4.1	Negative instances	29
4.5	Ontology extraction	29
4.5.1	Defining the unknown instances	30
4.5.2	Significance of the extracted facts	30
4.5.3	Concept extraction	30
4.5.4	Ontological relation extraction	31
4.5.5	Ontological instances extraction	31
5	Design of the solution	32
5.1	Concepts and relations assignment	32
5.2	Language independence	33
5.3	Data set processing	33
5.3.1	Linguistic processing	34
5.3.2	Features	34
5.4	The classifier learning process	35
5.4.1	Candidate text instances	36
5.5	Ontology extraction	37
5.5.1	Concept extraction	37
5.5.2	Relation extraction	38
5.5.3	Instances extraction	39
5.5.4	Significance and the result information	39
5.6	Conclusion	40
6	Techniques	41
6.1	Training process	41
6.2	Feature extraction	41
6.2.1	Extraction of candidate phrases	41
6.2.2	Extraction of candidate relations	44
6.2.3	Computation of concept feature values	44
6.2.4	Computation of relation feature values	47
6.3	Machine learning	48
6.3.1	Decision tree	48
6.3.2	Random forests	50
6.3.3	Problems in the machine learning	52
6.4	Ontological elements extraction	52
6.4.1	Ontological concepts and hierarchy	52
6.4.2	Ontological instances	53

7	Implementation	55
7.1	Annotation	55
7.2	Linguistic processing module	56
7.3	Document processing module	57
7.4	Learning module	57
7.4.1	Feature extraction	58
7.4.2	Classifier learning	58
7.5	Enrichment module	61
7.5.1	OWL representation	62
7.6	Conclusion	64
8	Experimental results and evaluation	65
8.1	Testing environment	65
8.1.1	Data sets	65
8.1.2	Learning	65
8.2	Experimental results	69
8.3	Evaluation	72
8.3.1	Analysis of the learned decision trees	72
8.3.2	Summary	74
	Conclusion	75
	Bibliography	76
	List of Tables	80
	List of Figures	80
	List of Abbreviations	82
	Appendix A	83

Introduction

Motivation

The notion of the *Semantic Web* was envisioned by Tim Berners-Lee in 2000. It was presented as an extension of the current World Wide Web, where information has a semantic, machine-readable meaning. This should be achieved by insertion of a certain semantic metadata to the web pages. The semantic web transforms the World Wide Web into a shared information repository for both humans and machines with efficient content retrieval. The *ontologies* are considered as one of the pillars of the semantic web, because they represent semantic information. This information describes particular objects and how they relate to each other in a machine readable way.

High expectations have been made about the Semantic Web. Now, several years after, it seems that the idea of the Semantic Web as a shared information repository has been quite utopian. The core semantic technologies have been already proposed - *RDF*, *OWL*, *SPARQL*. However, there is still a lot of problems about their integration into the Word Wide Web on a large scale. Practical feasibility of the ideas of the semantic web in global measure is doubtful.

On the other hand, the semantic web has been proven more successful on a smaller scale. Semantic applications operating within small domains with domain-specific ontologies can profit from the semantic web approach. But one basic problem is still challenging - the semantic metadata generation.

One method how to automatically create a semantic metadata is a *semantic annotation*. Several ontology-based semantic annotation platforms have been proposed in recent years, mostly concentrating on small domains. They attach semantic information to semi-structured or unstructured texts. However, the underlying ontologies are the bottleneck of these mechanisms. Such ontology should cover a given domain and it usually needs to capture a certain shared vocabulary. Manual construction of this ontology is very time and resources consuming and thus expensive process. It is not surprising that induction an ontology semi-automatically from a set of input data is the next step of semantic annotation platforms evolution.

At present, a few steps have been made to solve this problem. Several methods to create an ontology automatically from text have been presented. However, these methods are often too general and most of them suffers of a lack of configurability and adaptability to a particular domain.

Goals of the thesis

In this thesis, a new idea of how to facilitate the process of an ontology engineering will be presented. We concentrate to an ontology used in the process of semantic annotation. We want propose a general method that will extract new elements of such ontology based on the unstructured text data. The proposed method should will be supervised, highly configurable and adaptable to a particular domain. We want to employ certain natural language processing and machine learning methods to extract new elements of the ontology.

We will provide a prototype implementation to show the properties of our method and to evaluate its performance. We will use an ontology from a job descriptions domain and a set of domain-related text documents for evaluation of our prototype implementation.

Thesis organisation

The structure of this thesis is as follows.

In chapter 1, we present the necessary background of the ontologies, semantic annotation and particularly ontology learning. We explain the used terminology and give the example of an annotation ontology.

In chapter 2, we provide the specification of the task of ontology enrichment. We explain the classification nature of this task and present the metrics for its performance evaluation.

In chapter 3, we present the related works in the area of ontology enrichment and semantic annotation. We provide an overview of currently used methods along with their limitations and experimental performances.

In chapter 4, the analysis of the task of ontology enrichment is explained and basic questions and problems are identified.

In chapter 5, we present the design of our method and we specify the approach to identified problems.

In chapter 6, we describe the techniques and methods used in this thesis, particularly natural language processing and machine learning methods.

In chapter 7, the high-level view of the prototype implementation is provided along with the architecture of the system.

In chapter 8, we present the performance evaluation of the prototype implementation.

1. Ontologies and ontology learning

In this section, we introduce ontologies, ontological elements and their roles in the Semantic Web. Moreover, we present the task of ontology learning and particularly an ontology learning from text.

1.1 Ontologies

In ancient greek, the term ontology meant “talking” about “being”. Nowadays, ontology is a philosophical discipline which can be described as the science of existence or the study of theory of being, existence and reality.

In computer science this term has diverged in some way from its original meaning. An ontology is not anymore a science, but a resource that is perceived as a formal representation of knowledge and that gives a structural framework to information.

Gruber [1] defined an ontology as an

“explicit specification of a shared conceptualisation”,

comprising a formal description of concepts, relations between concepts, and axioms on the relations in the domain of interest.

In other words, an ontology attempts to give a precise representation of real-world entities and relationships between them. Ontologies provide a key to structured data that are both machine and human understandable [2]:

“Ontologies serve as metadata schemas, providing a controlled vocabulary of concepts, each with explicitly defined and machine-processable”.

1.1.1 Ontological elements

An ontology describes the subject matter using the ontological elements such as concepts, instances, relations and axioms.

Ontological concepts

An ontological concept, or class, represents a set or class of entities within a certain domain.

Ontological instances

Ontological instances, or individuals, are the “ground level” objects that belong to a given domain and are attached to a certain concept. They could represent concrete objects such as persons, places and other real-word entities, as well as abstract objects.

Ontological relations

Relations define how ontological concepts and ontological instances are related to other concepts and instances. Usually, ontological concepts have a hierarchical tree-like structure. The *subclass-of* binary relation between concepts reflects the real-world generalisation and specialisation relationship. Another name for this relation is the IS-A relation and multiple and single inheritance could be applied on it.

In some contexts this hierarchical structure is called a **taxonomy** [12]. However, in the majority of contexts a taxonomy is perceived as a part of an ontology that represents a controlled vocabulary. In such case, an ontology could contain many taxonomies. In this paper, we will stick to the former perception of this term and we will call the relations related to the conceptual hierarchy as *taxonomic relations*.

We use two terms, **hypernym** and **hyponym**, to describe taxonomic relations. A concept is a **hypernym** of another concept if there is a taxonomic relation between these concepts and the first concept is the parent of the second concept in the hierarchy. A concept is a **hyponym** of another concept if there is a taxonomic relation between these concepts and the first concept is a child of the second concept. We also define the **basic ontological concepts** as the top-level concepts of the conceptual hierarchy that do not have any hypernym.

Depending on the ontology, other relations can exist between concepts. Within the non-taxonomical field, most ontologies include a special binary PART-OF relation that express composition, or meronymy, when one object is a part of another object. Other relations, typically referred to as **attributes**, express association between concepts and data values. In this paper, we call these relations as **non-taxonomic relations**.

Each relation and attribute has a **domain** and a **range**. A domain defines the concept a relation or an attribute is linked to. A range represents relation “datatype”, such as string or integer or another concept.

Ontological axioms

Ontological axioms are assertions in a logical form that put some constraints into an ontology or are used to deduct new information. Asserting the concept subsumption or equivalence are example of ontological axioms.

Lexical representations

An ontology can have a lexicon - a list of words in a given language with a knowledge of how the word is used [3]. Each ontological concept, relation, attribute can have several lexical representations that are defined by a lexical entry in the ontology lexicon.

To summarize, an ontology is a:

- set of concepts with attributes
- set of instances, each belonging to some concept
- hierarchy on this concepts

- a set of non-taxonomical relations between concepts
- a set of axioms and rules
- and depending to ontology a set of lexical representations of concepts, relations, attributes

1.2 Representation of ontologies

There are many formal languages used to encode the ontologies. We will stick to the generally accepted representation of ontologies in the World Wide Web by using the **OWL Web Ontology Language** endorsed by World Wide Web Consortium [7]. OWL is developed as a follow-on from the RDF technology.

In OWL, ontological instances are represented as a set of “*individuals*” and ontological concepts are represented as a set of “*classes*”. Taxonomical relations between classes are expressed through subclassing. However, there are also other possibilities how to express the taxonomic hierarchy, ex. by including the SKOS data model [9] into a ontology. Non-taxonomical relations are represented through “*properties*”.

The OWL language is composed of three sublanguages with different expressivity. *OWL Lite* is the less expressive one, *OWL DL* is more expressive with computational guarantees and *OWL Full* is the most expressive one but with no computational guarantees. In OWL Full, an OWL class can be also treated as an OWL individual. Most of reasoning software does not support every feature of OWL Full [8].

1.3 Semantic Web

The author of the term Semantic Web Tim Berners-Lee defined it as a

“web of data that can be processed directly and indirectly by machines.” [4]

The aim of the Semantic Web is to combine the information of the World Wide Web and provide semantic knowledge where most of data are in unstructured or semi-structured form. According to the W3C,

“the Semantic Web provides a common framework that allows data to be shared and reused across application, enterprise, and community boundaries.” [6]

Therefore, the Semantic Web is envisioned as an extension of World Wide Web. It describes techniques to make the meaning of Web information machine-readable with efficient content retrieval.

1.3.1 Role of ontologies in the Semantic Web

To achieve the possibilities of the Semantic Web, provided information should have a structured form with a set of inference rules used for automated reasoning [10]. Thus ontologies are ideal for this task because of their ability to give a framework to machine-understandable data.

According to Berners-Lee [5], ontologies can improve the functioning of the Web in many ways. They can be used in a simple fashion to improve the accuracy of Web searches. Moreover, other applications could use the ontologies to relate the information on a page to the associated knowledge structures and inference rules.

1.4 Semantic annotation

Semantic annotation is the process of attaching semantic information to various kind of text data. Mostly, it is ontology-based and the output information is machine-readable and formally connected to some ontology. The main goal of the semantic annotation is to enhance information availability, search improvement and customisation.

1.4.1 Annotation ontology

The annotation ontology is used in the process of semantic annotation and usually reflects the content of text data. This ontology often differs from the conception of an ontology presented in 1.1. Elements of such ontology represent important parts of text documents. The basic concepts form a set of categories, or "labels", for the interesting parts of texts. The descendents of these basic concepts in the hierarchy represent the terminology of a particular domain. And finally, the ontological instances do not represent the real life objects, but the occurrences of the ontological concepts in the text.

Annotation ontology example

As an example of an annotation ontology, we provide the ontology from the job descriptions domain. In the following text, we present the basic ontological concepts of this ontology along with the examples of linked concepts:

Domain represents a knowledge area the job applicant should have or have experience with (*"Java"*, *"programovací jazyk"*)

Level describes a level of a knowledge or another job characteristic (*"minimálně základní"*, *"dobrá"*)

Person represents a role of human being in a job description (*"klient"*, *"analytik"*)

Feature describes a job position characteristic (*"zázemí prosperující dynamické společnosti"*, *"stravenky"*)

Quality describes a quality of a job applicant (*"aktivní přístup"*, *"flexibilita"*)

Activity represents an activity related to certain Domain (*"znalost"*, *"zkušenost"*)

This ontology contains a non-taxonomic relation *HasLevel* connecting together the Activity and the Level and a non-taxonomic relation *HasDomain* connecting together the Activity and the Domain. With these relations we could express for example the fact that an Activity represented by *"znalost"* has a Level represented by *"dobrá"* and has a Domain represented by *"Java"*.

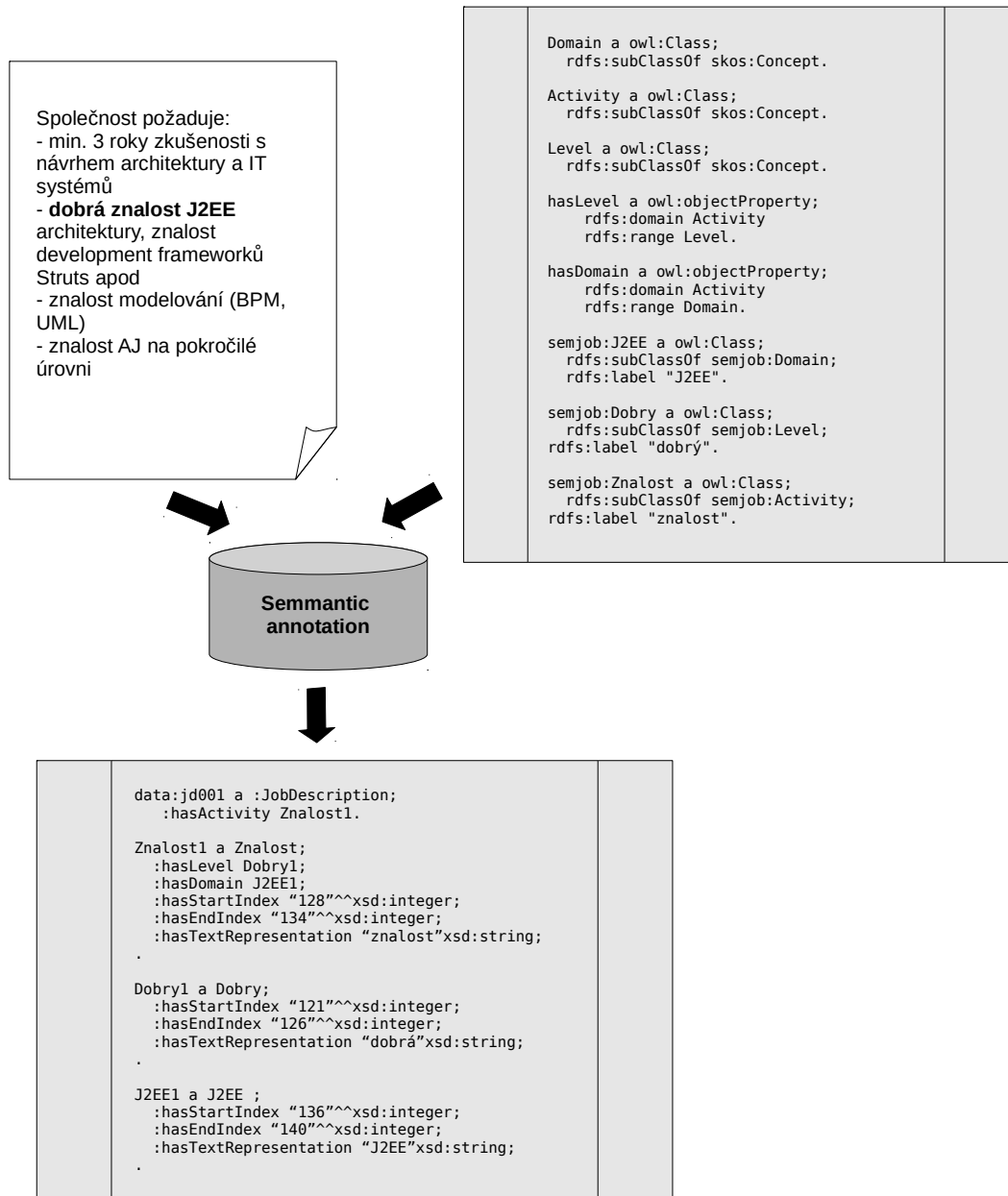


Figure 1.1: Semantic annotation

1.5 Natural language processing

Natural language processing (NLP) is the computerised approach to text analysis in order to process sentences in a natural language. Current approaches to NLP are based on statistical machine learning. The major NLP tasks include part-of-speech tagging, chunking, dependency parsing, word-sense disambiguation, semantic role labelling, named entity extraction and anaphora resolution.

1.6 Ontology learning from text

The term of **ontology learning** was originally coined by Madche and Staab in [11] and can be described as

“the act of acquisition of a domain model from data”.

Ontology learning requires relevant input data for a given domain. This input data can have many forms - structured (XML-DTD, UML), semistructured (XML, HTML) or unstructured. If ontology learning is based on unstructured data the term **ontology learning from text** is used [12].

Ontology learning includes different subtasks [12]:

- extraction of the relevant terminology,
- identification of synonym terms / linguistic variants (possibly across languages),
- formation of concepts,
- hierarchical organization of the concepts (concept hierarchy),
- learning relations, properties or attributes
- hierarchical organization of the relations (relation hierarchy),
- instantiation of axiom schemata,
- definition of arbitrary axioms

1.6.1 Extraction of terminology

This task is a prerequisite for every other task in ontology learning. The terms are textual representations of ontology concepts and relations and usually have a single-word or a multi-word form with a very specific meaning. The input of this task is a set of text documents and the output is a set of strings which are assigned to certain domain or concept [12].

1.6.2 Identification of synonym terms

The goal of this task is to group the terms with the same meaning, but different textual representations. In reality synonym terms have usually slightly different meaning, so it is not possible to create exact synonym groups. Rather the groups of terms with a related meaning are identified.

1.6.3 Formation of concepts

This task comprises a concept extraction from the set of group terms. The output of this task could also include other outputs such as several statistics related to concept extraction and other structures.

1.6.4 Formation of a concept hierarchy

The goal of this task is to create, extend or refine conceptual hierarchy or taxonomy from a set of concepts created in the previous step. This is typically performed together with their lexical representations.

1.6.5 Relations learning

This task include the induction of other non-hierarchical relations. First, the relation is identified in a text, mostly linking together two terms. Then a proper relation identifier is found together with its domain and range.

1.6.6 Other ontology learning tasks

Other ontology learning tasks comprises finding axiom schemata instantiations or other logical logical implications constraining the data. However, these tasks are not widely performed in the ontology learning context.

1.7 Ontology induction and enrichment

Typically, we make a difference between ontology induction and ontology enrichment. The goal of **ontology enrichment** is to add or modify existing ontology by performing one or several ontology learning tasks. In opposite, **ontology induction** means creation of an ontology from the scratch and usually combines multiple ontology learning tasks. However, the methods for both tend to be similar.

1.8 Conclusion

In this chapter, we introduced the notion of the ontologies in computer science, semantic web and ontology learning with the subtasks. In the following chapter, we want to specify our task of ontology enrichment in detail.

2. Task of ontology enrichment

In the previous chapter, we have described the basic terms in the domain of ontologies and ontology learning. In this chapter, we proceed on the specification of the task of ontology enrichment.

Aim of this thesis is to provide a general mechanism for an annotation ontology learning from text. In other words our mechanism for a given input text and a given input annotation ontology identifies important parts of a input text that could possibly represent new elements of the annotation ontology such as ontological concepts, relations and instances.

Primary, we want to extract new ontological concepts with their lexical representations from the relevant pieces of texts. These new ontological concepts will be attached to the basic ontological concepts in our annotation ontology. Moreover, we want to find more specific place of these new ontological concepts in the conceptual hierarchy. We want to achieve this by extraction of taxonomical relations from the text.

Each concrete concept occurrence in an input text could be represented by an ontological instance attached to this ontological concept. Thus we want to improve the process of semantic annotation by extracting this ontological instances along with their non-taxonomic relationships.

Concept and relations assignment is subjective to the current domain and user. In different domains, other concepts would probably be important. We want to handle this problem and a create a general and flexible mechanism, that could be applied generally in other domains too.

Our mechanism will be implemented for Czech language, but it should be general in the way that after small modifications if can be applied in other languages.

2.1 Assumptions and input

Our method should be generally applicable. However, we have chosen one concrete real-life domain to prove our method suitability and performance. The input of our method is formed by a set of domain related text documents and an input annotation ontology with respect to this domain.

2.1.1 Text documents

We use a real-life set of text documents in our mechanism. These documents originate from the domain of **computer science job descriptions**.

A job description is a list of general tasks, responsibilities and requirements of a work position. It has usually the form of a short text with the length of 40-60 lines.

Most job descriptions share a common structure, they have an brief introduction and a list of sections with headers. These sections usually include a job description section (“*Náplň práce*”), an offer section (“*Nabízíme*”, “*Společnost nabízí*” etc.), a requirement section (“*Požadujeme*” etc.) and a section with contact information. They have usually an outline form.

Společnost Sitewell s.r.o. se dlouhodobě specializuje na dodávky kompletního řešení prostorově orientovaných informačních systémů se zaměřením na správu majetku, provozně technické a územně identifikační informační systémy.

www.sitewell.cz

Programátor JAVA/J2EE

Požadujeme:

VŠ technického směru, SŠ s praxí;
Znalost GWT a JBOSS;
Hibernate či SQL/ORACLE výhodou;
Schopnost samostatné i týmové práce.

Nabízíme:

Práci na zajímavých projektech;
Zapojení do návrhu systému a jeho architektury;
Zajímavou, kreativní týmovou práci v mladém kolektivu,
příjemné, moderní prostředí;
Možnost profesního růstu;
Přístup k nejnovějším SW technologiím s možností podílet
se na jejich vzniku;
Dobré mzdové podmínky odpovídající praxi a znalostem
a dosaženým cílům;
Týden dovolené navíc.

Náplň práce:

Vývoj a údržba aplikace na platformě JAVA/J2EE podle zadání;
Implementace funkcí a změnových požadavků;
Komunikace a spolupráce ostatními členy technického týmu.

Místo pracoviště: Revoluční 551/6, Ústí nad Labem

Typ pracovního poměru: Práce na plný úvazek

Typ smluvního vztahu: Pracovní smlouva

Benefity: Dovolená navíc

Požadované vzdělání: Středoškolské nebo odborné vyučení
s maturitou

Požadované jazyky: Angličtina (Mírně pokročilá)

Zadavatel: Zaměstnavatel

Sitewell s.r.o.

Táborská 940/31, 14000 Praha - Nusle, Česká republika

Example of a job description

2.1.2 Annotation ontology

The input ontology describes the basic concepts of the job descriptions domain. We use the ontology as described in 1.4.1. For the application of our method only the declaration of the basic ontological concepts is required in the input ontology. However, the declaration of additional ontological sub-concepts of these basic ontological concepts can improve the performance of our method.

Concepts

Basic ontological concepts devoted by this thesis include: Domain, Level, Person, Feature, Quality and Activity.

In practice, because of the word phrase meaning ambiguity, one phrase could be attached to several concepts. For example the word “*analýza*” could represent an Activity and Domain too. Further, the concept occurrences in the text could overlap. For example the text “*spolupráce s týmem špičkových odborníků*” can be labeled by three concepts, the entire text could represent a Feature, the word “*špičkových*” could represent a Level and “*odborníků*” could represent a Person.

Instances

Ontological instances are primarily created in the process of semantic annotation. They represent specific concepts occurrences in the texts. For example, the ontological concept of type Level is represented by the term “*dobrý*”. This concept can have several occurrences in the texts with different lexical representations such as “*dobrého*”, “*dobrou*” etc. Each text occurrence is represented by one ontological instance of the concept “*dobrý*”.

In our method, we want to utilise the newly extracted concepts to improve the process of semantic annotation. We can accomplish this by creating new ontological instances based on the extracted concept occurrences.

Taxonomical relations

We primarily focus on the taxonomical relations extraction in our ontology. We recognise two taxonomical relations:

IS-A reflects a taxonomical concept hierarchy relation, when the first concept is a specification of second concept. This relation is reflexive, transitive and anti-symmetric. As an example, the IS-A relation could be between two concepts denoted as { “*stravenky*”, “*firemní benefity*” }.

COHYPO is a symmetric and reflexive relation that denotes two concepts sharing the same hypernym - parent concept in the taxonomy:

$$\text{COHYPO}(\text{concept1}, \text{concept2}) \Rightarrow \exists \text{concept3: IS-A}(\text{concept1}, \text{concept3}) \wedge \text{IS-A}(\text{concept2}, \text{concept3}).$$

As an example, the COHYPO relation could be between two concepts denoted as { “*stravenky*”, “*týden dovolené navíc*” }. Notice this relation expresses an implicit fact in the conceptual hierarchy and can be denoted as a composition of two IS-A relations. Therefore in our ontology no explicit denotation of this relation is needed.

If we look at the job description example above, we can assume that the phrase “*GWT*” is represented by a concept in our annotation ontology, but the phrase “*JBoss*” is not. We want to extract this phrase and create a related concept. We also want to find a proper place of this new concept in the hierarchy, thus the neighbouring term “*GWT*” can be used as the clue to find its location. We assume that the phrases occurring together in a sentence are related in some way. It could be a COHYPO relation in this case .

Non-taxonomical relations

Our mechanism should be adaptable to recognise other non-taxonomical relations too. Many non-taxonomical relations could exist in our ontology, we focused on detection of the following type of relations:

HAS-A Reflects a concept composition, when the second concept belongs to the first concept. An example of such relation could be the “*HasLevel*” relation linking together an Activity and a Level. This relation is realised in the text, if it connects together two instances of the ontological concepts. “*HasLevel*” relation could exist between an Activity concept occurrence denoted as “*znalost*” and an Level concept occurrence denoted as “*dobrá*”. Such relations would reflect the fact the first concept occurrence has a level denoted by second concept occurrence.

In our annotation ontology, HAS-A relations are defined between the basic ontological concepts. We have a relation “*HasLevel*” between an Activity and a Level and a relation “*HasDomain*” between an Activity and a Domain.

In the job description example above we can extract new ontological concepts represented by “*znalost*” and “*JBoss*”. Moreover, we can extract the non-taxonomical relation “*HasDomain*” relation connecting together the instances of these concepts. Thus we are able to create an ontological instance of the concept “*znalost*” with the property “*HasDomain*”. This property has value of the instance of the concept “*JBoss*”.

2.2 Machine learning task

The learning nature of our ontology enrichment problem leads us to machine learning. According to Tom M. Mitchell [27],

“a computer program is said to learn from experience *E* with respect to some class of tasks *T* and performance measure *P*, if its performance at tasks in *T*, as measured by *P*, improves with experience *E*”.

Machine learning techniques allows us to derive generalisation rules following the found evidences in the input data of the learning process . Generalisation is a property of a system that performs accurately on unseen data instances after the learning process. [30].

With machine learning techniques, we can treat our ontology enrichment problem as a **classification task**. Classification is a problem of identifying the category or a set of categories new unknown instance belongs to. We can use our

input ontology with a set of basic concepts as categories, or labels, for some important parts of texts. By analogy, ontological relations defined in our ontology can be used as the labels in the classification process.

In the machine learning process, we want to use a significant amount of input text data to derive classification rules of the classification mechanism. Then this classification mechanism will be used in the process of discovering new ontological concepts, instances and relations between these concepts and instances. Specifically, in the case of ontological concepts, we will use the classification mechanism to assign a basic ontological concept or none ontological concept to a given input phrase. Then we extract new ontological concept with its textual representation from this revealed part of text. In the case of ontological relations, we assign an ontological relation or none relation to a pair of word phrases. This relation then reflects a taxonomic relationship between two concepts or the non-taxonomic relationship between two ontological instances.

To summarise, our task is to design a classification mechanism based on machine learning that fulfils our requirements in the means of generality and domain adaptability.

2.2.1 Formal definition of the classification task

Formally, we define the classification task as follows.

Let C be a finite set of basic ontological concepts, W a set of word phrases, T be a finite set of taxonomical relations and R be a finite set of non-taxonomical relations.

In our ontology, C can be denoted as $\{D, L, P, F, Q, A\}$, where D represents *Domain*, L *Level*, P *Person*, F *Feature*, Q *Quality* and A represents *Activity*. In case of taxonomical relations, T can be denoted as $\{I, O\}$, where I represents *IS-A* relation and O represents *COHYPO* relation. Analogically for non-taxonomical relations, $R = \{H\}$ where H represents *HAS-A* relation.

Let a $s : W \rightarrow 2^C$ be a function that assigns a set of basic ontological concepts to a word phrase. Next, let $t : W \times W \rightarrow T \cup \{\emptyset\}$ be a function that for all word phrases w_1, w_2 such that $s(w_1) \cap s(w_2) \neq \emptyset$ assigns a taxonomical relation $t \in T$ or none relation.

Let $r : W \times W \rightarrow R \cup \{\emptyset\}$ be a function that for all word phrases $w_1, w_2 \in W$ assigns a non-taxonomical relation $r \in R$ or none relation.

These functions are known implicitly by an observer that defines $s(w)$ for $w \in W$, $t(w_1, w_2)$ and $r(w_1, w_2)$ for $w_1, w_2 \in W$.

The function s could assign a concept D to word phrase “*Java*”, the function t could assign for word phrases “*Eclipse*”, “*vývojové prostředí*” a taxonomical relationship I and for word phrases “*Eclipse*”, “*NetBeans*” a relationship O . The function r could assign a non-taxonomical relationship H to word phrases “*znalost*”, “*dobrá*”.

Definition. *Classification method.* Classification method is an algorithm that computes the functions $s' : W \rightarrow 2^C$ and $t', r' : W \times W \rightarrow R$.

Definition. *Classification method accuracy.* Classification method is accurate for a concept $c \in C$ and word phrase $w \in W$, if $c \in s'(w)$ and $c \in s(w)$. Classification method is accurate for relation $t \in T$ and word phrases $w_1, w_2 \in W$,

if $t'(w_1, w_2) = t(w_1, w_2)$ and analogically for relation $r \in R$ and word phrases $w_1, w_2 \in W$, if $r'(w_1, w_2) = r(w_1, w_2)$.

Definition. *Metric of classification method accuracy.* Metric of classification method accuracy for a concept c and a set of word phrases $W' \subset W$ is defined as $|W'_{acc}|/|W'|$ where W'_{acc} is a set of $w \in W'$ such as $s'(w)$ is accurate for concept c .

Analogically for the functions t' and r' : Let W_T be a subset of $W \times W$ where for some concept $c \in C$ and $\forall w_1, w_2 (w_1, w_2) \in W_T, c \in c(w_1) \cap c(w_2)$. Classification method accuracy metric for W_T is defined as $|W_{T_{acc}}|/|W_T|$ where $W_{T_{acc}}$ is a set of accurate $t'(w_1, w_2)$ assignments for $w_1, w_2 \in W_T$.

Let W_R be a subset of $W \times W$ where $\forall w_1, w_2 (w_1, w_2) \in W_R, \exists c_1, c_2 \in C, c_1 \in c(w_1)$ and $c_2 \in c(w_2)$. Classification method accuracy metric for W_R is defined as $|W_{R_{acc}}|/|W_R|$ where $W_{R_{acc}}$ is a set of accurate $r'(w_1, w_2)$ assignments for $w_1, w_2 \in W_R$.

Our goal is to design a classification method with as high accuracy metrics as possible. The method should be general, language independent and should capture the possible large flexibility of input word phrases. Further, it should also treat the subjectivity of concept and relations assignment. By assumption the input documents are originated from very specific domain, we omit the possibility of the term meaning ambiguity. The term meaning is defined by given domain.

We will explore existing machine learning mechanisms and choose the one that satisfies the most our requirements. If possible, several machine learning algorithms will be explored with their performance evaluating.

Prediction classes

In our classification method for some concept $c \in C$, the classifier prediction classes have two values corresponding to the positive classification class and the negative classification class: $\{ \text{"belongs-to-concept-c"}, \text{"not-belongs-to-concept-c"} \}$.

For the taxonomical and non-taxonomical relations classifier, multiple result classification classes can exist. With the set of n taxonomical relations T , $T = \{t_1, t_2, \dots, t_n\}$, the result classes are $\{ \text{"is-}t_1", \text{"is-}t_2", \dots, \text{"is-}t_n", \text{"none"} \}$. Analogically for the set of n non-taxonomical relations R , $R = \{r_1, r_2, \dots, r_n\}$, the result classes are $\{ \text{"is-}r_1", \text{"is-}r_2", \dots, \text{"is-}r_n", \text{"none"} \}$.

Feature extraction

Necessary part of each machine learning method is identification and extraction of input data characteristics. These characteristics are measurable properties of input data instances used in the classification rules derivation. We call them **features**, respectively **attributes**.

In our method, the input data has the form of word phrases. We have to found a set of features of these word phrases with the respect of classification classes - in our case basic ontological concepts and relations. Extracted characteristics will form a n -dimensional vector V in \mathbb{R}^n , where n is the number of characteristics.

Learning of a classification mechanism

Learning of a classification mechanism denotes the process of algorithmic training based on input learn data. Output of this process is an algorithm that implements the classification and maps data into classification classes. This algorithm is known as a **classifier**. This phase includes selecting the train data and proper machine learning algorithm for the result classifier.

Evaluation of performance

Performance of a classifier trained on a training set is evaluated experimentally on a different data set, called a **test set**. This experimental performance measures its ability to generalisation. To asses a classifier performance, we use our accuracy definition defined in section 2.2 as a criterion function.

Another criterion functions could be used in classification problems. Usually, their are based on the **confusion matrix**. A confusion matrix is showing the predicted and actual classifications. A confusion matrix is of size $L \times L$, where L is the number of different class values [29]. Instances of a predicted class are denoted in the columns of this matrix, while the rows represents the instances in an real class.

In the case of a binary classifier, the matrix will report the number of true positives classifications, true negatives classifications, false positives classifications and false negatives classifications.

		Actual class (observation)	
		Positive	Negative
Predicted class (expectation)	Positive	true positive (tp)	false positive (fp)
	Negative	false negative (fn)	true negative (tn)

Table 2.1: Confusion matrix for a binary classifier

True positives For the concept classifier, true positives (tp) represents the number of the items correctly labelled as "belong-to-concept-c". With our accuracy metrics, let W_{acc}^+ be the subset of W'_{acc} , where $c \in s'(w) \forall w \in W_{acc}^+$. True positives is the size of the set W_{acc}^+ .

For the taxonomic and non-taxonomic relation classifiers, true positives represents the number of the items correctly labelled as "*is-t*" respectively "*is-r*" for some relations $t \in T$, $r \in R$. Let $W_{T_{acc}}^+$ be the subset of $W_{T_{acc}}$ where $\forall (w_1, w_2)$ in this subset $\exists t \in T$, such as $t(w_1, w_2) = t$. True positives is the size of the set $W_{T_{acc}}^+$. Analogically for the set of nontaxonomical relations.

True negatives For the concept classifier, true negatives (tn) represents the number of the items correctly labelled as "not-belong-to-concept-c". Let W_{acc}^- be the subset of W'_{acc} , where $c \notin s'(w) \forall w \in W_{acc}^-$. True negatives is the size of the set W_{acc}^- .

For the taxonomic and non-taxonomic relation classifiers, true negatives represents the number of the items correctly labelled as "*none*". Let $W_{T_{acc}}^-$ be the subset of $W_{T_{acc}}$ where $\forall (w_1, w_2)$ in this subset, $t(w_1, w_2) = \emptyset$. True positives is the size of the set $W_{T_{acc}}^-$. Analogically for the set of nontaxonomical relations.

False positives For the concept classifier, false positives (fp) represents the number of the items incorrectly labelled as "belong-to-concept-c". Let W'_{inacc} be the set $W' \setminus W'_{acc}$ and let W'_{inacc+} be the subset of W'_{inacc} , where $c \in s'(w) \forall w \in W'_{inacc+}$. False positives is the size of the set W'_{inacc+} .

For the taxonomic and non-taxonomic relation classifiers, false positives represents the number of the items incorrectly labelled as "is-t" respectively "is-r" for some relations $t \in T, r \in R$. Let $W_{T_{inacc}}$ be the set $W_T \setminus W_{T_{acc}}$, let $W_{T_{inacc}+}$ be the subset of $W_{T_{inacc}}$ where $\forall (w_1, w_2)$ in this set, $\exists t \in T$, such as $t(w_1, w_2) = t$. True positives is the size of the set $W_{T_{inacc}+}$. Analogically for the set of nontaxonomical relations.

False negatives False negatives (fn) represents the number of the items incorrectly labelled as "not-belong-to-concept-c". Let W'_{inacc-} be the subset of W'_{inacc} , where $c \notin s'(w) \forall w \in W'_{inacc-}$. False negatives is the size of the set W'_{inacc-} .

For the taxonomic and non-taxonomic relation classifiers, false positives represents the number of the items incorrectly labelled as "none". Let $W_{T_{inacc}-}$ be the subset of $W_{T_{inacc}}$ where $\forall (w_1, w_2)$ in this set, $t(w_1, w_2) = \emptyset$. True positives is the size of the set $W_{T_{inacc}-}$. Analogically for the set of nontaxonomical relations.

The most commonly used criterion function are **accuracy**, **precision**, **recall**, **true negative rate** and **F-measure** [28],

Accuracy Accuracy represents the rate of correct predictions made by the classifier over a data set. The disadvantage of this criterion function is that accuracy puts equal costs for misclassification and assumes relatively uniform class distribution.

$$accuracy = \frac{tp + tn}{tp + tn + fp + fn} \quad (2.1)$$

Notice that this accuracy definition corresponds to our accuracy definition in section 2.2.

Precision Precision represents the rate of correctly classified positives over the total number of estimated positives.

$$precision = \frac{tp}{tp + fp} \quad (2.2)$$

Recall Recall represents the rate of correctly classified positives over the total number of positives.

$$recall = \frac{tp}{tp + fn} \quad (2.3)$$

True negative rate True negative rate is the rate of classified negatives over the total number of negatives.

$$\text{true negative rate} = \frac{tn}{tn + fp} \quad (2.4)$$

F-measure F-measure is used for calculating aggregate performance over precision and recall.

$$F - \text{measure} = \frac{2}{\frac{1}{\text{precision}} + \frac{1}{\text{recall}}} \quad (2.5)$$

2.3 Pre-annotation and the use of existing data sources

The purpose of our classification method is to identify new candidate ontology elements. So it is not necessary to extract elements that are already part of ontology. Already existing ontological concepts can be located in the input text document by some kind of semantic-annotation process and their evidence could be exploited in the classification process.

Our task is not to implement this pre-annotation mechanism, but our method should allow to possibly include them in the input of the classification process. By the reason of the maximum performance measurement of our method, we will also include existing ontology elements in the evaluation.

The relation information in the domain text documents is not complete and thus it is not possible to extract entire taxonomical relationships. Existing ontological data-sources such as DBpedia could be used to supply missing relations in the hierarchy. With the pre-annotation outputs included in the relation classification, we are able to insert the new ontology elements in the proper place of the taxonomical hierarchy.

2.4 Prototype implementation

Performance and characteristic of the proposed method will be the verified in the prototype implementation. Although our prototype implementation will operate with the job description ontology and Czech language, it should be implemented as language independent and adaptable to another domains with minor modifications. We expect that the prototype implementation will have low accuracy, thus it is necessary to establish initially design allowing further improvements of the method.

3. Related work

3.1 Methodologies

Creating ontology manually is very costly process and requires a lot of time and human resources. To facilitate this process, several methodologies for designing and building ontologies automatically have been proposed in recent years. We can divide this techniques into two main groups.

The first group employs **statistical methods** such as machine learning and clustering, mostly supervised. Machine learning treats the problem of ontology creation as the classification problem, whereas clustering method is based on similarity measure. Both groups require a large amount of input data. Machine learning methods use this data in the learning process and clustering methods use it in the process of similarity computing.

The second group of techniques is based on linguistic information extracting and includes various **lexico-syntactic pattern methods**. These methods use specific linguistic relations between words to determine ontological concepts and relations.

Some approaches use a combination of both methods. In the next section we describe key works in this area. Their overview is presented in the table 3.1

3.2 Overview of related work

3.2.1 Text-To-Onto

The pioneer work in the area of ontology enrichment is the framework Text-To-Onto [11]. The framework implements several tasks that facilitate ontology enrichment. Proposed method of ontology learning is semiautomatic and consists of several phases. The output of each phase has to be reviewed by an ontology engineer. The input of the method includes different kinds of resources such as plain text or semistructured text.

The first phase of an ontology enrichment is the extraction of lexical terms and corresponding ontological concepts. Input resources are processed on the morphological level, then the terms are extracted by statistical methods. From given set of concepts, a taxonomy is created by the hierarchical concept clustering based on similarity measure between lexical terms. The work also deals with ontology pruning and refinement. With the use of statistical measure tf/idf , the system cannot effectively extract domain-specific concepts. Further, identification of semantic relations is based on part-of-speech tags only, so the accuracy of extracted relations is limited. The method achieves the precision of 47.2 % and the recall of 2.1 % in the area of concept extraction.

3.2.2 OntoLT

Other approach is presented by Buitelaar [13]. OntoLT, a Protégé plugin, enables a user to interactively define mapping rules between XML-based linguistic

annotations and ontological concepts, instances and properties. This method employs statistical preprocessing to eliminate irrelevant text words. Mapping rules could be generated semi-automatically for all relevant words. An end user has to validate the generated Protégé classes and slot candidates. The plugin has been tested on the task of ontology extraction from a neurology corpus.

3.2.3 OTTO

Another framework, OTTO - An Ontology-based Framework for Text Mining, has been presented by Bloehdorn [14]. It enables semi-automatic ontology construction or enrichment. Three algorithms are implemented in this framework. First, a conceptual clustering algorithm used to build taxonomies. Then taxonomies are constructed by a combination of information from WordNet and Hearst patterns. Finally, non-taxonomic relations are determined by extraction of syntactic frames of input terms.

In the area of text clustering and classification with WordNet as an ontology, the method achieves several improvements. In the case of unsupervised text classification, F-measure is increased by 8 % . In the case supervised text classification, the method shows the improvement of F-measure up to 7 %.

3.2.4 C-Pankow

Another approach to the problem of ontology-based semantic annotation is C-PANKOW [15]. A web page is scanned for candidates terms defined by an ontology with a set of regular expression over words and part-of-speech tags. Then for each found term and each pattern string out of a certain pattern schema, a query for the Google API is generated. Few first result abstracts are downloaded and compared with the input web page. If the similarity with the source web page exceeds the given threshold, the possible candidate concept is revealed. The concept relevance increases with growing similarity between the web page and the abstract. The method achieves precision of up to 33 % and recall of 22 %.

3.2.5 Using Decision Trees and Text Mining Techniques for Extending Taxonomies

Method that uses a taxonomy as a decision tree was proposed by Witschel [17]. In this method, each taxonomy element is enriched by a set of features based on semantic description - a set of words used in a similar context. Then candidate terms are extracted from texts using Part-Of-Speech tags and regular expressions. Similarity between a candidate text element and an existing taxonomy element is computed by comparison of their semantic descriptions. A taxonomy tree is passed from the root to the leaves until the most similar node is found. The accuracy of this method is up to 14 %.

3.2.6 OnTeA

OnTea is an automated solution to semantic annotation of text presented by Laclavik [18]. In this solution, existing ontology elements are detected in text

documents with the use of manually created regular expressions. If an occurrence is found, a new or existing instance of an ontological concept is returned as the result. Precision of this method is 63.2 % and recall 83.1 %.

3.2.7 Learning syntactic patterns for automatic hypernym discovery

Another paper presents a new algorithm for extraction of hypernym relations [20]. First, a corpus is annotated with a set of known hypernyms obtained from WordNet. Then the set of lexico-syntactic patterns denoting the IS-A relation is learned from this corpus and the frequent patterns are extracted. Finally, a term adjacency to these patterns is used in the classification process for an unknown instance. This method shows F-measure up to 33,6 %.

3.2.8 Semantic Taxonomy Induction from Heterogenous Evidence

The enhancement of the previous method has been proposed by the same authors [19]. This work presents a probabilistic framework for taxonomy induction. It deals with two main problems in taxonomy learning: words sense disambiguity and heterogeneous source of evidences. A set of taxonomical relations occurrences is obtained from different classifiers: the hyponym classifier that was presented in the previous section 3.2.7 and the coordinate terms classifier based on the work of Ravichandran [21]. Then this information is combined and the result taxonomy is calculated as the most probable taxonomy that satisfies all occurrences. This method has precision of up to 68.3 % and recall of 21.1 %.

3.2.9 CRCTOL

The system CRCTOL proposed by Jiang and Tan [22] automatically creates ontology from domain specific texts. It employs a combination of statistical and lexico-syntactic patterns for key ontology concepts and taxonomic relationships extraction. Lexico-syntactic patterns are fixed, both Hearst patterns and term structure patterns are used. Extraction of non-taxonomic relations is based on the hypothesis that verbs indicate non-taxonomic relations between concepts. Syntactic trees obtained by text parsing allows to determine relations between concept nouns. The system shows precision up to 92.8 % and recall up to 4.1 % for concept extraction. In the case of non-taxonomic relations extraction, it achieves precision of 81.5% and recall of 55.5%. Precision of taxonomic relations extraction is 43.5 %.

3.2.10 OntoUSP

Previously mentioned methods mostly employed supervised techniques for ontology extraction and related tasks. Newer approach is revealed in the system OntoUSP [23]. It induces and populates a probabilistic ontology using only plain

texts with dependency parsing as the input. This system creates an IS-A hierarchy by forming logical expressions with the use of Markov logic. The system has 91 % accuracy on the GENIA dataset.

3.2.11 SOFIE

Another unsupervised system for automated ontology enrichment is the SOFIE system [26]. It parses natural language documents, extract ontological facts from them and composes these new facts into the ontology. Words are disambiguated to their most probable meaning by logical reasoning on the existing knowledge. As the newly extracted facts have to be logically consistent with the existing ontology, algorithm Weighted MAX-SAT is employed to deal with this problem. Results on different corpuses showed up to 94.7 % precision and 31.08 % recall.

3.3 Conclusion

The importance of the ontology enrichment task is reflected in large number of works dealing with this problematic. However, even after several years of studies this problem remains still the open and the ontology learning methods are hardly applicable in the practice. In this chapter we have presented several methods how to deal this problem based on different approaches with various performances, but it is evident that this problem is still relevant. In the next chapter, we want to define our approach to the ontology enrichment and describe its benefits.

	Input data	Input Language	Ontological elements learned	Degree of automation	Resource	Ontology enrichment or de novo generation	Learning Methods
Text-To-Onto	Dictionaries, unstructured text, semi-structured text	German	Concepts Taxonomic relations Non-taxonomic relations	Semi-automated	Domain ontology	Enrichment	Formal concept analysis Clustering Association rules
OntoLT	Unstructured text	German	Concepts Properties	Semi-automated	Domain ontology	N/A	Generating linguistic mapping rules using statistical methods
OTTO	Web pages	English	Concepts Taxonomic relations Non-taxonomic relations	Semi-automated	KAON	Enrichment	Clustering Lexico-syntactic patterns
C-Pankow	Web pages	English	Semantic annotations	Automated	Google API	N/A	Statistical method with similarity measure
Decision Trees and Text Mining	Unstructured Text	German	Taxonomic instances and relations	Automated	Existing taxonomies from GermaNet	Enrichment	Decision trees
OnTeA	Unstructured text (CV)	English	Taxonomical instances	Semi-automated	Domain Ontology	N/A	Regular pattern matching
Automatic hypernym discovery	Unstructured text	English	Taxonomical instances and relations	Semi-automated	WordNet	De novo Enrichment	Machine learning Different classifiers
Semantic taxonomy Induction	Unstructured text	English	Taxonomical instances and relations	Semi-automated	WordNet	De novo Enrichment	Probabilistic framework Different classifiers
CRCTOL	Unstructured text (terrorism domain)	English	Concepts Non-taxonomic relations Taxonomic relations	Semi-automated	WordNet	De novo	Lexico-syntactic patterns Statistical Methods Association rules
OntoUSP	Unstructured text (biomedical domain)	English	Concepts, Taxonomic relations	Automated	N/A	De novo	Statistical methods (Markov logic) Clustering
Sofie	Unstructured text, Semi-structured text	English	Ontological facts	Automated	Ontology with ontological facts Wikipedia	Enrichment	Lexico-syntactic patterns Weighted MAX-SAT

Table 3.1: Overview of ontology enrichment methods

4. Analysis of the problem

In the previous chapter, we have described our goals in the task of ontology enrichment. During the development of our machine learning mechanism, several questions need to be answered. In this chapter, we will look in more detail on these questions and provide an analysis of our ontology enrichment problem.

Our task of ontology enrichment can be divided into several subtasks. First, we want to identify important parts of text and assign them a set of basic ontological concepts from a given ontology. Then we want to extract new ontological concepts from these revealed pieces of text. Further, we want to determine hierarchical dependencies between these concepts by assigning them taxonomic relations. Finally, we want to create ontological instances for the given ontology. These instances will be represented by concrete occurrences of some ontological concepts in a text document and we will link them by non-taxonomic relations.

4.1 Concepts and relations assignment for a particular user

The first problem we encounter is how to handle the fact the assignment of concepts to word phrases, respectively relations to words phrases pairs, is subjective to a given domain and a particular user. In different domain, other word phrases could be important and the ontology could contain a different set of basic concepts. Thus we need to take into account that functions c , r and t are defined by a particular user and are not known explicitly. Our classification mechanism should be configurable in the way that it could be used by other users and applied on other domains too.

It turns out that we can use a partial representation of the functions s , r and t in the learn data. Our implemented functions s' , r' and t' should be as much accurate as possible on the rest of data with the respect of s , r and t . In this way we can handle these problems as well as the fact that definition of functions s , r and t is implicit only. We can use a data set with marked functions assignments in the learning process. Thus we should be able to made generalisations with respect to a particular user.

4.2 Language independence

Another problem we encounter is the language independence of the method. We have not ti use the techniques adapted to a particular natural language. Instead of this, we want to capture various linguistic information from the text data and use it in the classification process. Such linguistic information is different in different natural languages, so our method should be configurable to capture this variability.

4.3 Data set processing

Machine learning task includes the phase of data set processing. In our case, the data set consists of unstructured text documents. The goal of this step is extraction of meaningful information that will be used in the ontology enrichment process.

4.3.1 Features extraction

Feature extraction is the process of creating measurable characteristic from word phrases. They will be treated by our classification mechanism both in the training and the classification process for unknown instances. While the feature extraction from the structured data is relatively simple task, it is really challenging with the unstructured data. The computer is not able to “understand” the word phrases meaning. Therefore we must employ some techniques that are able to transform the information contained in text data to machine readable form such as a vector of n -real numbers. To implement this process, we need to answer several fundamental questions.

Features identification

The first important question is the identification of the features we are able to extract from text and that could be relevant in the classification process. This features should not only have the capability to detect important and unimportant parts of texts, but also distinguish between the text parts categories with respect of ontological concepts and relations.

Intuitively, such characteristics could be the linguistic characteristics of words and sentences. Another type of characteristics could be the simple textual characteristics such as the phrase length, the place of occurrence in the sentence or entire text. Next suggested type of characteristic are the characteristics related to the neighbourhood of word phrases.

In the process of concepts and relations detection, different characteristics could be used for each concept or relation. The feature that could be useful in the identification of a Domain in the text, could not at all be relevant for the extraction of a Level. We must handle this problem and design our mechanism to respect the feature diversity between concepts and relation types.

Precise selection of the feature set is necessary because of another aspect. The feature set size has the impact on the classification result. With the growing feature set, the classifier performance is decreasing. This phenomenon is known as *the curse of dimensionality* [31]. As we are expecting that the feature set could have many features, we must be prepared to deal with this problem.

Feature values acquiring

The acquiring of the feature values we have chosen is another problem that we have to solve. We need to find mechanisms that will automatically extract the feature values from the word phrases. Our data set has a form of unstructured texts. We need to transform this unstructured texts into a more complex structure that will provide a direct access to the text words and sentences.

To access the linguistic information contained in the text, we must employ some natural language processing techniques and tools. The output of this techniques and tools should be processed and included in the features values extraction process.

Further non-linguistic information will be extracted from the text, so we need to combine this information and transform it in the way that it will be directly readable by the classifier.

4.4 The classifier learning process

The classifier learning process in our method is the process of training and evaluating the classification mechanism. It should be able to recognise new ontology concepts and relations in the text. This process brings several questions. First, we must design our classification mechanism to meet the requirements given by the functions s' , r' and t' . Then we must choose a proper classification algorithm and along with the training algorithm. The chosen mechanism should be suitable for our method.

The function s' returns for one text phrase a set of basic concepts $C_1 \subset C$. To reduce the complexity of a classifier computing s' , we can divide this computation in n steps, where $n = |C|$. Otherwise the set of classification classes will grow exponentially with the size of C . Thus we will train n classifier, one for each basic concept in C . The functions r' and t' returns only one relation from the set R or T , therefore we need to create only one classifier that compute each function.

4.4.1 Negative instances

Another problem in our machine learning process is the identification of negatives instances. While the positive instances are defined implicitly by the functions c , r and t , the negative instances of concepts and relations are not defined. The positive specifications of concepts and relations could be described by the natural language of a particular user, but the negative specification is set only by the current situation, when the concept or relation description does not fit the current word phrase(s).

Our classification mechanism should handle this situation and create possibly artificial negative instances, which will be used in the classifier learning as the substitutes for the missing negative instances.

4.5 Ontology extraction

After the process of our classification mechanism training, we want to derive new ontological facts from the input texts. The output of the classification needs to be processed and the ontological representations of newly revealed facts have to be created. Before we can do this, we need to solve several problems.

4.5.1 Defining the unknown instances

Next important question is the identification of the word phrases as candidate concepts or the pairs of word phrases as candidate relations for the input of the classification process. The interesting word phrases are the part of sentences and they are not initially highlighted in any way. We could not make any presumptions about their form and occurrences.

We want to design a general mechanism that will be able to select this interesting parts of texts. The mechanism will operate independently on the ontology domain. We assume that not all candidate input phrases could be selected by this mechanism, but it should handle the majority of the interesting input phrases.

4.5.2 Significance of the extracted facts

Extracted concepts and relations does not have the same significance. If a new concept is revealed by our mechanism only in one document instance from the entire set, it could supposedly be a classification error. Such concept is probably not as significant as a concept or relation revealed in many documents. Analogically for the relations extraction. We want to create a model that reflects the real significance of the classification mechanism output. This model should be relatively dependent on the performance of our classification algorithm, as with higher performance the revealed concepts are more important.

4.5.3 Concept extraction

Selected candidate word phrases are the input for the concept extraction, the output is formed by a subset of basic ontological concepts to word phrase could possibly belong to. But before the concept extraction itself, we must answer several questions.

Names and lexical representations

We need to deal with the fact that we have extracted only lexical representation of a potential ontological concept in the classification process. Each ontological concept has a defined name and also in our case, one concept could have several lexical representations. Thus in the extraction process, we must somehow derive the identifier and the lexical representation of the new concept.

Further, we could have potentially extracted several lexical representations belonging to the same concept. In the simplest case, these lexical representations are equal, but in some natural languages the words could have a rich verbal form. In this case, the extracted lexical representations attached to one concept could be slightly different and we should merge them into one lexical representation. In other cases, the lexical representations belonging to the same concept could be completely different, ex. in the case of synonym terms. We have to deal with these problems in the concept creation process.

Existing concept in the ontology

Another problem is the extraction of a concept that already exists in our ontology. The existing concept could be defined only partially in the ontology and we could

extracted its new lexical representation. We can simply ignore the extraction of concept that already exists in the ontology in the case the lexical representations are the same.

4.5.4 Ontological relation extraction

In the case of ontological relations extraction, the input is a pair of word phrases, that represent some concepts c_1 , c_2 in C and the output is determined by a relation assigned to these phrases. This step is consecutive to the previous step of ontological concepts identification in the text. Notice the concept identification does not necessary have to be our implemented concept extraction. We can use other techniques such as a semantic annotation tool to provide the input for this process. Moreover, we can combine multiple inputs that can help us to reveal the relations between already existing and newly revealed ontological concepts. As we are not able to extract a complete conceptual hierarchy from text data, this method can be very useful.

Taxonomical relations extraction

Taxonomical relation extraction enables us to unfold the conceptual hierarchy in our ontology. In the extraction of new ontological concepts, we give them some “initial labels” represented by the basic ontological concepts the new concept is attached to. With taxonomical relations extraction, we are able to include the new concept more precisely in a taxonomy by revealing its relationship with other ontological concepts.

Non-taxonomical relations extraction

Extraction of non-taxonomical relations allow us to deduct further information about the connection of ontological instances in text. Such relations are not relevant in the construction of a conceptual hierarchy, but they can be useful to refine the semantic annotation outputs.

4.5.5 Ontological instances extraction

Next step in the ontology learning process is the ontological instances extraction. While the conceptual hierarchy extraction mentioned earlier provide us an information about the taxonomical structure, the extraction of the ontological instances reveals another useful ontological facts about the occurrences of the ontological concepts in the concrete text instance. Along with non-taxonomical relation extraction, we are able to extract a complex information about the content of particular text instances.

5. Design of the solution

In the previous chapter, we have analysed the task of ontology enrichment and identified the problems that need to be solved before the implementation. In this chapter, we will introduce the design of our solution and our approach to these identified problems.

As we mentioned before, we can divide the process of ontology extraction into three tasks: the first step is the ontological concepts extraction, the next step is the relations extraction and the last optional step is the extraction of ontological instances.

In the previous works, the extraction of concepts has been approached using a variety of methods. Mostly the extraction of concepts was done together with the concept hierarchy extraction by looking for specific patterns in the texts. These patterns included the Hearst patterns and another lexico-syntactic patterns [20] [17]. Another approach is based on the use of statistical methods. In this approach, the concepts are identified by computing the relevances of document words based on term frequency–inverse document frequency (td/idf) and similar measures. Often these methods are combined [11] [22]. The approach presented in [15] uses the Google API to found the relevant concepts by counting the number of returned occurrences.

In the are of relation extraction, the related work is mostly concentrating on the extraction of taxonomical hierarchies. The taxonomical relations are extracted by looking for specific lexico-syntactic patterns as we mentioned above or by looking for a specific term structure [22]. Another method is to use the lexical dictionaries such as WordNet to determine the hyponym relations [14] [11]. For the non-taxonomic relations, extraction is mostly based on the fact that verbs indicate non-taxonomic relations between nouns [11] [22].

Some papers also deal with the word meaning disambiguation problem [14] [19], when one lexical phrase could have several meanings in the natural language. The meaning of a phrase is defined by a current context. As we assume that our domain is very specific, we do not provide a solution to this problem.

As we have mentioned in the previous chapter, we want to create a highly configurable and adaptable method. Therefore we have chosen the classification approach to the ontology enrichment. We are not looking for specific patterns in text data and we are not using any particular natural language techniques such as lexical dictionaries as the core of our method, but we want to train a classifier to found the relevant pieces of text and relations between them. Our classifier will recognise the implicit patterns based on the generalisation rules from the learning data set.

5.1 Concepts and relations assignment

In the previous chapter, we have revealed that the partial denotation of the functions c , r and t could be the ideal solution to make our method highly adaptable to a particular domain. In this way, we handle the problem that the definition of some important word phrases is subjective to a domain and an observer together with the problem of an implicit knowledge of these functions.

Other approaches to identify relevant parts of text exist, but most of them does not provide such range of configurability. In these approaches, same base presumption about the form of these parts of text is made. Mostly, these methods are looking for sequences defined by Part-Of-Speech tags rules [11] [15] [22] [17], specific syntactic dependencies such as verb-object relations [14] or the important parts of texts are defined by XPATH expressions over the linguistic annotation of the text[13], respectively regular expressions [18].

Next question is how to partially denote the functions c , r and t in the text data. Because this denotation has to be done manually, some easy operable tool that enables the text annotation is needed. We need to annotate both concepts and relations between them.

Such annotation tool could be *Brat* [33]. Brat meets our requirements in the means of operability. Moreover, both important word phrases and relationship between them could be denoted in a text by this tool.

5.2 Language independence

Another requirement on our method is the language independence. Most of ontology learning methods that we have described in the chapter 3 use some techniques adapted to a particular language. These techniques include the Hearst patterns [24] in the task of automatically learning conceptual hierarchy [14] [15] [22] [16]. Next, the machine readable dictionaries such as WordNet are used to reveal the taxonomical concept relations [25] [14] [16].

Our method will be implemented for Czech language, but with minor changes it should be applicable in other languages. We do not want use the techniques that rely on the concrete structures of a particular language, but we need that our method will be adaptable to these structures. The linguistic information among various languages is different as they do not share the same parts-of-speeches, grammatical tenses and cases. We want to achieve the language independence by separating and encapsulating the linguistic information of the text in the implementation. We need to configure what linguistic information we are extracting, but our classification mechanism will treat this information independently on the language.

5.3 Data set processing

In the training process, the data set is formed by annotated text documents with marked basic ontological concepts and relations between them. To create this data set, we need to combine the information of our annotation tool and the document text itself.

In the process of unknown instances classification, the data set is not the same for concept and relation extraction. In the case of concept extraction, the data set is formed by a set of unannotated text documents. In the case of relation extraction, it is composed of text documents with marked concept occurrences.

5.3.1 Linguistic processing

During the data set processing, we want to access linguistic information related to words and sentences. To achieve this, we will use some NLP techniques to extract this information. Almost all previously mentioned ontology enrichment methods use some kind of NLP processing. We can divide these approaches into two groups: the shallow parsing methods and the deep parsing methods. The shallow parsing methods extract the sentence constituents (noun groups, verbs, verb groups, etc.), but specify neither their internal structure nor their syntactic dependencies. These methods are used in the papers [11], [14] and [15]. The deep parsing methods extract also the roles and dependencies between sentence constituents. This approach is used in the works [13], [22] and [23].

In our method, we decided to use the deep parsing methods because of their ability to capture the relations between the sentence constituents. We assume that this information can be useful in the concept and relation identification in the text. As we operate with the Czech language, we do not have a wide selection of NLP tools that enable the deep parsing. One of such tools is *Treex* [34]. *Treex* enables all the levels of linguistic processing required by our method - tagging, lemmatisation, morphological analysis and parsing. The biggest disadvantage of this tool is the training corpus for syntactical analysis quietly different from the document instances from the job descriptions domain. This corpus have been trained on the journalist texts that are mostly composed of long, complex sentences, while our domain texts are composed of short and brief sentences often with a missing verb. We assume that the parsing analysis will potentially have worse performance in our method because of this problem. The overview of extracted linguistic attributes is provided in the table 5.1.

Method	Extracted linguistic attributes
Lemmatisation	Word lemma
Morphological analysis	Word part-of-speech
	Word case
	Word gender
Parsing	Word number
	Word dependency head
	Word dependency type

Table 5.1: Extracted data of the linguistic analysis

The output of the linguistic processing does not contain only the values of the linguistic attributes related to the words. Usually, it also provides the output in a semi-structured form that is machine readable. This output reflects the text structure in the means of words and sentences. We can use this output to build a structure providing necessary direct access to particular sentences and words.

5.3.2 Features

Features, or attributes, are the descriptors characterising the candidate phrase(s) in text data. These descriptors will be used in the classification process. The adjacency of a unknown instance to some basic ontological concept or a relation will be revealed according to their values.

The idea to create a descriptor or a set of descriptors that can be used to distinguish important and unimportant parts of text is well known in the domain of ontology learning. The pieces of text are filtered by the number of hit counts obtained from the Google API [15], by thresholding the domain relevance calculated by the frequency of occurrences [22] or by the word phrase adjacency to some semantic pattern [20] [11] [17].

In our method, we go behind this approach and we let our classifier to distinguish between important and unimportant parts of texts in the classification process base on the extracted set of features. We do not define either patterns or rules indicating the relevant phrases in the text, but we use our classifier to create such rules implicitly by deriving them from the learning data set.

Input of the process of the features extraction are the linguistically processed texts and the candidate word phrases of the text. These phrases could be both located manually or deduced by our mechanism in the unknown instance classification process.

Feature definition

We need to convert the extracted information related to a particular word phrase(s) to a classifier readable form. First, we must define what features will be used in our method for the extraction of a particular concept or a relation type.

Intuitively, we divide our features for a word phrase into two groups, *local features* and *global features*. Local features are mostly related to a particular word and represent various linguistic attributes. Global features are related to an entire phrase. They define for example the location of a piece of text in a sentence or text or describe its close neighbourhood.

In the case of relation extraction, the features for a pair of word phrases can be composed by the combination of the first word phrase features and the second word phrase features. Moreover, we could use the features representing the concepts the word phrases are belonging to, because this information is already known in the relation classification process. Another type of features can reflect word phrases textual relationship such as distance etc.

Feature extraction

With the structured output of the linguistic processing, the acquisition of the feature values for word phrases is relatively easy process. We must design this process to be configurable for different classifiers, including the concept and relation classifiers.

5.4 The classifier learning process

With the respect of requirements given by the functions s' , r' and t' , we decided to learn one classifier for each basic ontological concept and one classifier for each relation type, in our case the taxonomic and non-taxonomic relation type.

The input of the learning process will be represented by the extracted features along with the classification label. In the case of concept classifier learning, the features will be extracted from important parts of texts defined in the annotation

process. The assigned label will be the annotated basic ontological concept. In the case of relation classifier learning, the features will be extracted from a pair of word phrases denoted in the annotation process and the assigned label will be the relation between these phrases.

We have to choose a classifier method with good generalisation ability. It is possible to examine several classifiers, but starting with some easily interpretable classifier is preferable because of the possibility to visually check the learned mechanism.

The idea to use a classifier in ontology learning was used before in the related work. In the paper [17], the method that uses a taxonomy as a decision tree was introduced. The instance features in the classification have been defined as the set of semantic descriptions created from the co-occurrent words in some domain related corpus. For each candidate word phrase, the most similar semantic description is found in the taxonomy. This approach however achieves low accuracy and does not provide satisfying results. Its application in the process of semantic annotation is doubtful.

In the paper [20], a classifier is trained on the set of hyponym relations obtained from WordNet that are denoted in the learning data set. The list of all dependency paths between these hyponyms is created and common patterns are extracted. Each pattern is used as one dimension in the feature vector with the value of number of occurrences in the training corpus. This method has better accuracy than the previous, but the lexical database of hyponyms is required before applying the algorithm. The method lacks of configurability, because the patterns are learned from this lexical database and they are not adapted to current domain.

In our method, we use the training data set with annotated relevant pieces of text and relation between them. The annotation is done manually. This approach provides high ability of adaptation to a particular domain.

We decided to use primarily the concept of a decision tree as the core of our classification method beside other machine learning techniques. Decision trees are simple to understand and interpret. They provide us the possibility to check the suitability of the proposed method by inspecting the learned classifier.

5.4.1 Candidate text instances

As we mentioned in the previous chapter, the annotation process denotes only the positive instances of the classification, in our case the occurrences of the basic ontological concepts and relations. In the learning process, we need to specify how to select the negative instances from the text as we do not have the specification of a "non-concept" or a "non-relation".

This problem is closely related to the problem in the unknown instances classification mentioned in 4.5.1, when we have to create a mechanism that will choose the candidate word phrases instances from the text. The ideal state will be when the candidate word phrases selected by this mechanism will cover both the negative and positive instances in the process of classifier learning.

To solve both problems, we have made a base assumption about the important

word phrases representing a base relation concepts in the text:

- the words of one candidate phrases are located within one sentence
- there exists a syntactic dependency between the candidate phrase words

In our learning data set, the first assumption is true in all the cases. The second assumption is correct on near all the cases. The second assumption significantly relies on the correctness of dependency parsing in the linguistic processing phase. We assume that not all annotated training instances will be covered by this method, as some syntactic relations do not have to be correctly recognised by the dependency parsing process.

With these two assumptions, we are able to identify a candidate word phrase in a sentence as a subset of sentence words that linked together by some syntactic dependency. In the process of concept classifier learning, the negative instances for one sentence will be created as the set of sentence candidate phrases minus the set of annotated word phrases in the current sentence.

We can proceed analogically in the process of relation extraction. We can take all pair-wise combinations of concept lexical representations in a particular sentence as the candidate relations. In the process of relation classifier learning, we can take the set of candidates relations minus the set of annotated relations as negatives instances.

5.5 Ontology extraction

In this section, we describe our methodology for different tasks in the problem of ontology enrichment.

5.5.1 Concept extraction

In the process of the concept extraction, the input is formed by the identified candidate word phrases as we showed in the section 5.4.1. For each candidate word phrase and for each concept classifier, a feature values vector is computed. Then each concept classifier computes the adjacency of the particular candidate phrase to the basic ontological concept.

Interpretation of the output

With the classification result, we are able to determine the word phrase adjacency to the set of the basic ontological concepts. Before a new ontological concept could be created from the input word phrase, we have to define its identifier and lexical representation as we mentioned in 4.5.3. The same phrase could be extracted in slightly different forms given by conjugation etc.

It seems that the most simple solution could be to use the lemmas of each word in the phrase for both the identifier and the lexical representation. This approach provides us a possibility to merge the phrase distinct verbal forms into one lexical representation.

Another problem is to deal with the synonym terms represented by one concept. We can use some machine readable dictionary such as WordNet to determine

the word phrases relatedness, but none such high-developed dictionary exists for Czech language. We let this problem to be solved in the future work.

For the same reason we do not solve the problem of attaching a newly-revealed lexical representation to existing concept of an ontology. To attach some lexical representation to an existing concept, the meaning of the phrase is required.

5.5.2 Relation extraction

The input for the relation extraction is formed by the candidate pairs of word phrases as described in 5.4.1. Candidate word phrases have been obtained by the concept extraction process and possibly some kind of semantic annotation. Feature values are computed from these candidate relations and then classification is performed. The positive output of this step indicates that the candidate pair of word phrases represents some ontological relation.

Interpretation of the output

If we integrate the process of the semantic annotation before the process of the taxonomic relation classification, we are able to derive more precise position of new concepts in the conceptual hierarchy.

IS-A Let (w_1, w_2) be a phrase to be classified. If an IS-A relation is identified between the concepts the phrases belong to, four cases could occur:

- w_1 represents the new concept c_1 , w_2 represents the new concept c_2 . The concept c_2 is attached in the conceptual hierarchy as the hyponym of the basic ontological concept c_3 , $c_3 \in s'(c_2)$ and the concept c_1 is attached as the hyponym of the concept c_2
- w_1 represents the new concept c_1 , w_2 represents the existing concept c_2 . If there are not any hyponyms of the concept c_2 , the concept c_1 could be attached as the hyponym of c_2 in the conceptual hierarchy. Otherwise an human intervention is needed, because we do not know the precise location of c_2 in the conceptual hierarchy.
- w_1 represents the existing concept c_1 , w_2 represents the new concept c_2 . If c_1 is attached as the hyponym of the basic ontological concept c_3 , the concept c_2 could be attached as the hypernym of the concept c_1 and the hyponym of c_3 in the conceptual hierarchy. Otherwise the human intervention is needed, because we do not know the precise location of c_2 in the conceptual hierarchy.
- w_1 represents the existing concept c_1 , w_2 represents the existing concept. None action is taken.

In our prototype implementation, the extracted concepts are attached as direct descendants of their hypernyms. A human intervention is not the part of the method and is realised after the extraction with an external tool or by manual editation of the extracted ontological representation.

In the future work, the insertion into conceptual hierarchy without a human intervention should be explored. In our prototype implementation, we do not implement the insertion into the ontology, but we only want to create an ontological representation of the output that could be processed in some way later.

COHYPO If an COHYPO relation is identified, the insertion into the conceptual hierarchy is more simple. Let (w_1, w_2) be the classified phrase. If the location of concept c_1 represented by w_1 is known, the second concept c_2 is placed as the direct hyponym of c_3 , $c_3 \in s(w_1), s'(w_2)$. If both concept locations are unknown, the concepts c_1 and c_2 are placed as the hyponym of a basic ontological concept they are both belonging to.

5.5.3 Instances extraction

The extraction of instances is relatively simple process. It comprises the grouping of previously revealed ontological information from one text document. We want to use the previously extracted concepts and non-taxonomic relations to create meaningful information about the processed documents.

For each processed document, the extracted concepts from this document can be treated as the instances of these ontological concepts. These instances could have other properties defined by the non-taxonomical relations. As an example, the occurrence of a concept Activity represented as "*znalost*" could have linked Level represented as "*dobrá*" in some document instance. We create an ontological instance by connecting these two concepts together for the particular document instance.

5.5.4 Significance and the result information

Previously in 4.5.2, we have introduced the problem of extracted information significance. Now we propose a solution how to express this significance based on the performance of our classifier.

Concepts Lets focus on the extracted concepts first. Concept c is extracted by a classifier from a document d . This classifier has some measured performance expressed as precision p and recall r . If we look on the definition of precision and recall, we can find out that both of these measures have a natural interpretation in the terms of probability [35]. Precision may be defined as the probability that an concept is relevant for d given that it is returned by our classifier. Recall may be defined as the probability that an concept is returned by our classifier given that it is relevant for d :

$$p = P(c \text{ is relevant for } d | c \text{ is returned}) \quad (5.1)$$

$$r = P(c \text{ is returned} | c \text{ is relevant for } d) \quad (5.2)$$

Let D be the set of all document instances the classification is performed on. If some concept c was revealed in n documents from this set, we can estimate the probability $P(c \text{ is returned})$ as:

$$P(c \text{ is returned}) = \frac{n}{|D|} \quad (5.3)$$

Then from the Bayes formula we define the significance of c for d :

$$P(c \text{ is relevant for } d) = \frac{p}{r} \times \frac{n}{|D|} \quad (5.4)$$

Relations Analogically we can proceed for relations. Let t be a relation between two concepts as a result of classification in the document d . The significance of t for d can be expressed as:

$$P(t \text{ is relevant for } d) = \frac{p}{r} \times \frac{n}{|D|} \quad (5.5)$$

We see the relevance of a concept is higher with increasing number of occurrences. From these formulas, we can deduct that if precision is much greater than recall, there is possibly a high number of unrevealed concept occurrences in text data. In this case, the significance of the extracted concept is increasing. In the contrary, if precision is smaller than recall, the number of incorrectly returned concepts is higher and the significance of the extracted concept is decreasing.

5.6 Conclusion

We have decomposed the problem of the ontology enrichment into several sub-tasks we have described above. Now we are able to create a complete picture of our system. We want to divide our mechanism into several cooperating components. These components will include a component for processing the input text documents, a component that will parse the output from the annotation tool, a component that will create the features vector from both known and unknown instances, a classifier learning component, a component providing the classification framework for both relations and concepts and finally a component that will generate a machine readable output.

In the following chapter, we will describe the techniques and method used in our method in more detail.

6. Techniques

In the previous chapter, we have described our approach to the problems ensuing from the ontology enrichment task. This chapter is devoted to the more precise description of the techniques used in our method.

First we describe the techniques used for extraction of the features for both concepts and relations classification. Next we describe the existing concepts of machine learning - decision tree and random forests. Because these concepts are well known, we will provide just brief descriptions. Then the method of ontological output generation from the classification results will be specified in more detail.

6.1 Training process

The process of a classifier training consists of several steps. First, we choose the training and testing data sets, next a proper learning algorithm is applied on the training data set and finally the performance is evaluated on the testing data set. This process is repeated until the performance of the classifier is not longer significantly increasing. We can treat it like an optimisation problem in a multidimensional feature space.

Instances in the testing and training data set need to have specified the classification classes. These are provided in the process of text documents annotation. Annotation of text documents has to be done manually and is a very time-consuming process. Unluckily, we were not able to provide a sufficient amount of annotated text documents in the scope of this thesis, so our data sets are relatively small in comparing to standard data sets.

6.2 Feature extraction

As we have mentioned in the previous chapter, we are using a NLP tool to extract linguistic information from the text documents. Text documents are split into sentences and words during the NLP processing and their linguistic attributes are converted into a machine readable form. Linguistically processed text documents along with the original text documents are the input of the feature extraction process. In the process of training data set creation, this information is combined with the annotation information from the text documents.

The feature extraction process for one text document includes several phases. First, the candidate text phrases are extracted from the document sentences. Then the feature values are computed and transformed into a classifier-readable form and a classifier unknown instance is created. In the learning process, a class label is attached to these instances.

6.2.1 Extraction of candidate phrases

In the section 5.4.1, we have made two basic assumptions about the candidate phrases. We assume that one candidate phrase is located in one sentence. Moreover, we assume that there exists a syntactic relationship between the phrase

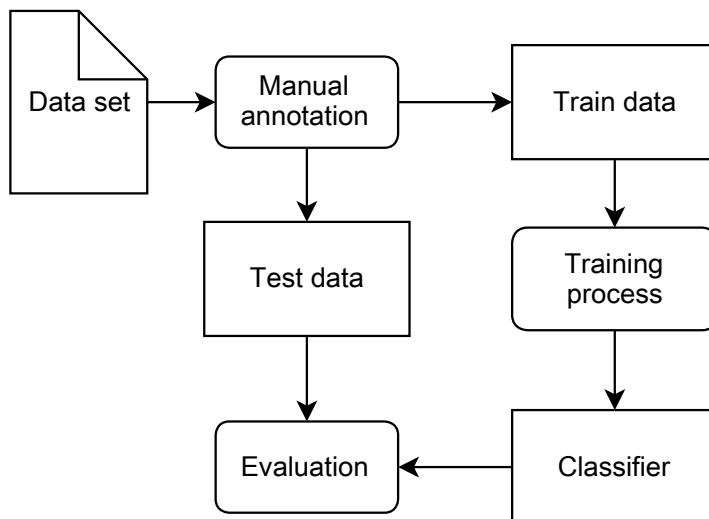


Figure 6.1: Workflow of the training process

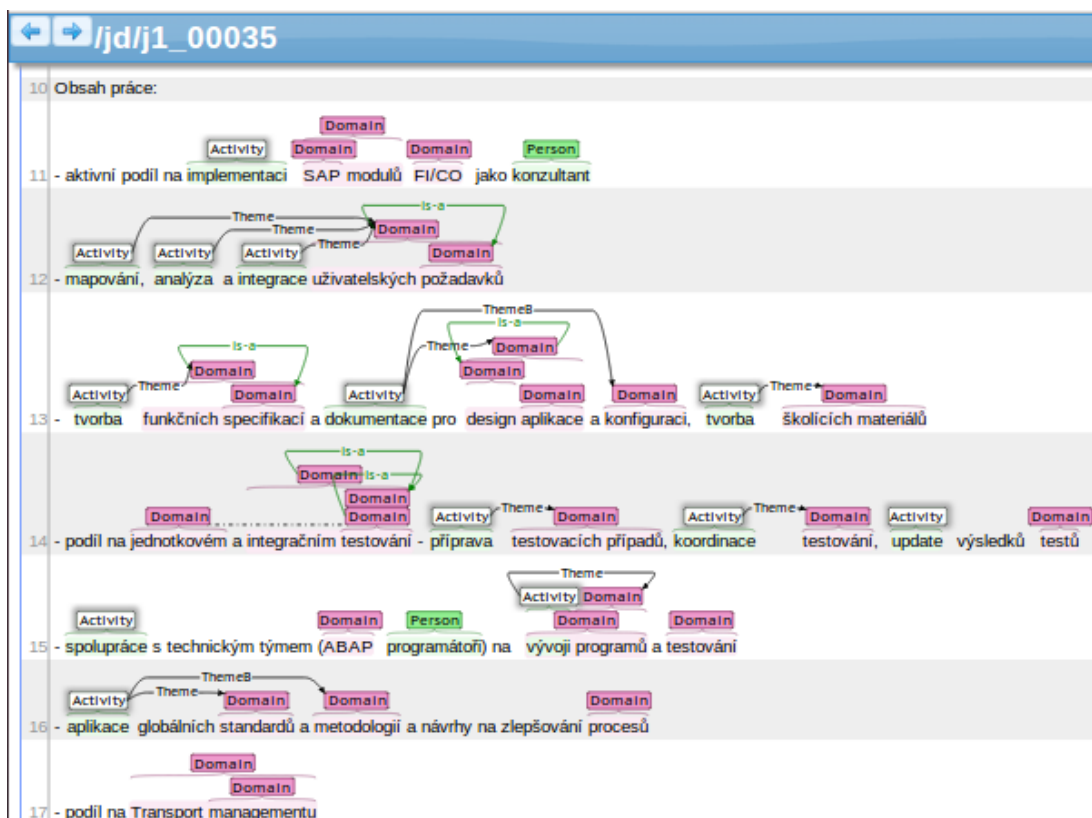


Figure 6.2: Annotation in the Brat tool

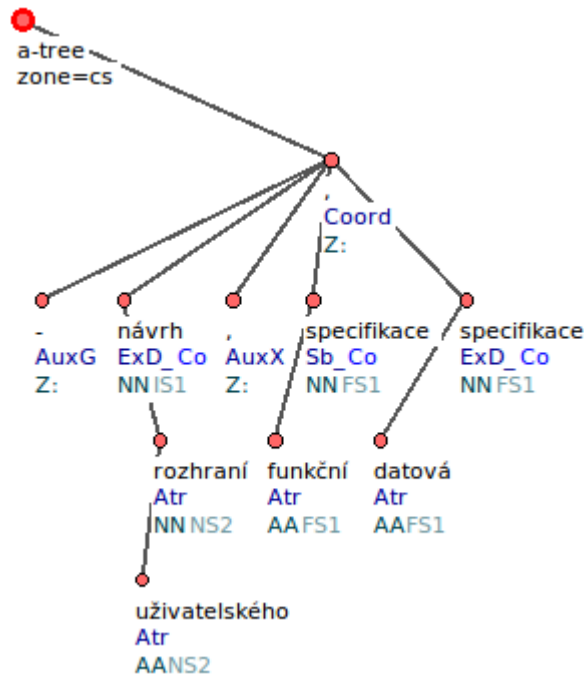


Figure 6.3: Syntax tree example

words. We use the NLP tool Treex to determine these relationships in a sentence.

The syntactic relationships of a sentence can be represented by a *syntax tree*. A syntax tree is an oriented, rooted tree. Each node in this tree except the root represents one morphological token of a sentence. The root node is a node with no linguistic representation and it has exactly one child node. Each oriented edge in this tree represents a syntax relationship between the nodes and has attached a label that defines the type of the relationship.

To acquire a set of phrases with words connected by syntactic relationships, we should take all connected subtrees of a dependency tree. However, we have made an observation of the annotated word phrases in our training data set. We have realised that the phrases in the documents of job description domain have usually short form without complicated syntax relationships. Thus we have decided to use only the subset of the set of connected subtrees of a syntax tree, concretely all descending paths. This reduces the set of candidate phrases to a set of descending strings from one node to another node in the dependency tree. If the annotated phrases had more complicated syntax structure, we would choose another approach.

The algorithm to create the set of all descending paths of the syntax tree is inductive. In the first step, we create the paths of length 0 by adding all nodes except the root to the set. In the i -th step of the algorithm, we take all paths of length of $i - 1$ and prolong them by one by adding a parent node of the path top level node, if it exists. We are finished if no path could be prolonged in a particular step. We define a *syntax string* as a syntax tree path obtained by this algorithm. The pseudocode of the algorithm is presented in Algorithm 1.

Algorithm 1 Candidate word phrases algorithm

```
function CREATECANDIDATEWORDPHRASES( $T$ )
   $C \leftarrow$  AddAllNodesExceptRoot( $T$ )
  AddAllPathsWithLength( $C, T, 1$ )
  return  $C$ 

function ADDALLPATHSWITHLENGTH( $C, T, pathLength$ )
   $P \leftarrow$  GetAllPathsWithLength( $T, pathLength - 1$ )
  if CouldBeProlonged( $P$ ) then
    for all  $p$  in  $P$  do
       $p' \leftarrow$  ProlongPath( $p$ )
      AddPath( $C, p'$ )
    AddAllPathsWithLength( $C, T, pathLength$ )
  else
    return

function PROLONGPATH( $path$ )
   $n \leftarrow$  FindTopLevelNode( $path$ )
   $newPath \leftarrow$  CreatePath( $n.parent, path$ )
  return  $newPath$ 
```

6.2.2 Extraction of candidate relations

Candidate relations are simply created as pair-wise combinations of two extracted concepts from one sentence. We distinguish the candidate relation for the taxonomic classifier and the relation for the non-taxonomic classifier. In first case, both word phrases in the relation are belonging to the same base ontological concept. In second case, no restriction is made to the base concepts of the phrases.

6.2.3 Computation of concept feature values

Each candidate phrase has assigned a vector representing the computed feature values. The vector contains the values of both local and global features. Each feature value could be either a real numbers or an enumeration value. Not all features are used in every classifier, for different concept or relations, different features could be relevant.

Global features

Global features are related to the entire phrase. We decided to use following global features:

Text relative position represents the relative position of the phrase in the sentence:

$$TextRelativePosition = \frac{W_{ch}}{T_{ch}} \quad (6.1)$$

where W_{ch} is the index of the first character in the word phrase and T_{ch} the number of characters in the text.

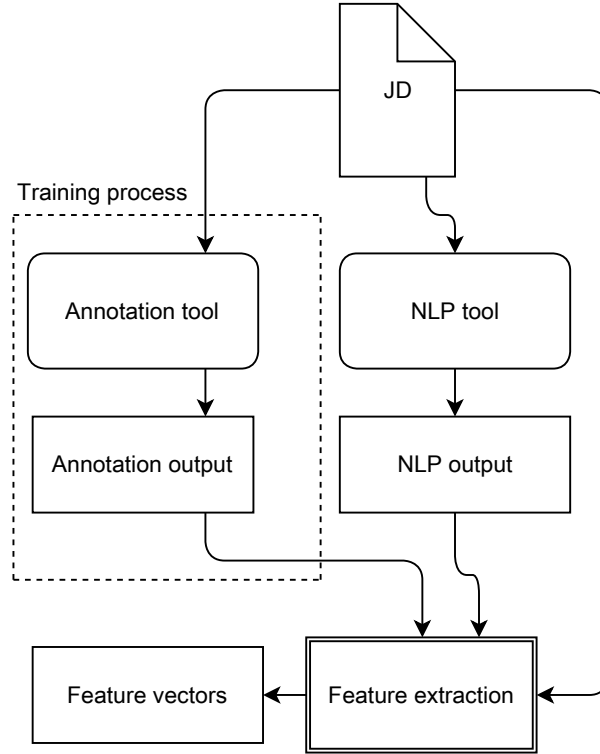


Figure 6.4: Input of the feature extraction process

Sentence relative position represents the relative position of the phrase in the sentence:

$$SentenceRelativePosition = \frac{W_s}{S_l} \quad (6.2)$$

where W_s is the phrase's first word index in sentence and S_l is the number of words in the sentence.

Sentence length represents the number of words in the word phrase sentence.

$$SentenceLength = S_l \quad (6.3)$$

Word count is the number of words of a candidate word phrase W_l :

$$WordCount = W_l \quad (6.4)$$

Paragraph number is the index of a paragraph of a candidate word phrase P_i :

$$ParagraphNumber = P_i \quad (6.5)$$

Part of list is a boolean attribute that is true for a candidate word phrase is a part of an *enumeration* - list of phrases separated by a some separator.

Separator before is a boolean attribute that is true if a separator precedes a candidate word phrase

Separator after is a boolean attribute that is true if a separator follows a candidate word phrase

Sentence ratio is the ratio of the phrase word count and the sentence word count.

$$SentenceRatio = \frac{W_l}{S_l} \quad (6.6)$$

We have observed that the location of some annotation types is correlated in the annotated text documents. For example, in the jobs description domain, a word phrase labelled with Activity is often located near the word phrase labelled with Domain. We decide to capture this fact and we have created a special global feature applied for some concepts that represents the distance between this phrases. We have defined the following global feature:

Correlated word phrase distance represents the distance of the defined correlated word phrases. The distance is computed as the length of the descendent syntax tree path between the nodes associated with phrases head words:

$$Distance = P_1 - P_2 \quad (6.7)$$

where P_1 is the path length from the root of syntax tree to the word phrase head and P_2 is the path length from the root of syntax tree to the correlated word phrase head. If no descended syntax tree path exists between the phrases, the distance is equal to -1 .

To determine the value of this feature in the process unknown instance classification, we must first make the classification of the correlated base concept and use the extracted instances in the process of feature creation.

Local features

Local features are related to a particular word. Because each word phrase could have different words count, the number of actually computed word phrase local features is not fixed. We define *an empty value* as the feature value when a feature is not applicable to current word because of the phrase length. We use this empty value also for all linguistic attributes that are not defined for a current word.

Before the computation of the feature values, we define *the word phrase head* as the top-level word in the syntax path that other words are directly or indirectly dependent to. We sort the words of the phrase in the way that reflects the passage of the syntax path from top to down and the head as the first word in the sorted words.

We decided to use following local features of the word phrase head:

Head part-of-speech is the grammatical part-of-speech of the head word.

Head gender is the grammatical gender of the head word.

Head case is the grammatical case of the head word.

Head number is the grammatical number of the head word.

Head dependency type is the syntax dependency type attached to the edge leading to the word.

Head proper noun represents the fact the head word is capitalised and it is not the start of sentence. We do not apply more intelligent proper noun recognition.

Head dependent word part-of-speech is a part-of-speech of the word the head is syntactically dependent to.

Head dependent word dependency type is a dependency type of the word the head is syntactically dependent to.

Analogically we define the local features for the last word in the syntax dependency string:

Last word part-of-speech is the grammatical part-of-speech of the word.

Last word dependency type is the syntax dependency type attached to the edge leading to the word.

Last word case is the grammatical case of the word.

For each other word of the syntax string we define the following local features:

Word i part-of-speech is the grammatical part-of-speech of the word with index i .

Word i dependency type is the syntax dependency type attached to the edge leading to the word with index i .

Word i word case is the grammatical case of the word with index i .

6.2.4 Computation of relation feature values

Each pair of word phrases is assigned a vector representing the computed feature values. Most of relation features are the same as the concept features, but in this case each feature is used twice - for the first word phrase of the relation and for the second word phrase of the relation.

Global features

We use the following global features:

Phrases distance is the textual distance of the relation phrases represented in number of words:

$$PhrasesDistance = W_{1_s} - W_{2_s} \quad (6.8)$$

where W_{1_s} is first word index in sentence of the first phrase and W_{2_s} is first word index in sentence of the second phrase.

Same entities is the boolean attribute that is true for word phrases belonging to the same base concept.

First word phrase entity is the base concept the first phrase belongs to.

Second word phrase entity is the base concept the second phrase belongs to.

Beside this global features, we use all the global features mentioned in 6.2.3 applied for the first and the second word phrase of the relation.

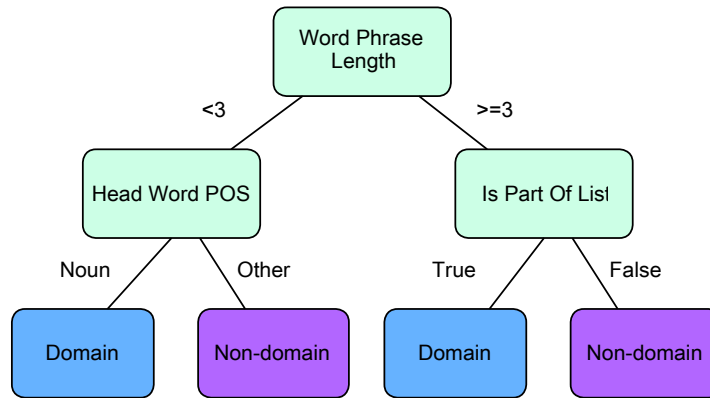


Figure 6.5: Example of a simple decision tree for Domain concept

Local features

Local features are related to the particular words of each relation phrase. We compute this features as the same way as mentioned in 6.2.3 for the first and the second word phrase of the relation.

6.3 Machine learning

We have decided to compare the performance of several classifiers, to decide which one is best suitable for our method. We have decided to start with the well-known concept of a decision tree [27] in our classification method because of their great generalisation capabilities and easy interpretability. Then we tried the random forest classifier and support vector machines.

6.3.1 Decision tree

A decision tree classifies instances by sorting them down the tree from the root to a leaf node, which determines the classification of the instance [27]. Suppose that we have a set of features, respectively attributes, F extracted from the word phrase w . Next suppose that we have a classification class $c \in C$. A decision tree for the classification class c is an oriented tree where each node except leaves have at least two successors. Each node except the leaves has assigned a feature $f \in F$ and each edge leading from this node has assigned a subset of the possible values of the feature f . The subsets assigned to the edges leading from one node form the disjoint subsets of the possible values of the node feature. Each leaf has assigned a boolean value *True* or *False*. This value represents the assignment or non-assignment of the classification class c of an unknown instance.

The assignment of the classification value for an unknown instance is computed by the controlled traversal of the decision tree from the root to the leaf. Suppose we have an unknown instance with the set of features F . In a particular node with assigned attribute f , the edge of traversal is decided according the value of f of the unknown instance. The traversal will then inductively continue from the node the edge is leading to.

Decision tree learning

There exist several decision tree learning algorithms, such as ID3, C4.5 and ASSISTANT [27]. We decided to use the algorithm C4.5 [37], which is an improvement of the ID3 algorithm [36]. Both ID3 and C4.5 build a decision tree from a set of training data using the concept of *information entropy* [27]. C4.5 adds several improvements to ID3. C4.5 is able to handle both discontinuous and continuous attributes, handle missing attributes values and adds pruning after the decision tree creation.

The learning algorithm for both ID3 and C4.5 algorithms is based on selecting which attribute to test at each node in the tree. This choice is determined by the attribute *information gain*, which determines how well a given attribute splits the learning data set [27].

Before we introduce the term of information gain, we define entropy as the measure of homogeneity of examples. Let S be the set of learn instances with positive and negative instances of some classification class $c \in C$.

Entropy is defined as

$$Entropy(S) = -p_+ \log_2 p_+ - p_- \log_2 p_- \quad (6.9)$$

where p_+ is the proportion of positive instances and p_- is the proportion of negative instances in the learn data set S . Notice the entropy is 0 if S contains only positive or only negative instances and entropy is 1 if the number of positive instances is equal to number of negative instances.

Information gain is the measure of effectiveness in classifying instances of an attribute. For an attribute $f \in F$ and learn data set S it is defined as

$$Gain(S, f) = Entropy(S) - \sum_{v \in Values(f)} \frac{|S_v|}{|S|} Entropy(S_v) \quad (6.10)$$

where $Values(f)$ is the set of all possible values for attribute f , and S_v is the subset of S for which attribute f has value v [27].

In the learning algorithm, the attribute for the current decision tree node is selected based on its information gain measure. The attribute with highest information gain provides the best prediction of the classification class.

The C4.5 algorithm has a divide and conquer strategy without the backtracking. During the learning algorithm, each node in the tree is associated with a set of instances. At the beginning, the root node is associated with the entire learn data set. In one step of the algorithm, one node of the decision tree is created. Let S be the set of instances associated to current node. If these instances are all positives or all negatives, a leaf is created with the assigned classification label according to this instances. Else the attribute with the highest information gain is selected for the test attribute of the current node. If the attribute is discrete, the assigned data set is split into subset of S , which share the same attribute value. If the attribute is continuous, the assigned data set is split into two subsets according to some attribute value threshold, which was determined in the information gain calculation. Then the divide and conquer approach is applied on the node successors with assigned subsets of S . The pseudocode of this algorithm is presented in Algorithm 2.

Algorithm 2 Basic C4.5 algorithm

```
function CREATETREE( $T$ )
   $f \leftarrow$  ComputeClassLabelFrequency( $T$ )
  if IsOneClass( $f$ ) then
     $newClass \leftarrow$  DetermineClass( $T$ )
    return NewLeaf( $newClass$ )
  else
     $N \leftarrow$  CreateNewNode
     $N.attr \leftarrow$  FindAttributeWithHighestGain( $T.attributes$ )
    if IsContinuousAttribute( $N.attr$ ) then
       $N.thres \leftarrow$  FindBestThreshold( $N.attr, T$ )
    for all  $value$  in  $N.attr.values$  do
       $T' \leftarrow$  SplitDataSet( $T, N.attr, value$ )
      AddChildren( $N, CreateTree(T')$ )
  return  $N$ 
```

Pruning

Pruning is one of the C4.5 improvements to the ID3 [37]. After the tree is created, the tree is traversed again and unhelpful branches are removed and replaced by a leaf with the dominating class. This improvement is based on the “Occam’s razor” - simpler trees are often better trees. If the pruning is applied in C4.5, one part of the training instances is left for the prune cross-validation to verify the accuracy of the pruned tree. Then the impact of the pruning is evaluated for each node and the performance is tested on the validation set. This type of pruning is called *reduced error pruning*. Pruning will always cause worse performance of the tree on the learning data set, but usually pruned tree has better performance on the test data set.

6.3.2 Random forests

Random forests is one of the ensemble methods for classification based on the decision trees. These methods generate many classifiers and aggregate their results. A random forest is a classifier constituted of a set of decision tree classifiers. In the classification process, the predicted class is simply computed as the majority of the predicted classes by individual trees.

Random tree induction

The induction algorithm of random forests has been proposed by Breilman [38]. Each decision tree in the set is independently trained using a random sample of the data set with replacement (“*bootstrapping*”) with the size equal to the size of the entire data set. Opposite standard decision trees, where each node is constructed using the best predictive attribute, in a random forest the attribute is chosen as the best attribute of the subset of randomly chosen attributes. After a tree is learned, no post-pruning is applied. In the induction process of a random forest, only two parameters are needed. First parameter is the number of generated trees N , the second parameter is the size of the random attribute subset in the node

creation process m . The pseudocode of this algorithm is presented in Algorithm 3.

Algorithm 3 Random forest induction algorithm

```

function CREATERANDOMFOREST( $T, m, N$ )
   $C \leftarrow$  NewArray
  for  $doi = 1$  to  $N$ 
     $T_i \leftarrow$  RandomSampleOfTrainingData( $T$ )
     $C_i \leftarrow$  CreateTree( $T_i, m$ )
     $C[i] \leftarrow C_i$ 
  return  $C$ 

function CREATETREE( $T, m$ )
   $f \leftarrow$  ComputeClassLabelFrequency( $T$ )
  if IsOneClass( $f$ ) then
     $newClass \leftarrow$  DetermineClass( $T$ ) return NewLeaf( $newClass$ )
  else
     $N \leftarrow$  CreateNewNode
     $A \leftarrow$  RandomMAttributes( $T.attributes, m$ )
     $N.attr \leftarrow$  FindAttributeWithHighestGain( $A$ )
    if IsContinunousAttribute( $N.attr$ ) then
       $N.thres \leftarrow$  FindBestThreshold( $N.attr, T$ )
    for all  $value$  in  $N.attr.values$  do
       $T' \leftarrow$  SplitDataSet( $T, N.attr, value$ )
      AddChildren( $N, CreateTree(T')$ )
  return  $N$ 

```

Random forests properties

Random forest is known to be one of the most efficient classification methods. Random forest are known as robust to overfitting, see 6.3.3. The main disadvantage of this method, and other ensemble algorithms, is the lack of interpretation.

During the induction of a particular tree of the random forest, the training data set is constructed by bootstrap replication. There is on average one third ($\frac{1}{e}$) of instances that have not been used used in the induction process for this particular tree [39]. These omitted instances are known as *out-of-bag instances*. Each instance has been omitted in the induction for about one third of trees. This instances could be used to determine the error rate of the random forest in the following way: Each instance is classified by the set of trees, for which it is an out-of-bag instance. Let c be its mostly predicted class. The proportion of times that c is not equal to the true class of the instance averaged over all cases is the *out-of-bag error estimate*.

The number of the generated trees is an important parameter of the algorithm. If this number is too small, some instances in the data set can be used in the prediction only one or even any times and the classification accuracy is decreased. On the other hand, high number of induced trees increase the computational complexity for both learning process and unknown instance classification. Thus this parameter must be chosen precisely with respect to the data set size.

6.3.3 Problems in the machine learning

The process of a classifier training brings several problems we describe in the following sections.

Imbalanced data set

The class distribution in the training and testing data sets usually does not have uniform distribution. We assume that in our case, unimportant parts of text are significantly dominating over the important parts of text. This fact implies a high degree of class imbalance, when there are many more instances of one class than others.

Standard classifiers often cause the problems on the unbalanced data set. To prevent this, two methods are used in the machine learning that alters the training balance: artificial up-sampling the minority class or artificial down-sampling the majority class [40].

After some experiments with learned classifiers, we decided to not to use the methods altering the training balance. In our case, majority of the negatives samples is filtered in the early steps of the classification algorithm and it seems that the classifier performances are not negatively influenced by the imbalance.

Overtraining

Overtraining, also known as overfitting, is a commonly known problem in the area of machine learning [32]. It occurs when the learned representation of training data describes more the noise instead of the underlying rules. Such classifier has great performance on the training data, but poor performance on the test data. There are several ways how to avoid this problem and different classifiers handle this problem in different way. Random forests are generally known as robust to overfitting because of their strategy that chooses the best attribute from a random subset of attributes[38]. Post-pruning in a decision tree makes the tree structure simpler and removes the branches that can lead to overfitting.

6.4 Ontological elements extraction

Next described technique is the creation of the ontological elements from the classification output. With the set of classifiers created by the machine learning techniques, we are able to recognise the important pieces of texts that represents ontological concepts along with relations between them.

6.4.1 Ontological concepts and hierarchy

In the classification process, we have recognised the pieces of text representing ontological concepts. Next task is to extract ontological concepts and find their position in the conceptual hierarchy.

First, we must create a concept identifier and assign it a lexical representation. For each extracted piece of text, we create a textual string containing the lemmatised and normalised word phrase form. Lemmatisation creates the lemma form of all words in the phrase. Normalisation capitalises the first character of the

string and remove the unnecessary elements in the lemmas such as explanations of the context. This string is then used as both the concept identifier and the lexical representation. Each concept has attached also some kind of metadata, as the relevance computed from the number of occurrences in the document text and precision and recall of the classifier.

Next we proceed to creation of the conceptual hierarchy. We take the IS-A relation as the primary relation between the word phrases and the COHYPO relation as the secondary. We will use an extracted COHYPO relation only if it adds a new ontological fact and we will create a node that is not yet the part of the hierarchy. This is important because the extracted hierarchical information may not be valid. Invalid extracted relations may create unnecessary cycles in the conceptual hierarchy. Although the cycles are allowed, we want to avoid them as possible.

First, we take all found IS-A relations and we assign to each concept its extracted hypernym concepts. If no extracted hypernym exists for a particular concept, we assign to it the base ontological concepts detected in the classification as the hypernym concepts. The IS-A relation extraction process is denoted in the picture 6.6.

Then we take all extracted COHYPO relations. If one concept of a relation does not have assigned the hypernym concept of the hierarchy, we assign it the hypernym of second concept.

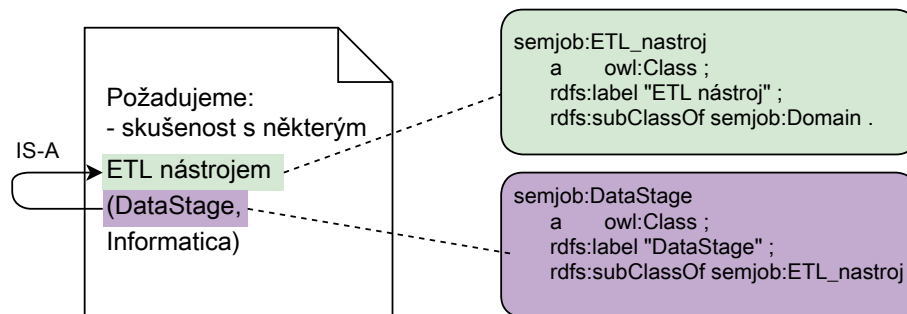


Figure 6.6: IS-A relation extraction

6.4.2 Ontological instances

We use all identified word phrases belonging to some concept as the ontological instances. Each instance has attached a concrete textual representation and the metadata such as the position in the text. Beside this, we take the extracted non-taxonomical relations and used them to link the instances together. The following process is denoted in the picture 6.7.

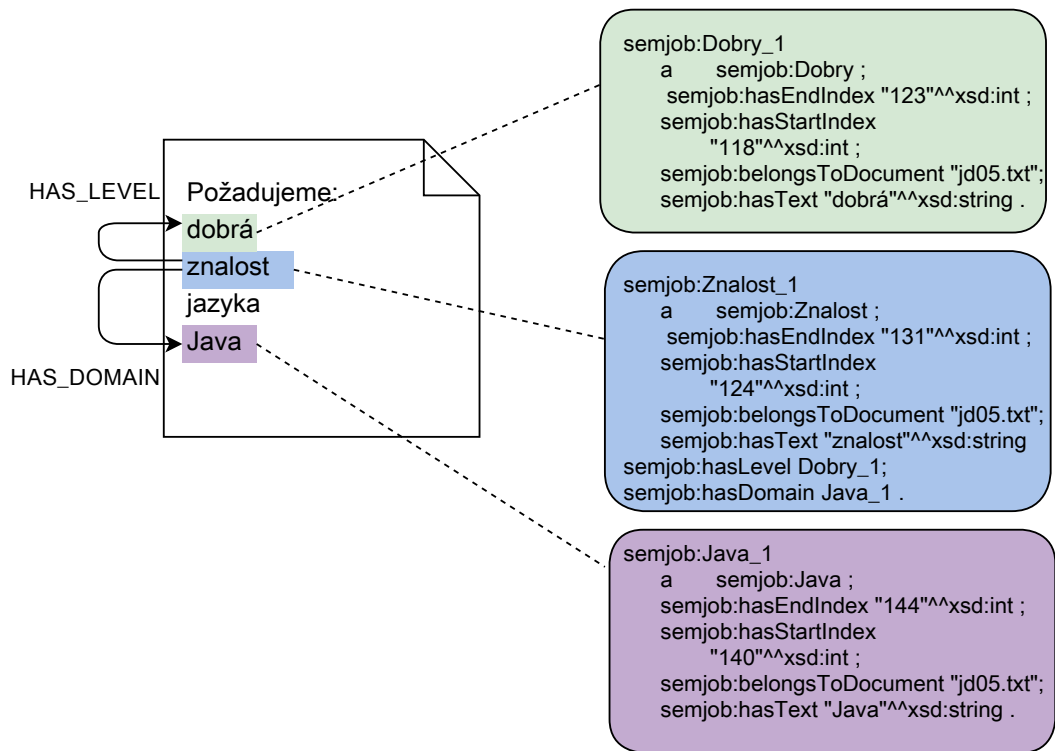


Figure 6.7: Ontological instances extraction

7. Implementation

In this chapter, we proceed to the details of the implementation of our method. The method is implemented in the Java programming language because of its platform-portability. We expect our method could be an extension of a semantic annotation tool and possibly became part of some client/server application.

We have used the Spring framework as the infrastructural support of the prototype implementation. Implementation of most of executive classes is encapsulated with an interface and we use the Spring to link the implementation to the proper interface. We also use the Spring to configure our implementation. In Spring configuration file, we define several parameters, such as concepts and relations names used in our method. For further details see the Programmer documentation on the attached DVD.

The application is divided into several cooperating modules: a linguistic processing module, a document processing module, a learning module and a module that performs the classification and processes the results.

7.1 Annotation

As we mentioned in the previous chapter, we use the Brat annotation tool to denote the interesting parts of texts and their relationships. The annotation process brings several problems.

First, we have to create a configuration file for this tool, where we define the annotation elements. The list of possible annotation types in Brat includes *entities*, *relations*, *events* and *attributes*. An entity is the annotation type for some span of texts, a relation denotes a binary relationship between two annotations. An event represents an n-ary association between the entities or other events. And finally an attribute is a multivalued flag for other annotation.

```
[entities]
JDEntity
    Feature
    Domain
    Level
    Quality
    Person

[relations]
# taxonomic relationship
is-a    Arg1:Domain, Arg2:Domain
cohypo  Arg1:Domain, Arg2:Domain, <REL-TYPE>:symmetric-transitive

[events]
Activity    Theme:Feature|Domain|Quality|Activity, Level:Level
```

Part of the Brat configuration file

We have created an annotation entity for each basic ontological concept except the Activity, which is represented by an event. In this way we are capable to denote the multinary relationships between the Activity, Domain(s) and Level. We use the relations to denote the other taxonomic and non-taxonomic relationships.

The manual annotation process is very time consuming. To minimise the error rate of the learning process, all the occurrences of a particular piece of text must be marked in the input text. Also it is not always evident how to annotate the text. For more complicated phrases, it is not easy to identify their span and relations. Sometimes the phrases are incomplete and with grammar and syntax errors. The annotator has to cope with these problems.

7.2 Linguistic processing module

This module is responsible for the processing of the Treex NLP tool output. We use the CoNLLX format as the format of this output. Each input text document has assigned one file with the linguistic processing results. In this file, each sentence token is on one line and the sentences are separated by an empty line. For each token, following fields are defined:

lemma contains the lemmatised form of the token

part-of-speech tag is the output of the POS tagger

head is the dependency head of current token -the parent of the token in the syntax tree

dependency relation is the dependency relation to the head

```

1 Naším můj^(přivlast.) PSZS7-P1----- _ 2 Atr
2 klientem klient _ NNMS7-----A---- _ 3 Pnom
3 je být _ VB-S---3P-AA--- _ 0 Pred
4 česká český _ AAFS1----1A---- _ 7 Atr
5 konzultační konzultační _ AAFS1----1A---- _ 7 Atr
6 IT it,t^(angl._to) _ PPNSX--3----- _ 7 Atr
7 společnost společnost^(*3ý) _ NNFS1-----A---- _ 3 Sb
8 působící působící^(*3it) _ AGFS1-----A---- _ 7 Atr
9 na na-1 _ RR--6----- _ 8 AuxP
10 mezinárodních mezinárodní _ AAIP6----1A---- _ 11 Atr
11 projektech projekt _ NNIP6-----A---- _ 9 Adv
12 . . _ Z:----- _ 0 AuxK

```

Example of a Treex processing output in the CoNLLX format

Linguistic processing output is read for each input text document and the linguistic attribute values are parsed from the CoNLLX fields. They are saved in a form that will be used in further processing and enables to access the document structure with sentences and words and the linguistic attributes.

Most linguistic attributes, such as gender, number, part-of-speech etc. have values defined by an enumeration. In the implementation, we must state these

values specifically for a particular language. We decided to use the representation of each attribute as an abstract class. A specific attribute along with the list of values of a particular language is defined in the implementation of the abstract class. A particular linguistic attribute value is then created by a factory for a each CoNLLX field. This value can be simple or structured. For example, the head field contains only the id of the dependency relation head, but the part-of-speech tag is formed by ten sub-fields.

7.3 Document processing module

This module is responsible for multiple tasks. First, it connects the linguistic information with the textual information and creates a complex structure information about a input text. This structure captures the sentence and words form, the values of linguistic attributes along with the syntax dependencies between the words. It also adds the positional information to each word and sentence - the start and end indexes.

T3	Domain	272	276	J2EE
T4	Person	241	250	kandidáta
T5	Level	230	240	seniorního
R1	has-a	Arg1:T4	Arg2:T5	
R2	has-a	Arg1:T2	Arg2:T3	
T6	Domain	410	417	systemu
T7	Domain	397	417	informačního systému
T8	Domain	384	396	architektury
T9	Domain	437	457	obchodních požadavků
T10	Activity	429	436	analýzy
E1	Activity:T10	Theme:T9		
T11	Activity	378	383	návrh
E2	Activity:T11	Theme:T12		
T12	Domain	384	417	architektury informačního systému
R3	is-a	Arg1:T12	Arg2:T8	

Example of an annotation output

Next task is to read the annotation output, which defines the important pieces of texts and relations between them. This annotation information is then applied on the previously described document structure and annotated word phrases and relations are created.

The last task is to create a set of candidate set of phrases and relations according to the syntax dependencies relations from the document structure. This set is the input of the feature extraction process described in the following section.

7.4 Learning module

Learning module has two main functions: to create a set of instances represented by the feature vector and to perform the proper machine learning algorithm and

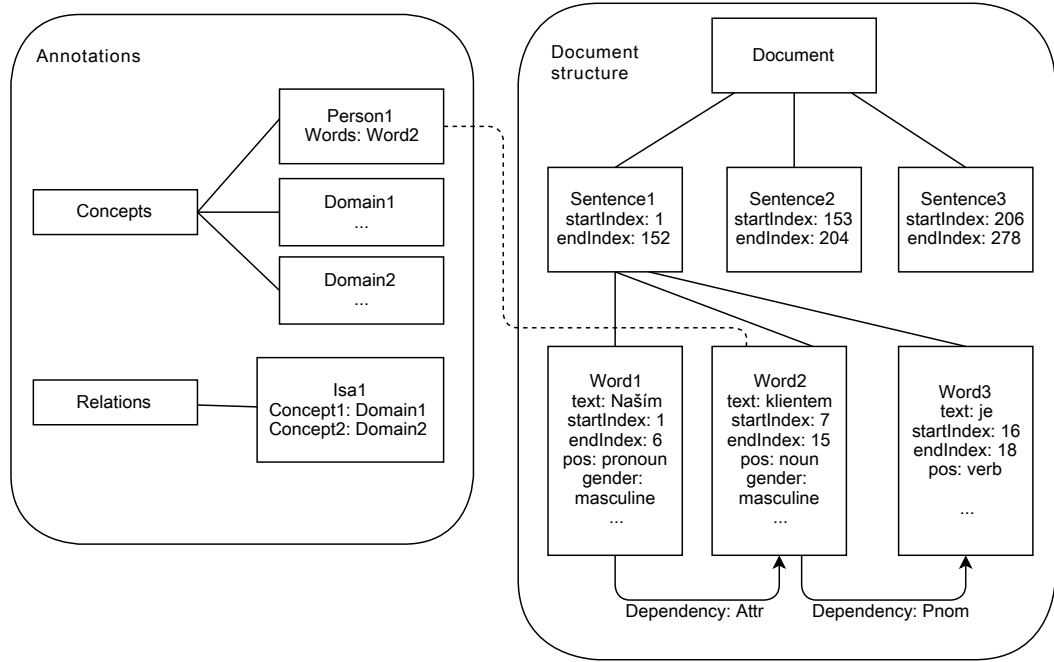


Figure 7.1: Annotations and document structure

performance evaluation. The input of this module is the set of annotated word phrases and relations and the output is the set of classifiers.

7.4.1 Feature extraction

We have defined the features of candidate relation and concepts in the previous chapter. In the learning module, for each candidate concept or relation the feature values are computed and a classifier candidate instance is created. We distinguish two types of candidate instances. First one is the *labelled instance* that has attached a label - classification class defined in the annotation process that can be positive or negative. This type of instances is exported into a data set and used in the classifier learning process. Second is the *unlabelled instance* that has no label and it is used in the unknown instance classification.

As we mentioned before, we create one classifier for each basic ontological concept and one classifier for each relation type - taxonomic and non-taxonomic. Within these classifiers, different features may be used. However, they share a majority of the features. We have decided to create a class responsible for extraction of the most common features, called the *basic instance creator*. Then we have made a couple of classes specialised to extraction of specific features called *specific instance creators*. Specific instance creators inherit from the basic instance creator. Each basic ontological concept or relation type has assigned a specific instance creator, which can reuse the features of a base instance creator or define new, more specific features.

7.4.2 Classifier learning

We have trained two classifiers for each learning case - a basic ontological concept and relation type - a decision tree and a random forest. We have also experiment-

	Activity	Domain	Feature	Level	Person	Quality
Text relative position	✓	✓		✓	✓	
Sentence relative position	✓	✓		✓	✓	✓
Sentence length	✓	✓	✓	✓	✓	✓
Word count	✓	✓	✓	✓	✓	✓
Paragraph number	✓	✓			✓	✓
Part of list	✓	✓	✓		✓	✓
Sentence ratio			✓			
Correlated word phrase distance		✓				
Separator before		✓				
Separator after		✓				
Head part-of-speech	✓	✓	✓	✓	✓	✓
Head gender	✓	✓	✓	✓	✓	✓
Head case	✓	✓	✓	✓	✓	✓
Head number	✓	✓		✓	✓	✓
Head dependency type	✓	✓	✓	✓	✓	✓
Head proper noun	✓	✓			✓	✓
Head dependent word part-of-speech	✓	✓		✓	✓	✓
Head dependent word dependency type	✓	✓		✓	✓	✓
Last word part-of-speech	✓	✓	✓	✓	✓	✓
Last word dependency type	✓	✓		✓	✓	✓
Last word case	✓	✓		✓	✓	✓
Word i part-of-speech	✓	✓	✓	✓	✓	✓
Word i dependency type	✓	✓	✓	✓	✓	✓
Word i word case	✓	✓	✓	✓	✓	✓

Table 7.1: Overview of features for concept classifiers

ed with support vector machines, but they have not proved satisfying results. We decided to use the well-known Weka[41] implementation of these classifiers.

Weka also adds an interesting feature to some of its classifier implementations. The train instances could have attached a *weight* that reflects their importance. This weight will be used in the learning process of a particular classifier and

	Taxonomic	Non-taxonomic
Phrases distance	✓	
Same entities	✓	✓
First word phrase entity		✓
Second word phrase entity		✓
Phrase text relative position	✓	✓
Phrase sentence relative position	✓	✓
Phrase sentence length	✓	✓
Phrase word count	✓	✓
Phrase paragraph number	✓	✓
Phrase part of list	✓	✓
Phrase sentence ratio	✓	
Phrase correlated word phrase distance	✓	✓
Phrase separator before	✓	✓
Phrase separator after	✓	✓
Phrase head part-of-speech	✓	✓
Phrase head gender	✓	✓
Phrase head case	✓	✓
Phrase head number	✓	✓
Phrase head dependency type	✓	✓
Phrase head proper noun	✓	✓
Phrase head dependent word part-of-speech	✓	✓
Phrase head dependent word dependency type	✓	✓
Phrase last word part-of-speech	✓	✓
Phrase last word dependency type	✓	✓
Phrase last word case	✓	✓
Phrase word i part-of-speech	✓	✓
Phrase word i dependency type	✓	✓
Phrase word i word case	✓	✓

Table 7.2: Overview of features for relation classifiers

influence the training output. We decided to use this feature and we increase the weight of the positive instances and decrease the weight of the negative instances. We hope this will increase the performance of our method.

Each learning case has created a data set containing the instances feature vectors in the ARFF format [42]. The two classifiers share this data set.

Decision tree

The implementation of the C4.5 algorithm in Weka is known as J48. This implementation enables to use several parameters modifying the learning process. First, we can use a pruned or an unpruned tree. We have decided to use the error reduced pruning as described in the section 6.3.1. We have set the size of the cross-validation data set to 10% of the entire train data set.

Another parameter of the J48 learning algorithm is the form of the decision tree. A decision tree can be either binary or multinary. If it is multinary, an enumeration node has multiple children, one for each possible value of the attached attribute. In the second case, an enumeration node has two children - one for the "primary" branch and second for the "else" branch. A numeric node has always two children. This parameter has impact on the form of the decision tree and also its accuracy. It seems that the binary form is more suitable for some classifiers and for another it is the multinary form.

J48 algorithm can handle the instance weights. They influence the calculation of the class probabilities during a decision tree node creation.

Random forest

The random forest training does not need almost any parameters. We only set the number of generated random trees. We have found this number by an empirical observation. Although the higher number of tree usually makes better prediction, it also increase the computational resources needed. The number of 30 trees seems to be the good compromise for us.

Random forest can also handle the instance weights. They are used in the process of bagging when the instances with the higher weight are used more often than the instances with the lower weight. They have also an impact on the out-of-bag error calculation.

Learning output

The learning output consists of the learned classifier and the instance creator used in the learning process. The specific text instance creator is later used in the unknown instance classification process. The learning process is connected with the evaluation, so the performance evaluation results are attached to the learning output. Then this result is serialised and stored on the hard disc.

7.5 Enrichment module

The enrichment module is responsible for the ontology enrichment. It uses the classifiers created in the learning module to extract new ontological elements. The input of this module is the set of linguistically processed documents and an input ontology and the output are the extracted concepts, relations and instances.

We can divide this module into two components. First component reads the input documents and performs the classification. It uses the linguistic processing module and the document processing module to create the set of unknown instances for the classification. The second component creates the ontological elements from the classification output. The classification output has the form of

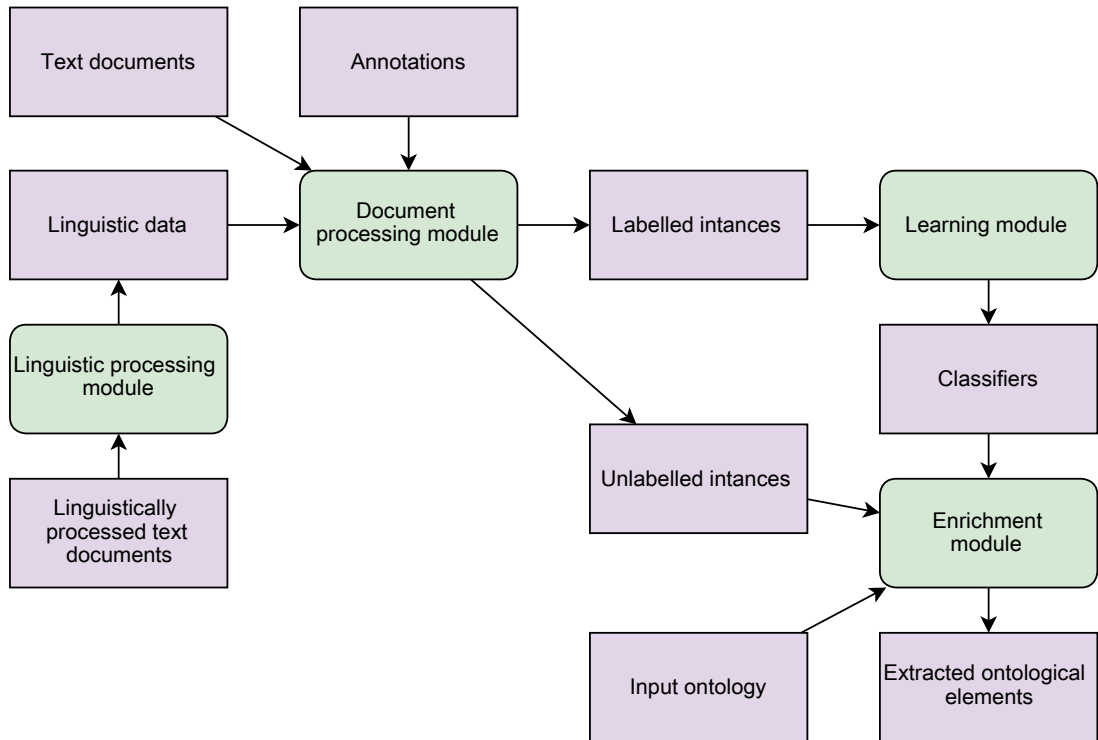


Figure 7.2: Workflow of the implementation modules

a set of text pieces with relations between them. We described the process of the extraction in the section 6.4. We use the OWL representation of the input ontology and we use the Jena framework [43] to read the input ontology and create the elements.

7.5.1 OWL representation

We decided to stick to the representation of an annotation ontology presented in the section 1.4.1. Each extracted ontological concept is represented by an *OWL class*. The attached lexical representation of this class is defined by a *rdfs:label* property and other metadata such as concept importance are stated in a *rdfs:comment* property.

The IS-A relationship is represented using the OWL class hierarchy. The COHYPO relation does not have specific denotation and is expressed only by IS-A relations. The HAS-A non-taxonomic relation is represented by the set of *OWL object property* between two OWL classes. For example, the *hasLevel* express the HAS-A relation between an Activity and a Level concept.

Finally, an extracted ontological instance is represented by an OWL individual of a particular OWL class that denotes some ontological concept. This individual has a set of properties defining its document position, such as *hasStartIndex* and *hasEndIndex* properties. Its textual representation is expressed in the *hasText* property.

```

semjob: a owl:Ontology.

semjob:JD a owl:Class;
  rdfs:subClassOf skos:Concept.

semjob:JDEntity a owl:Class;
  rdfs:subClassOf skos:Concept.

semjob:Domain a owl:Class;
  rdfs:subClassOf semjob:JDEntity.

semjob:Activity a owl:Class;
  rdfs:subClassOf semjob:JDEntity.

semjob:Level a owl:Class;
  rdfs:subClassOf semjob:JDEntity.

semjob:Quality a owl:Class;
  rdfs:subClassOf semjob:JDEntity.

semjob:Feature a owl:Class;
  rdfs:subClassOf semjob:JDEntity.

semjob:Person a owl:Class;
  rdfs:subClassOf semjob:JDEntity.

semjob:hasLevel a owl:ObjectProperty;
  rdfs:domain semjob:Activity;
  rdfs:range semjob:Level.

semjob:hasTheme a owl:ObjectProperty;
  rdfs:domain semjob:Activity;
  rdfs:range semjob:Domain.

semjob:hasStartIndex a owl:ObjectProperty;
  rdfs:domain semjob:JDEntity;
  rdfs:range xsd:integer.

semjob:hasEndIndex a owl:ObjectProperty;
  rdfs:domain semjob:JDEntity;
  rdfs:range xsd:integer.

semjob:hasText a owl:ObjectProperty;
  rdfs:domain semjob:JDEntity;
  rdfs:range xsd:string.

semjob:belongsToDocument a owl:ObjectProperty;
  rdfs:domain semjob:JDEntity;
  rdfs:range xsd:string.

semjob:hasActivity a owl:ObjectProperty;
  rdfs:domain semjob:JD;
  rdfs:range semjob:Activity.

```

Part of the input ontology used in the extraction

```

semjob:Znalost
  a      owl:Class ;
  rdfs:comment "Relevance:0.5363636363636364" ;
  rdfs:label "znalost^(*3ý)" ;
  rdfs:subClassOf semjob:Activity.

semjob:Komunikativni
  a      owl:Class ;
  rdfs:comment "Relevance:0.42499999999999993" ;
  rdfs:label "komunikativní" ;
  rdfs:subClassOf semjob:Level .

semjob:AJ
  a      owl:Class ;
  rdfs:comment "Relevance:0.3870192307692308" ;
  rdfs:label "aj" ;
  rdfs:subClassOf semjob:Domain .

semjob:Aj_817b66e1-c06f-426f-bf66-6d585381900
  a      semjob:AJ ;
  semjob:hasEndIndex "885"^^xsd:int ;
  semjob:hasStartIndex
    "883"^^xsd:int ;
  semjob:belongsToDocument "jd01.txt";
  semjob:hasText "AJ"^^xsd:string .

semjob:Komunikativni_8b3af065-42ba-4761-bb52
  semjob:Komunikativni;
  semjob:hasEndIndex "881"^^xsd:int ;
  semjob:hasStartIndex
    "868"^^xsd:int ;
  semjob:belongsToDocument "jd01.txt";
  semjob:hasText "AJ"^^xsd:string .

semjob:Znalost_2fc5de59-6cdd-457e-954e-373b9af60b22
  a      semjob:Znalost ;
  semjob:hasEndIndex "882"^^xsd:int ;
  semjob:hasLevel semjob:Komunikativni_8b3af065-42ba-4761-bb52;
  semjob:hasStartIndex
    "875"^^xsd:int ;
  semjob:belongsToDocument "jd01.txt";
  semjob:hasText "znalost"^^xsd:string ;
  semjob:hasTheme Aj_817b66e1-c06f-426f-bf66-6d5853819002> .

```

Example of the extracted ontology

7.6 Conclusion

We have described the main modules of our method and their workflows along with several implementation details. For further information, see the programmer documentation on the attached DVD.

8. Experimental results and evaluation

In the section 2.2.1, we have presented several evaluation metrics: accuracy, precision, recall and F-measure. In this chapter, we present the results of the experiments with our method and evaluate its performance with respect to these metrics.

8.1 Testing environment

Our testing environment consists of a set of annotated documents from the job descriptions domain. We have used the real-life job descriptions samples downloaded from different web portals for the annotation. These documents have been converted from the HTML format to the plain text and annotated in the annotation tool Brat by one annotator.

8.1.1 Data sets

We have decided to experiment with two different data set sizes to found their impact on the accuracy of the method. Unfortunately, because of the very time-consuming annotation process, we have not been able to provide a large number of annotated documents. We use relatively small data sets - the first one had 16 documents and the second one had 32 documents. These documents contains different counts of annotations.

We have trained the classifiers separately and each classifier has its proper data set. These data sets are presented in the tables 8.1 - 8.16. We report several statistical information such as the total number of annotations per data set and average number of annotations per document.

8.1.2 Learning

In the learning process, the data set is split into two parts, a test and a train part. The train part contains on average 70% of the input documents, the test part contains the remaining documents. The distribution between these two parts is computed based on a random number generator. Then a classifier is learned on the train data set.

After each classifier is learned, its performance is determined using the test data set. Each document from this set is processed and the classification is performed. The positive and negative classification results are compared with the annotated and unannotated parts of the texts. We use the textual comparison of the text pieces and only their full match is considered as the identity. We sum up the number of true positives, true negatives, false positives and false negatives annotations. Then the performance metrics are calculated for this text document and finally for the entire test data set.

Activity data set 1	Data set size	J48		Random Forest	
		Learn data set	Test data set	Learn data set	Test data set
Documents count	16	11	5	11	5
Positive annotations count	189	135	54	117	72
Negative annotations count	19299	14740	4559	13459	5840
Total annotations count	19488	14875	4613	13576	5912
Average positive annotations count per document	11.81	12.27	10.8	10.64	14.4

Table 8.1: Activity data set 1

Activity data set 2	Data set size	J48		Random Forest	
		Learn data set	Test data set	Learn data set	Test data set
Documents count	32	22	10	22	10
Positive annotations count	352	237	115	250	102
Negative annotations count	43425	27277	16148	31843	11582
Total annotations count	43777	27514	16263	32093	11684
Average positive annotations count per document	11	10.77	11.5	11.36	10.2

Table 8.2: Activity data set 2

Domain data set 1	Data set size	J48		Random Forest	
		Learn data set	Test data set	Learn data set	Test data set
Documents count	16	11	5	11	5
Positive annotations count	498	308	190	307	191
Negative annotations count	19038	11969	7069	11656	7382
Total annotations count	19536	12277	7259	11963	7573
Average positive annotations count per document	31.13	28	38	27.91	38.2

Table 8.3: Domain data set 1

Domain data set 2	Data set size	J48		Random Forest	
		Learn data set	Test data set	Learn data set	Test data set
Documents count	32	22	10	22	10
Positive annotations count	898	570	328	585	313
Negative annotations count	43709	30785	12924	26616	17093
Total annotations count	44607	31355	13252	27201	17406
Average positive annotations count per document	28.06	25.91	32.8	26.59	31.3

Table 8.4: Domain data set 2

Feature data set 1	Data set size	J48		Random Forest	
		Learn data set	Test data set	Learn data set	Test data set
Documents count	16	11	5	11	5
Positive annotations count	114	76	38	80	34
Negative annotations count	18674	13311	5363	12274	6400
Total annotations count	18788	13387	5401	12354	6434
Average positive annotations count per document	7.13	6.91	7.6	7.27	6.8

Table 8.5: Feature data set 1

Feature data set 2	Data set size	J48		Random Forest	
		Learn data set	Test data set	Learn data set	Test data set
Documents count	32	22	10	22	10
Positive annotations count	305	188	117	237	68
Negative annotations count	43544	32949	10595	28273	15271
Total annotations count	43849	33137	10712	28510	15339
Average positive annotations count per document	9.53	8.55	11.7	10.77	6.8

Table 8.6: Feature data set 2

Level data set 1	Data set size	J48		Random Forest	
		Learn data set	Test data set	Learn data set	Test data set
Documents count	16	11	5	11	5
Positive annotations count	61	40	21	40	21
Negative annotations count	19429	12533	6896	13009	6420
Total annotations count	19490	12573	6917	13049	6441
Average positive annotations count per document	3.81	3.64	4.2	3.64	4.2

Table 8.7: Level data set 1

Level data set 2	Data set size	J48		Random Forest	
		Learn data set	Test data set	Learn data set	Test data set
Documents count	32	22	10	22	10
Positive annotations count	123	82	41	86	37
Negative annotations count	43314	27375	15939	29943	13371
Total annotations count	43437	27457	15980	30029	13408
Average positive annotations count per document	3.84	3.73	4.1	3.91	3.7

Table 8.8: Level data set 2

Person data set 1	Data set size	J48		Random Forest	
		Learn data set	Test data set	Learn data set	Test data set
Documents count	16	11	5	11	5
Positive annotations count	78	57	21	51	27
Negative annotations count	19410	14440	4970	13704	5706
Total annotations count	19488	14497	4991	13755	5733
Average positive annotations count per document	4.88	5.18	4.2	4.64	5.4

Table 8.9: Person data set 1

Person data set 2	Data set size	J48		Random Forest	
		Learn data set	Test data set	Learn data set	Test data set
Documents count	32	22	10	22	10
Positive annotations count	174	123	51	133	41
Negative annotations count	44346	32682	11664	31387	12959
Total annotations count	44520	32805	11715	31520	13000
Average positive annotations count per document	5.44	5.59	5.1	6.05	4.1

Table 8.10: Person data set 2

Quality data set 1	Data set size	J48		Random Forest	
		Learn data set	Test data set	Learn data set	Test data set
Documents count	16	11	5	11	5
Positive annotations count	51	32	19	27	24
Negative annotations count	12830	8354	4476	8656	4174
Total annotations count	12881	8386	4495	8683	4198
Average positive annotations count per document	3.19	2.91	3.8	2.45	4.8

Table 8.11: Quality data set 1

Quality data set 2	Data set size	J48		Random Forest	
		Learn data set	Test data set	Learn data set	Test data set
Documents count	32	22	10	22	10
Positive annotations count	111	84	27	82	29
Negative annotations count	34611	26405	8206	25740	8871
Total annotations count	34722	26489	8233	25822	8900
Average positive annotations count per document	3.47	3.82	2.7	3.73	2.9

Table 8.12: Quality data set 2

Non-taxonomic rel. data set 1	Data set size	J48		Random Forest	
		Learn data set	Test data set	Learn data set	Test data set
Documents count	16	11	5	11	5
Positive annotations count	276	208	68	209	67
Negative annotations count	3063	2431	632	2364	699
Total annotations count	3339	2639	700	2573	766
Average positive annotations count per document	17.25	18.91	13.6	19	13.4

Table 8.13: Non-taxonomic relations data set 1

Non-taxonomic rel. data set 2	Data set size	J48		Random Forest	
		Learn data set	Test data set	Learn data set	Test data set
Documents count	32	22	10	22	10
Positive annotations count	479	340	139	338	141
Negative annotations count	6022	4175	1847	4420	1602
Total annotations count	6501	4515	1986	4758	1743
Average positive annotations count per document	14.97	15.45	13.9	15.36	14.1

Table 8.14: Non-taxonomic relations data set 2

Taxonomic rel. data set 1	Data set size	J48		Random Forest	
		Learn data set	Test data set	Learn data set	Test data set
Documents count	16	11	5	11	5
Positive annotations count	182	128	54	135	47
Negative annotations count	3157	2230	927	2296	861
Total annotations count	3339	2358	981	2431	908
Average positive annotations count per document	11.38	11.64	10.8	12.27	9.4

Table 8.15: Taxonomic relations data set 1

Taxonomic rel. data set 2	Data set size	J48		Random Forest	
		Learn data set	Test data set	Learn data set	Test data set
Documents count	32	22	10	22	10
Positive annotations count	304	228	76	219	85
Negative annotations count	6197	4790	1407	4340	1857
Total annotations count	6501	5018	1483	4559	1942
Average positive annotations count per document	9.5	10.36	7.6	9.95	8.5

Table 8.16: Taxonomic relations data set 2

8.2 Experimental results

In this section we provide the experimental results of the learning process. We have following learning cases: one for the base ontological concept classifiers - Activity, Domain, Feature, Level, Person and Quality, and one for each relation type - taxonomic and non-taxonomic. For each learning case, we report the accuracy metrics measured in the learning process.

In the process of a particular learning case performance optimisation, several classifiers have been trained. We have selected the best one according to F-measure criteria. The review of performance metrics of these classifiers is reported in the tables 8.17 - 8.24. We provide the results for two different data sets described in the previous section. The best trained classifier has specified an ID assigned in the learning process.

Activity	J48		Random Forest	
Data set size	16	32	16	32
Accuracy	0.992	0.995	0.991	0.993
Precision	0.718	0.663	0.629	0.594
Recall	0.516	0.478	0.541	0.618
F-measure	0.602	0.556	0.582	0.618
Classifier ID	1364919380644	1365000815311	1364918938716	1365005480784

Table 8.17: Activity classifiers performance

Domain	J48		Random Forest	
Data set size	16	32	16	32
Accuracy	0.972	0.996	0.976	0.981
Precision	0.472	0.369	0.525	0.449
Recall	0.353	0.524	0.387	0.310
F-measure	0.404	0.433	0.446	0.367
Classifier ID	1364919547967	1365007395475	1364917970156	1365007425732

Table 8.18: Domain classifiers performance

Feature	J48		Random Forest	
Data set size	16	32	16	32
Accuracy	0.989	0.983	0.994	0.994
Precision	0.211	0.258	0.389	0.241
Recall	0.211	0.291	0.206	0.206
F-measure	0.211	0.273	0.269	0.222
Classifier ID	1364918299217	1365008810966	1364918303598	1365008777942

Table 8.19: Feature classifiers performance

Level	J48		Random Forest	
Data set size	16	32	16	32
Accuracy	0.997	0.997	0.997	0.998
Precision	0.636	0.382	0.643	0.576
Recall	0.333	0.512	0.429	0.514
F-measure	0.437	0.438	0.514	0.543
Classifier ID	1364919675615	1365008319985	1364918583715	1365003263941

Table 8.20: Level classifiers performance

Person	J48		Random Forest	
Data set size	16	32	16	32
Accuracy	0.996	0.997	0.998	0.998
Precision	0.714	0.725	0.87	0.649
Recall	0.714	0.569	0.741	0.585
F-measure	0.714	0.637	0.8	0.615
Classifier ID	1364919592411	1365002191534	1364919599422	1365004690045

Table 8.21: Person classifiers performance

Quality	J48		Random Forest	
Data set size	16	32	16	32
Accuracy	0.991	0.992	0.993	0.997
Precision	0.188	0.161	0.23	0.467
Recall	0.316	0.333	0.125	0.241
F-measure	0.235	0.217	0.312	0.318
Classifier ID	1364922665921	1365010612141	1364919724051	1365002748478

Table 8.22: Quality classifiers performance

Non-taxonom. relation	J48		Random Forest	
Data set size	16	32	16	32
Accuracy	0.907	0.951	0.94	0.947
Precision	0.518	0.695	0.784	0.773
Recall	0.632	0.525	0.433	0.482
F-measure	0.571	0.598	0.558	0.594
Classifier ID	1364918592171	1365002299907	1364919418924	1365002302907

Table 8.23: Non-taxonomic classifiers performance

Taxonom. relation	J48		Random Forest	
Data set size	16	32	16	32
Accuracy	0.971	0.963	0.971	0.972
Precision	0.882	0.627	0.889	0.875
Recall	0.556	0.671	0.511	0.412
F-measure	0.682	0.65	0.649	0.56
Classifier ID	1364919608089	1365003710784	1364918275826	1365005336822

Table 8.24: Taxonomic relations classifiers performance

8.3 Evaluation

In the area of the concepts classifiers, the experimental results vary from learning case to learning case. The Feature and the Quality classifiers have the worst results, only nearly 22% of F-measure. The Person and Activity classifiers have the best results, they show up to the 80% of F-measure. The Domain classifiers shows up to 44% of F-measure and the Level classifiers shows up to 54% of F-measure.

In the area of the relation classifiers, the experimental results are quite similar and shows near 55 - 65% of F-measure for both taxonomical and non-taxonomical relation classifiers.

Based on these experimental results, we can made several interesting observations.

First, the two chosen classification techniques, the decision trees and random forests, do not show significant differences in classification accuracy. It seems they are both suitable for our classification method and they are both robust to the problem of an imbalance data set.

For every learning case, most of negative instances has been correctly filtered. All classifiers show high accuracy, from 97% to 99%. High accuracy does not guarantee the usable classification results, as we can see on the other performance metrics.

Next, the different data set sizes do not have significant impact on classification accuracy. With higher data set size, the performance of the classifier can be determined more precisely. On the other hand, the data set with size of 16 documents seems to be sufficient for most cases.

Other observation is made about the complexity of the word phrases. With the growing complexity, the classification accuracy is decreasing. We can see that on the example of the Person and Activity classifiers, where the terms are usually composed of one word. In the case of Feature concept, the word phrases are usually more complex and include several words. The complexity of a decision tree is proportional to the complexity of the word phrases. In this case, the learning data set does not contain enough of the positive instances to deduct correct generalisation rules.

Next observation is related to the observation above. A low frequency of the word phrases in the documents has negative impact on the classification accuracy. Combination of a low frequency and a complex word phrase may cause a wrong performance, as we can see in the case of the Quality classifiers.

8.3.1 Analysis of the learned decision trees

With high interpretability of the decision tree classifiers, we can investigate several interesting characteristics of our classification task. The overview of several properties of the learned decision trees is presented in the table 8.25.

First, we can see what features are important for a particular learning case. The importance of the features can be expressed as their information gain and the features with high information gain are located closer to the root of the decision tree. The feature in the root of a tree is **the most important feature** in the classification. We can see that this feature can change with the growing data set

size - with new learning instances, a different feature can become more important.

For a binary decision tree, we can similarly determine **the most important feature value** as the split condition of a particular feature in the decision tree node.

The complexity of the learning task is related to **the number of nodes in the decision tree**. With bigger tree, the complexity of the task is increasing. We can see that the Level and Person learning cases are the most simpler ones and the Domain learning case is the most complicated one. With growing data set, the tree complexity is growing too in most cases. In the case of Feature classifier, a multinary tree was used for the data set 2. Such tree has a huge number of leaves, because one particular leaf is created for every feature value.

		Data set 1	Data set 2
Activity	Root attribute condition	LastWordCase = NOMINATIVE	LastWordCase = NOMINATIVE
	Tree size	45	45
	Number of leaves	23	23
Domain	Root attribute condition	HeadCase = GENITIVE	Word2POS = 0
	Tree size	149	215
	Number of leaves	75	108
Feature	Root attribute condition	Word1POS = ADJECTIVE	Word1POS = ADJECTIVE
	Tree size	55	199
	Number of leaves	28	100
Level	Root attribute condition	LastWordPOS = ADJECTIVE	HeadDependent WordPOS = ADJECTIVE
	Tree size	17	39
	Number of leaves	9	20
Person	Root attribute condition	HeadGender = MASCULINE_ANIMATE	HeadGender = MASCULINE_ANIMATE
	Tree size	15	27
	Number of leaves	8	14
Quality	Root attribute condition	HeadCase = NOMINATIVE	SentenceLegth <= 12
	Tree size	59	687 ¹
	Number of leaves	30	622
Non-taxonomic relations	Root attribute condition	Phrase2Entity = Level	Phrase2Entity = Activity
	Tree size	61	65
	Number of leaves	31	33
Taxonomic relations	Root attribute condition	SameEntities = yes	SameEntities = yes
	Tree size	43	65
	Number of leaves	22	33

Table 8.25: Decision tree classifiers properties

¹The tree size is caused by the use of a multinary decision tree.

8.3.2 Summary

The performance of our classification method is satisfying for most of the learning cases. For several more complicated learning cases, such as Feature and Quality, the chosen features seems not to have enough generalisation capabilities and the classification performance is relatively low in comparing with other learning cases. However, the learning capacities of our method are not definitive and this handicap can be compensated by adding additional features to the classification.

Conclusion

We have presented a new method for the ontology enrichment from unstructured texts based on the natural language processing and machine learning techniques. This method is proposed as the extension of the semantic annotation process, where new ontological elements, such as ontological concepts and relations, are added to the annotation ontology. We have employed decision trees and random forest classifiers in the machine learning task. These classifiers are trained on a pre-annotated data set and used to identify new ontological elements in the texts. We have also tried to improve the process of semantic annotation by extracting a set of ontological instances attached to the newly extracted ontological concepts. An extracted ontology in our method has an OWL representation.

We have created a prototype implementation and evaluated the performance of our method on the job descriptions domain. Our method achieved F-measure from 21% to 70% for extraction of ontological concepts. In the area of ontological relations extraction, our method achieved F-measure up to 70%. These results are satisfying for the practical use.

However, several improvements can be made to our method. First, our method does not show sufficient results for the extraction of more complicated word phrases. The learning capacities could be increased by adding more specific features in the learning process. In the prototype implementation, we have used only single words or entire phrases in the classification features extraction. Feature extraction from more complex words groups or syntax tree segments can be used for the further improvement of the performance.

Next, the annotation process, necessary in learning phase, is very time consuming. Although the experiments have shown that relatively small learning data sets also provide good performance, facilitating of this process is very important in the practical use of the method.

Another limitation of our method is the resulting ontological representation. An extracted concept is represented by an OWL class. If we wanted to add other properties to this OWL class, an OWL Full representation would be needed. In OWL Lite and OWL DL, an OWL class could not have data properties except of some special meta-data ones. This problem can be resolved by a *reasoning transformation* that converts the extracted concepts represented by OWL classes to OWL individuals. Similarly, the extracted instances represented by OWL individuals could be converted to the individuals of a special OWL class representing an occurrence of a concept in the text.

Bibliography

- [1] GRUBER, Thomas R., et al. A translation approach to portable ontology specifications. *Knowledge acquisition*, 1993, 5.2: 199-220.
- [2] GÓMEZ-PÉREZ, Asunción; CORCHO, Oscar. Ontology languages for the semantic web. *Intelligent Systems, IEEE*, 2002, 17.1: 54-60.
- [3] HIRST, Graeme. Ontology and the lexicon. In: *Handbook on ontologies*. Berlin: Springer Berlin Heidelberg, 2009. p. 269-292. ISBN 978-3-540-70999-2.
- [4] BERNERS-LEE, Tim; FISCHETTI, Mark; FOREWORD BY-DERTOUZOS, Michael L. *Weaving the Web: The original design and ultimate destiny of the World Wide Web by its inventor*. Darby: DIANE Publishing Company, 2001. ISBN 0-7567-5231-0.
- [5] BERNERS-LEE, Tim, et al. The semantic web. *Scientific american*, 2001, 284.5: 28-37.
- [6] W3C Semantic Web Activity. *World Wide Web Consortium (W3C)* [online]. ©2013 [cited 2013-02-02]. URL:<<http://www.w3.org/2001/sw/>>
- [7] OWL 2 Web Ontology Language. *World Wide Web Consortium (W3C)* [online]. ©2013 [cited 2013-02-28]. URL:<<http://www.w3.org/TR/owl2-overview/>>
- [8] OWL WEB Ontology Language Guide. *World Wide Web Consortium (W3C)* [online]. ©2013 [cited 2013-02-28]. URL:<<http://www.w3.org/TR/2004/REC-owl-guide-20040210/>>
- [9] SKOS Simple Knowledge Organization System. *World Wide Web Consortium (W3C)* [online]. ©2013 [cited 2013-02-28]. URL:<<http://www.w3.org/2004/02/skos/>>
- [10] SHETH, Amit; LYTRAS, Miltiadis D. *Semantic web-based information systems: state-of-the-art applications*. Idea Group Publishing, 2007. ISBN 1-59904-428-5.
- [11] MAEDCHE, Alexander; STAAB, Steffen. Ontology learning for the semantic web. *Intelligent Systems, IEEE*, 2001, 16.2: 72-79.
- [12] BUITELAAR, Paul; CIMIANO, Philipp; MAGNINI, Bernardo. *Ontology learning from text: methods, evaluation and applications*. Ios PressInc, 2005. ISBN 978-1-58603-523-5.
- [13] BUITELAAR, Paul; OLEJNIK, Daniel; SINTEK, Michael. A protégé plug-in for ontology extraction from text based on linguistic analysis. In: *The Semantic Web: Research and Applications*. Springer Berlin Heidelberg, 2004. p. 31-44. ISBN 978-3-540-21999-6.

- [14] BLOEHDORN, Stephan, et al. An ontology-based framework for text mining. In: *LDV Forum-GLDV Journal for Computational Linguistics and Language Technology*. 2005. p. 87-112. ISSN 0175-1336.
- [15] CIMIANO, Philipp; GÜNTER, Ladwig; STEFFEN, Staab. Gimme'the context: context-driven automatic semantic annotation with C-PANKOW. In: *Proceedings of the 14th international conference on World Wide Web*. New York: ACM, 2005. p. 332-341. ISBN 1-59593-046-9.
- [16] CIMIANO, Philipp, et al. Learning taxonomic relations from heterogeneous sources of evidence. *Ontology Learning from Text: Methods, evaluation and applications*, 2005, 123: 59-73.
- [17] WITSCHERL, Hans Friedrich. Using decision trees and text mining techniques for extending taxonomies. In: *Proceedings of the Workshop on Learning and Extending Lexical Ontologies by Using Machine Learning Methods*. Bonn, 2005.
- [18] LACLAVIK, Michal, et al. Ontology based text annotation-OnTeA. *Frontiers in Artificial Intelligence and Applications*, 2007, 154: 311.
- [19] SNOW, Rion; JURAFSKY, Daniel; NG, Andrew Y. Semantic taxonomy induction from heterogenous evidence. In: *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2006, p. 801-808.
- [20] SNOW, Rion; JURAFSKY, Daniel; NG, Andrew Y. Learning syntactic patterns for automatic hypernym discovery. *Advances in Neural Information Processing Systems 17*, 2004. ISBN 978-0-26219-534-8.
- [21] RAVICHANDRAN, Deepak; PANTEL, Patrick; HOVY, Eduard. Randomized algorithms and NLP: Using locality sensitive hash functions for high speed noun clustering. In: *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, 2005. p. 622-629.
- [22] JIANG, Xing; TAN, Ah-Hwee. CRCTOL: A semantic-based domain ontology learning system. *Journal of the American Society for Information Science and Technology*, 2009, 61.1: 150-168. ISSN 1532-2890.
- [23] POON, Hoifung; DOMINGOS, Pedro. Unsupervised ontology induction from text. In: *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2010. p. 296-305. ISBN 978-1-932432-67-1.
- [24] HEARST, Marti A. Automatic acquisition of hyponyms from large text corpora. In: *Proceedings of the 14th conference on Computational linguistics-Volume 2*. Association for Computational Linguistics, 1992, p. 539-545.

- [25] DOLAN, William; VANDERWENDE, Lucy; RICHARDSON, Stephen D. Automatically deriving structured knowledge bases from online dictionaries. In: *Proceedings of the Pacific Association for Computational Linguistics (PACLING)*. 1993, p. 5-14.
- [26] SUCHANEK, Fabian M.; SOZIO, Mauro; WEIKUM, Gerhard. *SOFIE: a self-organizing framework for information extraction*. In: *Proceedings of the 18th international conference on World wide web*. ACM, 2009. p. 631-640. ISBN 978-1-60558-487-4.
- [27] MITCHELL, Tom M. *Machine learning*. Burr Ridge, IL: McGraw Hill, 1997. ISBN 0-070-42807-7.
- [28] OLSON, David L.; DELEN, Dursun. *Advanced data mining techniques*, Springer Verlag, 2008. ISBN 3-540-76916-1.
- [29] KOHAVI, Ron; PROVOST, Foster. Glossary of terms. *Machine Learning*, 1998, 30.2-3: 271-274. ISSN 0885-6125.
- [30] BISHOP, Christopher M., et al. *Pattern recognition and machine learning*. New York: Springer, 2006. ISBN 0-387-31073-8.
- [31] POWELL, Warren Buckler. *Approximate Dynamic Programming: Solving the curses of dimensionality*. Wiley-Interscience, 2007. ISBN 0-470-18295-6.
- [32] HAWKINS, Douglas M., et al. The problem of overfitting. *Journal of chemical information and computer sciences*, 2004, 44.1: 1-12. ISSN 0095-2338.
- [33] STENETORP, Pontus, et al. 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*. Association for Computational Linguistics, 2012. p. 102-107.
- [34] Treex Highly Modular NLP Framework. *Institute of Formal and Applied Linguistics* [online]. ©2013 [cited 2013-02-20]. URL:<<http://ufal.mff.cuni.cz/treex/>>
- [35] GOUTTE, Cyril; GAUSSIER, Eric. A probabilistic interpretation of precision, recall and F-score, with implication for evaluation. In: *Advances in Information Retrieval*. Springer Berlin Heidelberg, 2005. p. 345-359. ISBN 978-3-540-25295-5.
- [36] QUINLAN, John Ross. Induction of decision trees. *Machine learning*, 1986, 1.1: 81-106. ISSN 0885-6125.
- [37] QUINLAN, John Ross. *C4. 5: programs for machine learning*. Morgan kaufmann, 1993. ISBN 1-558-60238-0.
- [38] BREIMAN, Leo. Random forests. *Machine learning*, 2001, 45.1: 5-32. ISSN 0885-6125.
- [39] BREIMAN,, Leo. Bagging predictors. *Machine learning*, 1996, 24.2: 123-140. ISSN 0885-6125.

- [40] PROVOST, Foster. Machine learning from imbalanced data sets 101. In: *Proceedings of the AAAI'2000 workshop on imbalanced data sets*. 2000.
- [41] Weka 3: Data Mining Software in Java. *Computer Science Department, University of Waikato* [online]. [cited 2013-02-28]. URL:<<http://www.cs.waikato.ac.nz/ml/weka/>>
- [42] Attribute-Relation File Format (ARFF). *Computer Science Department, University of Waikato* [online]. [cited 2013-02-28]. URL:<<http://www.cs.waikato.ac.nz/ml/weka/arff.html>>
- [43] Apache Jena. *The Apache Software Foundation* [online]. ©2013 [cited 2013-02-28]. URL:<<http://jena.apache.org/>>

List of Tables

2.1	Confusion matrix for a binary classifier	19
3.1	Overview of ontology enrichment methods	26
5.1	Extracted data of the linguistic analysis	34
7.1	Overview of features for concept classifiers	59
7.2	Overview of features for relation classifiers	60
8.1	Activity data set 1	66
8.2	Activity data set 2	66
8.3	Domain data set 1	66
8.4	Domain data set 2	66
8.5	Feature data set 1	66
8.6	Feature data set 2	67
8.7	Level data set 1	67
8.8	Level data set 2	67
8.9	Person data set 1	67
8.10	Person data set 2	68
8.11	Quality data set 1	68
8.12	Quality data set 2	68
8.13	Non-taxonomic relations data set 1	68
8.14	Non-taxonomic relations data set 2	69
8.15	Taxonomic relations data set 1	69
8.16	Taxonomic relations data set 2	69
8.17	Activity classifiers performance	70
8.18	Domain classifiers performance	70
8.19	Feature classifiers performance	70
8.20	Level classifiers performance	70
8.21	Person classifiers performance	71
8.22	Quality classifiers performance	71
8.23	Non-taxonomic classifiers performance	71
8.24	Taxonomic relations classifiers performance	71
8.25	Decision tree classifiers properties	73

List of Figures

1.1	Semantic annotation	10
6.1	Workflow of the training process	42
6.2	Annotation in the Brat tool	42
6.3	Syntax tree example	43
6.4	Input of the feature extraction process	45
6.5	Example of a simple decision tree for Domain concept	48
6.6	IS-A relation extraction	53
6.7	Ontological instances extraction	54
7.1	Annotations and document structure	58
7.2	Workflow of the implementation modules	62

List of Abbreviations

ARFF	Attribute-Relation File Format
NLP	Natural language processing
OWL	Web Ontology Language
POS	Part-of-speech
RDF	Resource Description Framework
SKOS	The Simple Knowledge Organization System
SPARQL	SPARQL Protocol and RDF Query Language
W3C	World Wide Web Consortium

Appendix A

Content of the attached DVD

<code>/classifiers/</code>	Classifiers used in the enrichment application
<code>/documentation/</code>	User and programmer documentation
<code>/data/</code>	Text data for the prototype implementation
<code>/data/jds/input/</code>	Data set text documents and annotation files
<code>/data/jds/processed/</code>	Linguistically processed data set documents
<code>/data/jds/test/</code>	Sample data for the enrichment application
<code>/data/jds/test-processed/</code>	Sample data for the enrichment application
<code>/jar/</code>	Executable jar with the enrichment application
<code>/ontology/</code>	Ontology for the enrichment application
<code>/shell/</code>	Shells scripts
<code>/thesis/</code>	Text of this thesis in PDF format
<code>/workspace/</code>	Source codes of the prototype implementation