Chapter 1

Introduction

1.1 Web Information Extraction Systems for Web Semantization

There exist many extraction tools that can process web pages and produce structured machine understandable data (or information) that corresponds with the content of a web page. This process is often called Web Information Extraction (WIE).

In this paper we present a survey of web information extraction systems and we connect these systems with the problem of web semantization.

The paper is structured as follows. First we sketch the basic ideas of semantic web and web semantization. In the next two sections methods of web information extraction will presented. Then description of our solutions (work in progress) will continue. And finally just before the conclusion we will discuss the connection of WIE systems with the problem of web semantization.

1.1.1 The Semantic Web in Use

The idea of the Semantic Web [Berners-Lee *et al.*, 2001] (World Wide Web dedicated not only to human but also to machine – software agents) is very well known today. Let us just shortly demonstrate its use with respect to the idea of Web Semantization (see in next section).

The Fig. 1.1 shows a human user using the (Semantic) Web in three manners: a keyword query, a semantic query and by using a software agent. The difference between the first two manners (keyword and semantic query) can be illustrated with the question: "Give me a list of the names of E.U. heads of state." This example from interesting article [Horrocks, 2008] by Ian Horrocks shows the big difference between use of a semantic query language instead of keywords. In the semantic case you should be given exactly the list of names you were requesting without having to pore through results of (probably more then one) keyword queries. Of course the user have to know the syntax of the semantic query language or a special GUI[1] must be provided.

The last and the most important possibility (in the semantic or semantized setting) is to use some (personalized) software agent that is specialized to tasks of some kind like planning a business trip or finding the most optimal choice from all the relevant job offers, flats for rent, cars for sale, etc.

Both the semantic querying and software agents engagement is impossible without some kind of adaptation of the web of today in the semantic direction.

1.1.2 Web Semantization

The idea of Web Semantization [Dědek *et al.*, 2008c] consist in gradual enrichment of the current web content as an automated process of third party annotation tools making at least a part of today's web more suitable for machine processing and hence enabling it intelligent tools for searching and recommending things on the

[1] Such handy GUI can be found for example in the KIM project [Popov *et al.*, 2004].

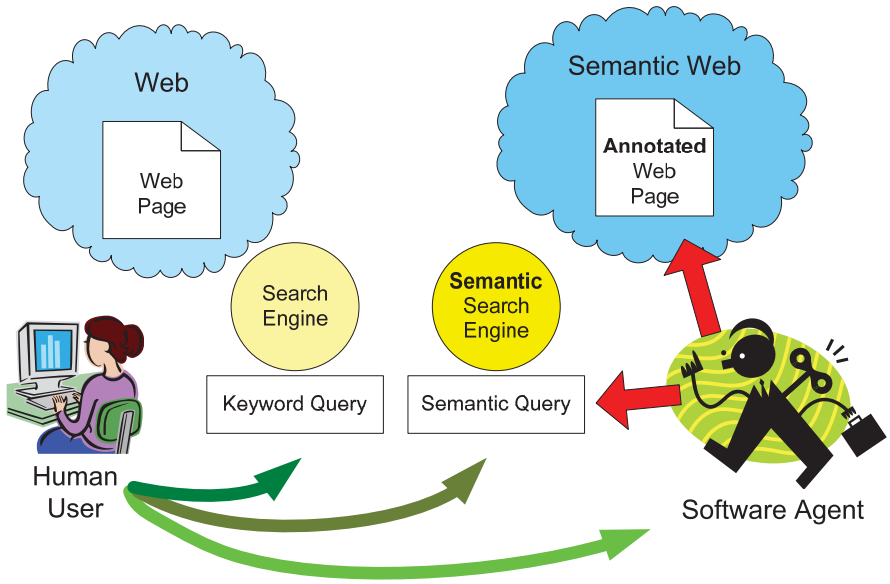


Figure 1.1: The Semantic/Semantized Web in Use

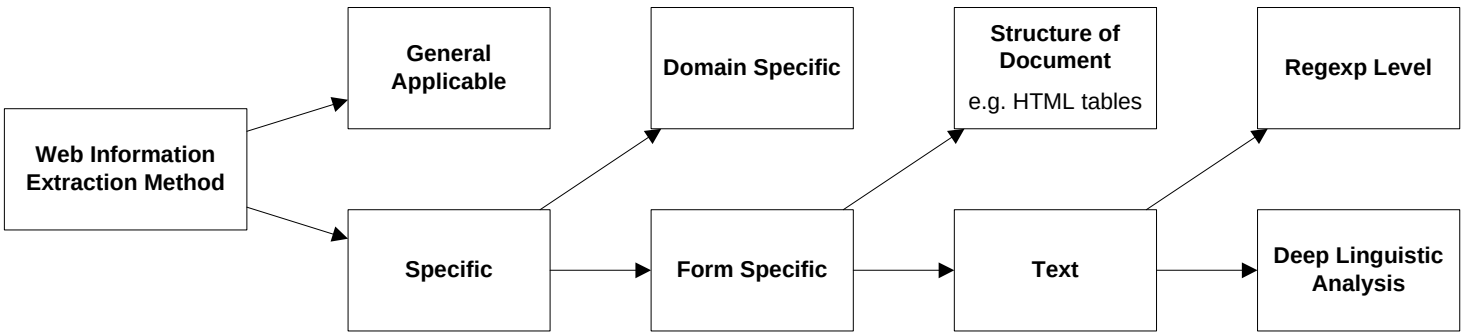


Figure 1.2: Division of extraction methods

web (see [Berners-Lee, 2008]).

The most strait forward idea is to fill a semantic repository with some information that is automatically extracted from the web and make it available to software agents so they could access to the web of today in a semantic manner (e.g. through semantic search engine).

The idea of a semantic repository and a public service providing semantic annotations was experimentally realized in the very recognized work of IBM Almaden Research Center: the SemTag [Dill *et al.*, 2003]. This work demonstrated that an automated semantic annotation can be applied in a large scale. In their experiment they annotated about 264 million web pages and generated about 434 millions of semantic tags. They also provided the annotations as a *Semantic Label Bureau* – a HTTP server providing annotations for web documents of 3rd parties.

1.2 Web Information Extraction

The task of a web information extraction system is to transform web pages into machine-friendly structures such as relational databases. There exists a rich variety of Web Information Extraction systems. The results generated by distinct tools usually can not be directly compared since the addressed extraction tasks are different. The extraction tasks can be distinguished according several dimensions: domain specificity, automation degree, techniques used, etc. These dimensions are analyzed in detail in the recent publications [Chang *et al.*, 2006a] and [Liu, 2007a]. Here we will concentrate on a little bit more specific division of WIE according to the needs of the Web Semantization (see in Sect. 1.5). The division is demonstrated on the Fig. 1.2 and should not be considered as disjoint division of the methods but rather as emphasizing different aspects of the methods. For example many extraction methods are domain and form specific at the same time.

The distinguishing between general applicable methods and the others that have meaningful application only in some specific setting (specific domain, specific form of input) is very important for Web Semantization because when we try to produce annotations in large scale, we have to control which web resource is suitable for which processing method (see in Sect. 1.5).

1.2.1 General Applicable

The most significant (and probably the only one) generally applicable IE task is so called *Instance Resolution Task*. The task can be described as follows: Given a general ontology, find all the instances from the ontology that are present in the processed resource. This task is usually realized in two steps: (1) Named Entity Recognition (see in Sect. 1.3.1), (2) Disambiguation of ontology instances that can be connected with the found named entities. Success of the method can be strongly improved with coreference resolution (see in Sect. 1.3.1).

Let us mention several good representatives of this approach: the SemTag application [Dill *et al.*, 2003], the KIM project [Popov *et al.*, 2004] and the PANKOW annotation method [Cimiano *et al.*, 2004] based on smart formulation of Google API queries.

1.2.2 Domain Specific

Domain and from specific IE approaches are the typical cases. More specific information is more precise, more complex and so more useful and interesting. But the extraction method has to be trained to each new domain separately. This usually means indispensable effort.

A good example of domain specific information extraction system is SOBA [Buitelaar *et al.*, 2008]. This complex system is capable to integrate different IE approaches and extract information from heterogeneous data resources, including plain text, tables and image captions but the whole system is concentrated on the single domain of football. Next similarly complex system is ArtEquAKT [Alani *et al.*, 2003], which is entirely concentrated on the domain of art.

1.2.3 Form Specific

Beyond general applicable extraction methods there exist many methods that exploit specific form of the input resource. The linguistic approaches usually process text consisting of natural language sentences. The structure-oriented approaches can be strictly oriented on tables [Pinto *et al.*, 2003] or exploit repetitions of structural patterns on the web page [Zhao *et al.*, 2005] (such algorithm can be only applicable to pages that contain more than one data record), and there are also approaches that use the structure of whole site (e.g. site of single web shop with summary pages with products connected with links to pages with details about single product) [Lerman *et al.*, 2004].

1.3 Information Extraction from Text-based Resources

In this section we will discuss the information extraction from textual resources.

1.3.1 Tasks of Information Extraction

There are classical tasks of text preprocessing and linguistic analysis like

Text Extraction – e.g from HTML, PDF or DOC,

Tokenization – detection of words, spaces, punctuations, etc.,

Segmentation – sentence and paragraph detection,

POS Tagging – part of speech assignment, often including lemmatization and morphological analysis,

Syntactic Analysis (often called linguistic *parsing*) – assignment of the grammatical structure to given sentence with respect to given linguistic formalism (e.g. formal grammar),

Coreference Resolution (or *anaphora resolution*) – resolving what a pronoun, or a noun phrase refers to. These references often cross boundaries of a single sentence.

Besides these classical general applicable tasks, there are further well defined tasks, which are more closely related to the information extraction. These tasks are domain dependent. These tasks were widely developed in the MUC-6 conference 1995 [Grishman and Sundheim, 1996] and considered as semantic evaluation in the first place. These information extraction tasks are:

Named Entity Recognition: This task recognizes and classifies named entities such as persons, locations, date or time expression, or measuring units. More complex patterns may also be recognized as structured entities such as addresses.

Template Element Construction: Populates templates describing entities with extracted roles (or attributes) about one single entity. This task is often performed stepwise sentence by sentence, which results in a huge set of partially filled templates.

Template Relation Construction: As each template describes information about one single entity, this tasks identifies semantic relations between entities.

Template Unification: Merges multiple elementary templates that are filled with information about identical entities.

Scenario Template Production: Fits the results of Template Element Construction and Template Relation Construction into templates describing pre-specified event scenarios (pre-specified "queries on the extracted data").

Appelt and Israel [Appelt and Israel, 1999] wrote an excellent tutorial summarizing these traditional IE tasks and systems built on them.

1.3.2 Information Extraction Benchmarks

Contrary to the WIE methods based on the web page structure, where we (the authors) do not know about any well established benchmark for these methods[2], the situation in the domain of text based IE is fairly different. There are several conferences and events concentrated on the support of automatic machine processing and understanding of human language in text form. Different research topics as text (or information) retrieval[3], text summarization[4] are involved.

On the filed of information extraction, we have to mention the long tradition of the Message Understanding Conference[5] [Grishman and Sundheim, 1996] starting in 1987. In 1999 the event of *Automatic Content Extraction (ACE) Evaluation*[6] started, which is becoming a track in the Text Analysis Conference (TAC)[7] this year (in 2009).

All these events prepare several specialized datasets together with information extraction tasks and play an important role as information extraction benchmarks.

1.4 Our Solutions

1.4.1 Extraction Based on Structural Similarity

Our first approach for the web information extraction is to use the structural similarity in web pages containing large number of table cells and for each cell a link to detailed pages. This is often presented in web shops and on pages that presents more than one object (product offer). Each object is presented in a similar way and this fact can be exploited.

As web pages of web shops are intended for human usage creators have to make their comprehension easier. Acquaintance with several years of web shops has converged to a more or less similar design fashion. There are often cumulative pages with many products in a form of a table with cells containing a brief description and a link to a page with details about each particular product.

Our main idea is to use a DOM tree representation of the summary web page and by breadth first search encounter similar subtrees. The similarity of these subtrees is used to determine the data region – a place where all the objects are stored. It is represented as a node in the DOM tree, underneath it there are the similar sub-trees, which are called data records.

We[8] have developed and implemented this idea [Eckhardt *et al.*, 2008] on the top of Mozilla Firefox API and experimentally tested on table pages from several domains (cars, notebooks, hotels). Similarity between subtrees was Levenshtein editing distance (for a subtree considered as a linear string), learning thresholds for decision were trained.

1.4.2 Linguistic Information Extraction

Our second approach [Dědek and Vojtáš, 2008c,d,b] for the web information extraction is based on deep linguistic analysis. We have developed a rule-based method for extraction of information from text-based web resources in Czech and now we are working on its adaptation to English. The extraction rules correspond to tree queries on linguistic (syntactic) trees made form particular sentences. We have experimented with several linguistic tools for Czech, namely Tools for machine annotation – PDT 2.0 and the Czech WordNet.

[2] It is probably at least partially caused by the vital development of the presentation techniques on the web that is still well in progress.

[3] e.g. Text REtrieval Conference (TREC)
`http://trec.nist.gov/`

[4] e.g. Document Understanding Conferences
`http://duc.nist.gov/`

[5] Briefly summarized in `http://en.wikipedia.org/wiki/Message_Understanding_Conference`.

[6] `http://www.itl.nist.gov/iad/mig/tests/ace/`

[7] `http://www.nist.gov/tac`

[8] Thanks go mainly to Du an Maru ? k and Peter Vojt .

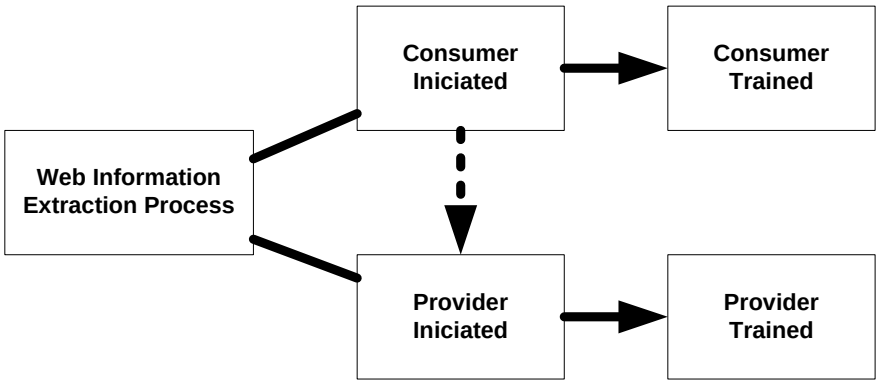


Figure 1.3: User initiative and effort

Our present system captures text of web-pages, annotates it linguistically by PDT tools, extracts data and stores the data in an ontology. We have made initial experiments in the domain of reports of traffic accidents. The results showed that this method can e.g. aid summarization of the number of injured people.

To avoid the need of manual design of extraction rules we focused on the data extraction phase and made some promising experiments [Dědek *et al.*, 2008a] with the machine learning procedure of Inductive Logic Programming for automated learning of the extraction rules.

This solution is directed to extraction of information which is closely connected with the meaning of text or meaning of a sentence.

1.5 The Web Semantization Setting

In this section we will discuss possibilities and obstructions connected with the employment of web information extraction systems in the process of web semantization.

One aspect of the realization of the web semantization idea is the problem of integration of all the components and technologies starting with web crawling, going through numerous complex analyses (document preprocessing, document classification, different extraction procedures), output data integration and indexing, and finally implementation of query and presentation interface. This elaborate task is neither easy nor simple but today it is solved in all the extensive projects and systems mentioned above.

The novelty that web semantization brings into account is the cross domain aspect. If we do not want to stay with just general ontologies and general applicable extraction methods then we need a methodology how to deal with different domains. The system has to support extension to a new domain in generic way. So we need a methodology and software to support this action. This can for example mean: to add a new ontology for the new domain, to select and train proper extractors and classifiers for the suitable input pages.

1.5.1 User Initiative and Effort

An interesting point is the question: Whose effort will be used in the process of supporting new domain in the web semantization process? How skilled such user has to be? There are two possibilities (demonstrated on the Fig 1.3). The easier one is that we have to employ very experienced expert who will decide about the new domain and who will also realize the support needed for the new domain. In the Fig 1.3 this situation is labeled as *Provider Initiated* and *Provider Trained* because the expert works on the side of the system that provides the semantics.

The second possibility is to enable ordinary users form outside to add a new domain to the semantization system. Such user is probably interested in semantic data form the domain so we call such user *Consumer*.

A cooperation of a consumer (possibly a domain expert) and a provider (system expert) on the support of the new domain can be considered as hybrid approach. This is represented with the dashed arrow in the Fig 1.3.

1.6 Conclusion and Future Work

In this paper we tried to show the complexity of the problem of web semantization in connection with the possibilities of web information extraction systems. Future work goes in several directions:

- Future development of WIE tools and work on their adaptability to new domains.
- Integration of WIE tools to the web semantization system.
- Development of the methodology and software to support the extension of the semantization system to a new domain for a non-expert user.

Chapter 2

Related Works

2.1 Based on ILP

There are many users of ILP in the linguistic and information extraction area. For example in [Konstantopoulos, 2003] ILP was used for shallow parsing and phonotactics. Authors of [Junker *et al.*, 1999] summarized some basic principles of using ILP for learning from text without any linguistic preprocessing. One of the most related approaches to ours can be found in [Aitken, 2002]. The authors use ILP for extraction of information about chemical compounds and other concepts related to global warming and they try to express the extracted information in terms of ontology. They use only the part of speech analysis and named entity recognition in the preprocessing step. But their inductive procedure uses also additional domain knowledge for the extraction. In [Ramakrishnan *et al.*, 2008] ILP was used to construct good features for propositional learners like SVM to do information extraction. It was discovered that this approach is a little bit more successful than a direct use of ILP but it is also more complicated. The later two approaches could be also employed in our solution.

2.2 Based on Dependency Linguistics

As stated in [Bunescu, 2007], the choice of the actual learning algorithm depends on the type of structural information available. For example, deep syntactic information provided by current parsers for new types of corpora such as biomedical text is seldom reliable, since most parsers have been trained on different types of narrative. If reliable syntactic information is lacking, sequences of words around and between the two entities can be used as alternative useful discriminators. But in our case deep linguistic parsing plays an essential role.

There are other approaches that use deep parsing, but they often use the syntactic structure only for relation extraction and either do not use machine learning at all (extraction rules have to be handcrafted) [Yakushiji *et al.*, 2001], [Fundel *et al.*, 2007], [Buyko *et al.*, 2009] or do some kind of similarity search based on the syntactic structure [Etzioni *et al.*, 2008], [Wang and Neumann, 2007] or the syntactic structure plays only very specific role in the process of feature selection for propositional learners [Bunescu and Mooney, 2007].

2.3 Based on Propositional Machine Learning

2.3.1 GATE Machine Learning

There is also a long row of information extraction approaches that use classical propositional learners like SVM on a set of features manually selected from input text. We do not cite them here. We just refer to [Li *et al.*, 2009] – using machine learning facilities in GATE. This is the software component (Machine Learning PR) to that we have compared our solution.

2.4 Semantic annotation

2.4.1 GATE

Last category of related works goes in the direction of semantics and ontologies. Because we do not develop this topic in this paper, we just refer to the ontology features in GATE [Bontcheva *et al.*, 2004], which can be easily used to populate an ontology with the extracted data. We discus this topic later in Section 5.2.7.

Chapter 3

Third Party Tools

3.1 Exploited Methods – Linguistics and ILP

In our solution we have exploited several tools and formalisms. These can be divided into two groups: linguistics and (inductive) logic programming. First we describe the linguistic tools and formalisms, the rest will follow.

3.1.1 GATE

GATE[1] [Cunningham *et al.*, 2002] is probably the most widely used tool for text processing. In our solution the capabilities of document and annotation management, utility resources for annotation processing, JAPE grammar rules [Cunningham *et al.*, 2000], machine learning facilities and performance evaluation tools are the most helpful features of GATE that we have used.

3.1.2 PDT and TectoMT

As we have started with our native language – Czech (a language with rich morphology and free word order), we had to make tools for processing Czech available in GATE. We have implemented a wrapper for the TectoMT system[2] [Žabokrtský *et al.*, 2008] to GATE. TectoMT is a Czech project that contains many linguistic analyzers for different languages including Czech and English. We have used a majority of applicable tools from TectoMT: a tokeniser, a sentence splitter, morphological analyzers (including POS tagger), a syntactic parser and the deep syntactic (tectogrammatical) parser. All the tools are based on the dependency based linguistic theory and formalism of the Prague Dependency Treebank project [Hajič *et al.*, 2006]. So far our solution does not include any coreference and discourse analysis.

Rozepsat PDT

3.1.3 Inductive Logic Programming

Inductive Logic Programming (ILP) [Muggleton, 1991] is a machine learning technique based on logic programming. Given an encoding of the known background knowledge (in our case linguistic structure of all sentences) and a set of examples represented as a logical database of facts (in our case tokens annotated with the target annotation type are positive examples and the remaining tokens negative ones), an ILP system will derive a hypothesized logic program (in our case extraction rules) which entails all the positive and none of the negative examples.

[1] http://gate.ac.uk/

[2] http://ufal.mff.cuni.cz/tectomt/

3.1.4 ILP tool

As an ILP tool we have used "A Learning Engine for Proposing Hypotheses" (Aleph v5)[3], which we consider very practical. It uses quite effective method of inverse entailment [Muggleton, 1995] and keeps all handy features of a Prolog system (we have used YAP Prolog[4]) in its background.

From our experiments (Section 5.3) can be seen that ILP is capable to find complex and meaningful rules that cover the intended information.

?? large amount of training data ??

As we do not have large amount of training data, there is no problem with excessive time demands during learning and the application of the learned rules is simple and quick.

[3] http://www.comlab.ox.ac.uk/activities/machinelearning/Aleph/

[4] http://www.dcc.fc.up.pt/~vsc/Yap/

Chapter 4

Datasets

Chapter 5

Extraction Method Based on ILP Machine Learning

5.1 Introduction

Automated semantic annotation (SA) is considered to be one of the most important elements in the evolution of the Semantic Web. Besides that, SA can provide great help in the process of data and information integration and it could also be a basis for intelligent search and navigation.

In this paper we present main results and reflections of our ongoing PhD project, a method for classical and semantic information extraction and annotation of texts, which is based on a deep linguistic analysis and Inductive Logic Programming (ILP). This approach is quite novel because it directly combines deep linguistic parsing with machine learning (ML). This combination and the use of ILP as a ML engine have following benefits: Manual selection of learning features is not needed. The learning procedure has full available linguistic information at its disposal and it is capable to select relevant parts itself. Extraction rules learned by ILP can be easily visualized, understood and adapted by human.

A description, implementation and initial evaluation of the method are the main contributions of the paper.

5.2 Implementation

Here we just briefly describe implementation of our system. The system consists of several modules, all integrated in GATE as processing resources.

5.2.1 TectoMT Wrapper (Linguistic Analysis)

TectoMT wrapper is a GATE component (processing resource), which takes the text of a GATE document, sends it to TectoMT linguistic analyzers, parses the results and converts the results to the form of GATE annotations.

Because TectoMT has to run as a separate process (it is implemented in Perl) and the initialization of TectoMT analyzers usually takes significant amount of time it would be very inefficient to start a new TectoMT instance for each document. Therefore the implementation currently offers two modes of execution: batch (TectoMTBatchAnalyser) and online (TectoMTOnlineAnalyser).

The batch mode is implemented similarly to the Batch Learning PR[1]. During the execution as a part of a corpus pipeline it only accumulates documents and the whole work is done as a batch when the last document is encountered. This also implies that TectoMTBatchAnalyser has to be the last PR in the pipeline because it produces no output in the time of execution (except for the last document).

[1] http://gate.ac.uk/userguide/sec:ml:batch-learning-pr

Client-server model of implementation is used in the online mode. A separate TectoMT server process is started at the time of initialization and GATE documents are processed in ordinary way at the time of execution. This means that (similarly to the previous case) each document is converted to the TectoMT readable format, sent to TectoMT and the result is converted back to GATE. The online mode of execution is a bit slower than the batch mode because additional time is spent on client-server communication (XML-RPC[2]).

5.2.2 PDT Annotations in GATE

Although GATE annotations are just highlighted pieces of text it is possible to use them to encode dependency tree structures. It is possible because each GATE annotation has a unique identifier (ID) and an arbitrary set of features (name-value pairs) can be assigned to it. The way how the PDT dependency trees are encoded in GATE is in fact the same as in the GATE wrapper for the Stanford Parser[3].

Three main constructs are used to capture an arbitrary configuration of a linguistic dependency tree:

tree nodes (usually corresponding to words (tokens) of a sentence)

edges (dependency relations between nodes)

node attributes (connected linguistic features like POS, gender, tense, case, etc.)

These constructs are encoded in GATE in the following way: tree nodes correspond to token annotations, node attributes are saved as token annotation features and edges are encoded as another special kind of annotations.

Two kinds of token annotations are used to represent two kinds of trees and tree nodes. "Token" annotation type is used for analytical tree nodes and "tToken" for tectogrammatical tree nodes.

Four kinds of edges (dependencies) are implemented by the TectoMT wrapper: analytical dependencies, tectogrammatical dependencies, aux.rf (auxiliary reference) and lex.rf (main lexical reference). The last two kinds (aux.rf and lex.rf) are used to connect tectogrammatical and analytical nodes. The implementation differs according to the cardinality of a dependency type. The first three kinds are of the cardinality one-to-many (one parent node can have many children nodes) and the last one (lex.rf) if of the cardinality one-to-one (one parent node has at most one child). Because of that lex.rf edges can be stored as features (with the name "lex.rf") of "tToken" annotations. Note that a GATE annotation feature can only have one value per annotation. In this case the annotation ID of the referenced "Token" annotation (referenced analytical node) is the value of the lex.rf feature.

One-to-many dependencies are stored as separate annotations (type names: "aDependency", "tDependency", "aux.rf") with a single feature called "args". Values of this feature are of Java type List<Integer> (list of integers). The list always contains just two items. The first one is the annotation ID of the parent annotation; the second one is the ID of the child annotation. Instead of using one list feature, two simple features (like "arg1", "arg2" or "parentID", "childID") could be used, but the implementation is consistent with the wrapper for the Stanford Parser (using the single list feature "args"), thus PDT dependencies are compatible with Stanford dependencies in GATE.

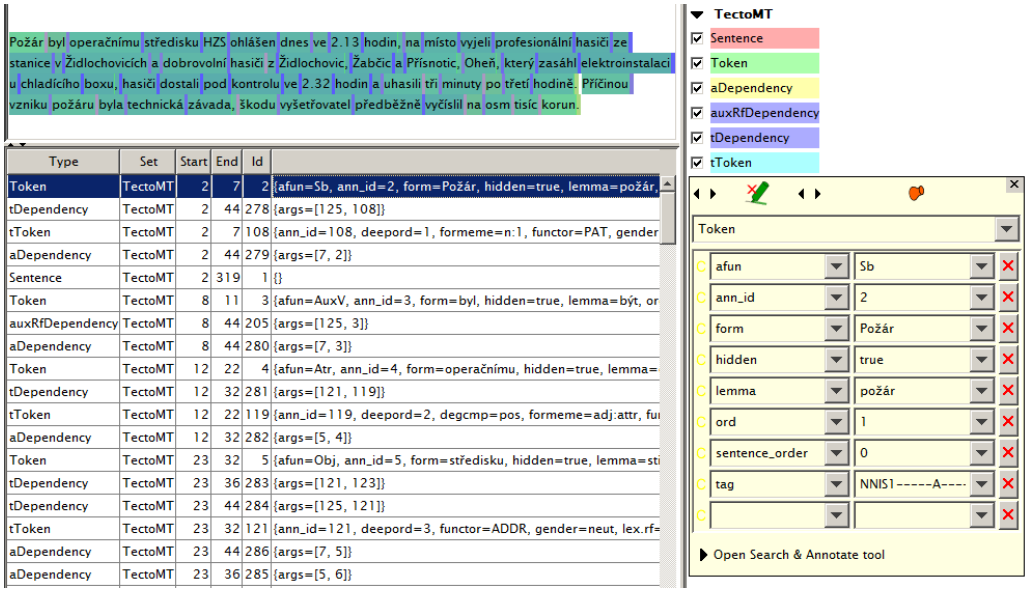
It is not simple to demonstrate the GATE representation of the dependencies in a static printed form; we can only show a GATE screenshot (Figure 5.1) that partly illustrates that.

5.2.3 ILP Wrapper (Machine Learning)

After a human annotator have annotated several documents with desired target annotations, machine learning takes place. This consists of two steps:

[2] http://www.xmlrpc.com/

[3] http://gate.ac.uk/userguide/sec:parsers:stanford



Type	Set	Start	End	Id	
Token	TectoMT	2	7	2	{afun=Sb, ann_id=2, form=Požár, hidden=true, lemma=požár,
tDependency	TectoMT	2	44	278	{args=[125, 108]}
tToken	TectoMT	2	7	108	{ann_id=108, deepord=1, formeme=n:1, functor=PAT, gender
aDependency	TectoMT	2	44	279	{args=[7, 2]}
Sentence	TectoMT	2	319	1	{}
Token	TectoMT	8	11	3	{afun=AuxV, ann_id=3, form=byl, hidden=true, lemma=být, or
auxRfDependency	TectoMT	8	44	205	{args=[125, 3]}
aDependency	TectoMT	8	44	280	{args=[7, 3]}
Token	TectoMT	12	22	4	{afun=Atr, ann_id=4, form=operačnímu, hidden=true, lemma=
tDependency	TectoMT	12	32	281	{args=[121, 119]}
tToken	TectoMT	12	22	119	{ann_id=119, deepord=2, degcmp=pos, formeme=adj:attr, fu
aDependency	TectoMT	12	32	282	{args=[5, 4]}
Token	TectoMT	23	32	5	{afun=Obj, ann_id=5, form=středisku, hidden=true, lemma=st
tDependency	TectoMT	23	36	283	{args=[121, 123]}
tDependency	TectoMT	23	44	284	{args=[125, 121]}
tToken	TectoMT	23	32	121	{ann_id=121, deepord=3, functor=ADDR, gender=neut, lex.rf=
aDependency	TectoMT	23	44	286	{args=[7, 5]}
aDependency	TectoMT	23	36	285	{args=[5, 6]}

Figure 5.1: PDT annotations in GATE (screenshot).

1. learning of extraction rules from the target annotations and

2. application of the extraction rules on new documents.

In both steps the linguistic analysis has to be done before and in both steps background knowledge (a logical database of facts) is constructed from linguistic structures of documents that are being processed. We call the process of background knowledge construction as *ILP serialization*; more details are presented below in Section 5.2.4.

After the ILP serialization is done, in the learning case, positive and negative examples are constructed from target annotations and the machine learning ILP inductive procedure is executed to obtain extraction rules.

In the application case a Prolog system is used to check if the extraction rules entail any of target annotation candidates.

The learning examples and annotation candidates are usually constructed from all document tokens (and we did so in the present solution), but it can be optionally changed to any other textual unit, for example only numerals or tectogrammatical nodes (words with lexical meaning) can be selected. This can be done easily with the help of *Machine Learning PR* (LM PR) from GATE[4].

ML PR provides an interface for exchange of features (including target class) between annotated texts and propositional learners in both directions – during learning as well as during application. We have used ML PR and developed our *ILP Wrapper* for it. The implementation was a little complicated because complex linguistic structures cannot be easily passed as propositional features, so in our solution we use the ML PR interface only for exchange of the class attribute and annotation id and we access the linguistic structures directly in a document.

5.2.4 ILP Serialization

In this section details about conversion of linguistic trees to ILP background knowledge (a Prolog logical database of facts) will be presented. Although the construction is quite strait forward it is worth describing because it makes it easier to understand the extraction rules found by the ILP learning procedure.

As mentioned in Section 5.2.2: three main constructs are used to capture an arbitrary configuration of a

[4] *Machine Learning PR* is an old GATE interface for ML and it is almost obsolete but in contrast to the new *Batch Learning PR* the LM PR is easy to extend for a new ML engine.

dependency linguistic tree: nodes, edges and node attributes. During the process of ILP Serialization these constructs are rendered to Prolog in following way.

A unique identifier (node ID) is generated for every tree node. The identifier is based on document name and GATE annotation ID (sentence order and node deep order are used outside of GATE, see PML→RDF transformation in Section 6.3.2.) These node IDs correspond to simple atoms and they represent tree nodes in the fact database. A node type (used by the ILP learning algorithm) is assigned to a node ID by predicates `Token(NodeID)` for analytical tree nodes and `tToken(NodeID)` for tectogrammatical tree nodes.

Tree nodes are connected by edges using binary predicates of the form:

`dependency_type_name(ParentNodeID, ChildNodeID)`

Note that the parent (governor) node always occupies the first argument and the child (dependant) node the second one. Predicate name *tDependency* is used for tectogrammatical dependencies and *aDependency* for analytical ones. There are also special kinds of dependencies that connect tectogrammatical and analytical nodes: *lex.rf* (main lexical reference) and *aux.rf* (auxiliary reference), in these cases tectogrammatical node occupies the first argument and analytical the second.

Node attributes are assigned to node IDs by binary predicates of the form:

`attribute_name(NodeID, AttributeValue)`

There are about thirty such predicates like *t_lemma* (tectogrammatical lemma), *functor* (tectogrammatical functor), *sempos* (semantic part of speech), *negation*, *gender*, etc. but minority of them is usually used in extraction rules.

Example of a serialized tectogrammatical tree is in Listing 1 it is the same tree as in Figure 7.3.

5.2.5 Connecting Linear GATE Annotations with Tree Nodes

5.2.6 Root/Subtree Preprocessing/Postprocessing

Sometimes annotations span over more than one token. This situation complicates the process of machine learning and this situation is often called as "chunk learning". Either we have to split a single annotation to multiple learning instances and after application we have to merge them back together, or we can change the learning task from learning annotated tokens to learning borders of annotations (start tokens and end tokens). The later approach is implemented in GATE in *Batch Learning PR* in the 'SURROUND' mode.

We have used another approach to solve this issue. Our approach is based on syntactic structure of a sentence and we call it "root/subtree preprocessing/postprocessing". The idea is based on the observation that tokens of a multi-token annotation usually have a common parent node in a syntactic tree. So we can

1. extract the parent nodes (in dependency linguistics this node is also a token and it is usually one of the tokens inside the annotation),
2. learn extraction rules for parent nodes only and
3. span annotations over the whole subtrees of root tokens found during the application of extraction rules.

We call the first point as *root preprocessing* and the last point as *subtree postprocessing*. We have successfully used this technique for the 'damage' task of our evaluation corpus (See Section 5.3 for details.)

5.2.7 Semantic Interpretation

Information extraction can solve the task "how to get documents annotated", but as we aim on the semantic annotation, there is a second step of "semantic interpretation" that has to be done. In this step we have to interpret the annotations in terms of a standard ontology. On a very coarse level this can be done easily. Thanks to GATE ontology tools [Bontcheva *et al.*, 2004] we can convert all the annotations to ontology instances with a quite simple JAPE [Cunningham *et al.*, 2000] rule, which takes the content of an

annotation and saves it as a label of a new instance or as a value of some property of a shared instance. For example in our case of traffic and fire accidents, there will be a new instance of an accident class for each document and the annotations would be attached to this instance as values of its properties. Thus from all annotations of the same type, instances of the same ontology class or values of the same property would be constructed. This is very inaccurate form of semantic interpretation but still it can be useful. It is similar to the GoodRelation [Hepp, 2008] design principle of *incremental enrichment*[5]:

> "...you can still publish the data, even if not yet perfect. The Web will do the rest – new tools and people."

But of course we are not satisfied with this fashion of semantic interpretation and we plan to further develop the semantic interpretation step as a sophisticated "annotation → ontology" transformation process that we have proposed in one of our previous works [Dědek and Vojtáš, 2008b].

5.2.8 How to Download

The project website[6] provides several ways how to get all the presented tools running. A platform independent installer, Java binaries and source codes are provided under the GPL license.

5.3 Evaluation

5.3.1 Dataset

We have evaluated our state of the art solution on a small dataset that we use for development. It is a collection of 50 Czech texts that are reporting on some accidents (car accidents and other actions of fire rescue services). These reports come from the web of Fire rescue service of Czech Republic[7]. The labeled corpus is publically available on the web of our project[8]. The corpus is structured such that each document represents one event (accident) and several attributes of the accident are marked in text. For the evaluation we selected two attributes of different kind. The first one is 'damage' – an amount (in CZK - Czech Crowns) of summarized damage arisen during a reported accident. The second one is 'injuries', it marks mentions of people injured during an accident. These two attributes differ. Injuries annotations always cover only a single token, while damage annotations usually consist of two or three tokens – one or two numerals express the amount and one extra token is for currency.

These two attributes differ in two directions:

1. Injuries annotations always cover only a single token while damage usually consists of two or three tokens - one or two numerals express the amount and one extra token is for currency.

2. The complexity of the marked information (and the difficulty of the corresponding extraction task) differs slightly. While labeling of all money amounts in the corpus will result in 75% accuracy for damage annotations, in the case of injured persons mentions there are much more possibilities and indications are more spread in context.

5.3.2 Comparison with Paum classifier

To compare our solution with other alternatives we took the Paum propositional learner from GATE [Li *et al.*, 2002]. The quality of propositional learning from texts is strongly dependent on the selection of right features. We obtained quite good results with features of a window of two preceding and two following token

[5] http://www.ebusiness-unibw.org/wiki/Modeling_Product_Models#Recipe:_.22Incremental_Enrichment.22
[6] http://czsem.berlios.de
[7] http://www.hzscr.cz/hasicien/
[8] http://czsem.berlios.de/

task/method	matching	missing	excessive	overlap	prec.%	recall%	F1.0%
damage/ILP	14	0	7	6	51.85	70.00	59.57
damage/ILP – lenient measures					74.07	100.00	85.11
dam./ILP-roots	16	4	2	0	88.89	80.00	84.21
damage/Paum	20	0	6	0	76.92	100.00	86.96
injuries/ILP	15	18	11	0	57.69	45.45	50.85
injuries/Paum	25	8	54	0	31.65	75.76	44.64
inj./Paum-afun	24	9	38	0	38.71	72.73	50.53

Table 5.1: Evaluation results

lemmas and morphological tags. The precision was further improved by adding the feature of *analytical function* from the syntactic parser (see the last row of Table 5.1).

Because we did not want to invest much time to this and the feature setting of the Paum learner was quite simple (a window of two preceding and following token lemmas and morphological tags). We admit that looking for better features could further improve the results of the Paum learner.

5.3.3 Cross validation

We used the 10-fold cross validation in the evaluation. Thanks to this technique the evaluation is simple. After processing all the folds each document is processed with some of the ten learned models such that the particular document was not used in learning of that model, so all documents are unseen by the model applied on them. At the end we just compare the gold standard annotations with the learned ones in all documents.

5.3.4 Results

Results of a 10-fold cross validation are summarized in Table 5.1. We used standard information retrieval performance measures: precision, recall and F_1 measure and also theirs lenient variants (overlapping annotations are added to the correctly matching ones, the measures are the same if no overlapping annotations are present).

In the first task ('damage') the methods obtained much higher scores then in the second ('injuries') because the second task is more difficult. In the first task also the root/subtree preprocessing/postprocessing improved results of ILP such that afterwards, annotation borders were all placed precisely. The ILP method had better precision and worse recall than the Paum learner but the F_1 score was very similar in both cases.

5.3.5 Examples of learned rules

In Figure 5.2 we present some examples of the rules learned from the whole dataset. The rules demonstrate a connection of a target token with other parts of a sentence through linguistic syntax structures. For example the first rule connects a root numeral (*n.quant.def*) of 'damage' with a mention of 'investigator' that stated the mount. In the last rule only a positive occurrence of the verb 'injure' is allowed.

Experience with human-designed rules.

5.4 Conclusion and Future Work

From our experiments can be seen that ILP is capable to find complex and meaningful rules that cover the intended information. But in terms of the performance measures the results are not better than those from a propositional learner. This is quite surprising observation because Czech is a language with free word order and we would expect much better results of the dependency approach than those of the position based approach, which was used by the propositional learner.

```
%[Rule 1] [Pos cover = 14 Neg cover = 0]
damage_root(A) :- lex_rf(B,A), has_sempos(B,'n.quant.def'), tDependency(C,B),
   tDependency(C,D), has_t_lemma(D,'investigator').

%[Rule 2] [Pos cover = 13 Neg cover = 0]
damage_root(A) :- lex_rf(B,A), has_functor(B,'TOWH'), tDependency(C,B),
   tDependency(C,D), has_t_lemma(D,'damage').


%[Rule 1] [Pos cover = 7 Neg cover = 0]
injuries(A) :- lex_rf(B,A), has_functor(B,'PAT'), has_gender(B,anim),
   tDependency(B,C), has_t_lemma(C,'injured').

%[Rule 8] [Pos cover = 6 Neg cover = 0]
injuries(A) :- lex_rf(B,A), has_gender(B,anim), tDependency(C,B),
   has_t_lemma(C,'injure'), has_negation(C,neg0).
```

Figure 5.2: Examples of learned rules, Czech words are translated.

Our method is still missing an intelligent semantic interpretation procedure and it should be evaluated on bigger datasets (e.g. MUC, ACE, TAC, CoNLL) and other languages. So far we also do not provide a method for classical relation extraction (like e.g. in [Bunescu and Mooney, 2007]). In the present solution we deal with relations implicitly. The method has to be adapted for explicit learning of relations in the form of "subject predicate object".

Our method can also provide a comparison of linguistic formalisms and tools because on the same data we could run our method using different linguistic analyzers and compare the results.

```
tToken(  id_jihomoravsky47443_243).
t_lemma( id_jihomoravsky47443_243, 'být'). %to be
functor( id_jihomoravsky47443_243, 'PRED').
sempos(  id_jihomoravsky47443_243, 'v').
tDependency( id_jihomoravsky47443_243, id_jihomoravsky47443_238).
tToken(  id_jihomoravsky47443_238).
t_lemma( id_jihomoravsky47443_238, ','). %comma
functor( id_jihomoravsky47443_238, 'APPS').
sempos(  id_jihomoravsky47443_238, 'n.denot').
gender(  id_jihomoravsky47443_238, 'nr').
tDependency( id_jihomoravsky47443_238, id_jihomoravsky47443_237).
tToken(  id_jihomoravsky47443_237).
t_lemma( id_jihomoravsky47443_237, 'vyčíslit'). %to quantify
functor( id_jihomoravsky47443_237, 'PAT').
sempos(  id_jihomoravsky47443_237, 'v').
tDependency( id_jihomoravsky47443_237, id_jihomoravsky47443_245).
tToken(  id_jihomoravsky47443_245).
t_lemma( id_jihomoravsky47443_245, 'předběžně'). %preliminarily
functor( id_jihomoravsky47443_245, 'MANN').
sempos(  id_jihomoravsky47443_245, 'adv.denot.grad.nneg').
tDependency( id_jihomoravsky47443_237, id_jihomoravsky47443_244).
tToken(  id_jihomoravsky47443_244).
t_lemma( id_jihomoravsky47443_244, 'vyšetřovatel'). %investigator
functor( id_jihomoravsky47443_244, 'ACT').
sempos(  id_jihomoravsky47443_244, 'n.denot').
gender(  id_jihomoravsky47443_244, 'anim').
tDependency( id_jihomoravsky47443_237, id_jihomoravsky47443_240).
tToken(  id_jihomoravsky47443_240).
t_lemma( id_jihomoravsky47443_240, 'osm'). %eight
functor( id_jihomoravsky47443_240, 'PAT').
sempos(  id_jihomoravsky47443_240, 'n.quant.def').
gender(  id_jihomoravsky47443_240, 'nr').
tDependency( id_jihomoravsky47443_240, id_jihomoravsky47443_242).
tToken(  id_jihomoravsky47443_242).
t_lemma( id_jihomoravsky47443_242, 'tisíc'). %thousand
functor( id_jihomoravsky47443_242, 'RSTR').
sempos(  id_jihomoravsky47443_242, 'n.quant.def').
gender(  id_jihomoravsky47443_242, 'inan').
tDependency( id_jihomoravsky47443_242, id_jihomoravsky47443_247).
tToken(  id_jihomoravsky47443_247).
t_lemma( id_jihomoravsky47443_247, 'koruna'). %crown
functor( id_jihomoravsky47443_247, 'MAT').
sempos(  id_jihomoravsky47443_247, 'n.denot').
gender(  id_jihomoravsky47443_247, 'fem').
tDependency( id_jihomoravsky47443_237, id_jihomoravsky47443_246).
tToken(  id_jihomoravsky47443_246).
t_lemma( id_jihomoravsky47443_246, 'škoda'). %damage
functor( id_jihomoravsky47443_246, 'PAT').
sempos(  id_jihomoravsky47443_246, 'n.denot').
gender(  id_jihomoravsky47443_246,'fem').
```

Listing 1: ILP serialization example

Chapter 6

Shareable Extraction Ontologies

6.1 Introduction

Information extraction (IE) and automated semantic annotation of text are usually done by complex tools and all these tools use some kind of a model that represents the actual task and its solution. The model is usually represented as a set of some kind of extraction rules (e.g. regular expressions), gazetteer lists or it is based on some statistical measurements and probability assertions (classification algorithms like Support Vector Machines (SVM), Maximum Entropy Models, Decision Trees, Hidden Markov Models (HMM), Conditional Random Fields (CRF), etc.)

In the beginning a model is either created by a human user or it is learned from a training dataset. Then, in an actual extraction/annotation process, the model is used as a configuration or as an input parameter of the particular extraction/annotation tool. These models are usually stored in proprietary formats and they are accessible only by the corresponding tool.

In the environment of the Semantic Web it is essential that information is shareable and some ontology based IE tools keep the model in so called extraction ontologies [Embley *et al.*, 2002]. Extraction ontologies should serve as a wrapper for documents of a narrow domain of interest. When we apply an extraction ontology to a document, the ontology identifies objects and relationships present in the document and it associates them with the corresponding ontology terms and thus wraps the document so that it is understandable in terms of the ontology [Embley *et al.*, 2002].

In practice the extraction ontologies are usually strongly dependent on a particular extraction/annotation tool and cannot be used separately. The strong dependency of an extraction ontology on the corresponding tool makes it very difficult to share. When an extraction ontology cannot be used outside the tool there is also no need to keep the ontology in a standard ontology format such as RDF or OWL. The only way how to use such extraction ontology is within the corresponding extraction tool. It is not necessary to have the ontology in a "owl or rdf file". In a sense such extraction ontology is just a configuration file. For example in [Labský *et al.*, 2009] (and also in [Embley *et al.*, 2002]) the so called extraction ontologies are kept in XML files with a proprietary structure and it is absolutely sufficient, there is no need to treat them differently.

6.1.1 Shareable Extraction Ontologies

In this paper we present an extension of the idea of extraction ontologies. We adopt the point that extraction models are kept in extraction ontologies and we add that the extraction ontologies should not be dependent on the particular extraction/annotation tool. In such case the extraction/annotation process can be done separately by an ordinary reasoner.

In this paper we present a proof of a concept for the idea: a case study with our linguistically based IE engine and an experiment with several OWL reasoners. In the case study (see Section 6.3) the IE engine exports its extraction rules to the form of an extraction ontology. Third party linguistic tool linguistically annotates an input document and the linguistic annotations are translated to so-called document ontology.

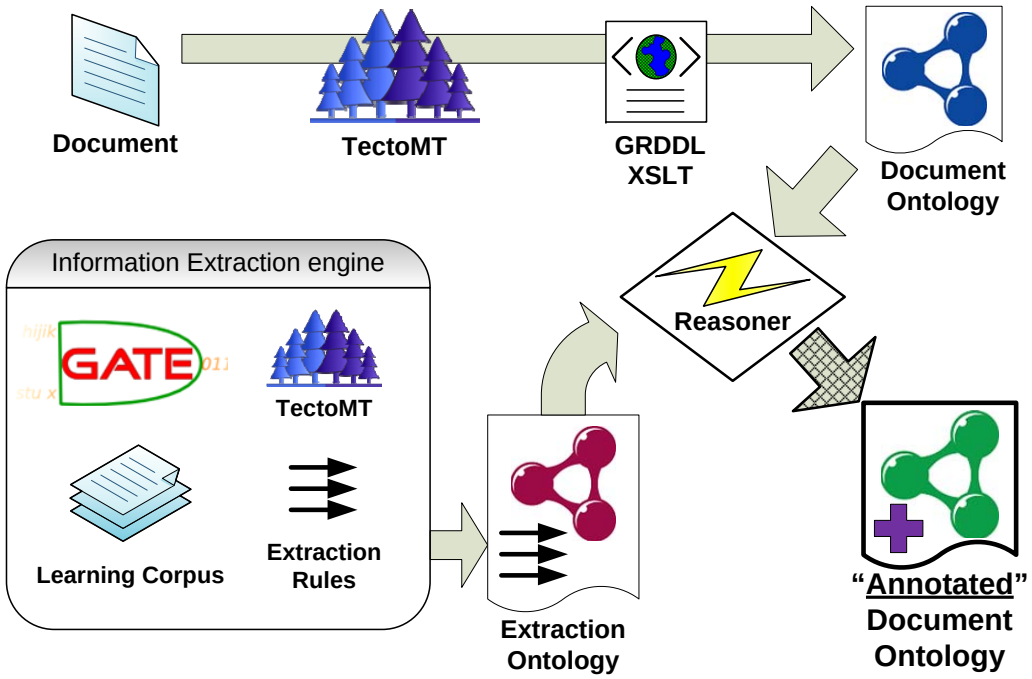


Figure 6.1: Semantic annotation driven by an extraction ontology and a reasoner – schema of the process.

After that an ordinary OWL reasoner is used to apply the extraction ontology on the document ontology, which has the same effect as a direct application of the extraction rules on the document. The process is depicted in Fig 6.1 and it will be described in detail in Section 6.3.2.

In Section 6.4 we present an experiment with several OWL reasoner and IE datasets to verify feasibility of the idea.

6.1.2 Contributions for Information Extraction

The paper combines the field of ontology-based information extraction and rule-based reasoning. The aim is to show a new possibility in usage of IE tools and reasoners. In this paper we do not present a solution that would improve the performance of IE tools.

We also do not provide a proposal of a universal extraction format (although a specific form for the rule based extraction on dependency parsed text could be inferred). This task is left for the future if a need for such activity emerges.

The main contribution and aim of the paper is a demonstration of the idea of tool independent extraction ontologies and the possibility to use reasoners for information extraction.

6.2 Related Work

Ontology-based Information Extraction (OBIE) [Wimalasuriya and Dou, 2010] or Ontology-driven Information Extraction [Yildiz and Miksch, 2007] has recently emerged as a subfield of information extraction. Furthermore, Web Information Extraction [Chang *et al.*, 2006b] is a closely related discipline. Many extraction and annotation tools can be found in the above mentioned surveys ([Wimalasuriya and Dou, 2010; Chang *et al.*, 2006b]), many of the tools also use an ontology as an output format, but almost all of them store their extraction models in proprietary formats and the models are accessible only by the corresponding tool.

In the literature we have found only two approaches that use extraction ontologies. The former one was published by D. Embley [Embley *et al.*, 2002; Embley, 2004] and the later one – IE system Ex[1] was

[1] `http://eso.vse.cz/~labsky/ex/`

developped by M. Labský [Labský *et al.*, 2009]. But in both cases the extraction ontologies are dependent on the particular tool and they are kept in XML files with a proprietary structure.

By contrast authors of [Wimalasuriya and Dou, 2010] (a recent survey of OBIE systems) do not agree with allowing for extraction rules to be a part of an ontology. They use two arguments against that:

1. Extraction rules are known to contain errors (because they are never 100% accurate), and objections can be raised on their inclusion in ontologies in terms of formality and accuracy.

2. It is hard to argue that linguistic extraction rules should be considered a part of an ontology while information extractors based on other IE techniques (such as SVM, HMM, CRF, etc. classifiers used to identify instances of a class when classification is used as the IE technique) should be kept out of it: all IE techniques perform the same task with comparable effectiveness (generally successful but not 100% accurate). But the techniques advocated for the inclusion of linguistic rules in ontologies cannot accommodate such IE techniques.

   The authors then conclude that either all information extractors (that use different IE techniques) should be included in the ontologies or none should be included.

Concerning the first argument, we have to take into account that extraction ontologies are not ordinary ontologies, it should be agreed that they do not contain 100% accurate knowledge. Also the estimated accuracy of the extraction rules can be saved in the extraction ontology and it can then help potential users to decide how much they will trust the extraction ontology.

Concerning the second argument, we agree that it is not always possible to save an extraction model to an ontology (at least not currently). But on the other hand we think that there are cases when shareable extraction ontologies can be useful and in the context of Linked Data[2] providing shareable descriptions of information extraction rules may be valuable. It is also possible that new standard ways how to encode models to an ontology will appear in the future.

6.2.1 Notes on Ontology Definitions

This short section briefly reminds main ontology definitions because they are touched and in a sense misused in this paper. The most widely agreed definitions of an ontology emphasize the shared aspect of ontologies:

> An ontology is a formal specification of a shared conceptualization. [Borst, 1997]

> An ontology is a formal, explicit specification of a shared conceptualization. [Studer *et al.*, 1998]

Of course the word 'shareable' has different meaning from 'shared'. (Something that is shareable is not necessarily shared, but on the other hand something that is shared should be shareable.) We do not think that shareable extraction ontologies will contain shared knowledge about how to extract data from documents in certain domain. This is for example not true for all extraction models artificially learned from a training corpus. Here shareable simply means that the extraction rules can be shared amongst software agents and can be used separately from the original tool. This is the deviation in use of the term 'ontology' in the context of extraction ontologies in this paper (similarly for document ontologies, see in Sect. 6.3.1).

6.3 The Main Idea Illustrated – a Case Study

In this section we will describe the main idea of the paper and we will illustrate it with a case study.

[2] `http://linkeddata.org/`

6.3.1 Document Ontologies

The main idea of this paper assumes that extraction ontologies will be shareable and they can be applied on a document outside of the original extraction/annotation tool. We further assert that the extraction ontologies can be applied by ordinary reasoners. This assumption implies that both extraction ontologies and documents have to be in a reasoner readable format. In the case of contemporary OWL reasoners there are standard reasoner-readable languages: OWL and RDF in a rich variety of possible serializations (XML, Turtle, N-Triples, etc.) Besides that there exists standard ways like GRDDL or RDFa how to obtain a RDF document from an "ordinary document" (strictly speaking XHTML and XML documents).

We call a 'document ontology' an ontology that formally captures content of a document. A document ontology can be for example obtained from the source document by a suitable GRDDL transformation (as in our experiment). A document ontology should contain all relevant data of a document and preferably the document could be reconstructed from the document ontology on demand.

When a reasoner is applying an extraction ontology to a document, it only has "to annotate" the corresponding document ontology, not the document itself. Here "to annotate" means to add new knowledge – new class membership or property assertions. In fact it means just to do the inference tasks prescribed by the extraction ontology on the document ontology.

Obviously when a document can be reconstructed from its document ontology (this is very often true, it is necessary just to save all words and formatting instructions) then also an annotated document can be reconstructed from its annotated document ontology.

6.3.2 Implementation

In this section we will present details about the case study. We have used our IE engine [Dědek, 2010] based on deep linguistic parsing and Inductive Logic Programming. It is a complex system implemented with a great help of the GATE system [Cunningham *et al.*, 2002] and it also uses many other third party tools including several linguistic tools and a Prolog system. Installation and making the system operate is not simple. This case study should demonstrate that the extraction rules produced by the system are not dependent on the system in the sense described above.

Linguistic Analysis

Our IE engine needs a linguistic preprocessing (deep linguistic parsing) of documents on its input. Deep linguistic parsing brings a very complex structure to the text and the structure serves as a footing for construction and application of extraction rules.

We usually use TectoMT system [Žabokrtský *et al.*, 2008] to do the linguistic preprocessing. TectoMT is a Czech project that contains many linguistic analyzers for different languages including Czech and English. We are using a majority of applicable tools from TectoMT: a tokeniser, a sentence splitter, morphological analyzers (including POS tagger), a syntactic parser and the deep syntactic (tectogrammatical) parser. All the tools are based on the dependency based linguistic theory and formalism of the Prague Dependency Treebank project (PDT) [Hajič *et al.*, 2006].

The output linguistic annotations of the TectoMT system are stored (along with the text of the source document) in XML files in so called Prague Markup Language (PML). PML is a very complex language (or XML schema) that is able to express many linguistic elements and features present in text. For the IE engine a tree dependency structure of words in sentences is the most useful one because the edges of the structure guide the extraction rules. An example of such (tectogrammatical) tree structure is in Fig. 6.2.

PML→RDF Transformation

In this case study PML files made from source documents by TectoMT are transformed to RDF document ontology by a quite simple GRDDL/XSLT transformation. Such document ontology contains the whole

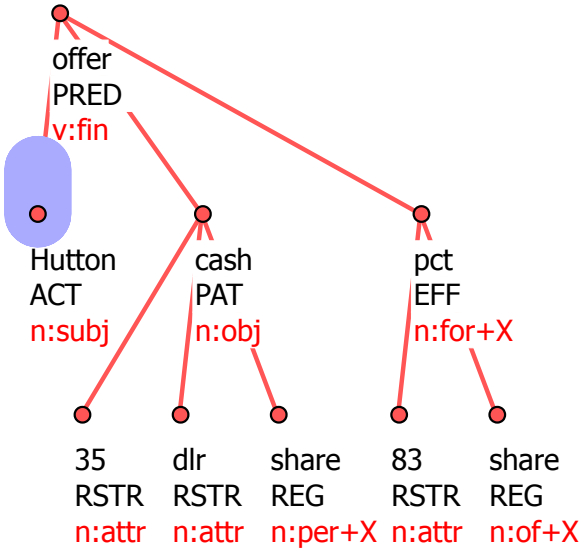


Figure 6.2: Tectogrammatical tree of the sentence: "Hutton is offering 35 dlrs cash per share for 83 pct of the shares." Nodes roughly correspond with words of a sentence, edges represent linguistic dependencies between nodes and some linguistic features (tectogrammatical lemma, semantic functor and semantic part of speech) are printed under each node. The node 'Hutton' is decorated as a named entity.

variety of PML in RDF format. An example of the transformation output is listed in Listing 2; the source linguistic tree is in Figure 7.3.

Rule Transformations

Extraction rules produced by the IE engine are natively kept in a Prolog format; examples can be seen in Fig. 6.3. The engine is capable to export them to the OWL/XML[3] syntax for rules in OWL 2 [Glimm *et al.*, 2009] (see in Fig. 6.5). Such rules can be parsed by OWL API[4] 3.1 and exported to RDF/SWRL[5] which is very widely supported and hopefully becoming a W3C recommendation. Fig. 6.4 shows the example rules in Protégé[6] 4 – Rules View's format. The last rule example can be seen in Fig. 6.6, it shows a rule in the Jena rules format[7]. Conversion to Jena rules was necessary because it is the only format that Jena can parse, see details about our use of Jena in Section 6.4. The Jena rules were obtained using following transformation process: OWL/XML → RDF/SWRL conversion using OWL API and RDF/SWRL → Jena rules conversion using SweetRules[8].

The presented rules belong to the group of so called DL-Safe rules [Motik *et al.*, 2005] so the decidability of OWL reasoning is kept.

Schema of the Case Study

A schema of the case study was presented in Fig. 6.1. The top row of the image illustrates how TectoMT (third party linguistic tool) linguistically annotates an input document and the linguistic annotations are translated to so-called document ontology by a GRDDL/XSLT transformation.

In the bottom of the picture our IE engine learns extraction rules and exports them to an extraction ontology. The reasoner in the middle is used to apply the extraction ontology on the document ontology and it produces the "annotated" document ontology, which was described in Section 6.3.1.

[3] http://www.w3.org/TR/owl-xmlsyntax/
[4] http://owlapi.sourceforge.net/
[5] http://www.w3.org/Submission/SWRL/
[6] http://protege.stanford.edu/
[7] http://jena.sourceforge.net/inference/#RULEsyntax
[8] http://sweetrules.semwebcentral.org/

```
%[Rule 1] [Pos cover = 23 Neg cover = 6]
mention_root(acquired,A) :-
        'lex.rf'(B,A), t_lemma(B,'Inc'),
        tDependency(C,B), tDependency(C,D),
        formeme(D,'n:in+X'), tDependency(E,C).
%[Rule 11] [Pos cover = 25 Neg cover = 6]
mention_root(acquired,A) :-
        'lex.rf'(B,A), t_lemma(B,'Inc'),
        tDependency(C,B), formeme(C,'n:obj'),
        tDependency(C,D), functor(D,'APP').
%[Rule 75] [Pos cover = 14 Neg cover = 1]
mention_root(acquired,A) :-
        'lex.rf'(B,A), t_lemma(B,'Inc'),
        functor(B,'APP'), tDependency(C,B),
        number(C,pl).
```

Figure 6.3: Examples of extraction rules in the native Prolog format.

```
#[Rule 1]
lex.rf(?b, ?a), t_lemma(?b, "Inc"),
tDependency(?c, ?b), tDependency(?c, ?d),
formeme(?d, "n:in+X"), tDependency(?c, ?e)
        -> mention_root(?a, "acquired")
#[Rule 11]
lex.rf(?b, ?a), t_lemma(?b, "Inc"),
tDependency(?c, ?b), formeme(?c, "n:obj"),
tDependency(?c, ?d), functor(?d, "APP")
        -> mention_root(?a, "acquired")
#[Rule 75]
lex.rf(?b, ?a), t_lemma(?b, "Inc"),
functor(?b, "APP"), tDependency(?c, ?b),
number(?c, "pl")
        -> mention_root(?a, "acquired")
```

Figure 6.4: Examples of extraction rules in Protégé 4 – Rules View's format.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Ontology [
    <!ENTITY pml "http://ufal.mff.cuni.cz/pdt/pml/" >
]>
<Ontology xmlns="http://www.w3.org/2002/07/owl#"
    ontologyIRI="http://czsem.berlios.de/onto ... rules.owl">
    <DLSafeRule>
        <Body>
            <ObjectPropertyAtom>
                <ObjectProperty IRI="&pml;lex.rf" />
                <Variable IRI="urn:swrl#b" />
                <Variable IRI="urn:swrl#a" />
            </ObjectPropertyAtom>
...
            <DataPropertyAtom>
                <DataProperty IRI="&pml;number" />
                <Variable IRI="urn:swrl#c" />
                <Literal>pl</Literal>
            </DataPropertyAtom>
        </Body>
        <Head>
            <DataPropertyAtom>
                <DataProperty IRI="&pml;mention_root" />
                <Literal>acquired</Literal>
                <Variable IRI="urn:swrl#a" />
            </DataPropertyAtom>
        </Head>
    </DLSafeRule>
</Ontology>
```

Figure 6.5: Rule 75 in the OWL/XML syntax for Rules in OWL 2 [Glimm *et al.*, 2009].

```
@prefix pml: <http://ufal.mff.cuni.cz/pdt/pml/>.
[rule-75:
        ( ?b pml:lex.rf ?a )
        ( ?c pml:tDependency ?b )
        ( ?b pml:functor 'APP' )
        ( ?c pml:number 'pl' )
        ( ?b pml:t_lemma 'Inc' )
     ->
        ( ?a pml:mention_root 'acquired' )
]
```

Figure 6.6: Rule 75 in the Jena rules syntax.

dataset	domain	language	number of files	dataset size	number of rules
czech_fireman	accidents	Czech	50	16 MB	2
acquisitions-v1.1	finance	English	600	126 MB	113

Table 6.1: Description of datasets that we have used.

How to Download

All the resources (including source codes of the case study and the experiment) mentioned in this demonstration are publically available on the web-page of our project[9] and detailed information can be found there.

6.4 Experiment

In this section we present an experiment that should serve as a proof of a concept that the proposed idea of independent extraction ontologies is working. We have selected several reasoners (namely Jena, HermiT, Pellet and FaCT++) and tested them on two slightly different datasets from two different domains and languages (see Table 6.1). This should at least partially demonstrate the universality of the proposed approach.

In both cases the task is to find all instances (corresponding to words in a document) that should be uncovered by the extraction rules. The extraction rules are saved in single extraction ontology for each dataset. The datasets are divided into individual document ontologies (owl files) corresponding to the individual documents. During the experiment the individual document ontologies are processed separately (one ontology in a step) by a selected reasoner. The total time taken to process all document ontologies of a dataset is the measured result of the reasoner for the dataset.

The actual reasoning tasks are more difficult than a simple retrieval of all facts entailed by the extraction rules. Such simple retrieval task took only a few seconds for the Acquisitions v1.1 dataset (including parsing) in the native Prolog environment that the IE engine uses. There were several more inferences needed in the reasoning tasks because the schema of the input files was a little bit different from the schema used in rules. The mapping of the schemas was captured in another "mapping" ontology that was included in the reasoning. The mapping ontology is a part of the publically available project ontologies[10] and a potentially interested reader can find the complete mapping ontology there.

6.4.1 Datasets

In the experiment we used two slightly different datasets from two different domains and languages. Table 6.1 summarizes some basic information about them.

czech_fireman

The fist dataset is called 'czech_fireman'. This dataset was created by ourselves during the development of our IE engine. It is a collection of 50 Czech texts that are reporting on some accidents (car accidents and other actions of fire rescue services). These reports come from the web of Fire rescue service of Czech Republic[11]. The labeled corpus is publically available on the website of our project[12]. The corpus is structured such that each document represents one event (accident) and several attributes of the accident are marked in text. For the experiment we selected the 'damage' task – to find an amount (in CZK - Czech Crowns) of summarized damage arisen during a reported accident.

[9] `http://czsem.berlios.de/`
[10] See "Data → ontologies" link on the project page `http://czsem.berlios.de/`
[11] `http://www.hzscr.cz/hasicien/`
[12] `http://czsem.berlios.de/`

reasoner	**czech_fireman**	stdev	**acquisitions-v1.1**	stdev
Jena	161 s	0.226	1259 s	3.579
HermiT	219 s	1.636	$\gg$ 13 hours	
Pellet	11 s	0.062	503 s	4.145
FaCT++	Does not support rules.			

Time is measured in seconds. Average values from 6 measurements. Experiment environment: Intel Core I7-920 CPU 2.67GHz, 3GB of RAM, Java SE 1.6.0_03, Windows XP.

Table 6.2: Time performance of tested reasoners on both datasets.

Acquisitions v1.1

The second dataset is called "Corporate Acquisition Events" and it is described in [Lewis, 1992]. More precisely we use the *Acquisitions v1.1* version[13] of the corpus. This is a collection of 600 news articles describing acquisition events taken from the Reuters dataset. News articles are tagged to identify fields related to acquisition events. These fields include 'purchaser' , 'acquired', and 'seller' companies along with their abbreviated names ('purchabr', 'acqabr' and 'sellerabr') Some news articles also mention the field 'deal amount'.

For the experiment we selected only the 'acquired' task.

6.4.2 Reasoners

In the experiment we used four OWL reasoners (namely Jena[14] ,HermiT[15] ,Pellet[16] and FaCT++[17]) and measured the time they needed to process a particular dataset. The time also includes time spend on parsing the input. HermiT, Pellet and FaCT++ were called through OWLAPI-3.1, so the same parser was used for them. Jena reasoner was used in its native environment and used the Jena parser.

In the early beginning of the experiment we had to exclude the FaCT++ reasoner from both tests. It turned out that FaCT++ does not work with rules[18] and it did not return any result instances. All the remaining reasoners strictly agreed on the results and returned the same sets of instances.

Also HermiT was not fully evaluated on the Acquisitions v1.1 dataset because it was too slow. The reasoner spent 13 hours of running to process only 30 of 600 files of the dataset. And we did not find it useful to let it continue.

6.4.3 Evaluation Results of the Experiment

Table 6.2 summarizes results of the experiment. The standard deviations are relatively small when compared to the differences between the average times. So there is no doubt about the order of the tested reasoners. Pellet performed the best and HermiT was the slowest amongst the tested and usable reasoners in this experiment.

From the results we can conclude that similar tasks can be satisfactorily solved by contemporary OWL reasoners because three of four tested reasoners were working correctly and two reasoners finished in bearable time.

On the other hand even the fastest system took 8.5 minutes to process 113 rules over 126MB of data.

[13]This version of the corpus comes from the Dot.kom (Designing infOrmation extracTion for KnOwledge Management) project's resources: `http://nlp.shef.ac.uk/dot.kom/resources.html`

[14]`http://jena.sourceforge.net`

[15]`http://hermit-reasoner.com`

[16]`http://clarkparsia.com/pellet`

[17]`http://code.google.com/p/factplusplus`

[18]`http://en.wikipedia.org/wiki/Semantic_reasoner#Reasoner_comparison`

This is clearly significantly longer than a bespoke system would require. Contemporary Semantic Web reasoners are known still to be often quite inefficient and the experiment showed that using them today to do information extraction will result in quite poor performance. However, efficiency problems can be solved and in the context of Linked Data providing shareable descriptions of information extraction rules may be valuable.

6.4.4 Repeatability

Our implementation is publicly available – source codes and the datasets can be downloaded from our project's web-page[19], so it should be also possible to repeat the experiment in a sense of the SIGMOD Experimental Repeatability Requirements [Manolescu *et al.*, 2008].

6.5 Future Work

In this paper (Section 6.3.1) we have described a method how to apply an extraction ontology to a document ontology and obtain so called "annotated" document ontology. To have an "annotated" document ontology is almost the same as to have an annotated document. An annotated document is useful (easier navigation, faster reading and lookup of information, possibility of structured queries on collections of such documents, etc.) but if we are interested in the actual information present in the document, if we want to know the facts that are in a document asserted about the real word things then an annotated document is not sufficient. But the conversion of an annotated document to the real world facts is not simple. There are obvious issues concerning data integration and duplicity of information. For example when in a document two mentions of people are annotated as 'injured', what is the number of injured people in the corresponding accident? Are the two annotations in fact linked to the same person or not?

In the beginning of our work on the idea of shareable extraction ontologies we planned to develop it further, we wanted to cover also the step from annotated document ontologies to the real world facts. The extraction process would then end up with so called "fact ontologies". But two main obstacles prevent us to do that.

1. Our IE engine is not yet capable to solve these data integration and duplicity of information issues. The real world facts would be quite imprecise then.

2. There are also technology problems of creating new facts (individuals) during reasoning.

Because of the decidability and finality constraints of the Description Logic Reasoning it is not possible to create new individuals during the reasoning process. There is no standard way how to do it. But there are some proprietary solutions like `swrlx:createOWLThing`[20] from the Protégé project and `makeTemp(?x)` or `makeInstance(?x, ?p, ?v)`[21] from the Jena project. And these solutions can be used in the future work.

6.5.1 SPARQL Queries – Increasing Performance?

There is also a possibility to transform the extraction rules to SPARQL[22] construct queries. This would probably rapidly increase the time performance. However a document ontology would then have to exactly fit with the schema of the extraction rules. This would be a minor problem.

The reason why we did not study this approach from the beginning is that we were interested in extraction *ontologies* and SPARQL queries are not currently regarded as a part of an ontology and nothing is suggesting it to be other way round.

Anyway the performance comparison remains a valuable task for the future work.

[19] `http://czsem.berlios.de/`
[20] `http://protege.cim3.net/cgi-bin/wiki.pl?action=browse&id=SWRLExtensionsBuiltIns`
[21] `http://jena.sourceforge.net/inference/#RULEbuiltins`
[22] `http://www.w3.org/TR/rdf-sparql-query/`

6.6 Conclusion – the Main Contributions

In the end of the paper we would like to summarize the main contributions of the paper.

- In the beginning of the paper we pointed out the draw back of so called extraction ontologies – in most cases they are dependent on a particular extraction/annotation tool and they cannot be used separately.
- We extended the concept of extraction ontologies by adding the shareable aspect and we introduced a new principle of making extraction ontologies independent of the original tool: the possibility of application of an extraction ontology to a document by an ordinary reasoner.
- In Section 6.3 we presented a case study that shows that the idea of shareable extraction ontologies is realizable. We presented implementation of an IE tool that exports its extraction rules to an extraction ontology and we demonstrated how this extraction ontology can be applied to a document by a reasoner.
- Moreover in Section 6.4 an experiment with several OWL reasoners was presented. The experiment evaluated the performance of contemporary OWL reasoners on IE tasks (application of extraction ontologies).
- A new publically available benchmark for OWL reasoning was created together with the experiment. Other reasoners can be tested this way.

We would like to conclude the paper by stating that only time will show if the fundamental idea of the paper will be useful but today it is at least a new use case for both: usage of IE tools and reasoners.

```
@prefix node: <http://czsem.berlios.de/ontologies/.../jihomoravsky47443.owl#node/> .
@prefix pml: <http://ufal.mff.cuni.cz/pdt/pml/> .

node:SCzechT-s4-n1 rdf:type pml:Node, owl:Thing;
                   pml:t_lemma "být" ; #to be
                   pml:sempos "v" ; pml:verbmod "ind" ;
                   pml:lex.rf node:SCzechA-s4-w4 ; pml:hasParent node:SCzechT-s4-root .
node:SCzechT-s4-n10 rdf:type pml:Node, owl:Thing;
                    pml:t_lemma "vyšetřovatel" ; #investigator
                    pml:negation "neg0" ; pml:sempos "n.denot" ;
                    pml:gender "anim" ; pml:number "sg" ;
                    pml:formeme "n:1" ; pml:functor "ACT" ;
                    pml:lex.rf node:SCzechA-s4-w9 ; pml:hasParent node:SCzechT-s4-n8 .
node:SCzechT-s4-n11 rdf:type pml:Node, owl:Thing;
                    pml:degcmp "pos" ; pml:t_lemma "předběžně" ; #preliminarily
                    pml:formeme "adv:" ; pml:sempos "adv.denot.grad.nneg" ;
                    pml:functor "MANN" ; pml:negation "neg0" ;
                    pml:lex.rf node:SCzechA-s4-w10 ; pml:hasParent node:SCzechT-s4-n8 .
node:SCzechT-s4-n12 rdf:type pml:Node, owl:Thing;
                    pml:t_lemma "osm" ; #eight
                    pml:number "pl" ; pml:numertype "basic" ;
                    pml:sempos "n.quant.def" ; pml:formeme "n:???" ;
                    pml:functor "PAT" ; pml:gender "nr" ;
                    pml:lex.rf node:SCzechA-s4-w13 ; pml:hasParent node:SCzechT-s4-n8 .
node:SCzechT-s4-n13 rdf:type pml:Node, owl:Thing;
                    pml:t_lemma "tisíc" ; #thousand
                    pml:number "sg" ; pml:functor "RSTR" ;
                    pml:gender "inan" ; pml:sempos "n.quant.def" ;
                    pml:numertype "basic" ; pml:formeme "n:???" ;
                    pml:lex.rf node:SCzechA-s4-w14 ; pml:hasParent node:SCzechT-s4-n12 .
node:SCzechT-s4-n14 rdf:type pml:Node, owl:Thing;
                    pml:t_lemma "koruna" ; #crown
                    pml:gender "fem" ; pml:sempos "n.denot" ;
                    pml:number "pl" ; pml:formeme "n:2" ;
                    pml:functor "MAT" ; pml:negation "neg0" ;
                    pml:lex.rf node:SCzechA-s4-w15 ; pml:hasParent node:SCzechT-s4-n13 .
node:SCzechT-s4-n7 rdf:type pml:Node, owl:Thing;
                   pml:t_lemma "," ; #comma
                   pml:gender "nr" ; pml:negation "neg0" ;
                   pml:sempos "n.denot" ; pml:functor "APPS" ;
                   pml:formeme "n:???" ; pml:number "nr" ;
                   pml:lex.rf node:SCzechA-s4-w7 ; pml:hasParent node:SCzechT-s4-n1 .
node:SCzechT-s4-n8 rdf:type pml:Node, owl:Thing;
                   pml:t_lemma "vyčíslit" ; #quantify
                   pml:is_member "1" ; pml:deontmod "decl" ;
                   pml:formeme "v:fin" ; pml:tense "ant" ;
                   pml:verbmod "ind" ; pml:aspect "cpl" ;
                   pml:is_clause_head "1" ; pml:functor "PAR" ;
                   pml:dispmod "disp0" ; pml:sempos "v" ;
                   pml:negation "neg0" ;
                   pml:lex.rf node:SCzechA-s4-w11 ; pml:hasParent node:SCzechT-s4-n7 .
node:SCzechT-s4-n9 rdf:type pml:Node, owl:Thing;
                   pml:t_lemma "škoda" ; #damage
                   pml:sempos "n.denot" ; pml:functor "PAT" ;
                   pml:gender "fem" ; pml:formeme "n:4" ;
                   pml:number "sg" ; pml:negation "neg0" ;
                   pml:lex.rf node:SCzechA-s4-w8 ; pml:hasParent node:SCzechT-s4-n8 .
```

Listing 2: RDF serialization example

Chapter 7

Usage of Annotations – Fuzzy ILP Document Classification

7.1 Introduction

The vast amount of information on the web increases the need of automated processing. Machine processing and machine understanding of textual information is especially difficult. In this paper we study the problem of classification of textual web reports. We are specifically focused on the situations in which structured information extracted from the reports is used for such classification. We present an experimental classification system based on usage of third party linguistic analyzers, our previous work on web information extraction and fuzzy inductive logic programming (fuzzy ILP).

The paper is based on a case study of seriousness classification of accident reports. Fig. 7.1 presents an example of an accident report from a message on the web. We would like to have a tool that is able to classify the accident's degree of seriousness in such a message.

Our solution is based on information extraction (see emphasized pieces of information that could be extracted from a report in Fig. 7.1) and on a fuzzy-based machine learning procedure that provides rules for classification of the reports.

Our experiments deal with texts in the Czech language but our method is general and it can be used with any structured linguistic representation. In this paper we do not provide many details about the information extraction part of the solution. We concentrate on the classification part and present a detailed study of the fuzzy ILP classification method (so-called 'Fuzzy ILP Classifier').

In our application we are facing the challenge of induction and/or mining on several occasions. First we

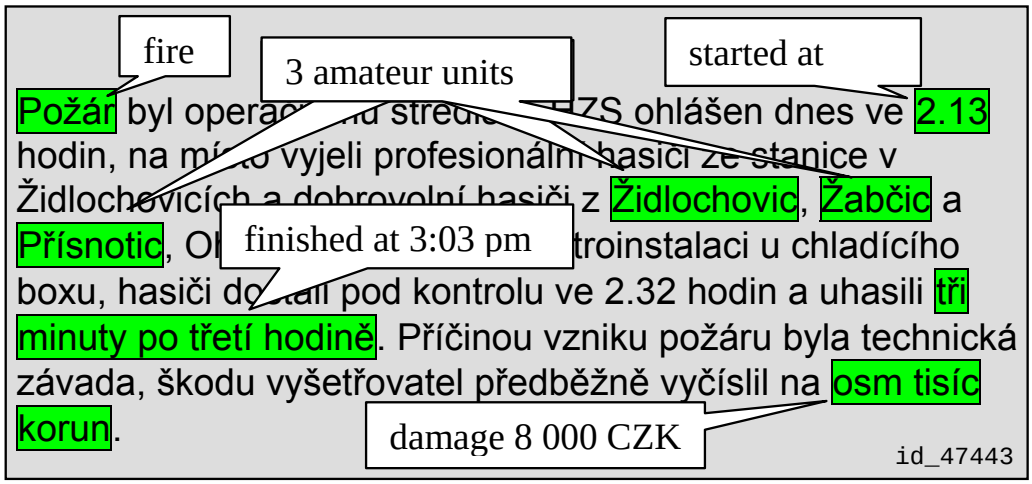


Figure 7.1: Example of analyzed web report.

need an inductive procedure when extracting attributes of an accident from text. Second (the subject of this paper) we need an inductive procedure when trying to explain an accident's degree of seriousness by its attributes. ILP is used as the inductive procedure in both of these places.

- During the information extraction phase, we exploit the fact that ILP can work directly with multirelational data, e.g., deep syntactic (tectogrammatical) trees built from sentences of the processed text.
- During the classification phase, we are experimentally using the Fuzzy ILP Classifier because it performs quite well on our dataset (see Section 7.6.1) and it is naturally capable of handling fuzzy data, which occurs when the information extraction engine returns confidence probability values along with extracted data. But in this paper we do not describe this situation; so only the approach is fuzzy in the present demonstration.

The main *contributions* of this paper are *formal models*, prototype *implementation* and experimental *evaluation* of the Fuzzy ILP Classifier.

The paper is organized as follows: In Section 7.2 we introduce some closely related works. Design of the experimental system is presented in Section 7.3, including a short description of the information extraction method and linguistic analyzers. Section 7.4 provides details about our case study, which is later used in examples. Formal models of the system (including ILP, fuzzy ILP and several translations of a fuzzy ILP task to classical ILP) are presented in Section 7.5, followed by a description of implementation of the models in the system. In Section 7.6 we present the main results of the work, and then we evaluate and compare the methods with other well-known classifiers. Section 7.7 concludes the paper.

7.2 Related Work

There are plenty of systems dealing with text mining and text classification. In [Reformat *et al.*, 2008] the authors use ontology modeling to enhance text identification. The authors of [Chong *et al.*, 2005] use preprocessed data from National Automotive Sampling System and test various soft computing methods to model severity of injuries (some hybrid methods showed best performance). Methods of Information Retrieval (IR) are numerous, with extraction mainly based on key word search and similarities. The Connection of IR and text mining techniques with web information retrieval can be found in the chapter "Opinion Mining in the book of Bing Liu [Liu, 2007b].

The Fuzzy ILP Classifier can be seen as an ordinary classifier for data with the monotonicity constraint (the target class attribute has to be monotonizable – a natural ordering has to exist for the target class attribute). There are several other approaches addressing the classification problems with the monotonicity constraint.

The CART-like algorithm for decision tree construction does not guarantee a resulting monotone tree even on a monotone dataset. The algorithm can be modified [Bioch and Popova, 2009] to provide a monotone tree on the dataset by adding the corner elements of a node with an appropriate class label to the existing data whenever necessary.

An interesting approach is presented in [Kotlowski and Slowinski, 2009]: first, the dataset is "corrected" to be monotone (a minimal number of target class labels is changed to get a monotone dataset), then a learning algorithm (linear programming boosting in the cited paper) is applied.

Several other approaches to monotone classification have been presented, including instance based learning [Lievens *et al.*, 2008] and rough sets [Bioch and Popova, 2000].

7.3 Design of the system

A general schema of our experimental system is shown in Fig. 7.2. We use our previously developed web information extraction tools based on third party linguistic analyzers (the upper two dashed arrows; this

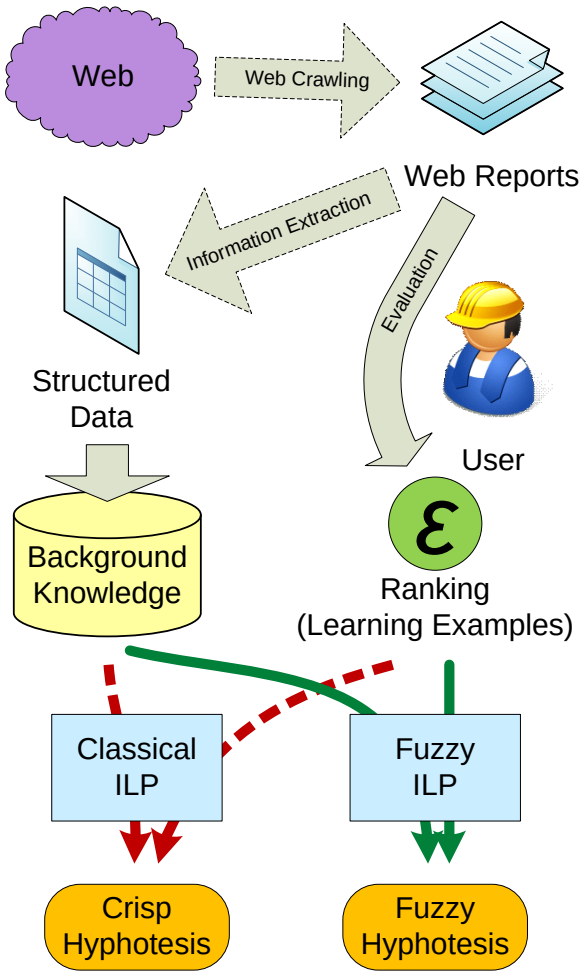


Figure 7.2: Schema of our system.

is not a subject of this paper). We extract some structured information from the web and the extracted information is then translated to an ILP knowledge base and, along with a user rating, it is used for the classification (we assume that a small amount of learning data is annotated by a human user). The classification is based on ILP and it could be *fuzzy* or *crisp*. Crisp denotes a straightforward application of ILP and fuzzy stands for the fuzzy method (subject of the paper), see in the next sections.

7.3.1 Linguistic Analysis

In this section we will briefly describe the linguistic tools that we have used to produce linguistic annotations of texts. These tools are being developed in the Institute of Formal and Applied Linguistics in Prague, Czech Republic. They are publicly available – they have been published on a CDROM under the title PDT 2.0 (Hajič *et al.* [2006] – the first five tools) and in (Klimeš [2006] – Tectogrammatical analysis). These tools are used as a processing chain. At the end of the chain they produce tectogrammatical dependency trees [Mikulová *et al.*, 2006] built up from the text.

Tool 1. Segmentation and tokenization consists of dividing the input text into words and punctuation (tokenization) and dividing a sequence of tokens into sentences (segmentation).

Tool 2. Morphological analysis assigns all possible lemmas and morphological tags to particular word forms (word occurrences) in the text.

Tool 3. Morphological tagging consists in selecting a single pair lemma-tag from all possible alternatives assigned by the morphological analyzer.

Tool 4. Collins' parser – Czech adaptation. Unlike the usual approaches to the description of English syntax, the Czech syntactic descriptions are dependency-based, which means that every edge of a syntactic tree captures the relation of dependency between a governor and its dependent node. Collins' parser gives the most probable parse of a given input sentence.

Tool 5. Analytical function assignment assigns a description (analytical function, in linguistic sense) to every edge in the syntactic (dependency) tree.

Tool 6. Tectogrammatical analysis produces linguistic annotation at the tectogrammatical level, sometimes called "layer of deep syntax". An example of a tectogrammatical tree can be seen in Fig. 7.3. Annotation of a sentence at this layer is closer to the meaning of the sentence than its syntactic annotation; thus, information captured at the tectogrammatical layer is crucial for machine understanding of natural language [Klimeš, 2006].

7.3.2 Web Information Extraction

After the content of a web resource is analyzed by the above-mentioned linguistic tools, the output linguistic data is obtained in the form of tectogrammatical trees. To achieve our objectives we have to extract information from this representation. Here we refer to our previous work [Dědek and Vojtáš, 2008a; Dědek *et al.*, 2008b]. A long path of tools, starting with web crawling and resulting in the extracted structured information, has been developed in our previous works. In Fig. 7.3, nodes of the tectogrammatical tree are decorated. A piece of information about the damage of 8000 CZK can be found there (the three nodes on the right). We have used ILP to learn rules that are able to detect these nodes. The extraction process requires human assistance when annotating the training data.

Note that our method is general and is not limited to Czech. It can be used with any structured linguistic representation.

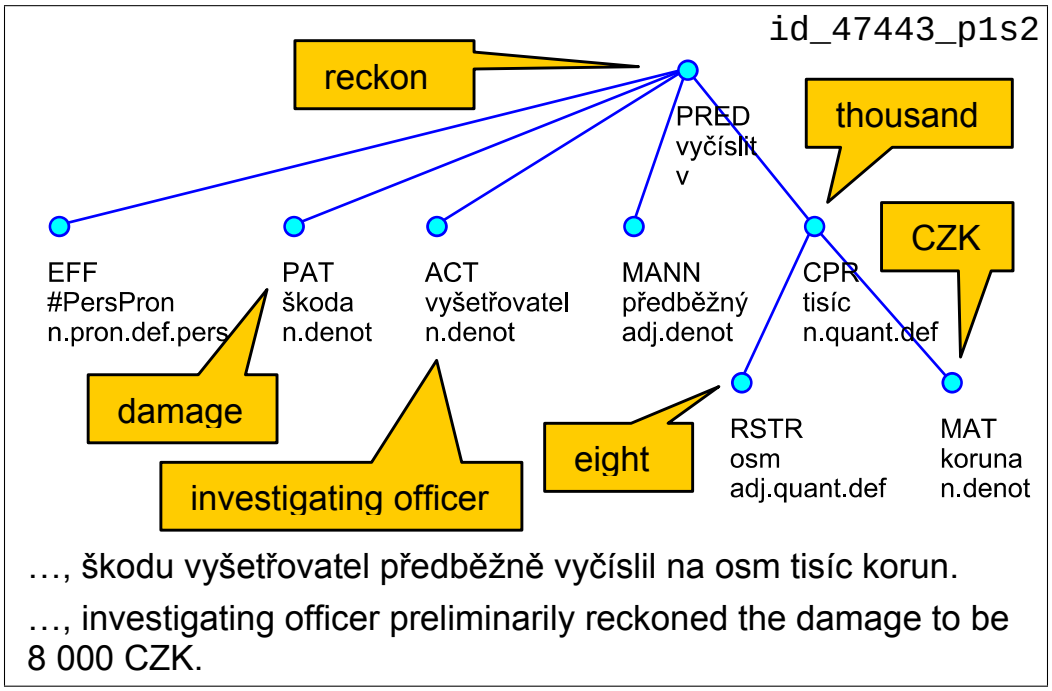


Figure 7.3: Example of a linguistic tree of one analyzed sentence.

7.4 The case study – accident seriousness classification

The main experiment presented in this paper leads to the seriousness classification of an accident presented in a web report. We use online fire department reports from several regions of the Czech Republic. These reports are written in Czech and can be accessed through the web of the General Directorate of the Fire and Rescue Service of the Czech Republic[1].

For our experiment we have selected a collection of 50 web reports. We have identified several features presented in these reports and manually extracted the corresponding values. This will be described in more detail in Section 7.4.1. To each report we have also assigned a value of overall ranking of seriousness of the presented accident, which is the target of the classification. The whole dataset can be downloaded from our Fuzzy ILP classifier's web page[2].

In this experiment we have not used any information extracted by our automated information extraction tools. Instead, we concentrate on the classification; the actual source of the information is not so important. The integration step still lies ahead.

7.4.1 Features of accidents

Table 7.1 summarizes all features (or attributes) that we have obtained from accident reports. Except for the attribute `type` (type of an accident – `fire`, `car_accident` and `other`), all the attributes are numerical and therefore *monotonizable* (see the next sections). There are cases in which the value of an attribute is unknown. We have decided to make evidence of this and keep the values `unknown` in the knowledge base. A brief explanation of each attribute follows.

- `size` is length of text of a particular report.
- `damage` is an amount (in CZK – Czech Crowns) of summarized damage arisen during a reported accident.
- `dur_minutes` is time taken to handle an accident.

[1] `http://www.hzscr.cz`
[2] `http://www.ksi.mff.cuni.cz/~dedek/fuzzyILP/`

attribute name	distinct values	missing values	monotonic
size (of file)	49	0	yes
type (of accident)	3	0	no
damage	18	30	yes
dur_minutes	30	17	yes
fatalities	4	0	yes
injuries	5	0	yes
cars	5	0	yes
amateur_units	7	1	yes
profesional_units	6	1	yes
pipes	7	8	yes
lather	3	2	yes
aqualung	3	3	yes
fan	3	2	yes
ranking	14	0	yes

Table 7.1: Accident attributes.

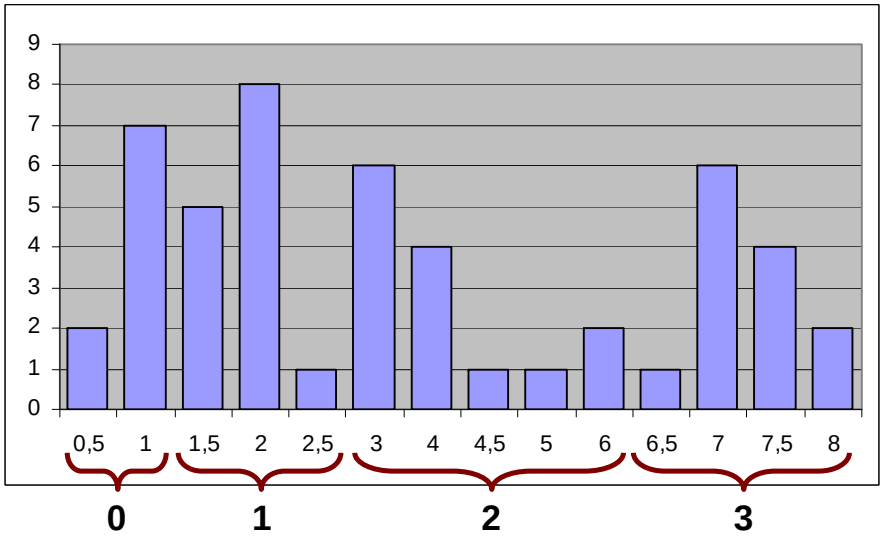


Figure 7.4: Frequencies of the seriousness ranking.

- `fatalities` and `injuries` are numbers of deaths and wound people sustained in an accident.
- `cars` is the number of vehicles damaged during an accident (mostly during car accidents).
- `professional_units` and `amateur_units` are numbers of fireman and volunteer units sent for a particular accident.
- `pipes` is a number of used fire hoses.
- `lather`, `aqualung` and `fan` (ventilator) indicates whether these devices were used.

A majority of accidents are of the type `fire` (52%) and `car_accident` (30%), the rest (type `other`, 18%) deals with ecological disasters, chemical accidents, etc.

7.4.2 Seriousness ranking

Values of the overall seriousness ranking attribute were stated from "a general impression" made by the texts with respect to particular attributes. The values have evolved to 14 distinct values in the range from 0.5 to

8. A histogram with frequencies of all these values is in Figure 7.4. We have divided the values into four approximately equipotent groups (see in Fig. 7.4) and these groups determine the target class attribute of the classification task.

7.5 ILP models and methods

To make the paper easier to read, we present a short description of the ILP techniques below.

7.5.1 Classical ILP

In our presentation of ILP we follow Dzeroski and Lavrac [2001] and Muggleton and de Raedt [1994].

Definition 1 (Classical ILP task)**.** A set of examples $E = P \cup N$, where P contains positive and N negative examples, and background knowledge denoted by B are given. The task of ILP is to find a hypothesis H such that

$$(\forall e \in P)(B \cup H \models e)$$

and

$$(\forall e \in N)(B \cup H \not\models e).$$

Typically, E consists of ground instances of the target predicate, in our case accident seriousness (see examples in Fig. 7.5). B typically consists of several predicates (relational tables), which describe properties of an object, in our case properties of an accident (see examples in Fig. 7.6). The background knowledge can also contain some rules. A hypothesis H typically consists of logic programming rules (see examples in Fig. 7.9). H added to B entails all positive examples and no negative examples. The main advantage of ILP is its multirelational character, namely, B can reside in several relational tables.

7.5.2 Fuzzy and GAP induction

In our presentation of fuzzy ILP we follow the paper of T. Horváth and P. Vojtáš [Horváth and Vojtáš, 2007] about fuzzy inductive logic programming. We use the approach of the fuzzy logic in a narrow sense, developed by J. Pavelka [Pavelka, 1979] and P. Hájek [Hájek, 1998]. Formulas are of the form φ, x (φ is syntactically the same as in the classical case), they are graded by a truth value $x \in [0, 1]$. A structure $\mathcal{M}$ consists of a domain M and relations are interpreted as fuzzy (we do not consider function symbols here). The evaluation $\|\varphi\|_{\mathcal{M}}$ of a formula φ uses truth functions of many-valued connectives (our logic is extensional and/or truth functional). The satisfaction $\models_f$ is defined by

$$\mathcal{M} \models_f (\varphi, x) \; iff \; \|\varphi\|_{\mathcal{M}} \geq x.$$

Definition 2 (Fuzzy ILP task)**.** A fuzzy set of examples $\mathcal{E} : E \longrightarrow [0, 1]$ and a fuzzy background knowledge $\mathcal{B} : B \longrightarrow [0, 1]$ are given. The task of fuzzy ILP is to find a fuzzy hypothesis $\mathcal{H} : H \longrightarrow [0, 1]$ such that

$$(\forall e_1, e_2 \in E)(\forall \mathcal{M})(\mathcal{M} \models_f \mathcal{B} \cup \mathcal{H})$$

we have

$$\mathcal{E}(e_1) > \mathcal{E}(e_2) \Rightarrow \|e_1\|_{\mathcal{M}} \geq \|e_2\|_{\mathcal{M}}.$$

That is, it cannot happen that

$$\mathcal{E}(e_1) > \mathcal{E}(e_2) \wedge \|e_1\|_{\mathcal{M}} < \|e_2\|_{\mathcal{M}},$$

or rephrased: if $\mathcal{E}$ is rating e_1 higher than e_2, then it cannot happen that e_1 is rated worse than e_2 in a model of $\mathcal{B} \cup \mathcal{H}$.

Typically, $\mathcal{E}$ consists of ground instances of the target predicate, which are classified in truth degrees, in our case degrees of seriousness of an accident. $\mathcal{B}$ typically consists of several fuzzy predicates (fuzzy relational tables), which describe properties of an object, in our case fuzzy properties of an accident – a degree of injury, a degree of damage, etc. A hypothesis $\mathcal{H}$ typically consists of a fuzzy logic program, which, when added to $\mathcal{B}$, prevents misclassification (what is better cannot be declared to be worse, nevertheless it can be declared as having the same degree – for more detailed discussion on this definition of fuzzy ILP we refer to the paper [Horváth and Vojtáš, 2007]).

Moreover, we use GAP – Generalized Annotated Programs – in practice, so graded formulas will sometimes be understood as annotated (with classical connectives and with a more complex annotation of the heads of rules). This is possible because in [Krajci *et al.*, 2004] we have shown that (some extension of) fuzzy logic programming is equivalent to (some restriction of) generalized annotated programs.

7.5.3 Translation of fuzzy ILP task to several classical ILP tasks

As far as there is no implementation of fuzzy ILP or GAP, we have to use a classical ILP system. Fortunately, any fuzzy ILP task can be translated to several classical ILP tasks (subject to some rounding and using a finite set of truth values).

In the following text, assume that all fuzzy sets take truth values only from a finite set of truth values $T : \{0, 1\} \subseteq T \subseteq [0, 1]$.

Definition 3 (Transformation of background knowledge)**.** A fuzzy background knowledge $\mathcal{B} : B \longrightarrow [0, 1]$ is given. For each predicate $p(x)$ in B we insert an additional attribute t to express the truth value, thus we have created a new predicate $p(x, t)$. We construct two classical background knowledge sets B_T^{raw} and B_T^{mon} as follows:

- The first (B_T^{raw}) is a direct coding of a fuzzy value by an additional attribute:
  If $\mathcal{B}(p(x)) = t,\ t \in T$, then we add $p(x, t)$ to B_T^{raw}.
- The second (B_T^{mon}) is obtained by a process called monotonization:
  If $\mathcal{B}(p(x)) = t,\ t \in T$, then for all $t' \in T,\ t' \leq t$ we add $p(x, t')$ to B_T^{mon}. This corresponds to the natural meaning of truth values t.

Additionally, example sets are constructed in two ways.

Definition 4 (Transformation of examples)**.** Given is a fuzzy set of examples $\mathcal{E} : E \longrightarrow [0, 1]$. For all $t \in T$ we construct two classical sets of examples E_t and $E_{\geq t}$ as follows:

- $E_t = P_t \cup N_t$, where $e \in P_t \ \ iff \ \ \mathcal{E}(e) = t$ and N_t is the rest of E.
- $E_{\geq t} = P_{\geq t} \cup N_{\geq t}$, where $e \in P_{\geq t} \ \ iff \ \ \mathcal{E}(e) \geq t$ and N_t is the rest of E.

These two translations create two classical ILP tasks for each truth value $t \in T$, the first one (*raw*) is crisp and the second one (*mon*) can be understood as (and translated back to) fuzzy ILP.

- The *raw ILP task* is given by B_T^{raw} and E_t for each $t \in T$. As a result, it produces a set of hypotheses H_t.
- The *mon ILP task* is given by B_T^{mon} and $E_{\geq t}$ for each $t \in T$. As a result, it produces a set of hypotheses $H_{\geq t}$ guaranteeing examples of a degree of at least t.

Note that among variable boundings in B there are no boundings on the truth value attribute which was added to each predicate; hence, there are no variable boundings on the truth value attribute in $H_{\geq t}$. We did not add an additional truth value attribute to the predicates in E.

Now we sketch the translation of the *mon ILP task* to GAP (fuzzy ILP) rules.

Theorem 1 (Translation of the *mon ILP task*)**.** A fuzzy ILP (or equivalent GAP) task is given by $\mathcal{E}$ and $\mathcal{B}$. Let us assume that C is the target predicate in the domain of $\mathcal{E}$ and for each $t \in T$, $H_{\geq t}$ is a correctly learned solution of the corresponding *mon ILP task* according to the definitions introduced above. We define $\mathcal{H}$ consisting of one GAP rule:

$$C(y) : u(x_1, \ldots, x_m) \leftarrow B_1(y) : x_1 \,\&\ldots\&\, B_m(y) : x_m,$$

here $B_1 : x_1 \& \ldots \& B_m : x_m$ is the enumeration of all predicates in B.

Assume that $B_1(y_1, t_1), \ldots, B_n(y_n, t_n)$ are some of the predicates in B (for simplicity enumerated from 1 to n, $n \leq m$). Then for each rule

$$R = C_{\geq t}(y) \Leftarrow B_1(y, t_1), \ldots, B_n(y, t_n)$$

in $H_{\geq t}$ ($C_{\geq t}$ is the monotonized target predicate) we give a constraint in the definition of u as follows:

$$U_R = u(x_1, \ldots, x_m) \geq t \text{ if } x_1 \geq t_1, \ldots, x_n \geq t_n.$$

Note that all x_i and y are variables, t_i and t are constants and $x_{n+1}, \ldots, x_m$ have no restrictions. We learn the function u as a "monotone" completion of the rules.

We claim that if all $H_{\geq t}$ were correctly learned by a classical ILP system, then, for the minimal solution u of all constraints U_R, the rule

$$C(y) : u(x_1, \ldots, x_m) \leftarrow B_1(y) : x_1 \,\&\ldots\&\, B_m(y) : x_m$$

is a correct solution of the fuzzy ILP task given by $\mathcal{E}$ and $\mathcal{B}$, for all $R \in H_{\geq t}$ and $t \in T$.

In our case `serious(Accident_ID)` is the fuzzy or GAP target predicate C. Let for example $t = 3$ and $H_{\geq t}$ is the same as in Fig. 7.9 (the last two rows correspond with $H_{\geq 3}$). Then `serious_atl_3(Accident_ID)` is the monotonized target predicate $C_{\geq t}$ and B_1, B_2 and u are realized as follows:

$B_1(y, t_1)$ …`fatalities_atl(Accident_ID, 1)`,

$B_2(y, t_2)$ …`damage_atl(Accident_ID, 1500000)`,

u is an arbitrary function of m arguments (m is the number of all predicates used in background knowledge) for which following restrictions hold true:
for $t = 3$:

$u(x_1, \ldots, x_m) \geq 3$ if $x_1 \geq 1$

$u(x_1, \ldots, x_m) \geq 3$ if $x_2 \geq 1500000$

and similar restrictions for $t = 1, 2$.

Our presentation is slightly simplified here and we freely switch between fuzzy and GAP programs, which are known to be equivalent, see in [Krajci *et al.*, 2004].

7.5.4 Implementation

In our experimental system we use two inductive logic approaches: crisp and fuzzy (as described above). Technically, the difference between the approaches consists in a different setting of the *ILP task*. Both can be done with a classical ILP tool. We have used "*A Learning Engine for Proposing Hypotheses*" (Aleph v5[3]), which we consider very practical. It uses the quite effective method of *inverse entailment* [Muggleton, 1995] and keeps all handy features of a Prolog system (it is supported by YAP and SWI Prolog) in its background.

We have compared results of the crisp and fuzzy approaches with other classification methods and, in our experiment, the fuzzy approach produced better results than many other methods, including the crisp one. See Section 7.6 for details.

[3] `http://www.comlab.ox.ac.uk/activities/machinelearning/Aleph/`

Crisp learning examples

```
serious_2(id_47443). %positive

serious_0(id_47443). %negative
serious_1(id_47443). %negative
serious_3(id_47443). %negative
```

Monotonized learning examples

```
serious_atl_0(id_47443). %positive
serious_atl_1(id_47443). %positive
serious_atl_2(id_47443). %positive

serious_atl_3(id_47443). %negative
```

Figure 7.5: Learning examples.

```
size(id_47443, 427).
type(id_47443, fire).
damage(id_47443, 8000).
dur_minutes(id_47443, 50).
fatalities(id_47443, 0).
injuries(id_47443, 0).
cars(id_47443, 0).
amateur_units(id_47443, 3).
profesional_units(id_47443, 1).
pipes(id_47443, unknown).
lather(id_47443, 0).
aqualung(id_47443, 0).
fan(id_47443, 0).
```

Figure 7.6: B_T^{raw} – crisp attributes.

```
damage_atl(ID,N) :- damage(ID,N), not(integer(N)). %unknown values

damage_atl(ID,N) :- damage(ID,N2), integer(N2), %numeric values
                    damage(N), integer(N), N2>=N.
```

Figure 7.7: Monotonization of attributes (damage_atl ← damage).

To use ILP for a classification task we have to translate the input data to the Prolog-like logic representation, as it was already described in previous sections. Here we will describe implementation details of constructing crisp and fuzzy knowledge bases and example sets.

In construction of a crisp example set E_t, the target predicate is denoted as **serious_t**. The letter **t** stands for the actual seriousness degree. We use multiple unary predicates **serious_0**, **serious_1**, etc., instead of one binary predicate **serious(ID,Degree)**. These two cases are equivalent and we have decided to use the unary option because a visual distinction between the multiple ILP tasks is then clearer.

In construction of a fuzzy (or monotonized) example set $E_{\geq t}$, the target predicate is denoted as **serious_atl_t**, see examples in Fig. 7.5.

In construction of a crisp background knowledge B_T^{raw}, we use a simple translation of the attribute names to the names of predicates and fill them with actual values. It is illustrated in Fig. 7.6).

In construction of a monotonized background knowledge B_T^{mon} we reuse the crisp background knowledge and add monotonization rules. An example for predicate **damage** is shown in Fig. 7.7. The first rule deals with **unknown** values and the second does the monotonization.

```
serious_0(ID) :- serious_atl_0(ID),
                 not(serious_atl_1(ID)), not(serious_atl_2(ID)), not(serious_atl_3(ID)).
serious_1(ID) :- serious_atl_1(ID),
                 not(serious_atl_2(ID)), not(serious_atl_3(ID)).
serious_2(ID) :- serious_atl_2(ID),
                 not(serious_atl_3(ID)).
serious_3(ID) :- serious_atl_3(ID).
```

Figure 7.8: Conversion rules for monotonized hypotheses (**serious_t** ← **serious_atl_t**).

Negations used in Fig. 7.7 and Fig. 7.8 are the standard Prolog *negations as failure*.

```
serious_0(A) :- fatalities(A,0), injuries(A,0), cars(A,1), amateur_units(A,0), lather(A,0).
serious_0(A) :- fatalities(A,0), cars(A,0), amateur_units(A,0), professional_units(A,1).
serious_1(A) :- amateur_units(A,1).
serious_1(A) :- damage(A,300000).
serious_1(A) :- type(A,fire), amateur_units(A,0), pipes(A,2).
serious_1(A) :- type(A,car_accident),dur_minutes(A,unknown),fatalities(A,0),injuries(A,1).
serious_2(A) :- lather(A,unknown).
serious_2(A) :- cars(A,0), lather(A,0), aqualung(A,1), fan(A,0).
serious_2(A) :- amateur_units(A,2).
serious_3(A) :- fatalities(A,2).
serious_3(A) :- type(A,fire), dur_minutes(A,unknown), cars(A,0), fan(A,0).
serious_3(A) :- injuries(A,2), cars(A,2).
serious_3(A) :- fatalities(A,1).

serious_atl_0(A).
serious_atl_1(A) :- injuries_atl(A,1).
serious_atl_1(A) :- dur_minutes_atl(A,21), pipes_atl(A,1), aqualung_atl(A,0).
serious_atl_1(A) :- damage_atl(A,8000), amateur_units_atl(A,3).
serious_atl_1(A) :- dur_minutes_atl(A,197).
serious_atl_1(A) :- dur_minutes_atl(A,unknown).
serious_atl_2(A) :- dur_minutes_atl(A,50), pipes_atl(A,3).
serious_atl_2(A) :- size_atl(A,1364), injuries_atl(A,1).
serious_atl_2(A) :- fatalities_atl(A,1).
serious_atl_2(A) :- size_atl(A,1106), professional_units_atl(A,3).
serious_atl_3(A) :- fatalities_atl(A,1).
serious_atl_3(A) :- damage_atl(A,1500000).
```

Figure 7.9: Crisp and monotonized hypotheses.

Once we have learning examples and background knowledge, we can run the ILP inductive procedure and obtain learned rules (a learned hypothesis). According to the kind of the ILP task (crisp or monotonized), we obtain the corresponding kind (crisp or monotonized) of rules (see e.g. in Fig. 7.9). But these rules cannot be used directly to solve the classification task. There are common cases when more than one rule is applicable to a single instance. So we have to select which one to use. For the monotonized hypothesis we select the one with the biggest return value. It is illustrated in Fig. 7.8. Such clear criterion does not exist for the crisp hypothesis, so we simply use the first applicable rule.

In the crisp case there are often many instances which cannot be classified because there is no applicable rule. In our experiment there was about a 51% of unclassified instances (see in the next section). It could be caused by the lack of training data, but the monotonized approach does not suffer from this shortage. We can always select the bottom value.

Another advantage of the monotonized approach is that the set of positive training examples is extended by monotonization.

7.6 Results

Fig. 7.9 summarizes obtained hypotheses learned from our data:

- a crisp hypothesis learned from E_t and B_T^{raw} and
- a monotonized hypothesis learned from $E_{\geq t}$ and B_T^{mon}.

In both cases, learning examples and background knowledge have the origin in the same data (the same accidents). The hypotheses differ in the form of the ILP task (crisp and monotonized). The crisp hypothesis uses only the crisp predicates, and the monotonized hypothesis uses only the monotonized predicates.

7.6.1 Evaluation

We have evaluated both ILP methods and compared them with other machine learning procedures used in data mining. To make the comparison clear and easy to perform, we have implemented an interface between the ILP methods and the Weka data mining software[4] [Hall *et al.*, 2009]. This interface makes it possible to use the ILP methods as an ordinary Weka classifier for any[5] classification task inside the Weka software. Our implementation is publicly available. The data, source codes and a platform-independent installer of the Crisp and Fuzzy ILP classifiers for Weka can be downloaded from our Fuzzy ILP classifier's web page[6]. This makes our experiment repeatable according to the SIGMOD Experimental Repeatability Requirements [Manolescu *et al.*, 2008].

For our experiment we used the Weka Experimenter and performed an experiment in which the Crisp and Fuzzy ILP classifiers were compared with five additional classifiers:

- Multilayer Perceptron [Bishop, 1996],
- Support Vector Machine classifier SMO [Keerthi *et al.*, 2001],
- J48 decision tree [Quinlan, 1993],
- JRip rules [Cohen, 1995] and
- Additive logistic regression LogitBoost [Friedman *et al.*, 2000].

We have evaluated all the methods two times by 10-fold cross validation. The obtained results (average values) are described by the graph in Fig. 7.10 and in Table 7.2 (with standard deviations and marked statistically significant values).

There is no clear winner in our experiment. But the Fuzzy ILP classifier proved better results than a majority of the methods on our data and the results are statistically significant in many cases. Very good results were also obtained using LogitBoost.

UCI datasets

The Fuzzy ILP classifier performed quite well on our dataset, but the next question is: How is it with other data, with more learning instances, and what about the time complexity? To answer these questions we performed another experiment. We selected several datasets from the University of California, Irvine, Repository of Machine Learning Databases (UCI) [Frank and Asuncion, 2010] and evaluated all the methods against them.

The list of selected datasets can be found in the legend of Table 7.3. All the datasets are monotonizable (the target attribute can be naturally ordered), so the fuzzy classifier could take advantage of that. Learning settings are the same as before (Table 7.2) except for settings of both ILP classifiers, which performed a little bit better with modified settings on a majority of the datasets (see in the legend of Table 7.3).

Table 7.3 compares the numbers of correctly classified instances on all the datasets. The last two columns show numbers of training and testing instances. The numbers of training instances are quite low; this is because the ILP classifiers are probably not capable of fully exploiting higher numbers of training instances and the difference between ILP classifiers and the others would be even a bit higher. This is demonstrated in Fig. 7.11 (for 'nursery' dataset only). It can be seen that when the number of training instances was under about 40, the fuzzy classifier performed better than some of the others (SMO, JRip and Multilayer Perceptron), but from about 60 training instances further both ILP classifiers performed worse than the others.

[4] `http://www.cs.waikato.ac.nz/ml/weka/`

[5] For the fuzzy ILP method, there is a requirement on the target (class) attribute: it has to be monotonizable (e.g. numeric).

[6] `http://www.ksi.mff.cuni.cz/~dedek/fuzzyILP/`

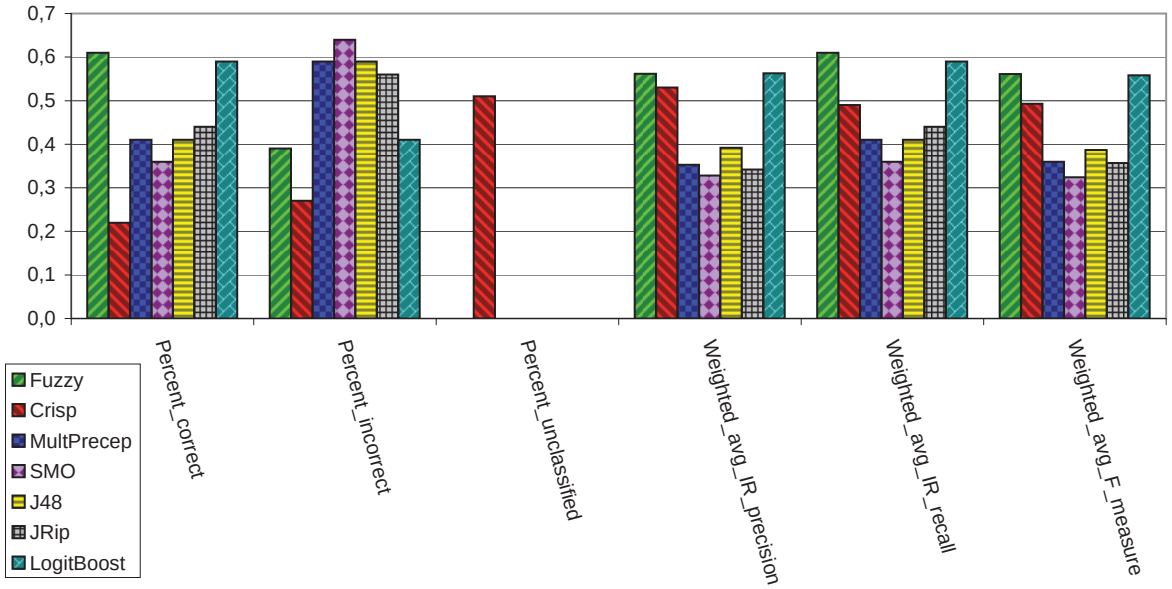


Figure 7.10: Evaluation of the methods – average values.

	Fuzzy	Crisp	MultPerc	SMO	J48	JRip	LBoost
Corr	0.61±.19	.22±.17 ●	.41±.19 ●	.36±.24 ●	.41±.22 ●	.44±.17 ●	.59±.26
Incor	.39±.19	.27±.24	.59±.19 ○	.64±.24 ○	.59±.22 ○	.56±.17 ○	.41±.26
Uncl	.00±.00	.51±.29 ○	.00±.00	.00±.00	.00±.00	.00±.00	.00±.00
Prec	.56±.24	.53±.37	.35±.20 ●	.33±.26	.39±.22	.34±.21 ●	.56±.28
Rec	.61±.19	.49±.32	.41±.19 ●	.36±.24 ●	.41±.22 ●	.44±.17 ●	.59±.26
F	.56±.20	.49±.33	.36±.19 ●	.32±.24 ●	.39±.21	.36±.19 ●	.56±.27

○, ● statistically significant increase or decrease

Legend:

Fuzzy ..... czsem.ILP.FuzzyILPClassifier ”
Crisp ..... czsem.ILP.CrispILPClassifier ”
MultPerc ..... functions.MultilayerPerceptron ’-L 0.3 -M 0.2 -N 500 -V 0 -S 0 -E 20 -H a’
SMO ..... functions.SMO ’-C 1.0 -L 0.0010 -P 1.0E-12 -N 0 -V -1 -W 1 -K \”functions.supportVector.PolyKernel -C 250007 -E 1.0\”’
J48 ..... trees.J48 ’-C 0.25 -M 2’
JRip ..... rules.JRip ’-F 3 -N 2.0 -O 2 -S 1’
LBoost ..... meta.LogitBoost ’-P 100 -F 0 -R 1 -L -1.7976931348623157E308 -H 0.1 -S 1 -I 10 -W trees.DecisionStump’

Corr ..... Percent correct
Inor ..... Percent incorrect
Uncl ..... Percent unclassified
Prec ..... IR precision, weighted average from all classes
Rec ..... IR recall, weighted average from all classes
F ..... F measure, weighted average from all classes

Table 7.2: Evaluation of the methods in 2 times 10-fold cross validation.

	Fuzzy	Crisp	MultPerc	SMO	J48	JRip	LBoost	train	test
car	.39±.03	.36±.03 •	.53±.02 ○	.57±.01 ○	.50±.02 ○	.51±.03 ○	.54±.02○	173	1554
wine	.44±.03	.42±.02 •	.48±.02 ○	.46±.02 ○	.47±.02 ○	.48±.03 ○	.52±.02○	160	1439
cmc	.79±.02	.77±.03 •	.89±.02 ○	.81±.01 ○	.88±.02 ○	.82±.03 ○	.85±.02○	147	1325
tae	.50±.12	.39±.11 •	.59±.11 ○	.55±.12 ○	.50±.12	.37±.11 •	.55±.11○	135	15
pop	.66±.09	.54±.17 •	.57±.13 •	.70±.06 ○	.70±.07 ○	.70±.06 ○	.66±.10	80	9
nurs	.79±.04	.68±.06 •	.81±.04 ○	.73±.04 •	.80±.04 ○	.79±.06	.83±.02○	52	12907

○, • statistically significant improvement or degradation

Legend:

train .....average number (±1) of training instances in each run
test .....average number (±1) of testing instances in each run

car .....http://archive.ics.uci.edu/ml/datasets/Car+Evaluation
wine .....red wine dataset http://archive.ics.uci.edu/ml/datasets/Wine+Quality
cmc .....http://archive.ics.uci.edu/ml/datasets/Contraceptive+Method+Choice
tae .....http://archive.ics.uci.edu/ml/datasets/Teaching+Assistant+Evaluation
pop .....http://archive.ics.uci.edu/ml/datasets/Post-Operative+Patient
nurs .....http://archive.ics.uci.edu/ml/datasets/Nursery

Learning parameters for both ILP methods:
set(noise,20). set(i,3). set(clauselength,13). set(search,heuristic). set(evalfn,wracc). set(samplesize,3).

Table 7.3: Evaluation of the methods on UCI datasets, **percent correct**, average values from 100 repetitions.

Fig. 7.12 demonstrates time complexity of the classifiers in the same experiment as in Fig. 7.11. Despite the fact that the Fuzzy ILP classifier was several times slower than the Crisp ILP classifier and even more than the others, it is still computable on current processors (e.g. P9600, 2.66 GHz, which we used) and the curve of time complexity did not grow rapidly during the experiment. Because ILP is a heuristic and iterative method, the time complexity can be quite directly managed by the setting of learning parameters.

7.7 Conclusion

In this paper we have presented a fuzzy system, which provides a fuzzy classification of textual web reports. Our approach is based on usage of third party linguistic analyzers, our previous work on web information extraction, and fuzzy inductive logic programming.

The main contributions are formal models, prototype implementation of the presented methods and an evaluation experiment. Our data and our implementation are publicly available on the Web. The first experiment evaluated performance of the presented methods and compared them with other machine learning procedures used in data mining on our data. The Fuzzy ILP classifier proved better results than a majority of the methods. The results are statistically significant in many cases. We see the advantage of the Fuzzy ILP classifier in the fact that monotonization leads to the extension of the learning domain and it utilizes the fact that the domain is or can be monotonically ordered.

In the second experiment, we evaluated all the methods on other datasets with more training instances and we also experimentally measured the time complexity of the methods. This experiment has shown that the fuzzy method is suitable mainly in situations with a small amount of training instances and in cases when the target attribute mostly respects the natural order of the remaining attributes. But this did not hold true for any of the later-used datasets. When comparing the fuzzy approach with the crisp one, the fuzzy approach always performed better in terms of correctness of the classification, but it was many times slower than all the methods in terms of time complexity.

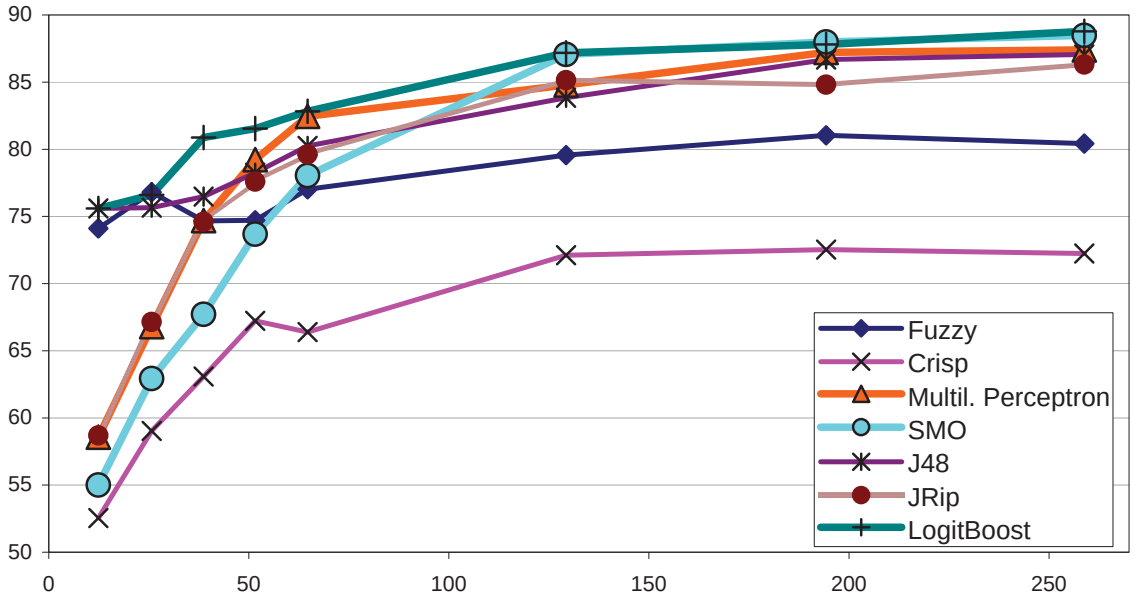


Figure 7.11: x-axis: number of training instances, y-axis: percent of correctly classified instances, average values from 10 repetitions, 'nursery' dataset.

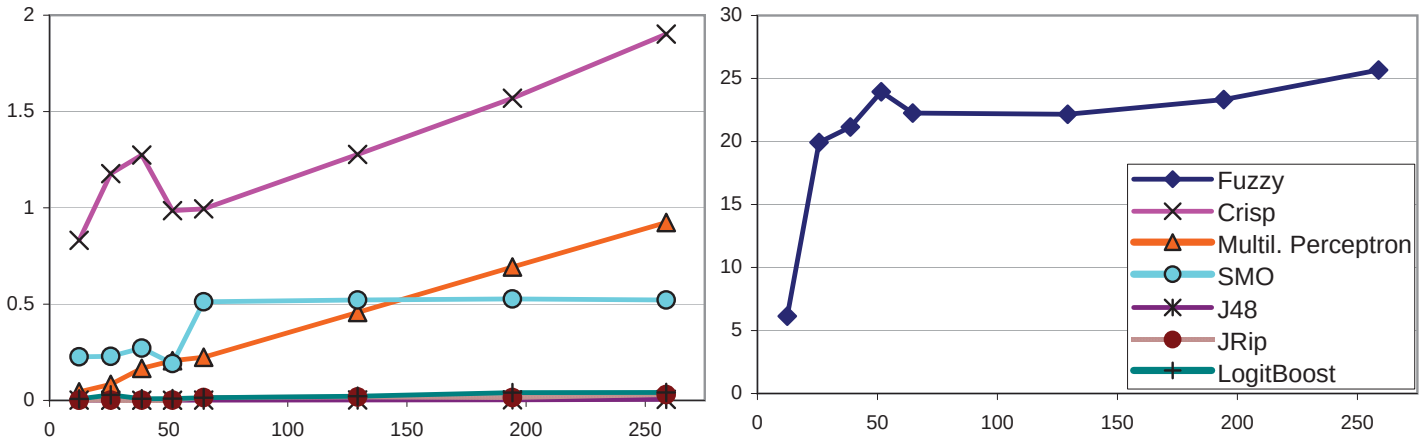


Figure 7.12: x-axis: number of training instances, y-axis: training time in seconds, average values from 10 repetitions, 'nursery' dataset.

Nomenclature

API	Application Programming Interface
GPL	GNU General Public License http://www.gnu.org/licenses/gpl.html
GRDDL	Gleaning Resource Descriptions from Dialects of Languages http://www.w3.org/TR/grddl/
JAPE	Java Annotation Patterns Engine http://gate.ac.uk/userguide/chap:jape
OWL	Web Ontology Language http://www.w3.org/TR/owl-primer/
PDT	Prague Dependency Treebank
PML	Prague Markup Language
RDF	Resource Description Framework http://www.w3.org/RDF/
RDFa	Resource Description Framework - in - attributes http://www.w3.org/TR/xhtml-rdfa-primer/
RPC	Remote Procedure Call
SPARQL	SPARQL Query Language for RDF http://www.w3.org/TR/rdf-sparql-query/
XML	Extensible Markup Language http://www.w3.org/XML/
XSLT	Extensible Stylesheet Language Transformations http://www.w3.org/TR/xslt

Bibliography

Stuart Aitken (2002), Learning Information Extraction Rules: An Inductive Logic Programming approach, in F. van Harmelen, editor, *Proceedings of the 15th European Conference on Artificial Intelligence*, IOS Press, Amsterdam.

Harith Alani, Sanghee Kim, David E. Millard, Mark J. Weal, Wendy Hall, Paul H. Lewis, and Nigel R. Shadbolt (2003), Automatic Ontology-based Knowledge Extraction from Web Documents, *IEEE Intelligent Systems*, 18:14–21.

Douglas E. Appelt and David J. Israel (1999), Introduction to Information Extraction Technology, A tutorial prepared for IJCAI-99, Stockholm, Schweden, URL `http://www.ai.sri.com/~{}appelt/ie-tutorial/IJCAI99.pdf`.

Tim Berners-Lee (2008), The Web of Things, *ERCIM News - Special: The Future Web*, 72:3, URL `http://ercim-news.ercim.org/content/view/343/533/`.

Tim Berners-Lee, James Hendler, and Ora Lassila (2001), The Semantic Web, A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities, *Scientific American*, 284(5):34–43.

Jan C. Bioch and Viara Popova (2000), Rough Sets and Ordinal Classification, in *ALT '00: Proceedings of the 11th International Conference on Algorithmic Learning Theory*, pp. 291–305, Springer-Verlag, London, UK, ISBN 3-540-41237-9.

J.C. Bioch and V. Popova (2009), Monotone Decision Trees and Noisy Data, Research Paper ERS-2002-53-LIS, Erasmus Research Institute of Management (ERIM), URL `http://econpapers.repec.org/RePEc:dgr:eureri:2002206`.

Christopher M. Bishop (1996), *Neural Networks for Pattern Recognition*, Oxford University Press, 1 edition, ISBN 0198538642, URL `http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/0198538642`.

K. Bontcheva, V. Tablan, D. Maynard, and H. Cunningham (2004), Evolving GATE to Meet New Challenges in Language Engineering, *Natural Language Engineering*, 10(3/4):349—373.

Willem Nico Borst (1997), *Construction of Engineering Ontologies for Knowledge Sharing and Reuse*, Ph.D. thesis, Universiteit Twente, Enschede, URL `http://doc.utwente.nl/17864/`.

Paul Buitelaar, Philipp Cimiano, Anette Frank, Matthias Hartung, and Stefania Racioppa (2008), Ontology-based information extraction and integration from heterogeneous data sources, *Int. J. Hum.-Comput. Stud.*, 66(11):759–788, ISSN 1071-5819, doi:http://dx.doi.org/10.1016/j.ijhcs.2008.07.007.

Razvan Bunescu and Raymond Mooney (2007), Extracting Relations from Text: From Word Sequences to Dependency Paths, in Anne Kao and Stephen R. Poteet, editors, *Natural Language Processing and Text Mining*, chapter 3, pp. 29–44, Springer, London, ISBN 978-1-84628-175-4, doi:10.1007/978-1-84628-754-1_3.

Razvan Constantin Bunescu (2007), *Learning for Information Extraction: From Named Entity Recognition and Disambiguation To Relation Extraction*, Ph.D. thesis, Department of Computer Sciences, University of Texas at Austin, URL http://www.cs.utexas.edu/users/ml/papers/razvan-dissertation.pdf.

Ekaterina Buyko, Erik Faessler, Joachim Wermter, and Udo Hahn (2009), Event extraction from trimmed dependency graphs, in *BioNLP '09: Proceedings of the Workshop on BioNLP*, pp. 19–27, Association for Computational Linguistics, Morristown, NJ, USA, ISBN 978-1-932432-44-2.

Chia-Hui Chang, M. Kayed, M. R. Girgis, and K. F. Shaalan (2006a), A Survey of Web Information Extraction Systems, *Knowledge and Data Engineering, IEEE Transactions on*, 18(10):1411–1428, URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1683775.

Chia-Hui Chang, Mohammed Kayed, Moheb R. Girgis, and Khaled F. Shaalan (2006b), A Survey of Web Information Extraction Systems, *IEEE Trans. Knowl. Data Eng.*, 18(10):1411–1428.

Miao Chong, Ajith Abraham, and Marcin Paprzycki (2005), Traffic Accident Analysis Using Machine Learning Paradigms, *Informatica*, 29:89–98.

Philipp Cimiano, Siegfried Handschuh, and Steffen Staab (2004), Towards the self-annotating web, in *WWW '04: Proceedings of the 13th international conference on World Wide Web*, pp. 462–471, ACM, New York, NY, USA, ISBN 1-58113-844-X, doi:http://doi.acm.org/10.1145/988672.988735.

William W. Cohen (1995), Fast Effective Rule Induction, in *In Proceedings of the Twelfth International Conference on Machine Learning*, pp. 115–123, URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.50.8204.

H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan (2002), GATE: A framework and graphical development environment for robust NLP tools and applications, in *Proceedings of the 40th Anniversary Meeting of the ACL*.

Hamish Cunningham, Diana Maynard, and Valentin Tablan (2000), JAPE: a Java Annotation Patterns Engine, Technical report, Department of Computer Science, The University of Sheffield, URL http://www.dcs.shef.ac.uk/intranet/research/resmes/CS0010.pdf.

Jan Dědek (2010), Towards Semantic Annotation Supported by Dependency Linguistics and ILP, in *Proceedings of the 9th International Semantic Web Conference (ISWC2010), Part II*, volume 6497 of *Lecture Notes in Computer Science*, pp. 297–304, Springer-Verlag Berlin Heidelberg, Shanghai / China, ISBN 978-3-642-17748-4, URL http://iswc2010.semanticweb.org/accepted-papers/219.

Jan Dědek, Alan Eckhardt, and Peter Vojtáš (2008a), Experiments with Czech Linguistic Data and ILP, in Filip Železný and Nada Lavrač, editors, *ILP 2008 - Inductive Logic Programming (Late Breaking Papers)*, pp. 20–25, Action M, Prague, Czech Republic, ISBN 978-80-86742-26-7, URL http://ida.felk.cvut.cz/ilp2008/ILP08_Late_Breaking_Papers.pdf.

Jan Dědek, Alan Eckhardt, and Peter Vojtáš (2008b), Experiments with Czech Linguistic Data and ILP, in Filip Železný and Nada Lavrač, editors, *ILP 2008 (Late Breaking Papers)*, pp. 20–25, Action M, Prague, Czech Republic, ISBN 978-80-86742-26-7.

Jan Dědek, Alan Eckhardt, Peter Vojtáš, and Leo Galamboš (2008c), Sémantický Web, in Václav Řepa and Oleg Svatoš, editors, *DATAKON 2008*, pp. 12–30, Brno, ISBN 978-80-7355-081-3, URL http://www.datakon.cz/datakon08/zvane.html#5.

Jan Dědek and Peter Vojtáš (2008a), Computing aggregations from linguistic web resources: a case study in Czech Rep. sector/traffic accidents, in Cosmin Dini, editor, *2nd International Conference on Advanced Engineering Computing and Applications in Sciences*, pp. 7–12, IEEE Computer Society, ISBN 978-0-7695-3369-8, URL http://www2.computer.org/portal/web/csdl/doi/10.1109/ADVCOMP.2008.17.

Jan Dědek and Peter Vojtáš (2008b), Computing aggregations from linguistic web resources: a case study in Czech Republic sector/traffic accidents, in Cosmin Dini, editor, *Second International Conference on Advanced Engineering Computing and Applications in Sciences*, pp. 7–12, IEEE Computer Society, ISBN 978-0-7695-3369-8, URL http://dx.doi.org/10.1109/ADVCOMP.2008.17.

Jan Dědek and Peter Vojtáš (2008c), Exploitation of linguistic tools in semantic extraction - a design, in Mieczysław Kłopotek, Adam Przepiórkowski, Sławomir Wierzchoń, and Krzysztof Trojanowski, editors, *Intelligent Information Systems XVI*, pp. 239–247, Academic Publishing House EXIT, Zakopane, Poland, ISBN 978-83-60434-44-4, URL http://iis.ipipan.waw.pl/2008/proceedings/iis08-23.pdf.

Jan Dědek and Peter Vojtáš (2008d), Linguistic extraction for semantic annotation, in Costin Badica, Giuseppe Mangioni, Vincenza Carchiolo, and Dumitru Burdescu, editors, *2nd International Symposium on Intelligent Distributed Computing*, volume 162 of *Studies in Computational Intelligence*, pp. 85–94, Springer-Verlag, Catania, Italy, ISBN 978-3-540-85256-8, ISSN 1860-949X, URL http://www.springerlink.com/content/w7213j007t416132.

Stephen Dill, Nadav Eiron, David Gibson, Daniel Gruhl, R. Guha, Anant Jhingran, Tapas Kanungo, Sridhar Rajagopalan, Andrew Tomkins, John A. Tomlin, and Jason Y. Zien (2003), SemTag and seeker: bootstrapping the semantic web via automated semantic annotation, in *WWW '03: Proceedings of the 12th international conference on World Wide Web*, pp. 178–186, ACM, New York, NY, USA, ISBN 1-58113-680-3, doi:http://doi.acm.org/10.1145/775152.775178.

Saso Dzeroski and Nada Lavrac, editors (2001), *Relational Data Mining*, Springer, Berlin, URL http://www-ai.ijs.si/SasoDzeroski/RDMBook/.

Alan Eckhardt, Tomáš Horváth, Dušan Maruščák, Róbert Novotný, and Peter Vojtáš (2008), *Uncertainty Issues and Algorithms in Automating Process Connecting Web and User*, volume 5327 of *Lecture Notes in Computer Science*, pp. 207–223, Springer Berlin / Heidelberg, ISBN 978-3-540-89764-4, URL http://dx.doi.org/10.1007/978-3-540-89765-1_13.

David W. Embley (2004), Toward semantic understanding: an approach based on information extraction ontologies, in *Proceedings of the 15th Australasian database conference - Volume 27*, ADC '04, pp. 3–12, Australian Computer Society, Inc., Darlinghurst, Australia, Australia, URL http://portal.acm.org/citation.cfm?id=1012294.1012295.

David W. Embley, Cui Tao, and Stephen W. Liddle (2002), Automatically Extracting Ontologically Specified Data from HTML Tables of Unknown Structure, in Stefano Spaccapietra, Salvatore T. March, and Yahiko Kambayashi, editors, *ER*, volume 2503 of *Lecture Notes in Computer Science*, pp. 322–337, Springer, ISBN 3-540-44277-4.

Oren Etzioni, Michele Banko, Stephen Soderland, and Daniel S. Weld (2008), Open information extraction from the web, *Commun. ACM*, 51(12):68–74, ISSN 0001-0782, doi:http://doi.acm.org/10.1145/1409360.1409378.

A. Frank and A. Asuncion (2010), UCI Machine Learning Repository, URL http://archive.ics.uci.edu/ml.

J. Friedman, T. Hastie, and R. Tibshirani (2000), Additive logistic regression: a statistical view of boosting, *Annals of statistics*, 28(2):337–374.

Katrin Fundel, Robert Küffner, and Ralf Zimmer (2007), RelEx—Relation extraction using dependency parse trees, *Bioinformatics*, 23(3):365–371, ISSN 1367-4803, doi:http://dx.doi.org/10.1093/bioinformatics/btl616.

Birte Glimm, Matthew Horridge, Bijan Parsia, and Peter F. Patel-Schneider (2009), A Syntax for Rules in OWL 2, in *Proceedings of the 6th International Workshop on OWL: Experiences and Directions (OWLED 2009)*, volume 529, CEUR.

Ralph Grishman and Beth Sundheim (1996), Message Understanding Conference-6: a brief history, in *Proceedings of the 16th conference on Computational linguistics*, pp. 466–471, Association for Computational Linguistics, Morristown, NJ, USA, doi:http://dx.doi.org/10.3115/992628.992709.

Petr Hájek (1998), *Metamathematics of Fuzzy Logic*, Kluwer.

Jan Hajič, Eva Hajičová, Jaroslava Hlaváčová, Václav Klimeš, Jiří Mírovský, Petr Pajas, Jan Štěpánek, Barbora Vidová-Hladká, and Zdeněk Žabokrtský (2006), Prague Dependency Treebank 2.0 CD–ROM, Linguistic Data Consortium LDC2006T01, Philadelphia 2006.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten (2009), The WEKA data mining software: an update, *SIGKDD Explor. Newsl.*, 11(1):10–18, ISSN 1931-0145, doi:http://doi.acm.org/10.1145/1656274.1656278.

Martin Hepp (2008), GoodRelations: An Ontology for Describing Products and Services Offers on the Web, in Aldo Gangemi and Jérôme Euzenat, editors, *EKAW*, volume 5268 of *Lecture Notes in Computer Science*, pp. 329–346, Springer, ISBN 978-3-540-87695-3.

Ian Horrocks (2008), Ontologies and the semantic web, *Commun. ACM*, 51(12):58–67, ISSN 0001-0782, doi:http://doi.acm.org/10.1145/1409360.1409377.

Tomáš Horváth and Peter Vojtáš (2007), Induction of Fuzzy and Annotated Logic Programs, *ILP: 16th International Conference, ILP 2006, Santiago de Compostela, Spain, August 24-27, 2006, Revised Selected Papers*, pp. 260–274, doi:http://dx.doi.org/10.1007/978-3-540-73847-3_27.

Markus Junker, Michaell Sintek, Michael Sintek, and Matthias Rinck (1999), Learning for Text Categorization and Information Extraction with ILP, in *In Proc. Workshop on Learning Language in Logic*, pp. 84–93, Springer, LNCS.

S. S. Keerthi, S. K. Shevade, C. Bhattacharyya, and K. R. K. Murthy (2001), Improvements to Platt's SMO Algorithm for SVM Classifier Design, *Neural Computation*, 13(3):637–649.

Václav Klimeš (2006), Transformation-Based Tectogrammatical Analysis of Czech, in *Proceedings of the 9th International Conference, TSD 2006*, number 4188 in Lecture Notes In Computer Science, pp. 135–142, Springer-Verlag Berlin Heidelberg, ISSN 0302-9743.

Stasinos Th. Konstantopoulos (2003), *Using ILP to Learn Local Linguistic Structures*, Ph.D. thesis, Faculteit der Wiskunde en Natuurwetenschappen, Groningen, URL `http://dissertations.ub.rug.nl/faculties/science/2003/s.t.konstantopoulos/`.

Wojciech Kotlowski and Roman Slowinski (2009), Rule learning with monotonicity constraints, in Andrea Pohoreckyj Danyluk, Léon Bottou, and Michael L. Littman, editors, *ICML*, volume 382 of *ACM International Conference Proceeding Series*, p. 68, ACM, ISBN 978-1-60558-516-1.

Stanislav Krajci, Rastislav Lencses, and Peter Vojtas (2004), A comparison of fuzzy and annotated logic programming, *Fuzzy Sets and Systems*, 144:173–192.

Martin Labský, Vojtěch Svátek, Marek Nekvasil, and Dušan Rak (2009), The Ex Project: Web Information Extraction Using Extraction Ontologies, in Bettina Berendt, Dunja Mladenic, Marco de Gemmis, Giovanni Semeraro, Myra Spiliopoulou, Gerd Stumme, Vojtěch Svátek, and Filip Železný, editors,

Knowledge Discovery Enhanced with Semantic and Social Information, volume 220 of *Studies in Computational Intelligence*, pp. 71–88, Springer Berlin / Heidelberg, URL http://dx.doi.org/10.1007/978-3-642-01891-6_5.

Kristina Lerman, Lise Getoor, Steven Minton, and Craig Knoblock (2004), Using the structure of Web sites for automatic segmentation of tables, in *SIGMOD '04: Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pp. 119–130, ACM, New York, NY, USA, ISBN 1-58113-859-8, doi:http://doi.acm.org/10.1145/1007568.1007584.

D.D. Lewis (1992), *Representation and learning in information retrieval*, Ph.D. thesis, University of Massachusetts.

Y. Li, K. Bontcheva, and H. Cunningham (2009), Adapting SVM for Data Sparseness and Imbalance: A Case Study on Information Extraction, *Natural Language Engineering*, 15(02):241–271, URL http://journals.cambridge.org/repo_A45LfkBD.

Yaoyong Li, Hugo Zaragoza, Ralf Herbrich, John Shawe-Taylor, and Jaz S. Kandola (2002), The Perceptron Algorithm with Uneven Margins, in *ICML '02: Proceedings of the Nineteenth International Conference on Machine Learning*, pp. 379–386, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, ISBN 1-55860-873-7.

S. Lievens, Bernard De Baets, and Kim Cao-Van (2008), A probabilistic framework for the design of instance-based supervised ranking algorithms in an ordinal setting, *Annals OR*, 163(1):115–142.

Bing Liu (2007a), *Web Data Mining*, Springer-Verlag, ISBN 978-3-540-37881-5, URL http://dx.doi.org/10.1007/978-3-540-37882-2.

Bing Liu (2007b), *Web Data Mining*, Springer-Verlag, ISBN 978-3-540-37881-5, URL http://dx.doi.org/10.1007/978-3-540-37882-2.

I. Manolescu, L. Afanasiev, A. Arion, J. Dittrich, S. Manegold, N. Polyzotis, K. Schnaitter, P. Senellart, S. Zoupanos, and D. Shasha (2008), The repeatability experiment of SIGMOD 2008, *SIGMOD Rec.*, 37(1):39–45, ISSN 0163-5808, doi:http://doi.acm.org/10.1145/1374780.1374791.

Marie Mikulová, Alevtina Bémová, Jan Hajič, Eva Hajičová, Jiří Havelka, Veronika Kolářová, Lucie Kučová, Markéta Lopatková, Petr Pajas, Jarmila Panevová, Magda Razímová, Petr Sgall, Jan Štěpánek, Zdeňka Urešová, Kateřina Veselá, and Zdeněk Žabokrtský (2006), Annotation on the tectogrammatical level in the Prague Dependency Treebank. Annotation manual, Technical Report 30, ÚFAL MFF UK, Prague, Czech Rep.

Boris Motik, Ulrike Sattler, and Rudi Studer (2005), Query Answering for OWL-DL with rules, *Web Semantics: Science, Services and Agents on the World Wide Web*, 3:41–60, ISSN 1570-8268, doi:http://dx.doi.org/10.1016/j.websem.2005.05.001, URL http://dx.doi.org/10.1016/j.websem.2005.05.001.

S. Muggleton (1995), Inverse Entailment and Progol, *New Generation Computing, Special issue on Inductive Logic Programming*, 13(3-4):245–286.

Stephen Muggleton (1991), Inductive Logic Programming, *New Generation Computing*, 8(4):295–318, ISSN 0288-3635, doi:http://dx.doi.org/10.1007/BF03037089, URL http://dx.doi.org/10.1007/BF03037089.

Stephen Muggleton and Luc de Raedt (1994), Inductive logic programming: Theory and methods, *Journal of Logic Programming*, 19:629–679.

Jan Pavelka (1979), On fuzzy logic I, II, III, *Zeitschr. Math. Logik und Grundl. Math.*, 25:45–52, 119–134, 447–464.

David Pinto, Andrew Mccallum, Xing Wei, and Bruce W. Croft (2003), Table extraction using conditional random fields, in *SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pp. 235–242, ACM Press, New York, NY, USA, ISBN 1581136463, doi:10.1145/860435.860479, URL http://dx.doi.org/10.1145/860435.860479.

Borislav Popov, Atanas Kiryakov, Damyan Ognyanoff, Dimitar Manov, and Angel Kirilov (2004), KIM – a semantic platform for information extraction and retrieval, *Nat. Lang. Eng.*, 10(3-4):375–392, ISSN 1351-3249, doi:http://dx.doi.org/10.1017/S135132490400347X.

J. Ross Quinlan (1993), *C4.5: programs for machine learning*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, ISBN 1-55860-238-0.

Ganesh Ramakrishnan, Sachindra Joshi, Sreeram Balakrishnan, and Ashwin Srinivasan (2008), Using ILP to construct features for information extraction from semi-structured text, in *ILP'07: Proceedings of the 17th international conference on Inductive logic programming*, pp. 211–224, Springer-Verlag, Berlin, Heidelberg, ISBN 3-540-78468-3, 978-3-540-78468-5.

Marek Reformat, Ronald R. Yager, and Zhan Li (2008), Ontology Enhanced Concept Hierarchies for Text Identification, *Journal Semantic Web Information Systems*, 4(3):16–43, ISSN 1552-6283.

Rudi Studer, V. Richard Benjamins, and Dieter Fensel (1998), Knowledge engineering: Principles and methods, *Data & Knowledge Engineering*, 25(1-2):161 – 197, ISSN 0169-023X, doi:DOI:10.1016/S0169-023X(97)00056-6, URL http://www.sciencedirect.com/science/article/B6TYX-3SYXJ6S-G/2/67ea511f5600d90a74999a9fef47ac98.

Rui Wang and Günter Neumann (2007), Recognizing textual entailment using sentence similarity based on dependency tree skeletons, in *RTE '07: Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pp. 36–41, Association for Computational Linguistics, Morristown, NJ, USA.

Daya C. Wimalasuriya and Dejing Dou (2010), Ontology-based information extraction: An introduction and a survey of current approaches, *Journal of Information Science*, 36(3):306–323, doi:10.1177/0165551509360123, URL http://dx.doi.org/10.1177/0165551509360123.

A. Yakushiji, Y. Tateisi, Y. Miyao, and J. Tsujii (2001), Event extraction from biomedical papers using a full parser., *Pac Symp Biocomput*, pp. 408–419.

Burcu Yildiz and Silvia Miksch (2007), ontoX - a method for ontology-driven information extraction, in *Proceedings of the 2007 international conference on Computational science and its applications - Volume Part III*, ICCSA'07, pp. 660–673, Springer-Verlag, Berlin, Heidelberg, ISBN 3-540-74482-5, 978-3-540-74482-5, URL http://portal.acm.org/citation.cfm?id=1793154.1793216.

Zdeněk Žabokrtský, Jan Ptáček, and Petr Pajas (2008), TectoMT: Highly Modular MT System with Tectogrammatics Used as Transfer Layer, in *Proceedings of the 3rd Workshop on Statistical Machine Translation*, pp. 167–170, ACL, Columbus, OH, USA, ISBN 978-1-932432-09-1.

Hongkun Zhao, Weiyi Meng, Zonghuan Wu, Vijay Raghavan, and Clement Yu (2005), Fully Automatic Wrapper Generation For Search Engines, in *WWW Conference*, pp. 66–75.