University of Economics, Prague
Faculty of Informatics and Statistics
Department of Information and Knowledge Engineering

Automated Analysis of the WWW Based on Reusable Resources

Habilitation Thesis

Author: **Vojtěch Svátek**

Prague, 2005

Preface

The emergence of World-Wide Web was the most important breakthrough of the last decades in the area of computing. Although it was originally developed for human 'consumption', its direct exploitation by software tools soon became necessity. The size and heterogeneity of web content and structure however presents an unprecedented challenge to existing technologies of data processing. One of principal problems of automated WWW analysis is to keep as much information as possible from the original content while maintaining tractability even for large sizes of data. Specialised research and industrial communities typically perceive the web space from different angles, and their mutual awareness is often limited. Most tools are developed from scratch and their principles are not thoroughly compared with the state of the art beyond a narrow field.

The central assumption in this thesis is that the notion of *reuse*, central to knowledge engineering, should be systematically applied to web analysis. This reuse appears at multiple levels. Rather obvious (and not unique to this project) is the implementation of simple *tools* in such a way that they can, at the syntactic level, be combined into (i.e., reused in) unforeseen and sophisticated *applications*. Similarly, the same sets of raw *web data* could be reused by multiple tools in the same or different application. The main contribution of the thesis is however connected with the reuse of knowledge *models*, i.e. *ontologies* and *problem solving methods*. It seems that these two types of models would enable to describe the web analysis tools and the web space itself such that individual tools can be tuned to particular forms of data and adequately combined.

The work described here has been carried out in several epochs, approximately from 2001 (though some very early results date to 1999) to 2005, at the Department of Information and Knowledge Engineering of the University of Economics, Prague. It has been supported by two Czech Science Foundation grants,

- 201/00/D045 "Knowledge model construction based on text documents" (2000-2003)
- 201/03/1318 "Intelligent analysis of WWW content and structure" (2003-2005)

and resulted in more than 30 reviewed publications.

It should be noted that a characteristic feature of research in *applied* computer science is its collective nature. The project this thesis is based upon is not an exception; it was virtually

impossible to formally separate my own research results with those provided by my younger teammates (especially PhD and MSc students) without destroying the logical structure of the text. Specific acknowledgments (in addition to the general ones below) are thus included in the respective sections. By my, necessarily subjective, assessment:

- my proper research contribution is concentrated in sections/chapters 3.3, 3.5, 4.2.1, the whole of 5, 7.1 and the whole of 8 (except 8.4.2)
- the results discussed in sections 3.1, 3.4, 4.2.2, 4.3, 7.2.1, 7.3 and 7.4 arose in tight collaboration between me and my colleagues
- finally, my role concerning the results in 3.2, 3.6, 3.7, 4.1 and 7.2.2 was rather that of occasional advisor.

The thesis starts with an introductory Chapter 1, which briefly presents the *state of the art* in three research areas closely connected to its own focus: *web information extraction* (and in more general, mining), *semantic web* and *web services*. The remainder of the thesis is broken down to two different parts.

Part I is devoted to the *implementation* and *experiments* with the *Rainbow* collection of web analysis (and associated) methods and tools, developed under my informal leadership between 2001–2005. Chapter 2 follows the evolution of this collection along a historical axis. Chapter 3, in turn, describes each constituent method/tool in a separate section. Finally, Chapters 4 and 5 are devoted to integration aspects: in actual applications and within a formal model, in turn.

In contrast, Part II focuses on *abstract models* of web data and analysis tools. It starts with an explanatory Chapter 6 on state of the art in knowledge modelling, in general. Chapter 7 describes a conceptual framework for describing web analysis tools, and the associated collection of ontologies. Finally, Chapter 8 deals with problem-solving methods (PSMs): among other, it describes my recent research on PSM-based composition of web analysis services carried out by means of simulated experiments.

As suggested above, I would like to thank my numerous senior as well as junior colleagues. The former, especially Petr Berka and Václav Snášel, mostly contributed to the current research with feedback on my rough ideas; I am also indebted to Petr for his initial idea of investing our effort to the WWW as subject of 'intelligent systems' research, and for hiring me to his *VŠEvěd* team as early as in 1998. Younger colleagues, especially Martin Kavalec, Jirka Kosek, Martin Labský, Ondřej Šváb, Miroslav Vacura and Filip Volavka, mostly implemented tools that enabled to test these ideas on real data while adding their valuable expertise as well. The part of the work regarding problem-solving methods was done in co-operation with Annette ten Teije from the Vrije Universiteit Amsterdam. I would also mention Vilém Sklenák, the Head of Department, who offered me favourable conditions for completing the thesis. My work was strongly supported by my family: my wife Miroslava, as well as my little daughters Anna and Jana, created a nice environment for me, in which I could both work and play with them. I was also strongly encouraged by my father and by my parents-in-law.

Finally (but most importantly), I owe my thanks to the God, who governs my professional carrier equally as the rest of my life: the emergence of the fabulous *Rainbow* research team and the successful completion of this thesis in due time are just two among the numerous blessings I experienced in the last few years.

Vojtěch Svátek, December 21, 2005

Contents

Chapter 1

How to Make the Web 'Computer-Understandable'?

The emergence of World-Wide Web was the most important breakthrough of the last decades in the area of computing. Its size has been growing almost exponentially, and at the same time, its underlying technology gradually evolved from purely textual, static HTML pages to pages with rich multimedia content, frequently generated from databases and made interactive thanks to scripts. Nevertheless, the web content is still meant to be exploited by *human users* upon display in browsers, which is often perceived as a shortcoming. The possibility of exploiting the web content (as an enormous repository of knowledge) in a 'computer-understandable' form would clearly be a big leap forward. By 'computer-understandable' we of course do not mean the mere capability to parse and display web pages (i.e. 'execute' their code) to the user, as the browsers only care of a tiny, fixed set of low-level formatting features. In order to transform the web content into a *semantically* structured representation, more sophisticated tools are needed.

In this chapter, we briefly discuss three substantially different technologies that, in a sense, aim at delivering the content of the web in a structured form suitable for software tools rather than just for humans. We also mention their mutual relationships as well as their role in the *Rainbow* project (as principal subject of this thesis).

1.1 Web Information Extraction and Mining

The task of transforming textual documents to structured (say, database) representation has been addressed earlier than the WWW appeared. There had been an important community that aimed at practical discovery of concepts and relations from especially news articles, focusing on information with military/security impact (terrorist attacks, accidents etc.) or business events. This community concentrated around the Message Understanding Conferences[1] (MUC). The discipline was eventually named *information extraction* (with acronym

[1] `http://www.itl.nist.gov/iaui/894.02/related_projects/muc/`

IE), and as it most important components, the following were typically considered:

1. Discovery of named entities such as names of persons, organisations or locations
2. Discovery of relations among these entities, e.g. that of a person being employed by an organisation
3. Discovery of scenarios, e.g. that of a company merging with another company thus becoming a third company, on a particular date.

With the growing amounts of HTML pages, it became evident that the web could be an even more important target for IE than traditional news resources. In addition to (mostly shallow) linguistic techniques previously employed in plain-text IE, there was a brand new option: to take into account the structure of HTML tags, which was, in some cases, almost as regular as that of databases. As soon as the delimiters and the semantic of individual 'fields' are established, content can be extracted and stored into e.g. a relational database in a straightforward manner. This was the idea of *wrappers* [9, 40, 43], where the semantic of 'fields' is typically identified by a human while the recurring HTML structures are automatically discovered by a software tool, within a feedback-looping interactive session.

Wrappers are, currently, a stable technology with a number of commercial applications. However, as they rely on the HTML structure that varies from one page to another, they are only efficient for very large pages/sites (e.g. encyclopedias) or frequently updated pages (e.g. news). For smaller pages, the benefits of automated extraction would not oughtweigh the cost of creating the wrapper. In addition, wrappers fail if the HTML code lacks in regularity. Therefore, alternative approaches appeared that aim at extracting useful content from *unseen* pages, taking advantage of various features including terms, punctuation as well as HTML tags. Their underlying principle is *inductive learning* of patterns describing the typical content as well as context of target information. The patterns may have the form of symbolic rules [18, 71] or e.g. probabilistic models [47, 54]. Learning-based IE tools have to be provided with a collection of labelled examples, i.e. typically pages with manually inserted semantic tags indicating the position of target information. As manual labelling is rather tedious, there have been attempts to eliminate it via statistical *bootstrapping* [55, 66] or via reuse of structured information available on the web itself [11, 17, 19].

Beside IE, there are also other data analysis techniques that were originally developed for plain text documents but soon became important in the context of the WWW. Prominent ones are classification and clustering. *Classification* is similar to information extraction in the sense that it aims at discovering the 'semantics' of textual—or non-textual—objects. This 'semantics' is however limited to assigning a class to the whole object given as input (in contrast to IE, where we do not know a priori the location of important information). As far as *clustering* is concerned, it 'only' results in groups (or hierarchies) of objects that are somehow (possibly, semantically) related. In order to discover some explicit semantics of the clusters, extra steps are needed.

Apart from techniques originally developed for the analysis of documents, there are also techniques borrowed from other areas of data analysis. For example, *association mining*

was first used for knowledge discovery from tabular data, and only then transformed to text mining (for frequent associations among terms) and then straightforwardly to web mining. *Graph analysis* had been used in many disciplines before but the huge structures of web hyperlinks recently became one of its primordial targets.

It might be useful to distinguish the notions of deductive vs. inductive web mining (though their boundary is not always clear). The *deductive web mining* (DWM), first introduced in [84], covers all activities where *pre-existing patterns are matched with web data*, be they of textual, graph-wise or, say, bitmap nature. DWM thus subsumes web information extraction, and differs from inductive web mining (such as the previously mentioned association mining in text), which aims at discovery of *previously unseen*, *frequent* patterns in web data. This does not mean that the 'pre-existing patterns' in DWM have necessarily been hand-crafted: inductive learning of patterns (or analogous structures/models) is merely viewed as an activity separate from DWM ('reasoning').

In the *Rainbow* project, information extraction and other (mostly deductive) web mining tools have been developed. Most of them are described in section 3.

1.2 Semantic Web

The idea of semantic web is to use the web infrastructure for storing and retrieving structured information and knowledge, which can subsequently be used for formal reasoning. Abundant literature has recently been published about the semantic web current state and prospects (e. g. the monographs [6, 27, 76]), so we will limit ourselves to a very brief and simplified description. The current language for semantic web data is *RDF*[2], which allows to express triples in the form "*subject* has *object* as value for *property*". Subjects, predicates and (most) objects are web resources identified with a URL. The vocabulary extension of RDF, called *RDF Schema* (RDFS) allows to define classes of resources and relationships among them. RDFS can be viewed as a simple *ontology*[3] language; for more inference-oriented applications, it can be extended with the constructs of the *Web Ontology Language* (OWL). OWL allows to specify relatively complex 'axioms', for example, to define necessary and sufficient conditions valid for all individuals that are members of a given class. Using the axioms, reasoning tools based on *description logics* are able to check the satisfiability of classes and to derive implicit subclass-class links. Finally, the upmost layer of the semantic web currently supported with formal languages (especially, SWRL[4]) is the *rule* layer. Rules allow additional inferences, mainly based on Horn logic.

While the initial concept of semantic web assumed manual creation of knowledge annotations (in RDF), most recent research efforts are aimed towards large-scale *automated* [17, 21] or at least *semi-automated* [34] annotation of websites. Automation of the annotation process is typically based on IE techniques mentioned above. An alternative is to generate annotations from the same database resources that have been used to generate this HTML code [57,

[2] `http://www.w3.org/RDF`

[3] For a somewhat more thorough account of the notion of ontology see chapter 6.

[4] `http://www.daml.org/2003/11/swrl`

91]; this approach is faster and more reliable but only works for web pages generated from databases and requires access to those databases. Extraction of structured information from the text or other resources can actually be viewed as 'ontology population', i.e. providing instances of classes and relations from the ontology.

In *Rainbow*, current semantic web languages have been explicitly used in the bicycle information project, where information extracted from the websites was converted to RDF based on an RDFS document (ontology), see section 4.3.

1.3 Web Services

Althougth web services[5], strictly spoken, are not necessarily concerned with the 'ordinary' (HTML-based) web, we mention them here as they represent a sort of alternative—but also complement—to the technologies described above, i.e. to information extraction and to semantic web. The principle of web services is that of distributed computation; in contrast to earlier approaches for remote calling among procedures, such as CORBA[6], their interface relies on relatively transparent (though verbose) XML-based languages. In addition, they are explicitly meant to be advertised on the web, in the form of UDDI descriptions[7]. This gives web services a degree of 'openness' unmatched by previous approaches.

In many situations, web services can side-step the problem of machine-processability of web content. Instead of semantically annotating the content of a large and highly structured page, we can create a web service that provides the same information (from the underlying database) dynamically on request from the client. Moreover, web services can provide the results of complex parametric calculations that are impossible to be stored in semantic annotations beforehand. On the other hand, web services as such cannot replace semantic web annotation when the user needs to automatically integrate information from multiple distributed (possibly, small) resources that have first to be discovered.

Other two problems of (original) web services is their stateless nature, and the difficulty of discovering the appropriate service in a large distributed WWW environment: the search in UDDI repositories so far relied on keyword matching or navigation in simple hierarchies. The first problem is typically overcome by combining multiple simpler services into a more complex application. Service composition can be carried out via manual programming (this is mostly the case in commercial applications) or more automatically. The latter option relies on semantic annotations similar to annotations of web page content. Semantic annotation of web services also represents the solution to the second problem, that of discovering unseen services on the web. Semantically annotated, discovered, composed and executed web services are simply called *semantic web services* [52, 62, 67]. As the provision of semantic annotations for web services becomes, with the growing number of such services, similarly tedious as the annotation of web page content, IE-based approaches recently appeared [36, 68].

[5] `http://www.w3.org/2002/ws`
[6] `http://www.corba.org`
[7] `http://www.uddi.org`

In the *Rainbow* project most tools have been implemented as web services. Furthermore, semantic web service technology (namely, automated annotation and composition of such services) has been investigated, see section 8.4.

Part I

Rainbow: Web Analysis via Reusable Services and Data

Chapter 2

History of the *Rainbow* Project

2.1 Initial Motivation

The way information is presented on the web typically combines multiple types and representations of data. Free text is interleaved with images and structured lists or tables, pages are connected with hyperlinks, labelled with URLs (often containing meaningful tokens) and endowed with explicit meta-data (in specialised tags). Different methods of web data analysis, focusing each on a different data type/representation, may provide complementary and/or supplementary information. Reducing the analysis on a single method, which is typically done in specialised text categorisation or information extraction projects, may thus lead to significant information loss. On the other hand, a monolithic application encompassing many methods would be impossible to maintain (in view of permanent changes in web data standards and conventions) as well as reuse in different domains. The only solution thus seems to be to combine multiple, relatively independent, web-analysis services based on diverse principles (statistics, linguistic, graph theory etc.) and equipped with hand-crafted or inductively trained knowledge bases.

This situation was starting point for the *Rainbow* project; in the rest of the chapter, we give an overview of this project according to its characteristic time periods. Most partial projects and achievements mentioned in this section are elaborated on more thoroughly in the rest of the thesis.

2.2 First Phase: Infrastructure and Ad-Hoc Tools

The *Rainbow* project started in 2001 as an informal joint venture of a group of researchers and students at the University of Economics, Prague. Their joint interest was the rich but heterogeneous and unreliable content of the WWW, in particular, that of websites of smaller companies. In the first phase of the project (2001–2003), the methods and tools were mostly designed for non-specific business websites. There were two main implemented outcomes:

- A method for extraction of 'company profile' sentences based on *lexical indicators* discovered via statistical analysis of web directory headings and associated pages, see [38] and section 3.1.
- A method for discovery of navigational structure of company websites, based on graph-theoretical analysis of *local link topologies*, see [98] and section 3.4.

Besides, less sophisticated tools for e.g. page classification based on *URL strings* (see section 3.5) or for collection of the content of selected *META tags* (see section 3.6) were developed. The overall *web service infrastructure* was also built in this phase, as well as a simple *result browser*, which eased manual experiments combining multiple tools [80]. The state of the project in the end of the first phase was summarised in [82].

2.3 Second Phase: Extraction Tools, Bicycle Application and Knowledge Models

In the second phase (2003–2005), the project was undertaken with support of the CSF grant no. 201/03/1318, in collaboration with the Technical University of Ostrava and the Vrije Universiteit Amsterdam (as foreign partner). In the course of this phase, a more restricted area was chosen for experiments, namely, that of websites offering *bicycle products*. The main (implemented) achievements were:

- An *information extraction* tool employing Hidden Markov Models and a 'wrapper ontology' for extraction of bicycle product offers from online catalogues, see [45] and section 3.2
- A collection of tools for *image analysis*, based on singular value decomposition (SVD), colour histograms and image dimensions, see [46] and section 3.7
- A *procedural application* that calls individual analysis tools, collects results and converts them to the RDF format (as standard language of the semantic web), see [88] and section 4.2.2
- A *result repository*[1] with simple HTML interface allowing for end-user *search and navigation*, see [87] and section 4.3
- By the collaborating group at VSB-TU Ostrava, the text-and-XML indexing and query tool *AmphorA* has been adapted for provision of *source data* using the web service technology [3].

A general diagram of the implemented architecture is at Fig. 2.1.

Another stream of research focused on *abstract descriptions* of analysis tools. Different types of knowledge models were subsequently developed:

[1] Based on *Sesame*, see `http://openrdf.org`.

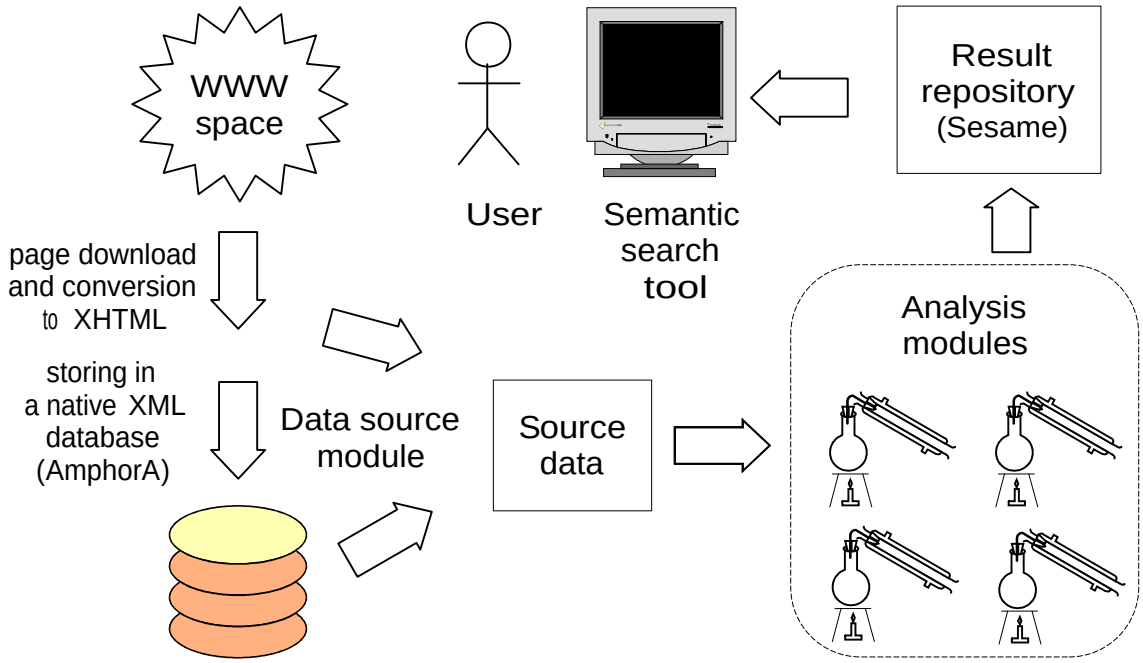

Figure 2.1: Diagram of the *Rainbow* architecture

- A four-dimensional *descriptive framework* called TODD (for task-object-data-domain as its four dimensions)
- A collection of *ontologies* capturing various aspects of the web space
- A collection of *problem-solving methods* defining different ways of solving the tasks of classification, extraction and retrieval.

The coverage of the models was tested through re-describing real (own as well as others') applications in a semi-formal language [84]. Furthermore, simulations of automated composition of more complex applications from simple components were carried out for a specific task of pornography classification [85, 86]; component simulations were derived from real tools developed within a PhD thesis [95].

2.4 Future Plans

As the exploitation of web content and structure is by far not a resolved problem, the motivation for extending the outcomes (and possibly, duration) of *Rainbow* project is high. The most interesting research goals can be summarised as follows:

- To *automatically build* web analysis applications on the fly, using ontological descriptions of individual tools

- To effectively combine *inductively learnt information extraction models* with *wrapper ontologies*, for newly addressed domains.
- To make full benefit of the available *XML indexing and querying* technology as preprocessor to knowledge-based analysis, capable of providing an appropriate XML environment to the knowledge-based web analysis tools.

We envisage to elaborate on these issues in the framework of several newly commenced EU-funded projects.

Additional bibliography for the project as well links to implemented tools can be found at `http://rainbow.vse.cz`.

Chapter 3

Web Analysis Methods in *Rainbow*

Although the main focus of this thesis is the *reuse* of web analysis methods (and models), an important part of the *Rainbow* project has been original *development* of analysis methods and tools. The actual developers were different project team members, mostly undergraduate or PhD students. Some of them worked under direct supervision of the author, while other were only indirectly influenced by the author as *Rainbow* coordinator. Namely:

- Experiments with indicator learning and company profile extraction (section 3.1) were carried out by Martin Kavalec, as part of his PhD thesis
- Application of statistical method on information extraction from product catalogues (section 3.2) is the topic of PhD thesis of Martin Labský
- A motivating experiment discussed in connection with the visual pattern method (section 3.3) was carried out by Jiří Bráza
- Analysis of webgraph of company sites (section 3.4) was the subject of two subsequent MSc theses: by Martin Sajal and by Filip Volavka; Filip was author of the ultimate implementation
- The web service for URL-based classification of documents in company[1] websites (mentioned in section 3.5) was implemented by Vladimír Vávra
- The web service oriented on explicit meta-data and the experiments carried out in this respect (section 3.6) are due to Pavel Kupka, being part of his MSc thesis
- Finally, various experiments with image analysis (section 3.7) were carried out by Miroslav Vacura, Pavel Praks and Martin Labský.

The descriptions and results of most methods were previously published as conference or workshop papers. In this chapter, all of them are briefly presented in separate sections of varying length.

[1] The earlier experiments with generic websites were carried out in collaboration with Petr Berka.

3.1 Extracting Company Profile Using Lexical Indicators

The goal of this partial project[2] was relatively modest: to extract information about (mostly generic) *products, services and areas of competence of companies*, from the free text chunks embedded in web presentations. For this sort of information, an abundant reusable resource are web directories such as Yahoo! or Open Directory. We based our experiments on the 'Business' branch of *Open Directory* (`http://dmoz.org`). Both the hierarchy of the *directory headings* and the categorization of *links* listed in each node are valuable sources of information. From the categorization of web links we can obtain *labelled* training data for information extraction, while the hierarchy could be used as source for building a (lightweighted) ontology of the domain corresponding to the given branch. The latter issue is subject of section 7.4.

Mining Indicator Terms Through Directory Headings

The general description of the company profile, area of competence, products and services is usually not too extensive but stylistically well-formed. This favourises the use of *linguistic* techniques, in contrast to surface techniques (such as regular-expression-based), which are often used for information extraction from idiosyncratic, abridged documents (e.g. advertisements or medical records). Our assumption is that the *directory headings* (such as `.../Manufacturing/Materials/Metals/Steel/...`) coincide with the generic names of products and services—let us nickname them *informative terms* here—offered by the owners of the pages referenced by the respective directory page. By matching the headings with the page full texts, we obtain sentences that contain the informative terms. The terms situated near the informative terms in the syntactical structure of the sentence are candidates for *indicator terms*, provided they occur frequently on pages from various domains. The resulting collection of indicator terms can be, conversely, the basis of extraction patterns for discovering informative terms in previously unseen pages.

The knowledge asset embedded in web directories is the judgement of human indexers who have assigned the pages under the particular heading(s). Naturally, informative terms on the page need not always correspond to the existing directory headings, e.g. due to synonymy. As consequence, our method will extract (without the help of a thesaurus) only a fraction of the sentences with informative terms. This however does not disqualify the method, since, in this training phase, we aim at discovering indicator terms rather than at identifying the informative terms themselves. The small degree of completeness of the method is actually compensated by the hugeness of the material available[3] in the directories. Namely, the 'Business' subhierarchy of Open Directory that we have exploited in our experiments points to approximately 150,000 pages overall, each of these containing the 'heading' terms (from the referencing node or one of its ancestors) in two sentences, on the average.

We tested the training phase of our method on a sample of 14,500 sentences[4] containing the

[2] See also papers [38] and [39].

[3] As we dispense with manual labelling, processing a larger sample of data is merely the matter of computer time/storage.

[4] I.e. about 5% of the total of such sentences.

'heading' terms. The syntactical analysis has been carried out using the free *Link Grammar Parser*[5][70]. Our working hypothesis was that the abovementioned indicative function is, in most cases, conveyed by *verbs* (and verb phrases). Therefore, in the initial experiments, the verbs that occurred the closest (in the parse tree) to informative terms have been counted, arranged into a frequency table, and ordered by ratio of their relative frequency of occurrence near some informative term to their relative frequency in general. Eight[6] most promising verbs have been chosen for the experimental collection. Most of these are likely to be associated with informative terms, e.g. 'our assortment *includes*...', 'we *manufacture*...', 'in our shop you can *buy*...'.

To test the precision of extraction based on these indicators, 130 sentences containing some indicators were randomly selected and each of them was *manually labelled*. The labelling amounted to the subjective estimation whether the sentence contains the target informative terms or not. This is sometimes difficult—e.g. due to missing context, special terminology and domain specific product names; see for example the sentence:

> We are equipped to run any grade of corrugated from E-flute to Triplewall, including all government grades.

Therefore, some unclear sentences were labelled with '?' and then counted once as negative and once as positive test cases. Some sentences contained the company name but no usable information on the products, e.g.

> Industrial Metals Inc. is committed to provide you with exceptional service.

Although named entities are often valued in the information extraction field, we considered these sentences as negative test cases, too, since we focus on *generic* names of products/services or of their providers. The testing results are in Tab. 3.1. Together with some ad-hoc inspections, they suggest that some general[7] verbs–such as 'use' or 'include'–need to be extended to more complex *phrases*, possibly again via selecting the neighbouring terms with frequent occurrence. Also, clearly, certain nouns and noun phrases could play the role of indicators, too.

Due to the tedium of the abovementioned manual labelling, we are not able to measure directly the *coverage* of a collection of indicators: this would amount to considering the full set of sentences in the selected sample of web pages. An indirect measure of coverage, which can be obtained automatically, is the number of pages in the sample that contain one or more indicators from the collection. On the pages directly referenced by directory nodes, this measure was rather low, between 10-20%; however, if we manually pre-filtered out pages

[5]The choice was motivated partly by the immediate availability of the parser, partly by the hypothesis that a linked-based parser could support the presumed 'navigation' over the dependency structures better than parsers based on constituent grammars.

[6]We hope to build a more comprehensive collection using a larger sample of pages, and possibly more domain-specific collections for sub-branches of 'Business'.

[7]Even the verb 'to be', which has no significance of its own, could presumably be the starting point for finding useful indicator phrases.

Table 3.1: Test of the indicative verbs

indicator	−	?	+	precision
include	8	4	18	60–73%
provide	9	3	28	70–78%
offer	6	1	21	75–79%
specialize	0	1	18	95–100%
(other)	3	5	5	38–77%
total	26	14	90	77–80%

with no or minimal free-text content (such as intro or menu pages), the proportion increased to 70-80%: the fact that this result was obtained for a collection of *eight* indicators suggests that the cross-domain variability of these terms might be relatively limited. Note that even if a set of indicators could not directly be used for systematic filling of information extraction templates due to low coverage, it could still be acceptable for the discovery of new terms for the *ontology of products and services*, see section 7.4.

Related Work

Li, Zhang and Yu also use the Link parser and describe in [48] how to learn mapping from the link grammar to RDF statements. Their work shows advantages of link grammar over constituent grammar for this task and demonstrates feasibility of this task.

While directories have already been used for learning to classify *whole documents*, by Mladenic, [56], their use for *information extraction* seems to be innovative.

There is also some similarity to Brin [11], which targets on automated discovery of extraction patterns using *search engines*. The patterns can be used to find relations, such as books, i.e. pairs (author, title). However, the patterns are simply based on characters surrounding the occurrence of the investigated relation. In comparison, we aim at finding less structured information, for which such simple patterns wouldn't be sufficient.

Finally, the use of bootstrapping and other statistical methods for information extraction has also been presented e.g. in [55] and [66].

3.2 Statistical Extraction from Product Catalogues

Product catalogues are the heart of most company websites. Although the number of companies relying on form-based (sometimes even web-service-based) access to catalogues is slowly increasing, small and medium companies typically find plain HTML pages (with navigational access) as the most rational option. The information about product names, properties and prices is structured to tables, lists or paragraphs, which can be analysed by *information extraction* (IE) techniques. The best known IE projects focusing on product information is

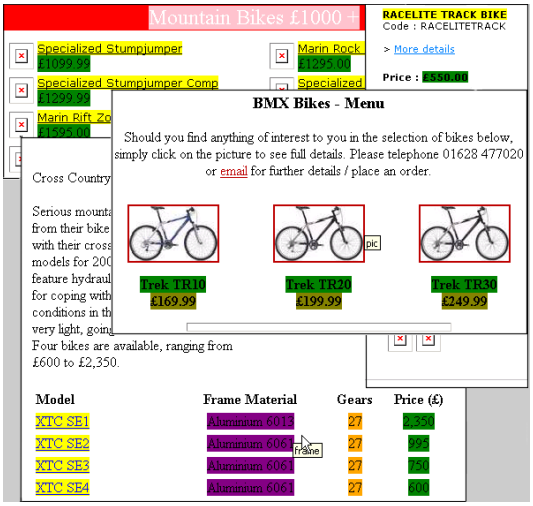

Figure 3.1: Samples of annotated training data

CROSSMARC[8].

Since the structure of catalogues is rather diverse from one to another, and emphasis is put on attractive presentation rather than on document logic, *wrapper-based* approaches [40, 43], which only work well on database-like pages with globally-regular structure, can hardly be applied. Likewise, the catalogues do not contain continous, linguistically sound text, which could be processed by *traditional NLP* techniques such as complete parsing. As the most feasible option then remains IE relying on complex *inductively trained models*, be they statistical or rule-based.

As typical in IE, we have to solve at least two constituent problems: identification (annotation) of partial data items[9] and their assignment (as 'slots') to instances of 'product offer' class from an underlying ontology. As discussed below, we so far used a trained statistical model for the former, and a simple heuristic algorithm for the latter.

Experimental Data

As training and testing data for our extraction models, we manually annotated 133 product catalogues randomly chosen from bike shop websites in the UK. The documents were picked from the Google Directory node *Sports-Cycling-Bike Shops-Europe-UK-England*. Each document contains from 1 to 50 bike offers; there were more than 900 instances of 'bike offer' in the data, overall. Manual annotation, carried out by means of simple interactive tool made for this purpose, covered different 'slots' of 'bike offer', distinguished by different colours; see examples of annotated pages at Fig. 3.1. The six most frequent slots are enumerated in the first column of Table 3.2. The labelled collection is available from `http://rainbow.vse.cz`.

[8] `http://www.iit.demokritos.gr/skel/crossmarc`

[9] They correspond or are analogous to traditional *named entities* (cf. the MUC conferences, `http://www.itl.nist.gov/iaui/894.02/related_projects/muc`).

Table 3.2: 10-fold cross-validation results

Slot	*Recall*	*Precision*	*# instances*
name	77.9 78.6 83.63	63.5 65.6 62.1	927
price	98.9 99.1 98.8	89.5 88.9 86.9	971
picture	69.0	89.6	359
speed	86.8	93.6	186
size	83.2	93.7	173
year	98.1	70.0	160

Annotation Using Hidden Markov Models

Hidden Markov Models (HMMs) are finite state machines augmented with state transition probabilities and lexical probability distributions for each state [65]. Text is modelled as a sequence of tokens (in our case including words, punctuation and formatting symbols). When applying an HMM to text, the given sequence of tokens is assumed to have been generated by that model. Provided some states of the model are associated with semantic slots (to be filled in with extracted text), we are interested in recovering the most probable state sequence that would have generated our text, and thereby obtaining its most probable semantic interpretation. This task is effectively solved by the *Viterbi algorithm* [65].

Before applying HMMs, we transformed each document into a sequence of HTML block elements (e.g. paragraphs, table cells) that directly contain potentially interesting data (in our case, any text or images). Furthermore, certain inline HTML tags were substituted with abstract tag classes, e.g. `<important>` was used in place of `<b><u><em><big><font>`, and several common web page patterns were identified with manual rules and replaced using dedicated symbols, e.g. `<addtobasket>` was used in place of forms that satisfied a set of manually-defined rules.

Experiments were carried out using three different HMM architectures denoted as 'naive', 'word N-gram' and 'submodel'; descriptions of the architectures are in [87]. The results presented in Table 3.2 for the *name* and *price* slots were obtained using the naive, word n-gram and submodel approaches respectively. The remaining slots do not exhibit significant internal structure and currently we have their results just for the naive model. Precision and recall was measured on a per-token basis. All results were obtained using 10-fold cross-validation on the whole set of labelled 100 documents, with the presented values averaged. Both non-naive approaches to modelling slot values however suffer from data sparseness, which probably causes the degradation of precision in some cases.

Instance Composition

While the size of data was acceptable for training HMMs for discovery of individual slots (such as bike name, price or picture), we would need much more data to learn how to *compose* them into whole *instances of product offers*—this task that can be, in the IE terminology, characterised as *template extraction*. Clearly, it is only this task that makes the whole extraction

effort sensible.

In the first approximation, we are using a rather toy algorithm for grouping the labels produced by annotation. The algorithm processes annotations sequentially and exploits information on required/optional slots and their allowed cardinality, defined by means of a tiny 'presentation' ontology. Essentially, a slot (i.e., annotated item) is added to the currently assembled (bike) instance unless it would cause inconsistency; otherwise, the current instance is saved and a new instance created to accomodate this slot and the following slots. Despite acceptable performance on error-free, hand-annotated training data, where the algorithm correctly groups about 90% of names and prices, this 'baseline' approach achieves very poor results on automatically-annotated data: on average, less than 50% of corresponding names and prices are matched properly, often for trivial reasons. We plan to replace the 'toy' algorithm with a more sophisticated version, which would be reasonably robust on automatically annotated data. Namely, the most critical problems of the 'baseline' algorithm are connected with *missing slots*, *multiple different references* to a single slot, and with *transposed tables*; for some of these, partial solutions have recently been suggested by IE research (e.g. [19, 23]) and could be reused.

Related Work

Recently reported IE tools for semantic web are *S-CREAM* [34] and *MnM* [97]. They pay significant attention to efficient coupling of training data mark-up and subsequent automated extraction of new data. *Armadillo* [19] is probably the most advanced information extraction tool explicitly addressing the semantic web standards such as RDF. Its strong point is bootstrapping, which minimises the human annotation effort. In contrast to most previously published results, we focus on *company websites*; presumably, they exhibit less transparent logical structures and fewer data replications than e.g. computer science department pages or bibliographies, the domains most favoured by web IE applications. CROSSMARC [63] focused on product sites, it however did not use a presentation ontology (it rather relied on terminological ontologies and NLP techniques), and did not seem to pay particular attention to presentation of extracted results in semantic web format (see section 4.3).

3.3 Extraction of Triples based on Visual Patterns

In this section we present a rather ambitious approach to web IE. To the difference of previous ones it has only been designed at conceptual level and not yet implemented (apart from a simple motivating experiment). It is however one of most promising ones for future work. The key ideas are: to *decompose* the mapping between HTML source code and 'semantic messages' into multiple parts, and to employ a *generic data model*—the (RDF–like) SPO triple.

Multi–Layer Mapping

Since the choice of HTML tags is always constrained by the outlook in the browser, we propose to model the outlook (in terms of 'visual' relations and properties such as 'above', 'tabular–left–to', 'emphasized' or 'indented') as an intermediate layer between the HTML source code and the semantic model. Though the separated mappings are still *n–to–m*, the *n, m* are likely to be smaller than if the HTML structures were directly matched with 'semantic messages'[10]. To further reduce complexity, we identified a single, very simple model applicable on the majority of messages: the *subject–predicate–object* (SPO) triple. It expresses that "the *value* (i.e. object) of *property* (i.e. predicate) X for *entity* (i.e. subject) Y is Z". The wide usability of this structure seems to be endorsed by its adoption for the *Resource Description Framework* (RDF)[11].

The leftmost part of Fig. 3.2, relating HTML source patterns to visual patterns, represents by itself a hard problem we do not address here. The middle part maps visual patterns to the SPO triple: the *object* part of HTML code is likely to 'follow' (in a varying visually–topological sense) the *predicate* part, e.g.:

- (X: short font–emphasized text) *above* (Y: short non–font–emphasized text) → (X=P, Y=O)
- (X: short text) *followed–with* (colon) *followed–with* (Y: short non–font–emphasized text) → (X=P, Y=O)
- (X: short font–emphasized text) *tabular–left–to* (Y: short non–font–emphasized text) → (X=P, Y=O)

The *subject* is usually not referenced in HTML code; typically, it is the company itself or one of its products/services, which can be expressed in RDF e.g. by means of an anonymous resource:

```
<rdf:Description about="http://www.XY.com">
 <dc:References rdf:resource="_anon1"
   a:Email="info@XY.com" />
</rdf:Description>

<rdf:Description about=
"http://www.XY.com/catalog#item3">
 <dc:References rdf:resource="_anon2"
   a:Price="800" />
</rdf:Description>
```

The rightmost part of diagram, relating the generic model to specific messages, is treated in more detail in the following section.

[10] Earlier we empirically identified common 'messages' on business sites, such as company profiles, contact info or catalogues.

[11] http://www.w3.org/RDF/

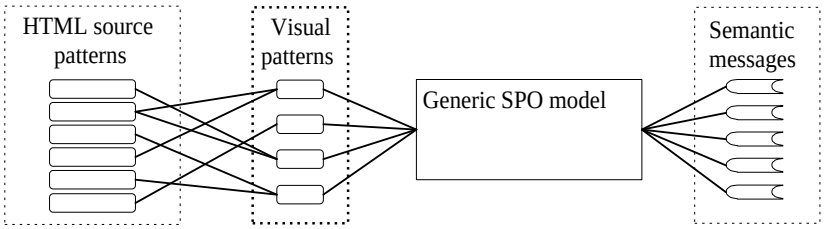

Figure 3.2: Multi–layer mapping

Configuring the SPO Model

The generic SPO model is reflected in the structure of *SPO extractor* (SPOE), the universal model we propose for detection and extraction of specific types of semantic messages:

Definition 1 *An SPO extractor is a tuple* (S, P, O, V)*, where*

- S *(the subject specification) is a logical expression.*
- P *(the predicate specification) is a pair* $(Pred, Lex)$*, where* $Pred$ *is a semantic predicate and* Lex *either a lexical pattern or 'nil'.*
- O *(the object specification) is an information extractor.*
- V *specifies the subset of visual patterns*[12] *applicable on the given extraction task.*

Indicative *lexical patterns* are understood as clue for identifying the HTML code corresponding to P; they may however be left out—e.g. an address at a company homepage could be considered as 'contact address' even without preceding pattern such as 'Contact:'. *Semantic predicates* should be defined as ontological properties, i.e. valid RDF resources. The nature of *information extractors* may vary from e.g. identity function ('pick up the whole content of element') to complex linguistic or statistical models. The *logical expression* specifying the subject could be just a default value such as 'current page' or 'website homepage', or could return different values depending e.g. on the semantic class of current page.

A simple algorithm for discovery of 'implicit RDF statements' for semantic predicate $Pred$ in a given HTML page Pg may, for example, look like this (looping omitted for brevity):

1. Let (S, P, O, V) be an SPOE, P=$(Pred, Lex)$, O=$Extr$.
2. Find in Pg an occurrence of Lex (possibly using a fulltext index) in the form of XPath address $Addr$.
3. Find in Pg an instance I of a visual pattern $Vis \in V$, such that the P–part of I contains $Addr$.
4. Apply $Extr$ on the O–part of I, yielding Obj.
5. Set $Subj$ to the resource specified in S.

[12] In the sense of the previous section; a single library of visual patterns could be reused by different SPO extractors.

6. Return the RDF triple $(Subj, Pred, Obj)$.

Small 'predicate–object' patterns could also be embedded in the code of the 'object' part of a larger pattern: e.g. 'price' in a 'catalog' or 'e–mail' in 'contact info'. This could be modelled by meta–predicates, and exploited by a more complex, recursive algorithm.

Experiments on Data

In the first try we focused on contact information data, i.e. postal and/or email address, see Table 3.3. We randomly selected 101 links from the Business category of *Open Directory*[13], and visually examined the HTML code and outlook of the respective websites. We found some form of contact information within 60 sites: either at the main page or at a page accessible via an appropriately labelled link (such as 'Contact us' or 'About us').

For all occurrences of *address*, we assigned the possibility of their automatic extraction to one of the categories:

1. 'Inextractable': contact info as image, or hidden in text without any indication as to what the address could be, where it begins and/or ends.
2. 'Extractable using a lexical pattern': a lexical indicator (such as 'Mail:' or 'Contact:') preceded the address, in most cases inside its own ('block' or at least 'inline') HTML element.
3. 'Extractable but no external indication': no lexical indicator was present but the address was structured enough to be possibly extracted using advanced (statistical) methods; the HTML structure could at least help to estimate the boundaries.

The simplest approach to *e–mail* extraction would surely be to employ a wrapper class for `<a href="mailto:XXX">`: this would work in approx. 50 % of cases. When, however, the address does not have the form of hyperlink and its end coincides with that of an appropriate HTML tag, an SPO extractor (using lexical indicators such as 'E–mail:') would still work (11 % of cases).

A by–product of our survey was a small statistics of use of explicit *metadata*. Some form of it occurred on nearly half of the pages, structured metadata (Dublin Core) however appeared just once, and only 4 pages contained metadata with contact information: this manifests the importance of WebIE. For contact information, 'implicit RDF metadata' on output might look like

```
<rdf:Description about="http://www.XY.com">
  <dc:Creator>Joe Bowen</dc:Creator>
  <imp:email>info@XY.com</imp:email>
  <imp:addr>42 StreetX, Bigcity, 111 54</imp:addr>
  <imp:phone>1-800-123-456</imp:phone>
</rdf:Description>
```

[13] `http://dmoz.org/Business`

Extractable address	50	83 %
– using lexical indicators (SPOE)	21	35 %
– only using advanced methods	29	48 %
Non-extractable address	4	7 %
Contact info w/o address	6	10 %
Extractable email:	38	63 %
– using a simple 'mailto:' wrapper:	31	52 %
– using lexical indicators (SPOE):	27	45 %
– only using lexical indicators (SPOE):	7	11 %
Contact info w/o email	22	37 %
Any metadata present	29	48 %
Metadata with contact info present	4	7 %

Table 3.3: Results for the 60 sites with contact info available

and later be converted to 'real–world' facts.

Discussion

We hope that exploitation of SPO structures could focus extraction on promising parts of the HTML code, and thus increase the accuracy and reduce the complexity of conventional IE methods. The fact that same visual patterns (expressing the P–O relationship) could be reused for a wide range of 'semantic messages' may alleviate the training of extraction models. We could also use analogy with Hearst patterns [35] in free, linguistically contiguous text, e.g. 'such as', which are often used to extract taxonomic or even non-taxonomic relationships among concepts, and can be used to bootstrap further extraction [17].

Related Work

Visual patterns in HTML code or rendering have already been suggested for IE by several authors. Their form was typically tailored to tree structures [15], tables [30] forms [28], or to larger blocks of text obtained from scanned documents via OCR [33]. None of these projects however proposed a generic triple-based model as ours.

3.4 Analysis of Webgraph of Company Sites

Topology-based web analysis (also known as webgraph analysis) is typically much faster than other methods, since it deals with substantially smaller data (hyperlinks represented as source-target pairs); those have often been collected in advance, in the site download phase. However, it offers limited material for semantic interpretation unless combined with other types of analysis, such as that of URL strings, free text, HTML code or metadata. For example, for company sites, the role of page as 'hub' in a topology may be positively

correlated with its (HTML-based) classification as 'product catalogue'; the result obtained by more sophisticated methods may thus be confirmed or questioned. We hypothesise that topology analysis may be beneficial for two interconnected tasks: semantic classification of pages, and identification of pages belonging to the same logical document. Analysis of data suggested that most information for either task can be derived from the discovery of user-oriented navigation structure. The assumption of existence of such structure represents quite widely reusable 'prior knowledge'.

In the initial survey, we collected the topologies of 75 randomly selected company websites. By their predominant structure, we distinguished the following types:

- Single file (i.e. page without links, or with outward links only)
- Hierarchy with limited connectivity of pages in the same layer.
- Hierarchy with complete connectivity of a set of sibling pages in the second, and possibly subsequent, levels of hierarchy. The union of such sets will be denoted as *navigation structure* (NS), since it enables easy navigation over mutually related topics; the individual sets (possibly multiple ones at the same level, but each with a different parent page) will be called *NS components*.
- Chain (sequence) of pages
- Sites committed to MM Flash (beyond the reach of our topology analysis).

Among the 47 NSs, 38 had their components only in the second layer of the hierarchy, the remaining 9 also in one or more subsequent layers. Since the NS patterns were frequent and conspicuous (could be determined by topology alone), we took their discovery as starting point.

We only sketch the basic steps of the NS discovery algorithm. More details can be found in [98]:

1. Collection of initial set I of candidate pages. Beginning from a start-up page, pages referenced by links are iteratively added. Pages beyond the current server are not considered.
2. Adjacency matrix and minimal-distance matrix of I is constructed. Depth of each page is computed as minimal distance from the start-up page.
3. In each set of pages with same depth, NS components are subsequently sought, as sets of at least three pairwise interconnected pages. Their union is output as the resulting navigation structure.

A side-product of the algorithm is the compactness measure of the website, computed from the minimal-distance matrix. It roughly expresses the accessibility of information, and thus indirectly the quality of website design. As a follow-up step, a tentative form of website 'core'

discovery, as a particular type of logical document, was also implemented. The 'core' only contains the pages that point to all pages from the NS. The algorithm has been implemented as a collection of PHP scripts, with both HTML and web service interface. The former also provides webgraph visualisation and displays statistical information (e.g. adjacency and minimal-distance tables).

For an experiment, 42 sites of 'organisations offering products' (nicknamed as OOP) were randomly chosen from the 'Business' branch of Open Directory . The size of the sites ranged between 1 and approx. 200 pages, the average value was 52. For 20 of them (47.6%), a NS was identified; most of the other sites, namely 15 (35.7%), consisted of a single page. The NSs consisted of one (most often, always in depth 1) up to 9 components; the average value was approx. 2. The average size of NS component was approx. 6, which corresponds to 'reasonable' size of menu bar. Unfortunately, the website core discovery part of the algorithm did not scale well to larger sites, since unacceptably many useful content pages were excluded. More detail on the experiment can be found in [98].

Semantic classification of pages has been designed at informal level but not implemented.

Related Work

Attardi [8] uses topology analysis as filtering method for content-based page classification. Links to frequently referenced pages are understood as 'structural links' and such pages are not submitted to classification. This low-cost approach is however unapplicable to many company pages as they may contain important information even on pages that belong to the navigation structure. For small websites, all pages may even be eliminated. Mathieu and Viennot [53] described the structure of the web using two different structures: link topology as well as file system hierarchy (derived from local parts of URLs). They ordered 8 millions of URLs according to the directory/file structure and computed the adjacency matrix. Clusters along the diagonal of the matrix, such that most links from the cluster again lead to the cluster, are understood as logical websites. The method is however sensitive to file system structuring rules. Such rules are often disobeyed, and actual file system structure may even be hidden behind 'virtual URLs'.

3.5 URL-Based Classification of Web Documents

First experiments with classification of company URLs [79] were already carried out in the context of *VŠEvěd* project [10], which was pre-cursor of *Rainbow*. The goal was to assign the documents retrieved by the *VŠEvěd* meta-search engine into meaningful categories, without the need of downloading their full text; earlier studies [73] indeed showed that humans can make significant deductions about the content of URLs[14].

[14]The situation however significantly worsened in the last few years due to massive proliferation of dynamic web technology, which often obscurs the URLs.

URL-Oriented Document Typology in *VŠEvěd*

We used three different viewpoints on web pages typology:

1. type of the document (TypD) - we adapted here the typology given in the Dublin Core set of description elements [99] (recognised types are e.g. information about persons, information about projects, bibliographical information etc.)
2. sphere of the document (TypS) - this typology roughly corresponds to first level domains (e.g. academical domain or commercial domain)
3. technological type of the page (TypT) - this typology is based on syntactical elements in the page (e.g. on-line forms, indexes).

The links could be grouped according to the types of pages, in *VŠEvěd*. The three basic ways of grouping (that correspond to the three viewpoints on web pages typology) are extended by so called 'waterfall grouping'. This option uses TypD as a first grouping criterion, links with no recognised TypD are grouped according TypS and the remaining links are grouped according to TypT.

Building the Knowledge Base

The heuristics that recognize different types of pages were based on an analysis of about 100,000 URLs pointing both to English and Czech servers. In order to obtain a large set of generic URLs required for the initial frequency analysis, we exploited two sources and merged (syntactically correct) URLs from both.

- We submitted extremely general (to say, "empty") *queries to search engines* – e.g. in the form `+domain:com` in the case of AltaVista, and *extracted* (by means of dedicated text extractors) only the URLs of the "hits" returned.
- To eliminate the bias incurred by the particular structure of search engines indices, we also scanned *the cache of an HTTP proxy server*.

The URLs have been parsed to their constituent parts (hostname, directories, filename etc.) and to individual terms, abstract (e.g. calendar) concepts have been deduced from overly specific data, and, finally, the resulting structured collections of terms and concepts have been filtered by means of frequency analysis[15], while discounting multiple occurrences of the same term from the same server (for details see [79]). The (alphabetically reordered) lists of terms and concepts served as a base for manual formulation of category-recognition rules.

We also performed a frequency analysis of terms occurring in the URLs and formulated rules for terms with relative frequency above 0.05%. To have an idea of the relative frequencies of

[15] In addition to unigram analysis, bigrams have been examined, too. This has yielded e.g. the compound terms "about us" or "yellow pages".

Table 3.4: Fragment of lists of terms and frequencies from directory path and filename

Directory occurrences	**%**	**Filename occurrences**	**%**
about	0.17	about	0.33
access	0.15	ad	0.43
academic	0.12	all	0.32
ad	0.11	archive	0.16
admissions	0.05	art	0.22
ads	0.21	arts	0.14
all	0.11	banner	0.32
and	0.18	bar	0.18
archive	0.07	bar	0.18
archives	0.12	bio	0.26
art	0.19	blank	0.20
articles	0.08	black	0.12
...		...	

terms within the sample, see the fragment from the beginning of the lists for both directory and filename, at Table 3.4. We can hypothesise that the most frequent terms typically belong to three semantic groups:

- terms related to the page category (in the sense studied in this paper)
- terms related to an (extremely general) subject topic
- terms with no semantic meaning of its own; they can however bring some information if associated with other terms.

Problem of Ambiguous Terms

In the process of manual formulation of recognition rules, *search engines* have served again, namely to verify the reliability of key terms. Special queries of the sort `+url:<term>` (or `+url:<term> -host:<term>`) have been posed, and the lists of hits visually inspected (only the first 20–30 per query, which should suffice to identify significant deviations from the main, expected, meaning). Some very frequent terms, such as "art" (article, but also page about art), "bio" (biography, but also page about biology), "cat" (catalogue, but also page about cats) or "pub" (list of publications, publicity page, or page about restaurants) have thus been labelled as "ambiguous", and submitted to further processing. The disambiguation was carried out using Inductive Logic Programming, and relied on *textual snippets* returned by search engines. As this goes beyond web space analysis in 'virgin' state, we do not describe it any further; details are in [81].

Testing the Knowledge Base

We focused our experiments on testing the functionality of typological grouping of hits returned by search engines queried by VSEved. At first, we used the same set of 28 queries that were used to test the system Clever [16]. The queries are rather frequent english terms like "alcoholism", "lyme disease" or "stamp collecting". VSEved used AltaVista search engine to get the results, first 10 (most relevant) hits returned for each query were then grouped both according single criterion (TypD, TypS and TypT respectively) and according to the waterfall criterion. The second set of experiments was oriented on more specific queries. We used as queries terms taken from titles of papers presented at the DATASEM'98 conference (Czech conference on databases). We formulated 11 Czech and 11 English queries. Again, we performed single grouping and waterfall grouping on results obtained when querying the Czech search tools AltaVista and Kompas. The results of both experiments are summarised in Table 3.5. Sections CLEVER, DATASEM CZ and DATASEM EN correspond to the three sets of queries. We show both absolute and relative number of classified links. The rows 'single' show number of pages in each typology (one page can belong to all three typologies), rows 'waterfall' show what was the first typology a page was classified. So. e.g. in the row 'CLEVER waterfall' 154 pages were classified according to the TypD typology, from the remaining 120 pages 100 pages were classified according to the TypS typology, 5 pages were only classified according to the TypT typology and 15 pages remained unclassified.

Dataset		TypD	TypS	TypT	Unknown
CLEVER:	Single	154	242	16	
	Single (%)	56,2	88,3	5,8	
	Waterfall	154	100	5	15
	Waterfall (%)	56,2	36,5	1,8	5,5
DATASEM CZ:	Single	33	45	1	
	Single (%)	38,4	52,3	1,2	
	Waterfall	33	31	0	22
	Waterfall (%)	38,4	36,0	0,0	25,6
DATASEM EN:	Single	8	76	7	
	Single (%)	7,8	73,8	6,8	
	Waterfall	8	68	2	25
	Waterfall (%)	7,8	66,0	1,9	24,3

Table 3.5: Tests of URL-based classification

Web Service Implementation

Later, a classifier specifically tuned to URLs of *company* websites was implemented using an analogous approach. It is available as a component of the *Rainbow* system[16]. It has however not been thoroughly tested on data, since the role of default page classifier was eventually attributed to a Bayesian classifier, for the practical use in bicycle application.

[16]The web service interface is available at `http://rainbow.vse.cz/doc/services/anaurlservice.html`

3.6 Cropping the Explicit Meta-Data

Explicit metadata embedded in META tags (or, ideally, RDF) are an important resource of information in web pages, as they provide the most important information items about the site content, which may be quite hard to obtain from the full HTML code. The main problem is however their relative scarcity. By a the statistical survey[17], the distribution of META tag types was as follows: the *keywords* tag was present in 11.8% of the sample, *description* in 9.6%, *generator* in 7.5%, *robots* in 3.9%, *author* in 3.3%; all other tags had frequency below 1%.

The simple META tag 'analyser' designed within *Rainbow* was thus parametrised so as to extract the content of *keywords*, *description* and *author* tags only (as *generator* and *robots* are not related to the page semantics). The way of its integration into the visualisation tool is shown in section 4.1.1, and an experiment considering META tags is described in section 4.2.1.

At the time of the above mentioned survey (1997), RDF had not existed yet. However, with the growing popularity of semantic web technology, the RDF annotations already present as part of web pages should be taken into consideration during their automated analysis.

3.7 Analysis of Image Information

The analysis of image information was used, within the *Rainbow* project, as part of the application on bicycle product catalogues, see section 3.2. All images used in our experiments come from a collection of 133 HTML documents chosen from the Google Directory *Sports-Cycling-BikeShops-Europe-UK-England*. Each document contains from 1 to 50 bicycle offers, and about 61% of offers include a bicycle picture. There are typically 3–4 documents from the same shop in the data. Together there are 1624 occurrences of 900 unique images[18]. Sample images are shown in Figure 3.3.

In our data, most repeating images are advertisement banners and images used for page layout. Of the 1624 image occurrences, there were 598 bicycle images (positive examples) and the remaining 1026 images were considered negative. Positive examples of bicycle images include those that were not part of any bicycle offer labeled for extraction.

Three different types of features were used for classification:

1. Image similarity based on *Latent Semantic Indexing*; the image bitmaps were transformed to vectors of features and reduced via Singular Value Decomposition, for more details see [64]. The classification was done using the K-nearest neighbour approach with respect to positively labelled examples. The error rate of the classifier was 26.7%.

[17] `http://vancouver-webpages.com/META/bycount.shtml`
[18] The image collection is available from `http://rainbow.vse.cz`.

Figure 3.3: Sample images sorted by similarity to the first image

2. *Image size* modelled using a 2-dimensional normal distribution. The error rate of the classifier was 6.2%.

3. *Colour histogram* constructed according to the so-called HSV colour space, yielding 162 different colour groups. The error rate of the classifier was 5.2%.

All results were measured using 10-fold cross-validation, where images were split so that no images from a single HTML document could appear both in training and test data. Tools from WEKA were used for actual classification based on multiple features of the same type. Combination of different types of features is discussed in section 4.1.2. Details on the experiment are in [45] and [46].

Chapter 4

Ad Hoc Integration of *Rainbow* Methods and Tools

In this chapter we subsequently discuss the different ways of integrating analysis results in *Rainbow*. We roughly divide them into two categories, denoted as horizontal and vertical integration, respectively. They can be characterised as follows:

1. In *horizontal integration* (section 4.1), the outputs of multiple methods returned for the same object in data are contrasted, compared, alternated or combined.
2. In *vertical integration* (section 4.2), the outputs of one method are used as input for another method.

Finally, we devote a separate section (4.3) to an experiment with converting the results of web site analysis to RDF and presenting them to the user in the form of navigational search interface.

Note that a more sophisticated way of horizontal integration is outlined in Chapter 5, where a general formal model is presented. Furthermore, a more advanced (especially, more flexible) method of both horizontal and vertical integration, based on problem-solving methods, is discussed in Chapter 8.

4.1 Horizontal Integration

As horizontal integration, we first (section 4.1) present a simple parallel display of information obtained by different methods for the same physical *page*. This method, implemented by J. Kosek, leaves the actual integration upon the user; it was developed for testing purposes, so as to ease manual experiments with the tools in question. Second (section 4.1.2), we discuss our experience with integrating multiple methods that are applied together on the same object, namely, *image*; the bulk of the work has been done by M. Labský, P. Praks and M. Vacura.

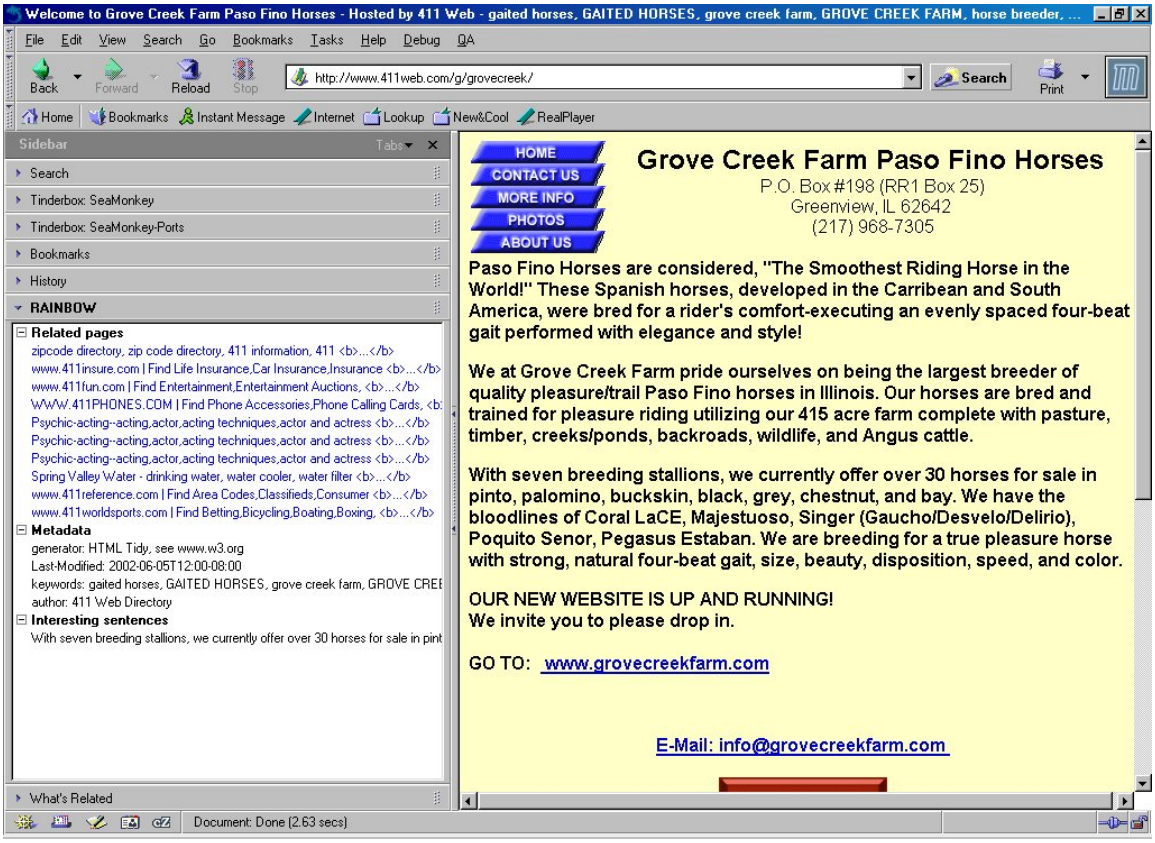

Figure 4.1: Screenshot of Mozilla with the *Rainbow* navigation pane

4.1.1 Parallel Result Display in Navigation Interface

A (testbed) *navigation interface* to the results of early *Rainbow* tools has been built in the form of plug-in panel in the open-source Mozilla browser. Every time a new page is open in the browser, the page is downloaded and pre-processed (provided it is not yet in the source-page database), and the analysis modules are invoked. Their results are just displayed, no integration is performed. Fig. 4.1 shows a screenshot of the browser equipped with the *Rainbow* navigation pane. The sections of the pane correspond to the listing of 'similar' pages provided by the Google server (as an additional, external web service), to the content of selected META tags (raw explicit metadata), and to the sentences provided by linguistic analysis[1].

The navigation interface was used e.g. for experiments presented in section 4.2.1.

[1]The presence of 'offer' (one of the 'indicators' found by web directory mining, see section 3.1) as the main verb in the sentence was basis for its selection.

4.1.2 Integration of Image Analysis Methods

For the bicycle product catalogue application, we built an image classifier that used all of the features described in section 3.7: the similarity score, size score based on width and height, and the 162-dimensional HSV histogram vector. We also tried to add the image's occurrence count within the same HTML document, since count above 1 seemed to be a good predictor of the image not depicting a product. However, this feature was dominated by the other features and did not bring further improvement.

The best performing classifier within the WEKA workbench was the PART decision list, with an error rate of 2.6%. Results for all single-feature classifiers[2] and the combined classifier are summarised in Table 4.1.

Table 4.1: Image classification results

	Similarity	Size	Size2	Histogram	Combined
Error rate (%)	26.4	6.2	3.2	5.2	2.6

On our image collection, the classification results seem to be promising for the purpose of further IE from web pages. However, good results are largely due to the collection's specific nature, which was especially exploited by the size- and histogram- based classifiers. For most bike shop pages, product images tend to have specific sizes – we could identify one cluster of larger product images that appeared in product detail pages, and another cluster of smaller product images, typically found in product listings. Furthemore, most product images seemed to have a similar ratio between their width and height, which could be well modelled by the normal distribution utilised by the size-based classifier. On the other hand, non-product images were often advertisements, web page graphics (such as logos, headers, buttons or menus), and images of other products including "picture not available" images.

Advertisments often had standardized sizes, and only rarely resembled product images in size. Page graphics (e.g. manufacturer's or shop's logos) sometimes matched product images in size, however they were often distinguished by either the histogram or similarity classifiers. The task of the histogram classifier was also relatively easy since most product images had a white (or very light) background, and only a small portion was on dark backgrounds. The "hardest" images in our collection were those of "non-bicycle" products, such as bicycle accessories (often depicted together with a part of bicycle). For these images, the LSI similairity's contribution was most important.

4.2 Vertical Integration

During the experiments with *Rainbow* components, vertical integration also appeared. We will first discuss the mostly manual experiment (done by the author of the thesis himself) with locating pages containing general company descriptions with the help of URL analysis. Then we will mention the procedural control structure for extraction of bicycle product information

[2]Size2 includes the original image width and height as attributes.

(by O. Šváb), which implements and refines the former experiment. Note that the application of image analysis as support for information extraction (section 4.1.2) can also be viewed as vertical integration.

4.2.1 From URL Analysis to Extraction of Company Descriptions

In this experiment we focussed on the task of extracting general company descriptions (profiles) from websites of 'organisations offering products or services' (we therefore use the acronym OOPS). The descriptions delimit the areas of expertise of the company and the generic types of products and services. They can be presented as free–text paragraphs, as HTML structures such as lists or tables, or as content of META tags such as *keywords* or *description*[3]. The profile usually occurs either immediately at the main page of the site, or at a page directly referenced by this page. The URL of such profile–page is very likely to be 'indicative'. This favourises 'navigational' access: rather than analysing exhaustively the whole website, the link can be followed from the main page.

In our experiment, we randomly chose 50 websites, whose main page was referenced by the 'Business' part of Open Directory[4]. Of these, 36 sites could be considered as directly belonging to 'OOPS'; other were multi–purpose portals or information pages. We exploited the *Rainbow* system in its state to date (end 2002), i.e. the source data module to acquire and pre–process the pages, the linguistic (section 3.1) and metadata (section 3.6) analysis to extract information, and the navigational assistant (section 4.1.1) to view the results. The simple task–specific knowledge base for linguistic analysis only contained the 10 most frequent 'indicators' such as 'offer', 'provide' or 'specialise in'. Since the URL analyser (section 3.5) was not yet operationally connected to *Rainbow*, we simulated its behaviour by observing the links on the page and following them manually. As 'knowledge base' for URL–based detection of profile page, we set up[5] a collection of four significant strings, namely: *about*, *company*, *overview*, *profile*.

Table 4.2 lists the numbers of websites (among the 36 in the pre–selected sample) that contained the target information in the respective forms. For free text, we distinguish the cases where the presence of company profile was only verified *visually*, those where it has been detected by *Rainbow* (denoted as 'FTD'), and, most specifically, those where the target terms from free text (successfully detected by *Rainbow*) were not contained in META tags. Note that, in this particular row, the fourth column is not equal to the sum of first and third, since cases when information in free text and META tags matched *across* the two pages also had to be ignored.

We can see that profile information is often contained in META tags as well as in free text, but quite rarely in structured HTML form. It might seem that the added value of linguistic analysis is low compared to META tags (which are available more easily). Note however

[3]The former are lists of terms while the latter are either lists or free–text paragraphs.

[4]`http://www.opendir.org`

[5]We arrived at this collection after having processed the first 20 pages and have not changed it afterwards. Admittedly, this is not a sound cross–validation methodology, it could be however tolerated given the triviality of the task.

Table 4.2: Presence of 'profile' information about the company

Information present in	Main page	Follow–up (profile) page	*Only* follow–up page	Overall	Overall (% of sample)
META tags	28	10	4	32	89
Free text	11	15	12	23	64
Free text, discovered (FTD)	8	8	7	15	42
FTD while not in META	2	5	5	3	8
HTML–structured text	3	3	3	6	17

Table 4.3: Link to the profile page

Recognisable by	Cases	Cases (%)
Both anchor text (or ALT) and URL	15	62
Anchor text, not URL	4	17
URL, not anchor text	2	9
Neither anchor text nor URL	3	12
Total (page exists)	24	100

that in most cases, META tags contain an unsorted mixture of keywords including both generic and specific terms related to the company, products as well as customers. Parsing the free–text sentences can help distinguish among semantically different categories of important terms.

Table 4.3 shows the availability and accessibility of a specialised 'profile' page. We can see that only 3 of the 24 pages did not have the link denoted by one of the *four* terms from our set. Analysis of URL (as specific data structure) is more-or-less redundant here since the same information can mostly be obtained from the *anchor* text of the link, or from the ALT text if there is an image instead of textual anchor. Never mind, our hypothesis of 'informative' URLs has been confirmed.

4.2.2 Multiway Extraction of Bicycle Information

As part of the bicycle catalogue application, we implemented a simple sequential algorithm starting with one (or more) address of web page and ending up with upload of information about bike offers, companies and metadata as *RDF files* into the *Sesame* repository (see section 4.3).

There are four analysis tools called by the algorithm. They are listed in the order as they

run:

- *AmphoraWS* [3] is used for collecting the addresses of Bike Product Catalogues. *AmphoraWS* gets one address where are potentially links to Bike Product Catalogues. The role of initial web page is played by the web page listed in the respective node of the Google Directory. *AmphoraWS* returns list of web pages potentially consisting Bike Offers.
- *Page classifier*: we want to distinguish among 'profile' web page about the companies themselves, product catalogues pages, and other web pages (which will not be processed further). The classification can be based either on the analysis of structure of *URL* with *URL analyser* tool or on the analysis of word unigrams with a *Naive Bayesian classifier*[6].
- *Information Extraction (IE) tool.* Web pages classified as Bike Product Catalogues are currently processed with a tool based on *Hidden Markov Models*, and individual bicycle product offers are extracted.
- *Linguistic analysis* as the second IE tool is to process web pages classified as profile web page for extracting information about companies.

Next phase consists of transformation of extracted information into *RDF*. This basically means that information are integrated around *Uniform Resource Identificators*. In the last phase, the *RDF data* are uploaded into *Sesame repository* endowed with a dedicated *HTML search interface*, see [87] and section 4.3.3.

4.3 *Rainbow* as Semantic Web Annotator

The topics related to semantic web are ubiquitous in the thesis, for example, applications of ontologies are discussed in several chapters. Here we discuss the relation between the principles of the *Rainbow* project and the semantic web in narrow sense: RDF annotations that can be processed by common semantic web tools. We again illustrate it on the example from the bicycle product domain; the implementation described is due to O. Šváb.

Note that the use of semantic-web-inaware IE methods means that the results have to be transformed from their original format to RDF. An alternative is to consider the 'triple-based' view to an IE method itself. An example of such (not-yet-implemented) approach was shown in section 3.3.

4.3.1 RDF Schema for Bicycle Sale Domain

The information extraction tools described in section 3.2 are currently (in the best case) able to yield instances of *retail offers*[7], typically consisting of *name* of a bike, its *price*,

[6] This latter tool is not described in this thesis, as it is merely an implementation (by O. Šváb) of well-known principles

[7] We use this term instead of 'bike offer', so as to cover separately-sold bike components.

details on its *components* (such as fork, frame, rear derailer etc.) and its *picture*. This information thus has to be covered by the underlying schema for the result repository. We are using the RDF format, which gives us useful flexibility when dealing with incomplete and imprecise data; hence, our data schema has the form of *RDF Schema* ontology. In addition to information produced by the HMM, the schema also covers some information about the *company* that offers the bicycle to be extracted other tools described in Chapter 3, e.g. a more linguistic-oriented (free-text) analyser, META-tag analyser or URL analyser, as well as by HMMs trained for a different sub-domain. Finally, we need to represent *metadata* associated with the extracted facts, such as "Statement XY has certainty 0.75" or "Statement XY was produced by URL analysis module".

Examples of information triples (in free-text form, to avoid syntax issues) are "Company X offers bike Y". "Bike Y has name Rockmachine Tsunami", "Bike Y has fork Z". "Fork Z has name Marzocchi Air", "Price of bike Y is 2500."

The *RDF schema* of our domain is shown in graphical form on Fig. 4.2. It uses four namespaces: `bike` dealing with bikes themselves, `comp` dealing with (not necessarily 'bike') companies, `pict` dealing with pictures on web pages, and `meta` dealing with metadata on extracted statements. The central point of the schema is the concept of RetailOffer. It corresponds to an offer of BikeProduct (whole bike or component) by a Company; it is also associated with the Name under which and Price for which it is offered, and URL of associated Picture. URI of particular RetailOffer corresponds to the URL of catalogue item containing the offer[8]. BikeProduct is superclass of all bike products. Note that BikeProduct and its subclasses only have 'types' of products as their instances, not individual physical entities. Such 'type' of product can be offered for different prices and even under slightly different names (associated with the given instance of RetailOffer) and accompanied with different pictures, while BikeProduct itself has a 'canonical' name, specified e.g. by its manufacturer. Finally, our way of representing *metadata* for extracted information is based on *reification* and inspired by [12]. The metadata should cover information on which *analysis module* the statement was obtained from, or its *certainty factor*. Metadata are grouped under an abstract class called Meta.

4.3.2 RDF Repository and Query Language

As RDF repository we chose *Sesame*, developed by the Dutch company *Aduna* (earlier *Aidministrator*), see `http://sesame.aidministrator.nl`, mainly because of its adherence to current RDF recommendations by W3C and some features of its original query language, SeRQL. *Sesame* also already proved scalable to larger quantities of data. A known weaker point of *Sesame* is limited support for dynamic schema integration; since we deal with a single RDF Schema fully under our control, this aspect is not of central importance.

[8] Typically the place from where the core information was extracted.

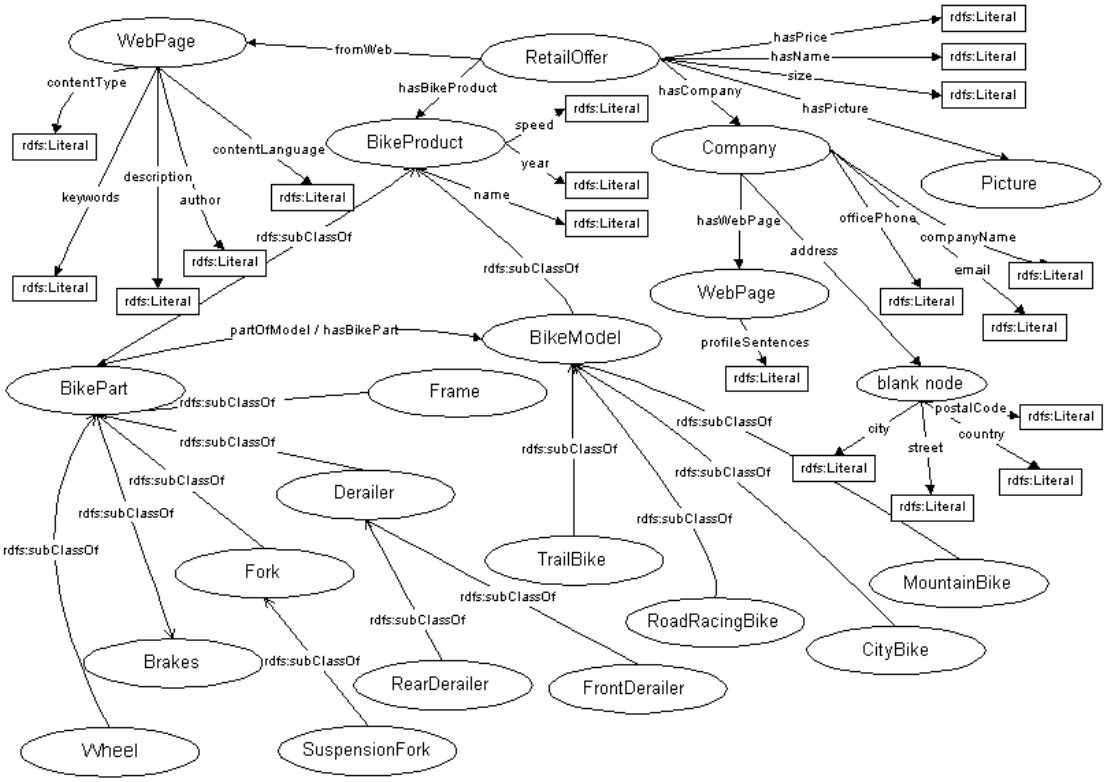

Figure 4.2: RDF schema of bicycle domain

SeRQL[9] ("*Sesame* RDF Query Language", pronounced as 'circle') is a declarative *query language* over RDF and RDF Schema. Its central part is the 'select-from-where'[10] construct similar to SQL. The 'select' part lists the variables to be output. All of them must appear in the 'from' part, which defines the part of RDF graph to be searched, by means of *path expressions*. Finally, the 'where' part includes an arbitrary selection pattern, and the 'using' part defines the relevant namespaces.

Let us demonstrate the syntax and semantics of *SeRQL* on a query from our application domain, which would read in plain English:

Find all retail offers of bicycles whose name begins with "Trek" and price is between 700 and 950. Output the bike name, price and picture, as well as the website and name of company that makes the given offer. Retrieve the retail offer even if the URL of picture is not known.

```
select name, price, picture, web, company
from
{x} <serql:directType> {<bike:RetailOffer>};
      <bike:hasPrice> {price};
     [<bike:hasPicture> {picture}];
```

[9] `http://www.openrdf.org/doc/sesame/users/ch06.html`; it is probably the most influential predecessor of SpaRQL, the language currently endorsed by W3C.

[10] There is an alternative 'construct-from-where' option, which yields RDF triples rather than plain result tables.

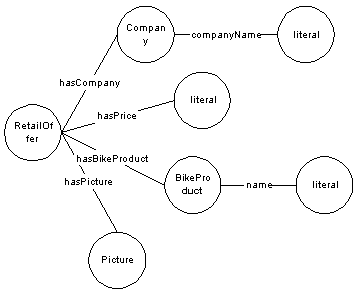

Figure 4.3: RDF graph for example path expression

```
      <bike:hasBikeProduct> {y},
{y} <bike:name> {name},
{x} <bike:hasCompany> {} <rdf:type> {<comp:Company>};
                         <comp:companyName> {company};
                         <comp:hasWebPage> {web}
where name like "Trek*"
      and price >= "700"^^<xsd:double>
      and price <= "950"^^<xsd:double>
using namespace
 comp = <!http://rainbow.vse.cz/schema/company.rdfs#>,
 bike = <!http://rainbow.vse.cz/schema/bikes.rdfs#>
```

The path expression from the example is graphically depicted at Fig. 4.3. In the 'from' part of sample query, all its constituent triples are listed, taking advantage of *SeRQL* shortcut notation: incomplete triples following the semi-colon symbol refer to the subject from preceding triple (here, the `x` variable). Note the brackets around the triple referring to `picture`: this part of graph is *optional*. Support for *optional path expressions* was our major reason for choosing *SeRQL* among three query languages applicable in *Sesame* to date (see [89] for in-the-context comparison): there is obviously a strong need for optional items when dealing with incomplete data extracted from HTML pages.

4.3.3 HTML Query Interface

In order to make our RDF repository available for a casual user, we prepared a domain-specific *HTML interface* with several SeRQL *query templates*. The templates *shield* the user from the syntax of the query language, and offer a simple form of *navigational retrieval*.

Template-based access to bike data relies on two-stage querying. The template for *initial query* (specifying its 'from' part) is quite complicated, rich in optional path expressions; its final shape is tuned by the user, who may refine the 'select' clause (variables), 'from' clause (optional or not), and 'where' clause (comparisons). The results of initial query are the starting point for *follow-up querying*. The user can then reformulate any of the two steps.

At Fig. 4.4 we see a screenshot of query interface after execution of both steps. The initial query (in the upmost pane) corresponded to that from example above, and yielded (in the

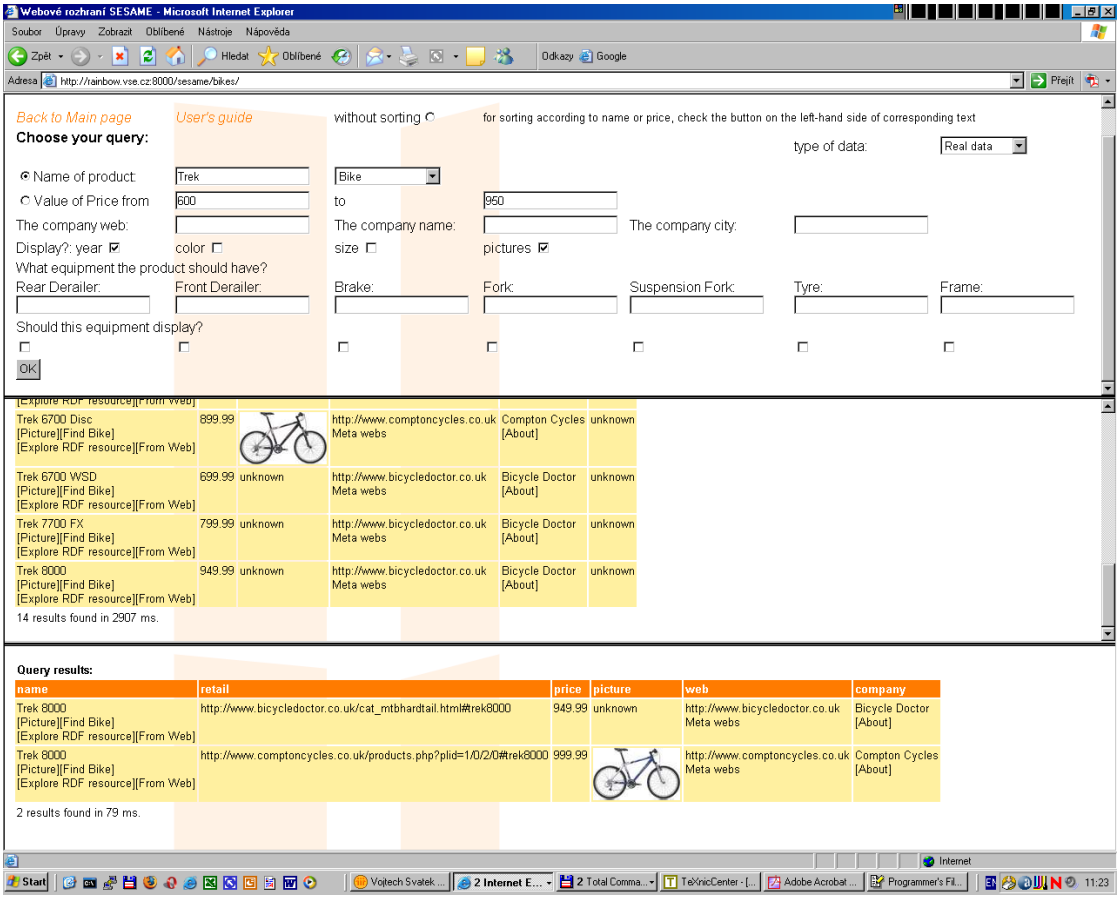

Figure 4.4: HTML interface to bike-offer RDF repository

middle pane) a collection of bicycle offers of desired make and within the chosen price range. As follow-up query, the user clicked on the 'Find bike' link within the 'Trek 8000' offer made by Bicycle Doctor for 949.99 pounds (second in the result list). The lowest pane then displayed both offers of this bicycle present in the repository, the latter (by Compton Cycles) being more expensive but accompanied with a picture. Analogously e.g. company information or enlarged picture can be displayed.

The HTML interface is available at `http://rainbow.vse.cz:8000/sesame/`.

Chapter 5

Formal Model of Horizontal Integration

We propose a model that associates a resource with a collection of properties that belong to three types (closed-domain properties, object properties and content properties) distinguished by the range of their values. The evaluation of 'correct' value assignment differs according to property type, the results can however be aggregated into a single table. We hypothesise that such a model is particularly useful in horizontal integration, when multiple complementary/supplemantary procedures can be applied on different data structures related to same underlying entities. This is typical for websites (hence the choice of illustrative example), which are known to exhibit a high degree of redundancy in presentation, but the model can presumably be generalised to other types of resources, too.

5.1 Resources, Properties, Values and Meta–Information

Let us first formally define the important notions and explain their role in the whole model.

Definition 2 *Let $\mathcal{R}$ be the* universe of information resources *(further only resources). Let $R_t \subseteq \mathcal{R}$ be a* set of resources *of same* type *t.*

An information resource can be any unique entity that carries some information content. In the case of web, it can be for example a physical web page, a hyperlink, or a whole website. 'Physical web page', 'hyperlink', and 'website' can be considered as types of resources.

Definition 3 *Let $P = P_{CD} \cup P_O \cup P_{Cn}$ be a set of* properties *relevant for a set of resources R; P_{CD}, P_O and P_{Cn} are pairwise disjoint. P_{CD} will be called* closed–domain properties*, P_O* object properties *and P_{Cn}* content properties*.*

Every p from $P_{CD} \cup P_{Cn}$ has an associated value set*, V_p. Every p from P_O has an associated resource type t_p.*

We assume that a fixed set of properties can characterise the resources (of a given type) from different aspects, e.g. to indicate the *author* of a given page, the *direction* ('upward', 'downward' etc.) of a given hyperlink, or the *startup page* of a website. The first is an example of a content property, the second of a closed–domain property, and the third of an object property. While the value set of closed–domain properties is assumed as enumerated, the value set of content properties is derived from some standard open data type. For simplicity, let us assume that content properties have character strings for values—this is the typical type of target information in information extraction. The model can thus be understood as unifying, from a very particular viewpoint, the paradigms of *information retrieval* (values of object properties refer to retrieved resources), *text classification* (values of closed-domain properties can be understood as classes of resources) and *information extraction* (values of closed-domain properties correspond to semantically labelled text extracted from the content of resources).

Note that the model is not limited to meta–information in the usual sense, i.e. 'information about the resource itself', but extend it to any information that could be acquired from the resource and is 'somehow' related to it. Typically, it is the case of information about the entity that 'owns' and/or 'created' the resource, e.g. about the company a website is devoted to. This is not a conceptual problem, since such 'second–order' meta–information can be directly captured by *composed* properties, e.g. 'location–of–company–owning–the–site'.

Definition 4 *For each $r \in R, p \in P$, the* reference value *of p for r will be denoted as $Ref(r,p)$.*

- *If $p \in P_{CD} \cup P_{Cn}$ then $Ref(r,p) \in V_p$.*
- *If $p \in P_O$ then $Ref(r,p) = r_k \in R_t$, where t is the resource type associated with p.*

Note that we do not introduce the reference value in the sense of 'ontologically true' value, but just as the value to which other values would be compared. The reference value could be even 'N/A' (i.e., 'not available') if the real value cannot be derived from the resource content in principle.

Now, we will introduce the notion of *meta–information set*, which corresponds to a (possibly partially) filled 'template' over the set of properties.

Definition 5 *A* meta–information set *of resource $r \in R$, $\mathcal{M}_r$, is a set*

$$\{(p_1, v_1), (p_2, v_2), \ldots, (p_k, v_k)\}$$

where $\{(p_1, p_2, \ldots, p_k)\}$ are all properties relevant for R.

A reference meta–information set *of resource $r \in R$, $\mathcal{M}_r^{Ref}$, is a set*

$$\{(p_1, Ref(r,p_1), (p_2, Ref(r,p_2)), \ldots, (p_n, Ref(r,p_n))\}$$

where $\{(p_1, p_2, \ldots, p_n)\}$ *are all properties relevant for* R. *Every resource has* exactly one *reference meta–information set.*

For simplicity, we assume that a meta–information set always covers *all* properties, some of them may however have the value 'N/A'.

Definition 6 *In a given* context, *every* p *from* P_{Cn} *has an associated* value–acceptability relation $Acc_p \subseteq V_p \times V_p$.

The value–acceptability relation may, depending on the nature of the property, amount e.g. to:

- strict equality: $Acc_p(v, Ref(r,p))$ iff $v = Ref(r,p)$ (e.g for a company registration code)
- term permutation (e.g. for keyword lists)
- term superset with length restriction (e.g. for person names—if the page author is 'John Smith', we could possibly accept the value 'Dr. John Smith', sometimes even 'page written by John Smith' but not a long sentence).

The acceptability relation may not be fixed for the given property: it may vary according to 'context'. The notion of context may, in practice, correspond e.g. to an *evaluation session* for different meta–information acquisition procedures. We may thus bias the evaluation toward 'strictness' or 'sloppiness', and obtain coarser or finer distinctions of the procedures' quality. Alternatively, the context may be the *syntactical standard* for true meta–information.

Definition 7 *A meta–information set of resource* $r \in R$, $\mathcal{M}_r$, *is* correct *for* r *in property* p *if it contains a pair* (p, v) *such that:*

1. *if* $p \in P_{CD} \cup P_O$ *then* $v = Ref(r,p)$
2. *if* $p \in P_{Cn}$ *then* $Acc_p(v, Ref(r,p))$.

Note that we chose to require strict identity for properties from $P_{CD} \cup P_O$. In principle, we could introduce some 'acceptability relation' even for these. For example, relaxed criteria for object properties might require, in a certain context, merely sub/super–object (part–of) relation to hold (instead of identity), and similarly, for closed–domain properties, the sub/superclass relationship in a hierarchy could fulfil such role.

Definition 8 *Given two meta–information sets* $\mathcal{M}_{r_i}$, $\mathcal{M}'_{r_i}$ *of the same resource* r_i, $\mathcal{M}_{r_i}$ *is* superior *(or equal) to* $\mathcal{M}'_{r_i}$ *with respect to* r_i, $\mathcal{M}_{r_i} \geq \mathcal{M}'_{r_i}$ *if* $\mathcal{M}_{r_i}$ *is correct for* r_i *in every property in which* $\mathcal{M}'_{r_i}$ *is correct for* r_i.

This enables to compare the performance of two meta–information acquisition procedures on the same resource (e.g. web document).

5.2 Example

Let us demonstrate our scheme on a 'toy' example: a hypothetical web page of a small toy producer (Fig. 5.1).

```
<html>
<head>
<meta author="John Black">
<meta description="Production and sale of wooden and textile toys">
</head>
<body>
<h1>BlackWood Ltd.</h1>
<p>An opportunity for lovers of original hand-carved and hand-knitted toys.
BlackWood specialises in toys for children aged over 6 years,
and in collectioners' items.</p>
<p>In our shop you can find:</p>
<ul>
<li>wooden animals from 4,-
<li>knitted dolls from 8,-
<li>toy furniture collection, special offer for just 120,-
</ul>
<hr>
<it>Page maintained by J. Black, last modification Feb 28, 2002.</it>
</body>
</html>
```

Figure 5.1: Example resource: page of a toy producer

Let us consider this page as our 'current resource' on which the procedures will be evaluated. Let the *reference meta-information set* wrt. this resource be:

$$\begin{aligned}\mathcal{M}^{Ref} = \{ \quad & (p_1 = \textit{page\_type}, \text{'Main information page'}) \\ & (p_2 = \textit{page\_author}, \text{"John Black" }) \\ & (p_3 = \textit{name\_of\_company\_referenced\_by\_page}, \text{"BlackWood Ltd."}) \\ & (p_4 = \textit{domain\_of\_competence\_of\_company\_referenced\_by\_page}, \\ & \text{"toys"}) \\ & (p_5 = \textit{pricelist\_location}, \text{'/html/body/ul'}) \\ & (p_6 = \textit{contact\_address\_location}, \text{'N/A'}) \ \}\end{aligned}$$

p_1 is a closed–domain property, p_2, p_3 and p_4 are content properties, and p_5 and p_6 are object properties. The acceptability relations for content properties will be defined as follows (for simplicity we write $Acc_n(x, y)$ instead of $Acc_{p_n}(x, y)$):

- $Acc_2(x, y)$ holds (for sequences of terms) if x contains the last term from y and no more than three other terms.

- $Acc_3(x, y)$ holds (for sequences of terms) if x contains the first term from y and no more than one other term.
- $Acc_4(x, y)$ holds (for sequences of terms) if x contains all terms from y and less than $|y|$ other terms.

We will consider four *meta–information acquisition procedures* Pr_1, Pr_2, Pr_3 and Pr_4. For instructiveness, we will assume that[1]

- Pr_1 is a Naive Bayes page categoriser operating on unigram representation.
- Pr_2 is an extractor of META tag content, equipped with heuristics mapping META attributes on target meta–information properties.
- Pr_3 is a linguistic, parser–based information extractor equipped with a database of domain–neutral lexical indicators.
- Pr_3 is an HTML–and–punctuation–based information extractor equipped with a database of domain–neutral lexical indicators.

The meta–information sets produced by the procedures will be ($\mathcal{M}(k)$ denoting the output of the $k-$th procedure, for the given resource):

$$\begin{aligned}\mathcal{M}(1) = \{ \quad & (p_1 = \textit{page\_type}, \text{'Main information page'}) \\ & (p_2 = \textit{page\_author}, \text{'N/A'}) \\ & (p_3 = \textit{name\_of\_company}\ldots, \text{'N/A'}) \\ & (p_4 = \textit{domain\_of\_competence}\ldots, \text{'N/A'}) \\ & (p_5 = \textit{pricelist\_location}, \text{'N/A'}) \\ & (p_6 = \textit{contact\_address\_location}, \text{'N/A'}) \ \}\end{aligned}$$

(the keyword–based categoriser is clearly designed for classification only, and does not care about structural patterns; assignment to 'Main information page' might be due to the presence of several 'promotion' keywords such as 'opportunity', 'lovers' or 'original')

$$\begin{aligned}\mathcal{M}(2) = \{ \quad & (p_1 = \textit{page\_type}, \text{'N/A'}) \\ & (p_2 = \textit{page\_author}, \text{"John Black"}) \\ & (p_3 = \textit{name\_of\_company}\ldots, \text{'N/A'}) \\ & (p_4 = \textit{domain\_of\_competence}\ldots, \\ & \text{"Production and sale of wooden and metallic toys"}) \\ & (p_5 = \textit{pricelist\_location}, \text{'N/A'}) \\ & (p_6 = \textit{contact\_address\_location}, \text{'N/A'}) \ \}\end{aligned}$$

(assuming that the meta–attribute 'description' is tentatively mapped on the property *domain_of_competence...*)

[1] These hypothetical procedures roughly correspond to some of the tools developed within the *Rainbow* project.

$\mathcal{M}(3) = \{$ (p_1 = *page_type*, 'N/A')
(p_2 = *page_author*, "J. Black")
(p_3 = *name_of_company...*, "BlackWood")
(p_4 = *domain_of_competence...*,
"toys for children aged over 6 years, collectioners' items")
(p_5 = *pricelist_location*, '`/html/body/ul`')
(p_6 = *contact_address_location*, 'N/A') }

(the value of *page_author* is identified with the subject that 'maintains' the object 'page'; the value of *name_of_company...* is identified with the subject that 'specialises' in *something*—which is, in turn, identified with the value of *domain_of_competence...*; finally, the *pricelist_location* is identified with the HTML element immediately following that with the phrase '...you can find')

$\mathcal{M}(4) = \{$ (p_1 = *page_type*, 'Main information page')
(p_2 = *page_author*, "Page maintained by J. Black")
(p_3 = *name_of_company...*, "BlackWood Ltd.")
(p_4 = *domain_of_competence...*, 'N/A')
(p_5 = *pricelist_location*, '`/html/body/ul`')
(p_6 = *contact_address_location*, 'N/A') }

(the *page_type* is recognised by presence of free–text paragraphs as well as a list—a mixture presumably typical for 'Main information page'; the value of *page_author* is identified with the slanted text in the page footer; the value of *name_of_company...* is identified with the text in the topmost header, containing the pattern 'Ltd.'; finally, the *pricelist_location* is identified with the unordered HTML list containing estimated 'price–patterns' in each item).

Taking into account the acceptability relations for p_2, p_3 and p_4, the 'correctness scores' of the meta–information sets for the individual properties are as follows:

Property	Pr_1	Pr_2	Pr_3	Pr_4
page_type	1	0	0	1
page_author	0	1	1	0
name_of_company...	0	0	1	1
domain_of_competence...	0	0	0	0
pricelist...	0	0	1	1
contact...	0	0	0	0

From the table we can deduce that e.g.:

- $\mathcal{M}(4)$ is superior to $\mathcal{M}(1)$
- $\mathcal{M}(3)$ is superior to $\mathcal{M}(2)$

It is however clear that *complementarity* of procedures is more important than *superiority*. We can easily see that:

- no combination of procedures can correctly acquire all meta–information
- if we removed the 'inaccessible' $p4$ and 'inavailable' $p6$, the minimal combinations of procedures needed for correct acquisition would be $\{Pr_1, Pr_3\}$, $\{Pr_2, Pr_4\}$ and $\{Pr_3, Pr_4\}$.

Furthermore, it is more reasonable to remove the *closed–world assumption*, since ignorance should not be treated the same as error. The original table will then look as follows:

Property	Pr_1	Pr_2	Pr_3	Pr_4
page_type	1	?	?	1
page_author	?	1	1	0
name_of_company...	?	?	1	1
domain_of_competence...	?	0	0	?
pricelist...	?	?	1	1
contact...	?	?	?	?

We could also construct the table for *pairs* of procedures. Here, in the case where both procedures return a value, we can either demand, for a correct result, that *both* are correct, or just that *at least one* is correct.

With the first interpretation, the table looks as follows:

Property	Pr_1,Pr_2	Pr_1,Pr_3	Pr_1,Pr_4	Pr_2,Pr_3	Pr_2,Pr_4	Pr_3,Pr_4
page_type	1	1	1	?	1	1
page_author	1	1	0	1	0	0
name_of_company...	?	1	1	1	1	1
domain_of_competence...	0	0	?	0	0	0
pricelist...	?	1	1	1	1	1
contact...	?	?	?	?	?	?

We can see that, in the first (more natural?) interpretation, the combinations involving Pr_4 now become inferior, since its incorrect claim of knowing the value or *page_author* invalidates the correct suggestion of Pr_2 or Pr_3, respectively. The combination $\{Pr_1, Pr_3\}$ then appears as clear 'winner'.

With the second interpretation, the table looks as follows (changes in boldface):

Property	Pr_1,Pr_2	Pr_1,Pr_3	Pr_1,Pr_4	Pr_2,Pr_3	Pr_2,Pr_4	Pr_3,Pr_4
page_type	1	1	1	?	1	1
page_author	1	1	0	1	**1**	**1**
name_of_company...	?	1	1	1	1	1
domain_of_competence...	0	0	?	0	0	0
pricelist...	?	1	1	1	1	1
contact...	?	?	?	?	?	?

The three candidate combinations would then become 'equally correct' again.

5.3 Discussion of the Model

The present model strikes for certain balance between simplicity and coverage. However, due to stress on the former, many important aspects of real-world meta-information acquisition tasks remain uncovered. Among the most important limitations (and topics for future research) are probably the following:

- The model does not treat different methods of combining the results of multiple procedures for the same property (such as voting)
- No uncertainty is allowed in the results of the procedures.
- The properties are single-valued.
- The model is static, it does not anticipate possible change of property values over time.

It might also be interesting to compare the typology of properties in our model with the inventory of *web ontology languages* such as OWL[2]. In OWL, content properties would correspond to *datatype properties with embedded data type*, while closed–domain properties would correspond to (object/datatype) *properties with enumerated class / data type*, respectively, since enumeration is possible, in principle, for both types of properties in OWL. Unlike web ontology languages, we however treat *class membership* simply as (closed–domain) properties, to keep the model general enough. Aside the 'enumeration' case, the notion of object property has equivalent meaning in our model and in OWL. Mapping (of semantically relevant elements) from OWL to our model seems to be rather straightforward: a structured ontology could be flattened to our model by replacing pairs of chained properties with new, composed, properties. Having done such mapping, we could then e.g. 'tap' on existing tools producing meta-information in the form of RDF triples.

Future work will address some of the limitations discussed above. Since the model is currently merely descriptive, we examine some existing algebraic formalisms that could extend

[2] `http://www.w3.org/TR/owl-features/`

it with more rigorous semantics. We would also like to elaborate on the problem of transformation/mapping to/from ontology languages suggested; an adequate method could possibly be adapted from state-of-the-art research on ontology transformation and matching[3].

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

Part II

Towards Reuse of Web Analysis Knowledge Models

Chapter 6

Overview of Knowledge Modelling

The discipline of knowledge modelling, though not too frequently mentioned outside its own community, is inherently present in many nowadays popular notions such as semantic web or ontologies. Its history and main principles are subject of this short chapter, from the seminal works deeply anchored in artificial intelligence research, to the more widespread (and often more 'mundane') semantic web research.

6.1 From Symbol Level to Knowledge Level

In the 60s and first half of 70s, published research in knowledge-based systems (KBSs) was almost uniquely devoted to concrete (mostly, logic-based) techniques of representing knowledge in symbolic form and performing (formally sound) inference steps over this representation. Little attention was paid to the fact that every KBS can be viewed, similarly as a human, as aiming to achieve some higher-level *goal* and using its 'knowledge' in the more abstract sense of the word (that of 'being familiar' with e.g. a certain subject matter). A. Newell, in its pioneering work [59], suggested to view a KBS both

- at the *symbol level*, describing e.g. its inputs/outputs, internal memory and control structures
- at the *knowledge level*, describing its goals, actions taken to achieve these goals, and the knowledge about its 'environment' and about itself.

Clearly, a single knowledge-level description can correspond to many different symbol-level descriptions. The (conceptually) same knowledge can be represented e.g. via rules, frames, decision trees or first-order theories. The (conceptually) same control structure, e.g. of finding an entity matching with a description, can be implemented e.g. as breadth-first search, depth-first search, or some form of informed search, over a structure of candidate entities.

Among the multiple ideas introduced by Newell, it was the concept of generic tasks and of implementation-independent methods used to complete these tasks that attained most

interest in the following almost 20 years—under the name of *problem-solving methods* (PSM).

6.2 Problem Solving Methods

The first generic method of problem solving, from which many successors took inspiration, was the model of *heuristic classification* formulated by Clancey [20]. It represents an abstraction over the reasoning structure of numerous diagnostic expert systems from different domains. Its essence are three 'primitive' *inferences* called 'abstract', 'match' and 'specialise', whose inputs/outputs are denoted as *knowledge roles*: 'Observables', 'Variables', 'Solution Abstractions' and 'Solutions'. The knowledge roles are, in a concrete application, mapped on domain concepts. For example, a medical expert system for treatment recommendation might acquire patient lab tests and other findings as 'Observables', it would *abstract* more general notions such as 'obesity' or 'hypertension' from them, *match* these with general categories of drugs such as 'diuretics' or 'β-blockers', and, finally, *specialise* the drug groups to concrete substances, drug brands and dosage, according to the context, e.g. availability on market and coverage by insurance.

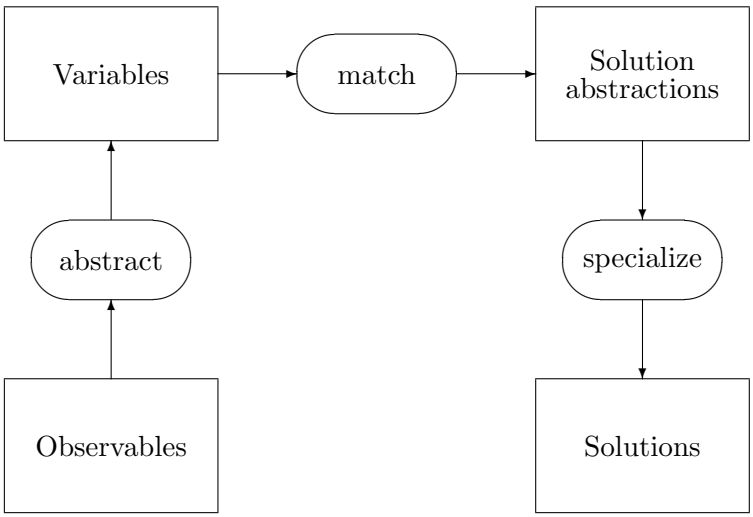

Figure 6.1: Inferences and knowledge roles in heuristic classification model

The CommonKADS methodology [69], fully developed in mid 90s, formulated complex guidelines for the design and usage of PSMs in the context of knowledge-based system development. The knowledge-level description of a KBS is viewed as consisting of three interconnected layers:

1. The *domain layer* describes the relevant domain concepts and relations independent of their use for reasoning.
2. The *inference layer* specifies the flow of inferences and data but not the control flow. It is typically expressed using inference diagrams such as that of heuristic classification

shown at 6.1. Knowledge roles in the diagram are mapped to concepts from the domain layer.

3. The *task layer* specifies the decomposition of tasks to subtasks and the algorithmic control structure. The lowest level of tasks in the decomposition tree corresponds to the inferences from the previous layer.

While the domain layer has to be assembled individually for every application, with respect to the domain[1], the remaining two layers are to large degree domain-independent. Several libraries of generic tasks with abstract descriptions of methods for their solving (PSMs), subsequently arose. They contain skeletons of inference and task layers, typically divided into two large groups: for *analytic* tasks, such as classification, diagnosis, assessment or monitoring, and for *synthetic* tasks, such as configuration, planning or scheduling.

6.3 Ontological Engineering

Compared to the situation ten years ago, the familiarity with the notion of ontological engineering has widespread in the computer science community. As it is not the main topic of the thesis, we only explain it in the extent necessary for understanding other parts of the text, and with numerous simplifications. Interested reader can get an overview of ontological engineering proper from [31] and a broader overview of applied ontology topics from [72].

The notion of ontology appeared long ago in philosophy, where it denoted the discipline dealing with the nature of and types of being, possibly also a general theory of being. In computer science, ontologies (in plural) are formal models of reality, describing concepts, individuals and relations that 'exist' in a certain domain. They differ from other types of conceptual models (such as ER diagrams and thesauri) by the stress on clean logical semantics, which allows formal reasoning over them.

As an example of ontological definition, let us show (in Table 6.1) one from the Enterprise ontology [94], which formally represents the notion of 'sale offer' in the language called Ontolingua [32], the most prominent ontology language in the 90s. The class `Sale-Offer` is defined by means of a necessary and sufficient condition (`Iff-Def`), which is a conjunction of simple subclass relationship wrt. another class (`For-Sale`) and existence of an object (`?Le`, for 'legal entity') that is connected with the `Sale-Offer` instance by means of the `Specified-Potential-Customer` relation.

6.4 Semantic Web Ontologies and PSMs

Although the principles of ontological engineering are entirely independent of the semantic web environment, it is obvious that ontologies owe most of their current popularity to that of semantic web. While the use of languages such as Ontolingua was limited to a few tens

[1] Domain ontologies, as mentioned below, would serve as starting point.

Table 6.1: Concept definition in Enterprise ontology

```
(Define-Class Sale-Offer (?X)
"A For-Sale situation with a Specified-Potential-Customer"
:Iff-Def
 (And
  (For-Sale ?X)
  (Exists (?Le) (Specified-Potential-Customer ?X ?Le))))
```

of specialised (artificial intelligence) research groups, ontologies in the current semantic web language OWL[2] (or in simpler RDF Schema) are developed by thousands of researchers and practitioners, often without AI background. For the design of ontologies to become a widespread skill rather than a difficult art, many supporting methods and tools are required; see [78] for an overview. Let us first mention ontology *editors* such as Protégé[3] (with its OWL Plug-In component) or WebODE [7], which shield the user from the unfriendly syntax of the web ontology language[4] and allow to navigate along the conceptual paths among concepts, individuals and relations (called 'properties'). Furthermore, there are systematic *methodologies* such as METHONTOLOGY [31], and *upper-level ontologies* that can be reused as 'root' for more specific, application-oriented ones. Finally, there has recently been growing interest in detailed *design patterns* describing ontology constructs and their use cases, either in the sense of *logical structures* (such as 'using classes as values for properties' [60] or 'converting n-ary relations to binary ones' [61]) or *content structures* (such as 'participation of an agent in an event' [29]).

The role of PSMs on the semantic web has been much less investigated; potentially, their role could be that of templates for complex reasoning services for semantic web applications. A pioneering work has been done in the *UPML* project [26]; the most recent research is best represented by the work by ten Teije et al. [90], described in more detail in section 8.4.1.

[2] `http://www.w3.org/TR/owl-features/`

[3] `http://protege.stanford.edu`

[4] Its formal expressions are melted downto RDF triples, which are themselves serialised in XML.

Chapter 7

Framework and Ontologies for Web Space Analysis

7.1 TODD: a Conceptual Framework for Web Analysis

In section 1.1, *deductive web mining* (DWM) was defined as 'all activities where pre-existing patterns are matched with web data'; the patterns may be either hand-crafted or learnt. We proposed a framework that positions any DWM tool or service within a space with four dimensions:

1. Abstract *task* accomplished by the tool. So far, we managed to characterise any concrete DWM task as instance of either:

   - *Classification* of a web object into one or more pre-defined classes.
   - *Retrieval* of one or more web objects.
   - *Extraction* of desired information content from (within) a web object.

   The *Classification* of an object takes as input its identifier and the list of classes under consideration. It returns one or more classes.

   The *Retrieval* of desired objects takes as input the (syntactic) *type* of object and *constraints* expressing its class membership as well as (part–of and adjacency) relations to other objects[1]. It outputs the *identifiers* (addresses based on URIs, XPath expressions and the like) of relevant objects[2].

[1] For example: "Retrieve (the XPath addresses of) those HTML tables from the given website that are immediately preceded with a possible 'Product Table Introduction Phrase' (containing e.g. the expression `product*`)".

[2] In the description of this as well as other tasks, we omit auxiliary information on output, such as numerical measures of relevance or uncertainty. These are also typically output by DWM applications, including those developed in *Rainbow*.

The *Extraction* task takes as input the class of information to be extracted and the scope (i.e., an object) within which the extraction should take place[3]. It outputs some (possibly structured, and most often textual) *content*. In contrast to Retrieval, it does not provide the information about precise location from where the content was extracted[4].

2. Type of *object* to be classified or retrieved, or from which information is to be extracted. The types, such as Document, Hyperlink, or Phrase, represent an upper-level of abstraction of web objects; any class of web object considered in a DWM application should be subclass of such type. This is facilitated by the fact that types correspond to classes of our Upper Web Ontology (see section 7.2). The basic assumption is that the type of object is always known, i.e. its assignment is not by itself subject of DWM.

3. *Data type*[5] *and/or representation*[6], which can be e.g. full HTML code, plain text (without tags), HTML parse tree (with/without textual content), hyperlink topology (as directed graph), frequencies of various sub-objects or of their sequences (n-grams), image bitmaps or even URL addresses.

4. *Domain* to which the task is specific.

We thus denote the framework as '*task-object-data(type)-domain*' (TODD). Its dimensions are to high degree independent, e.g. *object type* is only partially correlated with *data type*. For example, a document may be classified based on its HTML code, URL, META tag content or position in topology. Similarly, a hyperlink can be classified based on its target URL or the HTML code of source document (e.g. the menu structure containing the respective `<a>` tag). Clearly, not all points of the 4-dimensional space are meaningful. For instance, a META tag content cannot directly be used to classify a hyperlink, since the relation of a META tag (being a special class of HTML document fragment) to a hyperlink is intermediated by a whole HTML document.

7.2 The Collection of *Rainbow* Ontologies

7.2.1 Basic Principles and Layers

In the construction of *Rainbow* applications, ontologies as shared formal conceptualization of a domain will enable *consistency checking of offered services* and provide support for the *integration of multiple types of analyses*. The initial version of *Rainbow ontologies* was built

[3]For example: "Extract the occurrences of Company Name within the scope of given Company Website".

[4]This is of course merely a knowledge-level view, which does not discourage relevant DWM applications from remembering such information for technical purposes.

[5]Readers familiar with semantic web terminology should avoid confusion of the object type / data type dimension of TODD with the distinction of object / datatype properties in ontology languages such as OWL, see section 6.4. There is no such analogy.

[6]We alternatively call this dimension 'web view', since the particular representation of data corresponds to a certain view of the complex structure of the web.

in 2003 (by M. Labský in cooperation with V. Svátek)[7], as a proof of concept rather than as a fully integrated component usable in on-line web analysis. As a matter of fact, the collection of *Rainbow* analysis tools has not yet grown to a size where ontological support for their integration becomes necessity. Subsequent re-engineering and use of the ontologies has thus been postponed to later phases. The main research contribution of the ontological engineering sub-project of *Rainbow* was thus the innovative adaptation of an ontology *merging* method, described in the section 7.2.2. Parts of the ontology were also rewritten in the form of Prolog clauses and used in the simulation of automated tool composition, which is described in section 8.5.1. Let us now describe the initial version of ontology collection.

In general, there are two kinds of concepts in *Rainbow* ontologies — syntactic *types* and semantic *classes*. Types currently considered are e.g. *Document*, *DocumentFragment*, *HyperLink* or *Phrase*. Among classes, there are e.g. *HProductCatalogue*, *TLeafDocument*, *HProductDescription* or *LSentence*. As outlined above, classes[8] differ from types in the sense that their identification is subject of analysis, while the identification of types is assumed to be known in advance (say, no *Rainbow* tool should be developed in order to distinguish a physical page from a collection of pages). Every class is subconcept of some general type.

In addition to concepts, there are also relations. Among the most widely used relations in *Rainbow* ontologies is the transitive *part–of* relation, e.g. *HProductDescription* may be *part–of* a *HProductCatalogue*. Concepts can also be *adjacent* to each other, they may be *identified–by* some other concepts etc. Inverse relations are defined where possible.

The three dimensions of the TODD model (i.e. apart from the task dimension), namely the distinction of *data types*, *object types* and application *domains* suggest a natural decomposition of the system of ontologies into four layers as depicted in Fig. 7.1. The upmost layer contains the *object types* themselves. Furthermore, the upper two layers are domain–independent and therefore reusable by applications from all *domains*, while the lower two layers add information specific to the domain of analysis, e.g. OOPS ('organisations offering products and services') sites or web pornography. Finally, the outer two layers contain concepts that are independent of the *data type* used for their recognition, while the inner two contain data-type-dependent concepts. Let us now characterise each of the four layers, in turn.

Upper Web Ontology.

The abstract *Upper Web Ontology* (UWO) provides a hierarchy of common Web–related concepts and relations that are shared by all analysis types and application domains. The UWO doesn't attempt to define an exhaustive description of the WWW. Instead, it provides a shared conceptual language for all *Rainbow* modules to build upon. The UWO only defines concepts that correspond to syntactic types as defined above, and only the most generic ones, which are likely to be frequently reused across individual analysis tools. Its UML diagram is

[7] In the DAML+OIL language (predecessor of OWL, see `http://www.daml.org/2001/03/daml+oil-index.html`); the whole collection is available from `http://rainbow.vse.cz`.

[8] Note that both types and classes in the specific *Rainbow* terminology were syntactically modelled as classes in DAML+OIL.

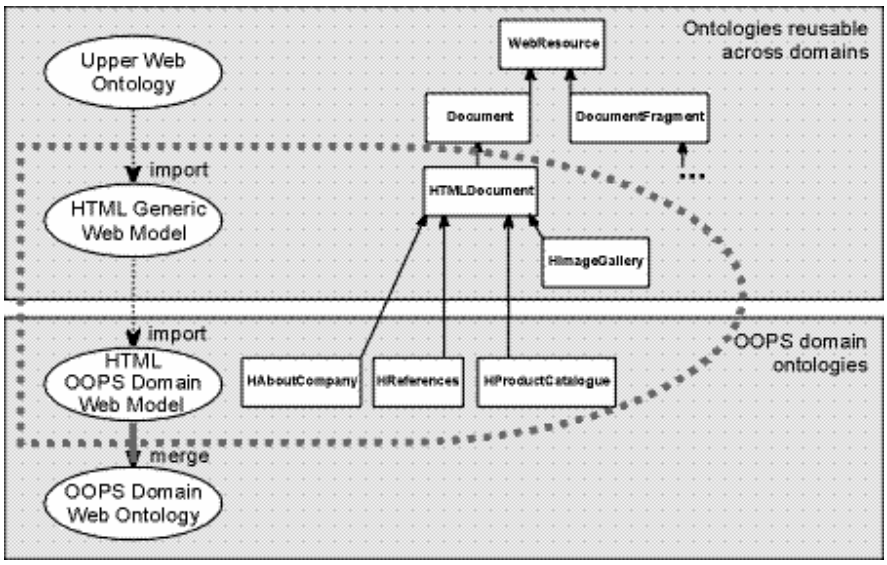

Figure 7.1: Structure of the *Rainbow* ontology system shown on HTML analysis example

at Fig. 7.2.

Partial Generic and Domain(-Specific) Web Models

For each way of analysis, *Partial Web Models* occupy the middle layers of the *Rainbow* ontology system. Concepts and relations defined in these models represent the *classes* and *types* specific to one data type. The partial web models consist of a *generic* and *domain–dependent* part. Elements introduced in the generic model are based on the UWO and are reusable across different application domains. On the other hand, for each data type there might be domain models specializing in different application domains. All of these domain models are then based on a single generic model and the common UWO. Concepts from the generic and domain models mostly correspond to *classes* of resources, but new *types* may be defined as well. In Fig. 7.1, the generic model and OOPS domain model for HTML analysis are depicted within the dashed area. Examples of concepts from these models and the UWO are shown on the right. Class names are prefixed with corresponding data types: 'H' for HTML structure, 'T' for topology etc.

Domain(-Specific) Web Ontologies

The central information contained in partial web models (both generic and domain) are the concept hierarchies of semantic *classes*. Each such hierarchy extends one of the classified syntactic types, such as *Document* or *DocumentFragment*, and presents a data–type restricted view of that type. In our approach, the integration of multiple analyses consists in *merging*

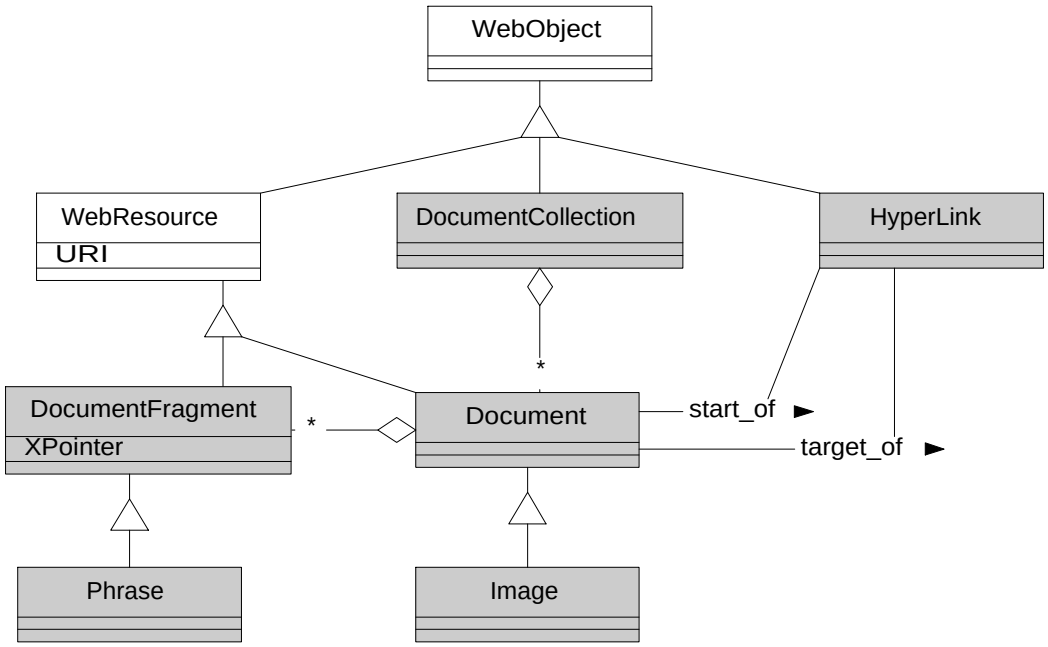

Figure 7.2: UML diagram of Upper Web Ontology

these class hierarchies[9]. The resulting class hierarchy is no more restricted to a single data–type view and is included in a *Domain Web Ontology* (DWO), as depicted in Fig. 7.1.

The DWO also imports all concepts from the UWO and partial models. Thus, the DWO is a *domain-specific ontology of web resources*, and it *integrates all data–type specific views* of the used analyses.

7.2.2 Merging the Partial Ontologies

We propose to derive the structure and content of the DWO based on labelled data, which may be acquired e.g. from training data previously used to train the individual modules. This labelled data comprises training web pages that are simultaneously labelled with classes used by different types of analyses (based on different data types).

Input Data to the Merge Process.

As mentioned earlier, *Rainbow* currently addresses semantic classes that are subclassed from the syntactic types such as *Document*, *DocumentFragment*, *DocumentCollection* or *HyperLink*. These four types of objects may have ontology–defined relations such as *part–of*; using these relations, we can — in a limited way — relate the concepts to each other and use this

[9]These is–a hierarchies will often be flat, since the analysis–specific *classes* are usually not richly structured before integration.

information as input to the merging process.

Before we start to merge concepts, we have to choose one *central* syntactic type to which all other (currently three) syntactic types will be related. For example, if the merge process is done for the Document as the central type, we regard the labeled input data as a set of training *Documents*. The input attributes used by the merge process are then of the following forms:[10]

- *is* <*Document* class>,
- *contains* <*DocumentFragment* class>,
- *part–of* <*DocumentCollection* class>
- (*source–of* or *target–of*) <*HyperLink* class>.

For other syntactic types, the same relations (or their inverses) may be used to derive equivalent attribute sets. For example, if DocumentFragment was chosen as the central type, we would regard the labeled data as a set of *DocumentFragments* and use attributes such as *part–of* <*Document* class>).

In further text, we will only consider *Document* as the central syntactic type. The labeled input data is therefore treated as a set of documents, which we will denote as D. Each document $d_i \in D$ has attributes given by the analysis–specific classes it is labeled with. Each of these attributes has one of the above four forms.

The Merge Process.

The task is essentially that of merging parts of multiple ontologies based on concept extensions. For its accomplishment we use an adapted FCA–Merge method introduced in [77], based on formal concept analysis (FCA). FCA–Merge is a semi–automatic method which originally integrates ontologies based on a set of natural language documents, in which references to ontology concepts are found using NLP techniques. In our approach, we use a set of training documents, where labeled instances of to–be–merged concepts already exist. Detailed description of our adaptation of FCA-Merge (carried out by M. Labský) can be found in [44].

The result of the merging process is a hierarchy of new concepts, each of which may be (1) an exact copy of its original analysis–specific concept, or (2) a concept that has been created by combining more of the original concepts together. FCA–Merge assists the human integrator and makes suggestions, but the ultimate modeling decisions are left to the integrator. Concepts can typically be combined when they have the same *extension* in the training data, resulting in them being replaced by a new concept in the role of their conjunction. If two or more of the original concepts have a significant overlap, a new concept, representing their conjunctions, may be added to the new ontology as well as the original ones. This takes place

[10] Relations *part–of*, *source–of*, *target–of* and defined in the UWO.

especially when there exist concepts in the source ontologies that are more specific than the potential new concept, and/or when the new concept has high support[11] in training data. The hierarchy of the new ontology is derived directly from the concept lattice and can be altered by the integrator. Typically, the resulting class hierarchy is deeper (as it is drawn from data) than the usually flat hierarchy of input classes.

The new concept hierarchy reveals previously hidden subsumption relations across the different analysis types. These empirically induced subsumptions may further be checked with formal ontology definitions of the original concepts. This checking could be made automatic and a conflict would signal either inconsistently labeled training data or a wrong definition of some of the original concepts.

The final DWO only consists of the merged classes, while all other information is simply imported from the partial web models and the UWO.

Example

In the following we present a 'toy' example taken from the domain of websites of car dealers. In the example, integration is done based on a set of only thirteen manually labeled documents. Nevertheless, we will be able to demonstrate some of the benefits of this method. Due to lack of space, we will consider only one kind of semantic classes: Document. The other three kinds of classes could be taken into account as described above. We also limit the integration to two types of analyses: *HTML structure* and *topology*.

Table 7.1: HTML documents classified by the topology and HTML structure analysis

I	TLE	THU	TLO	TRE	HAB	HRE	HPR	HIM
d1	×				×			
d2		×	×		×			
d3	×				×			
d4		×		×		×		
d5		×		×		×		
d6		×	×				×	×
d7		×		×			×	×
d8		×	×				×	×
d9		×	×				×	×
d10		×		×			×	
d11		×	×				×	
d12		×		×			×	
d13							×	

The above table shows thirteen documents classified by data-type specific classes. Topology–specific classes are prefixed with 'T', while HTML–specific classes have 'H' as their first letter:

- *TLeaf* (TLE) is a document with no links to other documents,
- *THub* (THU) has more than 1 link to other documents,

[11] We compute support as the number of positive documents divided by the total number of training documents.

- *TLocalHub* (TLH) is a subconcept of THub, most of whose links lead to the same IP address as its own,
- *TRemoteHub* (TRH) is a subconcept of THub, most of whose links lead to a different IP address than its own,
- *HAboutCompany* (HAB) is a document describing a company in general,
- *HProductCatalogue* (HPR) contains descriptions of offered products,
- *HReferences* (HRE) refers to customers' pages,
- *HImageGallery* (HIM) contains a set of similarly–sized images.

From the labelled set of documents, we are able to construct a concept lattice depicted in Fig. 7.3. Formal concepts that are on the same level of specificity as the source class hierarchies are shown in dark color. On the other hand, formal concepts more specific than original classes are drawn in light color.

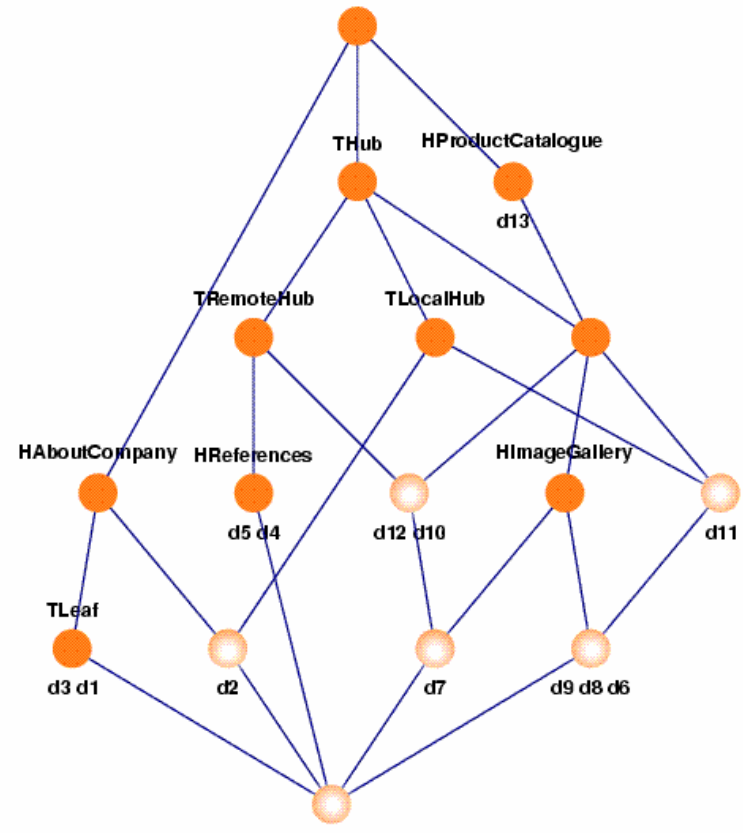

Figure 7.3: Pruned concept lattice derived for the 13 car dealer documents

We observe that no original concepts had the same extensions in the training data, as each formal concept directly corresponds to at most one ontology concept. Depending on the use

of the target ontology, all or most of the original classes could be included in it. In the concept lattice, we might reveal several interesting implications pertaining the analyzed domain (or, more precisely, the training data), such as $HImageGallery \Rightarrow HProductCatalogue$ or $HReferences \Rightarrow HRemoteHub$. It is up to the integrator what part of the induced hierarchy will be reflected in the target ontology.

We can also identify one new (no-name) concept which is a combination of *THub* and *HProductCatalogue* and whose specialization is *HImageGallery*. This concept might e.g. stand for product catalogues referring to child documents with detailed information about the sold products — let us denote it *ComplexProductCatalogue*[12] . Because of its interpretation and being within the specificity level of the original ontologies, *ComplexProductCatalogue* is an appropriate candidate for adding to the target ontology.

Last of all, formal concepts more specific than the original ontologies should be examined. *Frequently occurring combinations* of original ontology concepts may be included in the target ontology as well if they are considered useful by human integrators. For example, the formal concept with extension $d6, d8, d9$ and significant support of 0.23 may be considered interesting for certain applications and included in the target ontology as *GraphicalLocalCatalogue*.

Related Work

An analogous effort to our construction of *Rainbow* ontology was the *OntoWebber* project [37], in which a 'website ontology' was designed. It was however biased by its application on portal building (i.e. 'website synthesis'), and thus did not fully cover the needs of automated analysis.

Integration of ontologies for other domains has been addressed by numerous projects, see e.g. `http://www.ontologymatching.org` for links.

7.3 Ontologies in Web Information Extraction

While the previous section addressed the role of ontologies in the overall architecture of *Rainbow*, as instrument for integration of different analysis tools, in this section we discuss the role of ontologies in connection with a specific (and actually, most important) analysis task, that of information extraction from the content of websites.

In a general overview of ontology types, van Heijst [96] distinguishes among terminological, information and knowledge ontologies. *Terminological ontologies* are centered around human-language terms, without direct reference to real world. Their main constructs are synonym sets and (hyponymy/meronymy) hierarchies. *Information ontologies* and *knowledge ontologies* both deal with classes directly mapped to sets of entities (instances) in some universe of discourse. Knowledge ontologies however differ from information ontologies by presence

[12]The new concept is not specific to any type of analysis.

of formal axioms, most particularly, by the possibility to define the extent of a class via a logical expression over its properties (relations to other classes).

The range of models possibly appearing in different phases of web information extraction (WIE), as specific type of potential ontology application, seems to be analogous to the general categorisation. Stevenson & Ciravegna [75] already raised the issue of ontologies 'for customer service' that do not satisfy the needs of information extraction components, namely, they point out the contrast between *domain ontologies* suitable for reasoning over real-world objects (in the 'customer' application) and *linguistic ontologies* applicable on (presumably, continuous) text. This contrast however becomes less sharp when considering semi-structured web content in the form of lists, tables or forms. Ontologies directly usable for analysis of web structures are likely to borrow a lot from 'customer-service' ontologies, since the fragments of HTML code will often directly map on ontology classes, attributes/relations and instances. We will call them *presentation ontologies*, since their universe of discourse is that of web objects as *presented* on the web (e.g. bicycle offers encoded in HTML) rather than of real-world objects (real bicycles). Finally, at the level of plain text strings, *terminological ontologies* may come into play.

Let us illustrate this simple typology of WIE (uses of) ontologies on our experiments in the bicycle domain (described in section 3.2) and on related projects.

7.3.1 Ontologies in *Rainbow* Product Catalogue Application

Let us briefly recall the three uses of ontologies in connection with *Rainbow* information extraction tools, two of which were already presented in chapter 3.2.

- **Populating the Domain Ontology.** The ontology to be populated was expressed in RDF Schema, see section 4.3 for more detail. Fig. 7.4 (already shown in section 4.3) shows most of the ontology: it covers information on the product offer itself (as presented on the web), characteristics of the product, as well as those of the selling company. Consistently with the observation made in [75], this ontology links together pieces of information occurring nearby each other at a product catalogue page, as well as those located quite separately or even not directly present on the website and thus unlikely to be picked up by means of a single WIE procedure. Indeed, different parts of the ontology are assumed to be populated by different analysis methods within *Rainbow*.
- **Template-Filling with Presentation Ontology.** We assume that presentation ontologies will most likely be restricted to a smaller portion of the original domain, cut up according to web presentation factors. Our simple ontology already mentioned in section 3.2 and shown on Fig 7.5 is specific to product catalogues[13], and only contains one 'true class', that of Bike Offer; the remaining concepts are shrunk to its properties[14]. Note (in contrast to the domain ontology above) the direct link between product

[13] Similar presentation ontologies could be designed e.g. for company profile (pages) or contact info (pages); the former would presumably be more linguistic-oriented as the profile information is typically expressed by free text, cf. [38].

[14] It could thus be viewed as *information ontology* (i.e. a data schema) in the typology from [96].

offer information and information about bike parts. Although not 'deeply' ontologically related, they fit together in terms of presentation: the company hopes to sell the offered bike thanks to pointing out its equipment. The domain of product offers is simple enough (in terms of logical structure of presentation) to allow to keep only one class and to dissolve the remaining ones into properties. This assumption would certainly not hold for all domains where WIE might be applied; the presentation ontology then would have multiple 'class vertices', and the template-filling algorithm (see below) would be more sophisticated.

While the domain ontology was destinated for direct retrieval of structured information, our presentation ontology is tuned for 'template filling' by means of a simple sequential algorithm (assigning properties to the 'current' object as long as constraints are satisfied). The expressive power of the ad hoc 'ontology language' used is thus kept limited. The central features are the *uniqueness*, *multiplicity* and *optionality* of properties, the latter two indicated with the `*` and `?` symbols, respectively. In addition, *'sticky' properties* are distinguished: as soon as the value of sticky property is discovered on a page, it is filled to all objects extracted afterwards, until a new value is discovered for this property.

- **Lexical Taxonomy for Primary Annotation.** In our project we have not used a lexical taxonomy (even not a look-up gazetteer) in the primary annotation of bike names, prices, component names and the like; the annotation was entirely carried out by means of statistical models. However, a collection of more than 60 bicycle *categories* (in various sense) arose as side product of annotation, and was later arranged into a hierarchy (see part of it at Fig. 7.6). We could easily imagine adoption of a similar taxonomy, e.g. a domain-specific part of product taxonomy such as bicycle-specific part of UNSPSC[15], for automated annotation with possibility of conceptual abstraction upto an arbitrary level of taxonomy.

7.3.2 Ontologies in Other WIE Projects

The authors are not aware of a similar study related to WIE, offering a synoptic view of multiple ontology types and uses; our own experiments are presented as simple instantiations of more general ideas rather as for their own sake. Let us now position some advanced ontology-endowed WIE projects in the above-described framework. Embley [22, 23] uses the notion of 'extraction ontology' for conceptual schema with data frames hand-crafted by domain expert (i.e. presentation ontology). While [23] focuses on *HTML table* analysis (for offers of products, namely, cars), [22] deals with *free text*[16] (obituaries); the nature of 'extraction ontology' however remains the same. In the Armadillo [19] project, an (inductively learnt) presentation ontology allows to reuse a surface-logical structure from one resource to another, e.g. accross multiple bibliography resources from the same domain, containing data about overlapping sets of publications. A sort of presentation ontology is also used in the OntoBuilder project [28] aiming at 'deep web' information extraction. It defines layout rules

[15] `http://www.unspsc.org`

[16] But applies surface term-distance heuristics rather than sentence parsing.

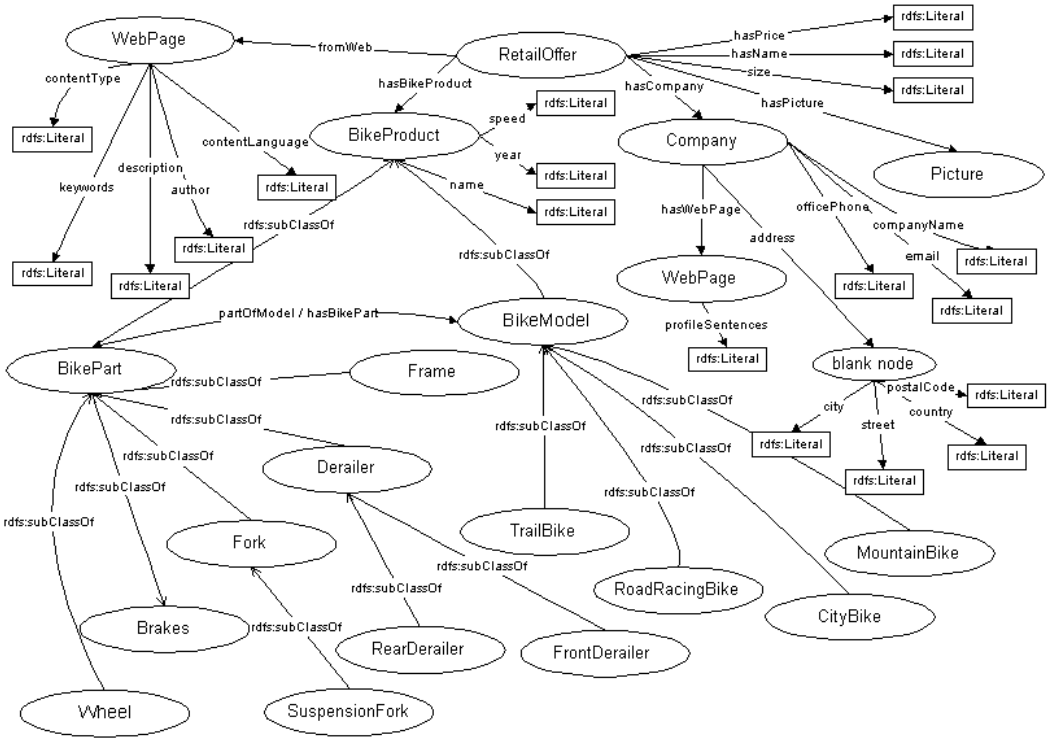

Figure 7.4: Part of RDF schema for the bicycle domain

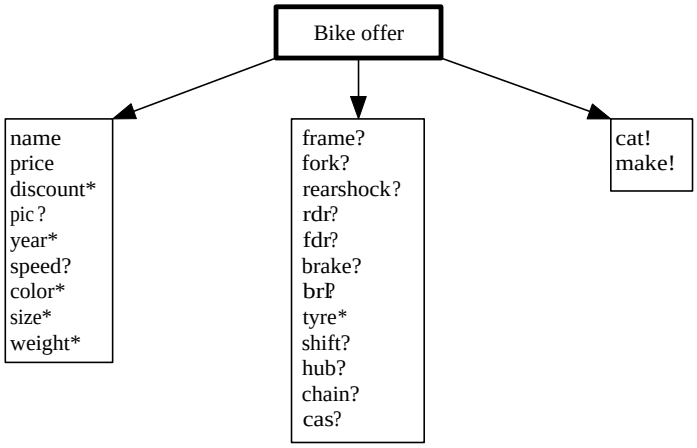

Figure 7.5: Bicycle offer presentation ontology

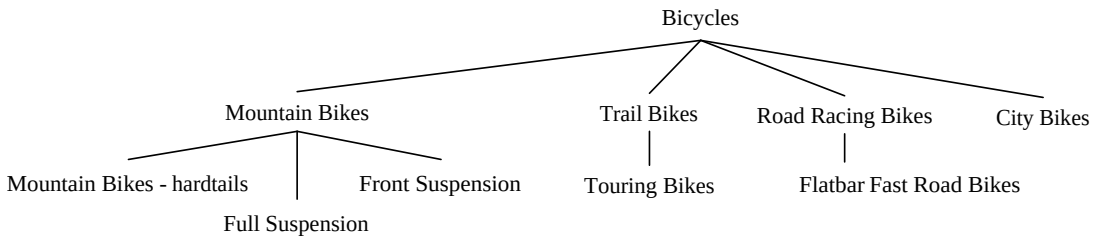

Figure 7.6: Fragment of empirical taxonomy for bicycle 'categories'

for HTML forms used as input to online databases. On the other hand, the Crossmarc project [63] is limited to terminological level (term sets mapped on semantic classes) in its usage of ontologies[17]. In the AeroDAML approach [41], a terminological ontology (WordNet) is used for annotation and a knowledge ontology (expressed in DAML) is populated by extraction results. Since the extraction method is named-entity recognition rather than structural IE, consistency-constraints are only applied at the level of target domain ontology rather than within a dedicated presentation ontology. Similarly, Maedche et al. [50] used an ontology engine (OntoBroker) to verify 'conceptual bridges' between terms extracted via shallow syntactic analysis. Ontologies are of course important in many IE projects that do not explicitly target the WWW but handle text structures similar to those of HTML elements; we do not cover them here.

7.4 Ontologies and Web Directories

This section follows up with section 3.1, in which we described the method of learning lexical indicators as simple information extraction patterns, with the help of a public web directory. In this section, we suggest to extend this approach to a bootstrapping cycle of information extraction and ontology learning[18]. A similar combination of information extraction and ontology learning has previously been described by Maedche [50]. The main novelty of our approach is however in the use of a public *web directory*.

7.4.1 Ontological Analysis of Web Directories

Web directory hierarchies are sometimes mistaken for ontologies; however, as already observed by Uschold [93], they are rarely valid taxonomies. It is easy to see that subheadings are often not specializations of headings; some of them are even not *concepts* (names of entities) but *properties* that implicitly restrict the extension of a preceding concept in the hierarchy. Consider for example `.../Industries/Construction_and_Maintenance/Materials_and_Supplies/Masonry_and Stone/Natural_Stone/International_Sources/Mexico`.

[17] Admittedly, its main focus is multi-linguality rather than HTML-centred WIE.
[18] An overview of ontology learning as such can be obtained from [14, 49].

Table 7.2: Examples of interpretation rules

Rule no.	Path pattern	Ontology relation
1	Subj/Prop	'Prop_Subj' *is-a* Subj (or, Prop *restricts* Subj *to* 'Prop_Subj')
2	Dom1/Dom2	Dom2 *is-part-of* Dom1
3	Obj1/Obj2	Obj2 *is-a* Obj1
4	Dom/Prop	'Prop_Dom' *is-part-of* Dom

Rule no.	Example
1	`Publishers/Academic_and_Technical`
2	`Security/National_Security`
3	`Electric_Motors/AC_Motors`
4	`Manufacturing/Electrical`

Semantic interpretation of a representative sample of directory paths revealed that

- terms and phrases in individual headings belong to quite a small set of *classes*, and
- surface 'parent-child' arrangement of headings belonging to particular classes corresponds (with a certain degree of ambiguity) to 'deep' ontological *relations*.

The result of this effort was a *meta-ontology of directory headings* plus a collection of *interpretation rules*. The diagram at Fig. 7.7 depicts the essence[19] of the *meta-ontology*. Boxes correspond to classes, full edges to named relations, and dashed edges to the class-subclass relationship. Reflexive binary relations are listed inside the respective boxes. Examples of informally expressed *interpretation rules* are in Tab. 7.2.

7.4.2 Coupling Information Extraction and Ontology Learning

Plain indicator terms, gathered by means of the fully automated technique described in section 3.1, are by themselves powerful enough to extract *sentences* that are likely to contain *some kind of* interesting information about the company. We can even, in many cases, access this information thanks to simple heuristics over the parse-tree, such as:

> If the immediate object of the *indicator verb* is a generic *set-semantic expression* such as 'range of', 'family of', 'assortment of' etc. then output the *indirect attribute* of the object; otherwise output the *object* itself.

Universal extraction patterns however impose strong assumptions on the whole collection of indicators. A more sensitive method should take account of the *classes* of indicators/headings

[19] For better readability, we have e.g. omitted the notion of 'Location', which may also be important to extract but is not directly related to the commercial profile of the company.

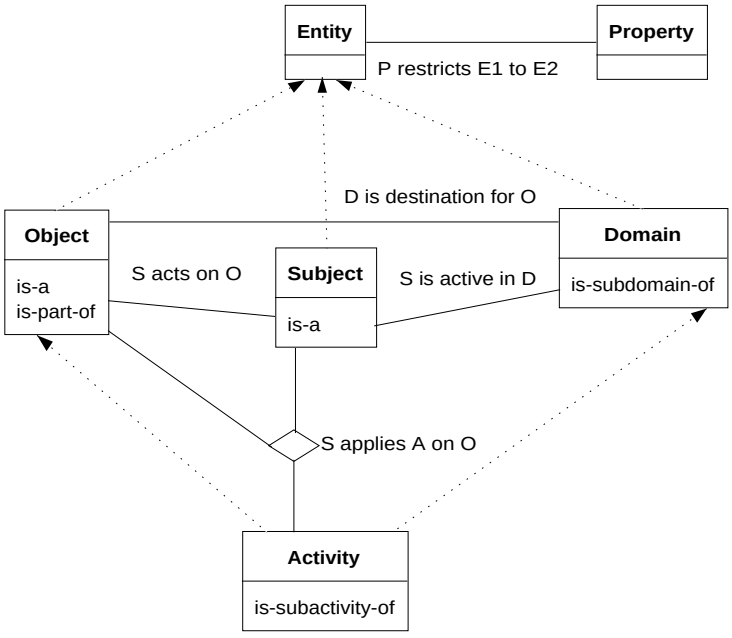

Figure 7.7: The ontology of web directory headings

revealed by ontological analysis. If we learn the indicators for each class of information (such as 'subjects', 'objects' or 'domains') separately, we could be able to perform true *information extraction* in the sense of filling database templates. Conversely, if the informative terms thus discovered coincide with the headings of directory nodes referencing the particular page, we can automatically 'restore the identity' of these headings. With the help of generic interpretation rules such as those shown in Tab. 7.2, fragments of true taxonomies (possibly several interconnected ones, for 'subjects', 'objects'..., as specified by the meta-ontology) could be built. We can understand this as a two-step *ontology learning* process using two resources: text and the hierarchies of headings. Obviously, the result of this process will still be rather incomplete, and should be enhanced using other ontology-learning techniques, taking into account co-occurrences (and linguistic dependencies) of terms in the text beyond the headings.

These two tasks represent a *closed loop*: as soon as we have classified the headings, we can learn class-specific indicators[20]. From the other side: as soon as we have class specific indicators, we can use them for the classification of headings. Since the first step in this loop has to be done by a human, a more viable approach seems to be that starting by *classifying the directory headings*. One more reason for this are some regularities and similarities in the structure of Open Directory: some of the headings could thus be even classified semi-automatically with the help of heuristic rules. Another interesting possibility is to classify the headings by matching them to a generic lexical ontology such as *WordNet*.

[20]The class-specific indicators will apparently be more complex than the current ones.

Chapter 8

Problem Solving Methods of Web Space Analysis

The idea of using Problem Solving Methods as models of web analysis applications is considered as one of major contributions of this thesis. We first briefly review the state of the art in PSM-style modelling of 'data-intensive' tasks, then describe a newly developed library of PSMs for web space analysis, demonstrate their usability to describe existing applications, suggest an approach to automated composition of tools relying on PSM-based templates, and, finally present a simulated experiment of such composition.

8.1 State of the Art

PSMs are mostly understood as specific for knowledge-intensive but 'data-temperate' tasks, which are common in Artificial Intelligence realms. The question whether *data-intensive tasks* could benefit from the introduction of PSMs has however come to the mind of several researchers.

In the *IBrow* project [1], operational PSM libraries have been developed for two areas of document search/analysis: Anjewierden [4] concentrated on *analysis of standalone documents* in terms of low-level formal and logical structure, and Abasolo et al. [2] dealt with information search in multiple external resources. Direct mining of websites was however not addressed; IBrow libraries thus do not cope with the problem of web heterogeneity and unboundedness, which motivated the development of the TODD framework. In contrast, the Armadillo system [19] attempts to integrate many website analysis methods; it currently relies on workflows manually composed from scratch by the user, although a template-based solution is also being envisaged. Besides, PSM-based solution has also been developed for task configuration in Knowledge Discovery in Databases (KDD) [24].

8.2 Library of Deductive Web Mining PSMs

The textual descriptions of three core *tasks* introduced in chapter 7.1 only specify the *input* and *output* of these tasks, in a style analogous to CommonKADS [69]. A natural next step towards reusable knowledge models thus seems to be the identification of appropriate *problem-solving methods* (cf. Section 6.2) and their representation in the form of inference and task knowledge.

Let us now present a collection of eight PSMs for DWM. It is rather tentative, yet seems to cover a large part of realistic cases; examples will be given in section 8.3.

For *Classification*, we could consider three PSMs. *Look-up based Classification* amounts to picking the whole content of the given object (cf. the Overall Extraction PSM below), and comparing it with content constraints (such as look-up table), which yields the class; for example, a phrase is a Company Name if listed in business register. *Compact Classification* also corresponds to a single inference, it is however not based on simple content constraints but on some sort of computation (e.g. Bayesian classification), which is out of the scope of the knowledge modelling apparatus. Finally, *Structural Classification* corresponds to classification of an object based on the classes of related objects (sub–objects, super–objects and/or neighbours). It is thus decomposed to *retrieval* of related objects, their *individual classification*, and, finally, evaluation of *global classification patterns* for the current object. It is therefore *recursive*[1]: its 'inference structure' typically contains full-fledged (Direct) Retrieval and Classification tasks.

For *Extraction*, there will be again three PSMs, rather analogous to those of Classification. *Overall Extraction* amounts to picking the whole content of the given object. *Compact Extraction* corresponds to a single inference based on possibly complex computation, which directly returns the content of specific sub-object/s of the given 'scope' object. Finally, *Structural Extraction* corresponds to extraction of information from an object via focusing on its certain sub-objects. Such objects have first to be *retrieved*, then lower-grained *extraction* takes place, and, finally, multiple content items possibly have to be *integrated*. Structural Extraction is thus equally recursive as Structural Classification.

Finally, let us first introduce two PSMs for the *Retrieval* task. The upper inference structure[2] at Fig. 8.1 corresponds to Direct Retrieval and the lower one to Index-Based Retrieval, respectively. The names of inferences (in ovals) are mostly borrowed from the CommonKADS library [69], while the knowledge roles are more DWM-specific. In *Direct Retrieval*, potentially relevant objects are first retrieved based on structural (parthood and adjacency) constraints, and then classified. Objects whose classes satisfy the class constraints are the output of the method. In the absence of class constraints, the method reduces to the 'specify' inference. In *Index-based Retrieval*, the (abstract) class constraints are first operationalised so that they can be directly matched with the content of objects. Then the objects are retrieved

[1] The notion of recursion previously appeared in knowledge modelling literature, e.g. in the form of Systematic Refinement as form of classification [92]. Here, however, the problem is more severe, since a recursively processed data structure appears in a dynamic rather than static role (in the CommonKADS sense).

[2] We did not show inference structures for Classification and Extraction, due to limited space as well as due to incompatibility of their structural variants with the CommonKADS notation, see below.

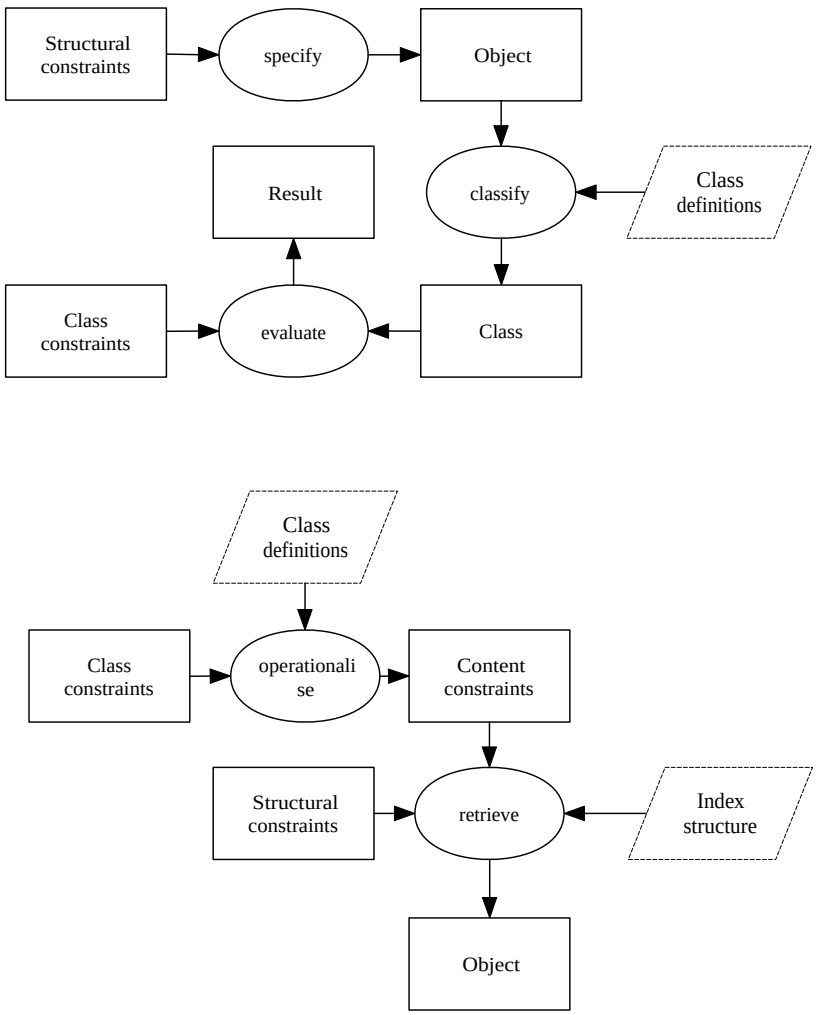

Figure 8.1: Inference structures of Retrieval PSMs

in an index structure (which is considered as separate from the web space itself), possibly considering structural constraints (provided structural information is stored aside the core index).

An interesting issue related to the representation of above PSMs is the possible interaction of different 'time horizons' in one application; static roles may become dynamic when changing the time scale. For example, a typical DWM application may first build an index of a part of the website (or learn class definitions from a labelled subset of objects), and then use the index to efficiently retrieve objects (or use the class definitions to classify further objects). This interaction deserves further study.

Traditional vs. DWM Classification

Among the three tasks, it is *Classification* that is most appropriate for comparison with existing PSM research. Classification problem solving was recently systematised by Motta&Lu [58]. Their taxonomy of classification problems is mainly derived from the presence (or absence) of a few key features:

1. *Whether the goal is to find one, all or the best solution.* This distinction can well be ported to the DWM context.
2. *Whether all observables are known at the beginning or are uncovered opportunistically (typically at some cost) during the problem solving process.* In DWM, the latter is typically the case (provided we interpret 'observables' as the web objects themselves); the cost is however only associated with download/analysis time, and its increase is smooth—unlike e.g. medical applications, where addition of a single examination may lead to abrupt increase of (financial or social) cost.
3. *Whether the solution space is structured according to a refinement hierarchy.* Presence of class hierarchy is quite typical in DWM; in the *Rainbow* project, it is reflected in concept taxonomies that constitute our ontology, see Section 7.1.
4. *Whether solutions can be composed together or each presents a different, self-contained alternative.* We believe that in DWM, elementary classification will mostly be carried out over disjoint classes, but can be superposed by multi-way classification with non-exclusive class taxonomies. We discuss this option below, in connection with the *refine* inference of Heuristic Classification.

Motta&Lu [58] also formulated a generic task-subtask decomposition *template*, which can be instantiated for different task settings:

1. First the observations have to be verified whether they are legal (*Check*).
2. All legal observations (⟨*feature,value*⟩-pairs) have to be scored on how they contribute to every possible solution in the solution space (*MicroMatch*).
3. Individual scores are then aggregated (*Aggregate*).
4. Candidate solutions are determined via aggregated scores (*Admissibility*).
5. Final solutions are selected among candidate solutions (*Selection*) .

Compared to this generic Classification template, our notion of DWM classification is slightly simplified and more goal-driven. Some parts of Structural Classification PSM can be mapped on the generic template: classification from lower level of recursion is similar to MicroMatch, while evaluation of global pattern unites the Aggregage, Admissibility and Selection steps. There is no Check step (since no observations are known a priori), but an extra step of Retrieval (since objects relevant for classification of current object have first to be determined).

We can also compare Structural Classification with the well-known Clancey's *Heuristic Classification* (HC) [20], consisting of *abstract*, *match* and *refine* inferences; note that HC was also chosen as the default PSM for classification by Motta&Lu [58], who defined all other models as its reductions. In (DWM) Structural Classification, the *abstract* inference is replaced with *classify* inferences applied on related (contained and/or adjacent) objects; this is due to the 'object-relation-object' (rather than 'object-feature-value') character of web data representation. The *match* inference from HC corresponds to 'evaluation of global classification patterns'. Finally, a *refinement* from general to case-specific solution might rather have the form of classification according to *multiple hierarchies* in DWM. The object is then assigned to the class that is defined as intersection of both original classes. For example, in the pornography application (section 8.3.1), an object classified as Image Gallery may also be independently classified as Scarce Text Fragment, which yields the class Porno Index. Another example was shown in section 7.2.2, where two taxonomies were merged, one based on HTML analysis and the other on link topology analysis. Based on large overlap in data, the 'HTML' concept of ProductCatalogue and the 'topological' concept of LocalHub were unified to the concept of "product catalogue referring to child documents with detailed information about the offered products" (say, 'ComplexProductCatalogue').

8.3 Example Descriptions of DWM Applications

Syntax of the Semi-Formal Language

Let us now describe concrete applications in terms of the TODD framework, including the mapping of tasks to PSMs. For this purpose, we will use an ad hoc semi-formal language with Prolog-like syntax. Its building blocks are *decompositions* of tasks ('heads of clauses') to ordered sequences of subtasks ('bodies of clauses'). Individual task descriptions ('literals') look as follows, respectively:

```
Cla?(<obj_var>, <obj_type>, <data_type>, <domain>, <classes>)
Ret?(<obj_var>, <obj_type>, <data_type>, <domain>, <constraints>)
Ext?(<obj_var>, <obj_type>, <data_type>, <domain>, <content>)
```

The 'predicate' (task name) corresponds to the first dimension in the TODD framework. An extra letter is used to distinguish the PSMs introduced in the previous sections: `ClaS` for Structural Classification, `ClaL` for Look-up based Classification, `ClaC` for Compact Classification; `RetD` for Direct Retrieval, `RetI` for Index-based Retrieval; `ExtS` for Structural Extraction, `ExtC` for Compact Extraction and `ExtO` for Overall Extraction. From the nature of the PSMs follows that each ClaS task can be decomposed to a structure including (among other) one or more subtasks of type Classification; analogously, each ExtS task can be decomposed to a structure including one or more subtasks of type Extraction. In the examples, the 'unification' of a 'goal' with a 'clause head' is always unique; the representation is only 'folded' for better readability.

The remaining three dimensions of the TODD model are reflected by the 'arguments' `<obj_type>`, `<data_type>` and `<domain>`:

- `<obj_var>` is variable referring to the 'current' object of the task instance: input object in the case of Classification and output object/s in the case of Retrieval. We use object variables (and object types) even for Extraction; however, here they only refer to the scope of extraction, not to a 'current' object as in Classification and Retrieval.
- `<classes>` is the list of classes distinguished in the classification task (beside named classes, we use the symbol `@other` for a 'complement' class).
- `<constraints>` is the list of logical expressions determining the set of objects to be retrieved; they correspond to the knowledge roles Class Constraints (class membership restrictions) and Structural Constraints (parthood/adjacency restrictions).
- `<content>` is the list of types of content information (datatype properties in semantic web terminology) to be extracted.

For simplicity, we ignore strictly procedural constructs such as selections or iterations, as well as the cardinality of input and output.

Overview of Applications Described

We first describe the *Rainbow* applications: pornography-recognition application [95] (Table 8.1) and two variants of bicycle offer extraction [45] (Tables 8.2 and 8.3). Then we, for better coverage, attempt to describe three DWM methods from the literature: the company website classification method by Ester et al. [25] (Table 8.4), the information extraction application for foundry websites by Krötzch & Rösner [42] (Table 8.5), and the bootstrapping approach to website information extraction by Ciravegna et al. [19] (Table 8.6) . The common aspect of all of them is the effort to overcome the limitations of single resource and/or single representation in web mining. However, the symbol-level principles of the methods are different: the first relies on probabilistic reasoning, the second on a mix of domain-specific heuristics, and the third on shallow NLP augmented with knowledge reuse. Due to limited space, we sometimes slightly simplify the structure of applications, without affecting their core principles.

8.3.1 Pornography-Recognition Application

The upper level of the pornography-recognition process is an instantiation of the *Structural Classification* PSM as discussed in the previous section. In order to classify the whole website (i.e. document collection), symptomatic 'out-tree' topology structures are first sought; their sources (local hubs) can possibly be identified with 'index' pages with image miniatures. To verify that, the hub is examined for presence of 'nudity' PICS rating in META tags (Look-up Classification PSM), for presence of indicative strings in the URL, and its whole HTML code

Table 8.1: TODD-based description of pornography application

```
ClaS(DC, DocCollection, _, Pornography, [PornoSite,@other]) :-
   RetD(D1, Document, topology, General, [D1 part-of DC, LocalHub(D1)]),
   ClaS(D1, Document, _, Pornography, [PornoIndex,@other]),
   RetD(D2, Document, topology, General, [D2 follows D1]),
   ClaS(D2, Document, _, Pornography, [PornoContentPage,@other]).
% classification of index page
ClaS(D, Document, _, Pornography, [PornoIndex,@other]) :-
   ClaL(D, Document, meta, Pornography, [PornoResource,@other]),
   ClaS(D, Document, url, Pornography, [PornoResource,@other]),
   RetD(DF, DocFragment, html-txt, General, [DF part-of D, ImgGallery(DF)]),
   ClaC(DF, DocFragment, freq, General, [ScarceTextFragment,@other]).
% classification of content page
ClaS(D, Document, _, Pornography, [PornoContentPage,@other]) :-
   ClaL(D, Document, meta, Pornography, [PornoResource,@other]),
   RetD(Im, Image, html-txt, General, [Im referenced-in D]),
   ClaC(Im, Image, image, Pornography, [PornoImage,@other]).
```

is searched for 'image gallery'-like structures with low proportion of text (which distinguishes pornography from regular image galleries). The analysis further concentrates on individual pages referenced by the hub, and attempts to identify a single dominant image at each of them. The images are then analysed by (bitmap) image analysis methods; in particular, the proportion of body colour and the central position of a dominant object are assessed. In the description, we omit the 'evaluation of global classification pattern' subtasks, for brevity; their inclusion would be straightforward.

8.3.2 Bicycle Application

Navigational Data Access

The start-up scenario for extraction of user-oriented information from bicycle-selling sites is centred around *navigation-based* access to individual pages; in the version described here, we use URL analysis for this purpose. Subsequently, statistical extraction (Hidden Markov Models) is applied to obtain structured information (products, company address), while phrasal patterns are applied on the (presumably) free text describing the overall company profile. All Retrieval tasks (for navigation-based access to pages as well as at the level of phrases in the last subtask) are mapped on the Direct Retrieval PSM. Most Extraction tasks shown correspond to Structural Extraction. However, at the lowest level, company address is obtained via Compact Extraction, and company description (sentences) are obtained via Overall Extraction. Product information is still a Structural Extraction; if we decomposed it further (not shown in the code), it would however consist of Compact Extraction followed with integration of individual information (names, prices and the like) to a more complex structure.

We omit other types of analysis included in the application (topology, META tags, images),

Table 8.2: TODD-based description of bicycle application with navigation

```
ExtS(DC, DocCollection, _, Bicycle, [products, comp_addr, comp_descr]) :-
   ExtS(DC, DocCollection, _, Bicycle, [products]).
   ExtS(DC, DocCollection, _, Company, [comp_descr]).
   ExtS(DC, DocCollection, _, Company, [comp_addr]).
% extraction of product information from catalogue pages
ExtS(DC, DocCollection, _, Bicycle, [products]) :-
   RetD(D, Document, url, Company, [D part-of DC, ProductCatalogue(D)]),
   ExtS(D, Document, html, Bicycle, [products]).
% extraction of company address from the contact page
ExtS(DC, DocCollection, _, Bicycle, [comp_addr]) :-
   RetD(D, Document, url, Company, [ContactPage(D)]),
   ExtC(D, Document, html, Company, [comp_addr]).
% extraction of general company profile from the profile page
ExtS(DC, DocCollection, _, Bicycle, [comp_descr]) :-
   RetD(D, Document, url, Company, [D part-of DC, ProfilePage(D)]),
   RetD(P1, Phrase, text, Company, [P1 part-of D, ProfilePhrase(P1)]),
   RetD(P2, Phrase, text, General, [Sentence(P2), P1 part-of P2]),
   ExtO(P2, Phrase, text, General, [comp_descr]).
```

Table 8.3: TODD-based description of bicycle application with index

```
...
ExtS(DC, DocCollection, _, Bicycle, [products, ...]) :-
   RetI(P, Phrase, text, Company, [P part-of DC, ProductCataloguePhrase(P)]),
   RetI(DF, DocFragment, html-tree, General, [DF contains P]),
   ExtC(DF, DocFragment, html, Bicycle, [products]),
   ...
```

for brevity.

Index-Based Data Access

Among alternative methods of page access, we are seriously considering the one taking full advantage of the available XML database engine (AmphorA), with its capability of term indexing combined with XML indexing. The parts of website suitable for detailed extraction can be efficiently detected via lexical indicators (e.g. phrases typically occurring nearby product catalogues): some sort of XML environment of the indicators can then be submitted to the extraction tool. Since the overall structure of the application is analogous to the previous one, we only show a fragment, in which Index-based Retrieval of indicative phrases plus Index-based Retrieval of 'mark-up environment' (in a native XML database storing the HTML trees) appears.

Table 8.4: TODD-based description of application by Ester et al.

```
ClaS(DC, DocCollection, _, Company, TopicSet) :-
   RetD(D, Document, topology, Company, [D part-of DC]),
   ClaC(D, Document, freq, Company, TopicSet),
   ClaC(DC, DocCollection, freq, Company, TopicSet).
```

Table 8.5: TODD-based description of application by Krötzch&Rösner

```
ExtS(DC, DocCollection, _, Foundry, [products, customers, certificates]) :-
   RetD(D, Document, html, Company, [D part-of DC, InfoPage(D)]),
   ExtS(D, Document, _, Foundry, [products]),
   ExtS(D, Document, _, Company, [customers]),
   ExtS(D, Document, _, Company, [certificates]).
% product information extraction
ExtS(D, Document, _, Foundry, [products]) :-
   RetD(DF, DocFragment, html, General, [DF part-of D, ContentTable(DF)]),
   ExtS(DF, DocFragment, html, Foundry, [products]).
% customer information extraction
ExtS(D, Document, _, Company, [customers]) :-
   RetD(P1, Phrase, text, Company, [P1 part-of D, CustomerPhrase(P1)]),
   RetD(P2, Phrase, parse-tree, General, [P2 depends-on P1]),
   ExtO(P2, Phrase, text, General, [customers]),
% certificate extraction
ExtS(D, Document, _, Company, [certificates]) :-
   RetD(P1, Phrase, text, Company, [P1 part-of D, QualityPhrase(P1)]),
   RetD(P2, Phrase, parse-tree, General, [CertName(P2), P2 depends-on P1]),
   ExtO(P2, Phrase, text, General, [certificates]).
```

8.3.3 Website Mining by Ester et al.

The method is not knowledge-based: it relies on Bayesian classification of individual documents (wrt. topics) over the feature space of terms, and then again on Bayesian classification, this time of the whole website (i.e. document collection) over the feature space of individual document's topics. Hence, the overall task pattern amounts to Structural Classification similar to the pornography-recognition task, while the embedded (Bayesian) classifications are Compact.

8.3.4 Company Profile Extraction by Krötzch&Rösner

The overall scheme is similar to the bicycle application, except that product information is only extracted from tables (via heuristics), while phrasal patterns are used in a finer way, to extract not just sentences but names of either customers or quality certificates.

8.3.5 Bootstrapping Information Extraction by Ciravegna et al.

The approach described in [19] heavily relies on knowledge reuse, thanks to the well-known redundancy of WWW information. We only describe the most elaborated part of the method, targeted at extraction of person names (additionally, various personal data and paper titles are extracted for the persons in question). First, potential names are cropped from the website, and checked against binary classification tools such as context-based named-entity recognisers (Compact Classification), as well as against public search tools (namely, online bibliographies, homepage finders and general search engines) that produce the same binary classification (person name - yes/no) as by-product of offering information on papers or homepages (i.e. Index-based Retrieval). Furthermore, for the results of general web search, the page from the given site is labelled as homepage if the name occurs in a particular (typically, heading) tag. The seed names obtained are further extended by names co-occurring in a list or in the same column of a table. Finally, potential person names from anchors of intra-site hyperlinks are added.

8.4 Template-Based Composition of DWM Services

8.4.1 Templates in Web Service Composition

Composition[3] of simple *web services* into sophisticated (distributed) applications recently became one of hottest topics in computer science research. The area of application for (composite) web-services is potentially quite wide. While the focus is most often on B2B transactions and financial services, the general paradigm appears useful even for less critical tasks such as organisation of scientific events [90] or information harvesting from the *surface web*, which is the focus of the thesis.

Three alternative research streams can be identified:

1. *Programming in the large*, i.e. composition of services by (more-or-less) traditional procedural programming in languages such as BPEL4WS[4], inspired by workflow research. This stream is the only one recognised by most of the industrial IT community, to date; its main advantage is perfect control over the choice and linkage of different services, at design time. This however, on the other hand, entails a rather low degree of flexibility at run time.
2. *Planning* in artificial intelligence style, based on pre- and post-conditions of individual services without pre-specified control flows, as in OWL-S [5]. This approach offers extreme flexibility; however, the results may be quite unpredictable if all conditions are not perfectly specified, which may often be difficult in real environments.
3. *Template-based* composition, in which concrete services are filled in run time into prefabricated templates [51, 90].

[3] Other terms such as 'choreography' or 'configuration' are also frequently used.
[4] `http://www-128.ibm.com/developerworks/library/ws-bpel`

Table 8.6: TODD-based description of an Armadillo application

```
ExtS(DC, DocCollection, _, CSDept, [names]) :-
   RetD(P1, Phrase, text, General, [P1 part-of DC, PotentPName(P1)]),
   % named entity recognition for person names
   ClaC(P1, Phrase, text, General, [PName,@other]),
   % use of public search tools over papers and homepages
   RetI(P2, Phrase, freq, Biblio, P1 part-of P2, PaperCitation(P2)]),
   RetI(D, Document, freq, General, [P1 part-of D, D part-of DC, PHomepage(D)]),
   RetD(DF1, DocFragment, freq, General,
         [Heading(DF1), DF1 part-of D, P1 part-of DF1),
   ExtO(P1, Phrase, text, General, [names]),
   % co-occurrence-based extraction
   RetD(DF2, DocFragment, html, General,
         [ListItem(DF2), DF2 part-of DC, P1 part-of DF2]),
   RetD(DF3, DocFragment, html, General,
         [ListItem(DF3), (DF3 below DF2; DF2 below DF3)]),
   ExtS(DF3, DocFragment, text, General, [names]),
   RetD(DF4, DocFragment, html, General,
         [TableField(DF4), DF4 part-of DC, P1 part-of DF4]),
   RetD(Q, DocFragment, html, General,
         [TableField(DF5), (DF5 below DF4; DF4 below DF5)]),
   ExtS(DF5, DocFragment, text, General, [names]),
   % extraction from links
   RetD(DF5, DocFragment, html, General,
         [IntraSiteLinkElement(DF5), DF5 part-of DC]),
   ExtS(DF5, DocFragment, text, General, [names]),
   ...
% extraction of potential person names from document fragments
ExtS(DF, DocFragment, text, General, [names]) :-
   RetD(P, Phrase, text, General,
         [DF contains P, PotentialPersonName(P)]),
   ExtO(P, Phrase, text, General, [names]).
```

Most recently, ten Teije et al. [90] suggested to view web service composition templates as analogy to *problem solving methods* (PSMs), i.e. abstract descriptions of knowledge-based reasoning scenarios, which have been intensely studied in the knowledge modelling community for nearly two decades (see section 6.2). In addition, they suggested to view the *configuration* of the template again as a kind of reasoning task, namely, that of *parametric design*. Each concrete application is assumed to be specified (in sufficient detail) merely as combination of values assigned to a fixed set of parameters. The configuration process is carried out by a so-called *broker* tool, and employs the *propose-critique-modify* (PCM) reasoning method[5], taking advantage of *background knowledge* of the broker. As the approach taken in this thesis directly builts on ten Teije's work, we describe the parametric design approach in more detail in the next subsection.

8.4.2 Configuration of Web Services as Parametric Design

Current approaches to Web service configuration are often based on pre/post-condition-style reasoning. Given descriptions of elementary Web services, and the required functionality of the composite Web service, they aim to try to construct a 'plan' of how to compose the elementary services in order to obtain the required functionality. In [90], we instead proposed a *knowledge intensive* approach to the creation of composite Web services. We described a complex Web service as a fixed template, which must be configured for each specific use. Web service configuration can then be regarded as *parametric design*, in which the parameters of the fixed template have to be instantiated with appropriate component services. During the configuration process, we exploit detailed knowledge about the template and the components, to obtain the required composite web service. Whereas in other work the main metaphor is "Web service configuration = planning" (i.e. generalised reasoning based on only component specifications), our approach is based on the metaphor "Web service configuration = brokering" (i.e. reasoning with specialised knowledge in a narrow domain). A *planner* is assumed to be *domain-neutral*: it is supposed to work on any set of components, simply given their descriptions. A *broker* on the other hand exploits specific knowledge about the objects it is dealing with. In the remainder of this section, we describe how such a broker can be equipped with configuration knowledge on how to combine these web services.

Parametric Design

Parametric design is a simplification of general configuration. It assumes that the objects to be configured (in our case: complex Web services) have the same overall structure that can be captured by templates. Variations on the configuration can only be obtained by choosing the values of given parameters within these templates. We will show that for specific type of web services, namely classification services, this is indeed possible.

An existing reasoning method (PSM) for parametric design is *Propose-Critique-Modify*, or PCM for short [13]. The PCM method consists of four steps:

[5] I.e. a 'meta-level' PSM with respect to that incorporated in the template itself.

- The *propose step* generates an initial configuration. It proposes an instance of the general template used for representing the family of services.
- The *verify* step checks if the proposed configuration satisfies the required properties of the service. This checking can be done by both pre/post-condition reasoning, or by running the service.
- The *critique* step analyses the reasons for failure that occurred in the verification step: it indicates which parameters may have to be revised in order to repair these failures.
- The *modify* step determines alternative values for the parameters identified by the critique step. The method then loops backto the verify step.

The propose-critique-modify method for Parametric Design requires specific types of configuration knowledge to drive the different steps of the configuration process. The question is whether this configuration knowledge (PCM knowledge) can be identified for large classes of Web services. It turns out that this is indeed possible for a specific class of web services, namely, *classification* ones.

Application on Classification Services

The common definition of classification is [74]: "Classification problems begin with data and identify classes as solutions. Knowledge is used to match elements of the data space to corresponding elements of the solutions space, whose elements are known in advance." More formally, classification uses knowledge to map observations (in the form of ⟨*feature,value*⟩-pairs) to classes.

Based on the work by Motta&Lu [58], we assume that classification services can be described in a single template. This template (see Section 8.2) consists of five steps: *Check*, *MicroMatch*, *Aggregate*, *Admissibility* and *Selection*.

This structure constitutes the overall template for classification services, which can be easily captured in current Web service description languages such as OWL-S [5]. Example values of *Admissibility* parameter are (see [90] for more):

- *weak-coverage*: All ⟨*feature,value*⟩ pair in the observations are *consistent* with the feature specifications of the solution.
- *strong-coverage*: All ⟨*feature,value*⟩ pair in the observations are *consistent* with the feature specifications of the solution and *explained* by them.
- *strong-explanative*: All ⟨*feature,value*⟩ pair in the observations are *consistent* with the feature specifications of the solution, *explained* by them, and all features specified in the solution are *present*.

The value of *Selection* parameter then decides whether e.g. the number of unexplained and missing features is considered in ranking candidate solutions.

The broker may employ e.g. the following pieces of knowledge:

- Propose knowledge for the *Admissibility* parameter: if many ⟨*feature,value*⟩ pairs are irrelevant then do not use *strong-coverage*.
- Critique knowledge for the *Selection* parameter: if the solution set is too small or too large then adjust the *Admissibility* or the *Selection* parameter.
- Modify knowledge for the *Admissibility* parameter: if the solution set has to increased (reduced) in size, then the value for the *Admissibility* parameter has to be moved down (up) in the following partial ordering:
  weak-coverage $\prec$ *strong-coverage* $\prec$ *strong-explanative*.

A prototype PCM broker has been successfully applied on real data in the domain of conference paper classification (for reviewer assignment).

8.4.3 *Rainbow* Applications as Composite Web Services

For the first composite application of *Rainbow*, a few hundred lines of Java code sufficed to weave together the tools cooperating in the analysis of bicycle websites (see section 4.2.2). However, with increasing number of available tools, composition by traditional programming soon becomes cumbersome. On the other hand, the space of suitable tools will hardly be as borderless as in semantic-web scenarios of information search, which are assumed amenable to planning approaches. The *template-based approach* thus looks as a reasonable compromise. The collection of *PSMs* abstracted from real *deductive web mining* applications, explained in section 8.2, could be basis for templates. Furthermore, individual components (services) can be positioned in the TODD *multi-dimensional space*, which could, among other, play a similar role as the space of template parameters from [90].

An important point is to evaluate the possibility to adapt the parametric design approach from [90] to the (specific features of) web analysis PSMs; this is the subject of the next subsection. Main focus will be on *classification*, which is the only task considered in [90] and also one of tasks studied in this thesis.

8.4.4 DWM Service Configuration as Parametric Design

Limitations of Fixed Template

As we outlined in section 8.2, the PSMs for deductive web mining tend to involve *recursion*: a reasoning process starting at one object is successively redirected to other objects in its part-hood or neighbourhood. This more-or-less disqualifies reasoning methods relying on a *single and entirely fixed feature template*, of which parametric design is a typical representative. There seem to be at least two possible solutions to this problem:

1. to allow for *multiple templates per task*, differing in the number of 'sibling' sub-tasks and degree of recursion, and to include *heuristics for template selection* as part of broker knowledge.

2. to modify the *parametric design algoritm* to involve, in addition to setting parameter values, also *template-restructuring operations* such as subtask replication and recursive unfolding (i.e. replacement of parameter with a whole template for processing a different object).

In the rest of this chapter, we outline the first solution, since it is easier to design and implement in its rudimentary form; it obviously oversimplifies many aspects of real-world settings.

Description of Templates

Table 8.7 shows five templates for the classification task (encoded in Prolog): the first amounts to single classification of the current object, the second aggregates two different ways of classifying the current object, the third and the fourth rely on another object (sub-object or adjacent object) in order to classify the current object, and the fifth combines direct classification of current object with its structural classification (via classification of another object). The arguments of the `templ` clauses amount to the following: template *identifier* (`sc#`), composed service *signature*, list of component services *signatures* (one for each 'empty slot'), list of ontological *constraints* among object types (classes). Each signature (i.e. `s()` structure) first defines the *task type* accomplished by the service; the numbers (`0`, `1`, ...) have the semantic of variables that either refer to objects or to slots themselves (`0` being the 'start-up' object of the composed service), and the Prolog variables `Tp#` correspond to types (or classes) of these objects. In addition to classification (`cla`) and retrieval (`ret`) services types, the templates also include slots for *auxilliary services* needed to accomplish the target classification task. As types of auxilliary services, we so far considered aggregation (`agr`), transformation (`tsf`) and iteration (not shown here). For example, the presence of sub-object of certain class determines the class of the super-object in a certain way. In particular, the certainty factor of classification of sub-object is *transformed* to certainty factor of classification of super-object; the data flow between the services is indicated by the `ref(SourceService,SourceObject)` construct[6]. Similarly, classification of the same object by different methods has to be compared and the result computed via *aggregation* (e.g. combining the certainty factors).

[6]This method of combining control flow and data flow is rather cumbersome and is likely to be replaced with a smarter one in a real web service composition language.

Table 8.7: Sample templates for classification task

```
templ(sc1,s(cla,0,0,Tp1,Tp2),
  [s(cla,0,0,Tp3,Tp4)],[subclasseq(Tp3,Tp1),subclasseq(Tp4,Tp2)]).
templ(sc2,s(cla,0,0,Tp1,Tp2),
  [s(cla,0,0,Tp3,Tp4),s(cla,0,0,Tp5,Tp4),
  s(agr,[ref(1,0),ref(2,0)],0,Tp4,Tp4)],
  [subclasseq(Tp3,Tp1),subclasseq(Tp5,Tp1),subclasseq(Tp4,Tp2)]).
templ(sc3,s(cla,0,0,Tp1,Tp2),
  [s(ret,0,1,Tp3,Tp4),s(cla,1,1,Tp5,Tp6),s(tsf,ref(2,1),0,Tp6,Tp2)],
  [subclasseq(Tp3,Tp1), rel(part,Tp4,Tp3), subclasseq(Tp4,Tp5)]).
templ(sc4,s(cla,0,0,Tp1,Tp2),
  [s(ret,0,1,Tp3,Tp4),s(cla,1,1,Tp5,Tp6),s(tsf,ref(2,1),0,Tp6,Tp2)],
  [subclasseq(Tp3,Tp1),rel(adj,Tp4,Tp3),subclasseq(Tp4,Tp5)]).
templ(sc5,s(cla,0,0,Tp1,Tp2),
  [s(cla,0,0,Tp3,Tp4),s(ret,0,1,Tp5,Tp6),s(cla,1,1,Tp7,Tp8),
  s(tsf,ref(3,1),0,Tp8,Tp4),s(agr,[ref(1,0),ref(4,0)],0,Tp4,Tp4)],
  [subclasseq(Tp3,Tp1),subclasseq(Tp5,Tp1),rel(part,Tp6,Tp5),
  subclasseq(Tp6,Tp7),subclasseq(Tp4,Tp2)]).
```

8.5 Simulation of Template Configuration and Execution

8.5.1 One-Shot Setting Without Broker Knowledge

As seen from the above discussion, there are several differences from the seminal work by ten Teije et al., most important:

- We do not have a single template but a choice of multiple ones
- For the individual template slots, we don't deal with a clearly defined family of different methods (variations of a method) but with a theoretically borderless space of applicable tools.

It was therefore natural to start with a fragment of the original Parametric Design model only, namely, with its *Propose* and *Verify* (in the sense of service execution) only. Although *broker knowledge* would be desirable (and was used by ten Teije) for the Propose phase, it was not indispensable, and we could perform service configuration based on the *signatures* in the templates only. The use of broker knowledge is only discussed in the following subsection.

One-Shot Setting Without Broker Knowledge

We implemented a collection of simple programs in Prolog consisting of:

1. the five templates discussed in the previous sections

2. four simulated 'websites' (inspired by real ones), in clausal form, an example is in Table 8.8
3. simplified services (incl. auxilliary ones) operating on 'website clauses' and equipped with meta-data
4. a *configuration tool* that selects and fills in the templates based on service meta-data
5. an *execution tool* that executes the filled template for a given data object
6. an 'ontology' (derived from the UWO and its sub-models, see section 7.2) containing definitions of basic concepts required for the composition and/or execution phase.

The whole setting is very rudimentary. The service slots in templates are limited to a single object on input and on output. The classification services only perform binary classification, i.e. they output a certainty factor for a single class on output (distinguishing it from its complement). The classes amount to pornography-relevant ones, such as pornography-containing site or pornography content page.

Composition Phase

Table 8.9 shows two examples of service composition. The first one suggests two ways of classifying a document as `pornoContentPage`, based on two different templates: either by directly classifying the document or by first retrieving and classifying its follow-up document and then transforming the certainty factor. The second one suggests to classify a site by retrieving and classifying its hub page.

Execution Phase

The composed services can then be *executed*. For example, we can call the already configured template `sc4` from above using the ID of input object, its initial class (e.g. just `document` as generic type) and the certainty factor of this class (it should be 1 in this case). The execution engine returns the ID of output object (for a classification task, it is identical to input object), its suggested class (here, `pornoContentPage`), and the certainty factor of this refined class. The results can be compared with 'gold standard' data and thus provide a simple form of *verification* of the configuration.

8.5.2 Towards a Complete Parametric Design Cycle

Tentative Broker Knowledge

While the initial configuration of the template ('propose' phase) could be accomplished using 'semantic signatures' of individual services only, its subsequent automated *modification*

requires additional knowledge. Tentative examples of such knowledge have been formulated in [85]. Compared to broker knowledge from [90], they also include template selection/reformulation knowledge in addition to slot-filling knowledge. They are limited to *Propose* knowledge, which is relevant for initial setting of parameters. Note that, in our multiple-template version, broker knowledge relates to *template selection* as well as to *specification of arguments* for all subtasks within the template:

- Templates with lower number of distinct objects (`X`, `Y`, `Z`, ...) should be preferred.
- Non-recursive templates should be preferred; moreover, look-up classification should be preferred to compact classification.
- Default partial ordering of data types with respect to object classification, for *Document* object (may be overridden in a domain context):
  frequency $\succ$ *URL* $\succ$ *topology*, *free_text* $\succ$ *metadata*
- URL-based or topology-based classification (as rather unreliable kinds of services) should never be used alone, i.e. can only be filled into a template with 'parallel' classification of same object, such as SC2 or SC4
- Default partial ordering of types of relations (`@rel`) to be inserted into classification template (may be overridden in a domain context):
  part-of $\succ$ *is-part* $\succ$ *adjacent*
- Preference of domains used in structural classification,with respect to the domain of current object: *same domain* $\succ$ *super-domain* $\succ$ *other_domain*.
- The class of object determined by a Classification sub-task should be (according to domain knowledge) sub-class of the class of objects determined by the immediately preceding Retrieval sub-task in the template.

These heuristics are merely tentative, to illustrate the variety of possible broker knowledge for DWM applications.

Example of Broker-Based Composition

Let us further show a hypothetical scenario of the use of broker knowledge, in connection with the pornography-recognition application from section 8.3.1.

Let us assume a web pornography ontology[7] grafted upon the *Upper Web Ontology* and containing among other the following description-logic axioms:

```
PornoSite same-class-as (WebSite and (has-part some PornoIndex))
PornoIndex same-class-of (LocalHub and (followed-by >1 PornoContentPage))
```

[7]This ontology has actually been developed in DAML+OIL, by the authors [83].

For an application recognising pornography websites, the broker would select the template SC3, which is simpler than SC4; neither SC1 nor SC2 would be applicable (assuming no service was able to recognise `PornoSite` by Look-Up or Compact Classification). In attempting to fill SC3 in, it would seek a class of related object that could help determine the class of current object. With the help of the first axiom, it finds out that `PornoIndex` could serve for the purpose (as part of sufficient condition); it will thus accordingly instantiate the Classification sub-task. Then it will determine, by the second axiom, a suitable class of objects to be retrieved in the preceding (Retrieval) sub-task as `LocalHub`; since this is not a pornography concept but generic concept, `Domain1` will be set to `General`. Finally, it finds out that `LocalHub` cannot be recognised as `PornoIndex` merely by Look-Up or Compact Classification. It will thus have to create another SC3 template, on the second level, in order to recognise `PornoIndex` by means of `PornoContentPage`s following it in the link topology.

Table 8.8: Incomplete example of simulated 'website' in clausal form

```
site(s2).
class(s2,nonporno).

page(p21). % index page with 5 html fragments and no pictures
url_of(u21,p21).
part(p21,s2).
textprop(p21,0.9). % proportion of text on page
part(f211,p21).
html_frag(f211). % fragment 1
part(f212,p21).
html_frag(f212). % fragment 2
part(f213,p21).
html_frag(f213). % fragment 3
part(f214,p21).
html_frag(f214). % fragment 4
part(f215,p21).
html_frag(f215). % fragment 5

page(p22). % text only page with 2 html fragments and no pictures
url_of(u22,p22).
part(p22,s2).
linkto(p21,p22).
textprop(p22,0.9). % proportion of text on page
part(f221,p22).
html_frag(f221). % fragment 1
part(f222,p21).
html_frag(f222). % fragment 2

page(p23). % page with 2 html fragments and 1 picture
url_of(u23,p23).
url_terms(u23,[hot]).
part(p23,s2).
linkto(p21,p23).
textprop(p23,0.8). % proportion of text on page
part(f231,p23).
html_frag(f231). % fragment 1
part(i2311,f231).
image(i2311).
body_color(i2311,0.1).
part(f232,p23).
html_frag(f232). % fragment 2
```

Table 8.9: Service composition dialogue

```
?- propose(cla, document, pornoContentPage).
Number of solutions: 2
 Template:       sc1
 Configuration:
  s(cla, 0, 0, document, pornoContentPage, cla_por_url)
 Template:       sc4
 Configuration:
  s(ret, 0, 1, document, document, ret_follows)
  s(cla, 1, 1, document, pornoContentPage, cla_por_url)
  s(tsf, ref(2, 1), 0, pornoContentPage, pornoContentPage, tsf_porno2)

?- propose(cla, doc_coll, porno_coll).
Number of solutions: 1
 Template:       sc3
 Configuration:
  s(ret, 0, 1, doc_coll, localhub, ret_localhub)
  s(cla, 1, 1, document, pornoContentPage, cla_por_url)
  s(tsf, ref(2, 1), 0, pornoContentPage, porno_coll, tsf_porno1)
```

Summary and Prospects

The presented habilitation thesis attempted to show that reuse of resources (software components and data collections) as well as conceptual models (ontologies and PSMs) is useful for efficient analysis of the web space. Most concrete examples were taken from the *Rainbow* project (`http://rainbow.vse.cz`), which is coordinated by the author.

As the exploitation of web content and structure is by far not a resolved problem, the motivation for extending the outcomes of the *Rainbow* project, presented in this thesis, is high. The most interesting research goals can be summarised as follows:

- To *automatically build* web analysis applications on the fly, using ontological descriptions of individual tools
- To effectively combine *inductively learnt information extraction models* with *wrapper ontologies*, for newly addressed domains.
- To make full benefit of the available *XML indexing and querying* technology as preprocessor to knowledge-based analysis, capable of providing an appropriate XML environment to the knowledge-based web analysis tools.

The author and his collaborators envisage to focus on these issues in the framework of several newly commenced EU-funded projects, from 2006 on.

Bibliography

[1] IBROW homepage, `http://www.swi.psy.uva.nl/projects/ibrow`

[2] Abasolo, C. et al.: Arcos, J.-L., Armengol, E., Gómez, M., López-Cobo, J.-M., López-Sánchez, M., López de Mantaras, R., Plaza, E., van Aart, C., Wielinga, B.: Libraries for Information Agents. IBROW Deliverable D4, IIIA, Barcelona, March 2001. Online at `http://www.swi.psy.uva.nl/projects/ibrow/docs/deliverables/deliverables.html`.

[3] Andrt M., Krátký, M., Svátek, V., Snášel, V.: AmphoraWS – webova sluzba pro vyhledavani ve strukturovanych dokumentech. [AmphoraWS – Web service for querying semi-structured data.] In: Datakon'04, Brno 2004.

[4] Anjewierden, A.: A library of document analysis components, IBrow deliverable D2b. Online at `http://www.swi.psy.uva.nl/projects/ibrow/docs/deliverables/deliverables.html`.

[5] Ankolekar, A. et al.: DAML-S: Semantic markup for web services. In: Proc. ISWC 2002, *LNCS* 2342, pp. 348–363.

[6] Antoniou, G., van Harmelen, F.: A Semantic Web Primer. MIT Press, 2004.

[7] Arpírez, J., C., Corcho, O., Fernández-López, M., Gómez-Pérez, A.: WebODE in a nutshell. *AI Magazine*, 24(3):37-48. Fall 2003.

[8] Attardi G., Gull A., Sebastiani F. Automatic Web Page Categorization by Link and Context Analysis. In: THAI-99, European Symposium on Telematics, Hypermedia and Artificial Inteligence, ed.: Chris Hutchison and Gaetano Lanzarone, Varese, 1999, 105-119.

[9] Baumgartner, R., Flesca, S., Gottlob, G.: Visual Web Information Extraction with Lixto. In: Proc. VLDB 2001.

[10] Berka P., Sochorová M., Svátek V., Šrámek D.: The VSEved System for Intelligent WWW Metasearch. In: (Rudas I. J., Madarasz L., eds.:) INES'99 – IEEE Intl. Conf. on Intelligent Engineering Systems, Stara Lesna 1999, 317-321.

[11] Brin, S.: Extracting Patterns and Relations from the World-Wide Web. In WebDB Workshop at EDBT'98.

[12] Broekstra, J., Ehrig, M., Haase, P., van Harmelen, F., Kampman, A., Sabou, M., Siebes, R., Staab, S., Stuckenschmidt, H., Tempich, C.: A Metadata Model for Semantics-Based Peer-to-Peer Systems. In: Proceedings of the WWW'03 Workshop on Semantics in Peer-to-Peer and Grid Computing, Budapest, 2003.

[13] Brown, D., Chandrasekaran, B.: Design problem solving: knowledge structures and control strategies. *Research notes in AI*, 1989.

[14] Buitelaar, P., Cimiano, P., Magnini, B., Eds.: Ontology Learning and Population, Series information for Frontiers in Artificial Intelligence and Applications, IOS Press, 2005.

[15] Burget, R.: Hierarchies in HTML Documents: Linking Text to Concepts, In: 15th Int'l Workshop on Database and Expert Systems Applications, Zaragoza, IEEE CS, 2004, 186-190.

[16] Chakrabarti, S., et al.: Automatic resource compilation by analyzing hyperlink structure and associated text. In: Proc. 7th Intl. World Wide Web Conf., 1998.

[17] Cimiano, P., Staab, S.: Learning by Googling. *SIGKDD Explorations*, 6(2), pp. 24-33.

[18] Ciravegna, F.: LP2, an Adaptive Algorithm for Information Extraction from Web-related Texts. In: Proceedings of the IJCAI-2001 Workshop on Adaptive Text Extraction and Mining, Seattle, WA, 2001.

[19] Ciravegna, F., Dingli, A., Guthrie, D., Wilks, Y.: Integrating Information to Bootstrap Information Extraction from Web Sites. In: IJCAI'03 Workshop on Intelligent Information Integration, 2003.

[20] Clancey, W. J.: Acquiring, representing, and evaluating a competence model of diagnostic strategy. In: The Nature of Expertise, Lawrence Erlbaum Press 1988.

[21] Dill, S., Eiron, N., Gibson, D., Gruhl, D., Guha, R., Jhingran, A., Kanungo, T., Rajagopalan, S., Tomkins, A., Tomlin, J., Zien, J.: SemTag and Seeker: Bootstrapping the semantic web via automated semantic annotation. In: Proc. WWW2003, Budapest 2003.

[22] Embley, D.W., Campbell, D.M., Jiang, Y.S., Liddle, S.W., Lonsdale, D.W., Ng, Y.K., Smith, R.D.: Conceptual-model-based data extraction from multiple-record Web pages. *Data and Knowledge Engineering*, Volume 31, Issue 3 (November 1999).

[23] Embley, D.W., Tao, C., Liddle, S.W.: Automatically extracting ontologically specified data from HTML tables with unknown structure. In: ER2002, Tampere 2002, 322-337.

[24] Engels, R., Lindner, G., Studer, R.: Providing User Support for Developing Knowledge Discovery Applications; A Midterm report. In: S. Wrobel (Ed.) *Themenheft der Künstliche Intelligenz*, (1) March, 1998.

[25] Ester, M., Kriegel, H.P., Schubert, M.: Web Site Mining: a new way to spot Competitors, Customers and Suppliers in the World Wide Web. In: Proc. KDD 2002.

[26] Fensel, D., Benjamins, V. R., Motta, E., Wielinga, B.: UPML: A Framework for knowledge system reuse. In Proceedings of the International Joint Conference on AI (IJCAI-99), Stockholm, Sweden, July 31 - August 5, 1999.

[27] Fensel, D., Hendler, J., Liebermann, H., Wahlster, W. (eds.): Spinning the Semantic Web. MIT Press, 2003.

[28] Gal, A., Modica, G.A., Jamil, H.M.: OntoBuilder: Fully Automatic Extraction and Consolidation of Ontologies from Web Sources. In: Proc. ICDE 2004.

[29] Gangemi, A.: Ontology Design Patterns for Semantic Web Content. In: International Semantic Web Conference 2005: 262-276.

[30] Gatterbauer, W., Krüpl, B., Holzinger, W., Herzog, M.: Web Information Extraction Using Eupeptic Data in Web Tables. In: RAWS 2005, September 14-16, 2005, Prague.

[31] Gomez-Perez, A., Fernandez-Lopez, M., Corcho, O.: Ontological Engineering. Springer 2004.

[32] Gruber, T. R.: A Translation Approach to Portable Ontology Specifications. *Knowledge Acquisition*, 5 (2), 1993, 199-220.

[33] Gu, X., Chen, J., Ma, W., and Chen, G.: Visual Based Content Understanding towards Web Adaptation. In: Proceedings of the Second international Conference on Adaptive Hypermedia and Adaptive Web-Based Systems. Lecture Notes In Computer Science, vol. 2347. Springer-Verlag, London, 164-173.

[34] Handschuh, S., Staab, S., Ciravegna, F.: S-CREAM - Semi-automatic CREAtion of Metadata. In: Proc. EKAW-2002, LNCS, Springer, 2002.

[35] Hearst, M.: Automatic Acquisition of Hyponyms from Large Text Corpora. Proc. of theFourteenth International Conference on Computational Linguistics, Nantes, France, 1992.

[36] Heß, A., Johnston, E., Kushmerick, N.: ASSAM: A Tool for Semi-automatically Annotating Semantic Web Services. In: Proc. International Semantic Web Conference 2004, 320-334.

[37] Jin, Y., Decker, S., Wiederhold, G.: OntoWebber: Model-Driven Ontology-Based Web Site Management. In: 1st International Semantic Web Working Symposium (SWWS'01), Stanford University, Stanford, CA, July 29-Aug 1, 2001.

[38] Kavalec, M., Svátek, V.: Information Extraction and Ontology Learning Guided by Web Directory. In: ECAI Workshop on NLP and ML for ontology engineering. Lyon 2002.

[39] Kavalec, M., Svátek, V., Strossa, P.: Web Directories as Training Data for Automated Metadata Extraction. In: Semantic Web Mining, Workshop at ECML/PKDD2001, Freiburg 2001, 39-44.

[40] Knoblock, C. et al.: Modeling Web Sources for Information Integration. In: Proceedings of the 15th National Conference on Artificial Intelligence (AAAI–98).

[41] Kogut, P., Holmes, W.: AeroDAML: Applying Information Extraction to Generate DAML Annotations from Web Pages. In: K-CAP 2001 Workshop Knowledge Markup & Semantic Annotation, 2001.

[42] Krötzch, S., Rösner, D.: Ontology based Extraction of Company Profiles. In: Workshop DBFusion, Karlsruhe 2002.

[43] Kushmerick, N., Weld, D. S., Doorenbos, R.: Wrapper Induction for Information Extraction. In: Intl. Joint Conference on Artificial Intelligence (IJCAI), 1997.

[44] Labský, M., Svátek, V.:.Ontology Merging in Context of Web Analysis. In: Workshop DATESO03, TU Ostrava, 2003.

[45] Labský, M., Svátek, V., Šváb, O., Praks P., Krátký, M., Snášel, V.: Information Extraction from HTML Product Catalogues: from Source Code and Images to RDF. In: 2005 IEEE/WIC/ACM International Conference on Web Intelligence (WI'05), IEEE Computer Society, 2005.

[46] Labský, M., Vacura M., Praks P.: Web Image Classification for Information Extraction. In: First International Workshop on Representation and Analysis of Web Space (RAWS-05), online `http://CEUR-WS.org/Vol-164`.

[47] Lafferty, J., McCallum, A., Pereira, F.: Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In: Proc. ICML-2001.

[48] Li, Y., Zhang, L., Yu, Y.: Learning to Generate Semantic Annotation for Domain Specific Sentences. In: K-CAP 2001 Workshop on Knowledge Markup & Semantic Annotation, October 21, 2001, Victoria B.C., Canada.

[49] Maedche, A.: Ontology Learning for the Semantic Web. Kluwer, 2002.

[50] Maedche, A., Neumann, G., Staab, S.: Bootstrapping an Ontology-Based Information Extraction System. Studies in Fuzziness and Soft Computing, editor J. Kacprzyk. In: Intelligent Exploration of the Web, P.S. Szczepaniak, J. Segovia, J. Kacprzyk, L.A. Zadeh, Springer 2002.

[51] Mandell, D. J., McIlraith, S. A.: Adapting BPEL4WS for the Semantic Web: The Bottom-Up Approach to Web Service Interoperation. In: Proc. ISWC2003.

[52] Martin, D. et al.: OWL-S 1.0 Release. Online at `http://www.daml.org/services/owl-s/1.0/`.

[53] Mathieu F., Viennot L.: Local Structure in the Web. In: Poster Session of the International World-Wide Web Conference, Budapest 2003.

[54] McCallum, A., Freitag, D.: Information extraction with hmms and shrinkage. In: Proceedings of the AAAI-99 Workshop on Machine Learning for IE, 1999.

[55] McCallum, A., Nigam, A.: Text Classification by Bootstrapping with Keywords, EM and Shrinkage. In: ACL'99 Workshop for Unsupervised Learning in NLP, 1999.

[56] Mladenic, D.: Turning Yahoo into an Automatic Web-Page Classifier. In: Proc. 13th European Conference on Aritficial Intelligence, ECAI'98, 473-474.

[57] Motik, B., Maedche, A., Volz, R.: A Conceptual Modeling Approach for Semantics-Driven Enterprise Applications. In: Proc. CoopIS/DOA/ODBASE 2002, 1082-1099.

[58] Motta, E., Lu, W.: A Library of Components for Classification Problem Solving. In: Proceedings of PKAW 2000, The 2000 Pacific Rim Knowledge Acquisition Workshop, Sydney, Australia, December 2000.

[59] Newell, A. 1982. The knowledge level. *Artificial Intelligence*, 18:87-127.

[60] Noy, N., ed.: Representing Classes As Property Values on the Semantic Web. W3C Working Group Note, 5 April 2005, online at `http://www.w3.org/TR/swbp-classes-as-values/`.

[61] Noy, N., Rector, A. (eds.): Defining N-ary Relations on the Semantic Web: Use With Individuals. W3C Working Draft, 21 July 2004, online at `http://www.w3.org/TR/swbp-n-aryRelations/`.

[62] Patil, A., Oundhakar, S., Sheth, A., Verma, K.: METEOR-S Web service Annotation Framework, Proceedings of the World Wide Web Conference, July 2004.

[63] Pazienza, M.T., Stellato, A., Vindigni, M.: Combining ontological knowledge and wrapper induction techniques into an e-retail system. In: ECML/PKDD workshop ATEM, Cavtat-Dubrovnik, 2003.

[64] Praks, P., Dvorský, J., Snášel, V.: Latent semantic indexing for image retrieval systems. In: Proceedings of the SIAM Conference on Applied Linear Algebra (LA03), Williamsburg, USA.

[65] Rabiner, L.R.: A tutorial on hidden Markov models and selected applications in speech recognition. In: *Proceedings of the IEEE*, 77(2), 1989.

[66] Riloff, E., Jones, R.: Learning Dictionaries of Information Extraction by Multi-Level Bootstrapping. In: Proc. 16th Nat. Conf. Artificial Intelligence (AAAI-99).

[67] Roman, D., Keller, U., Lausen, H., de Bruijn, J., Lara, R., Stollberg, M., Polleres, A., Feier, C., Bussler, C., and Fensel, D.: Web Service Modeling Ontology, *Applied Ontology*, 1(1): 77–106, 2005.

[68] Sabou, M.: Learning Web Service Ontologies: an Automatic Extraction Method and its Evaluation. In: [14].

[69] Schreiber, G., et al.: Knowledge Engineering and Management. The CommonKADS Methodology. MIT Press, 1999.

[70] Sleator, D., Temperley, D.: Parsing English with a Link Grammar. In: Third International Workshop on Parsing Technologies, August 1993.

[71] Soderland, S.: Learning Information Extraction Rules for Semi–Structured and Free Text. Machine Learning, Vol. 34, 1999, 233–272.

[72] Staab S., Studer R., eds.: Handbook on Ontologies. International Handbooks on Information Systems, Springer 2004.

[73] Stanyer D., Procter R.: Human Factors and the WWW: Making sense of URLs. In: (Brewster S., Cawsey A., Cockton G., eds.:) Human-Computer Interaction – Interact'99 (Vol.2). The British Computer Society, 1999, 59-60.

[74] Stefik, M.: *Introduction to knowledge systems*, Morgan Kaufmann, 1995.

[75] Stevenson, M., Ciravegna, F.: Information extraction as a Semantic Web technology: Requirements and promises. In: Workshop on Adaptive Text Extraction and Mining (ATEM03) held with ECML/PKDD 2003, Cavtat 2003.

[76] Stuckenschmidt, H., van Harmelen, F.: Information Sharing on the Semantic Web. Springer 2005.

[77] Stumme, G., Maedche, A.: FCA–Merge: A Bottom–Up Approach for Merging Ontologies. In: IJCAI '01 - Proceedings of the 17th International Joint Conference on Artificial Intelligence, Morgan Kaufmann 2001.

[78] Svátek, V.: Design Patterns for Semantic Web Ontologies: Motivation and Discussion. In: 7th Conference on Business Information Systems, Pozna 2004.

[79] Svátek, V., Berka, P.: URL as starting point for WWW document categorisation. In: (Mariani J., Harman D.:) RIAO'2000 – Content-Based Multimedia Information Access, Paris, 2000.

[80] Svátek, V., Berka, P., Kavalec, M., Kosek, J., Vávra, V.: Discovering company descriptions on the web by multiway analysis. In: New Trends in Intelligent Information Processing and Web Mining (IIPWM'03), Zakopane 2003. Springer-Verlag, 'Advances in Soft Computing' series, 2003.

[81] Svátek, V., Kavalec, M.: Supporting Case Acquisition and Labelling in the Context of Web Mining, in (Zighed D., Komorowski J., Zytkow J.:) Principles of Data Mining and Knowledge Discovery - PKDD2000. Springer, 2000, pp. 626-631.

[82] Svátek, V., Kosek, J., Labský, M., Bráza, J., Kavalec, M., Vacura, M., Vávra, V., Snášel, V.: Rainbow - Multiway Semantic Analysis of Websites. In: 2nd International DEXA Workshop on Web Semantics (WebS03), IEEE Computer Society 2003.

[83] Svátek, V., Kosek, J., Vacura, M.: Ontology Engineering for Multiway Acquisition of Web Metadata. LISP-2002-1 Technical Report, 2002. Available from `http://rainbow.vse.cz/papers.html`.

[84] Svátek, V., Labský, M., Vacura, M.: Knowledge Modelling for Deductive Web Mining. In: International Conference on Knowledge Engineering and Knowledge Management (EKAW 2004), Whittlebury Hall, Northamptonshire, UK. Springer Verlag, LNCS, 2004.

[85] Svátek, V., ten Teije, A., Vacura, M.: Web Service Composition for Deductive Web Mining: A Knowledge Modelling Approach. In: Znalosti 2005, High Tatras, 2005.

[86] Svátek, V., Vacura, M.: Automatic Composition of Web Analysis Tools: Simulation on Classification Templates. In: First International Workshop on Representation and Analysis of Web Space (RAWS-05), online `http://CEUR-WS.org/Vol-164`.

[87] Šváb, O., Labský, M., Svátek, V.: RDF-Based Retrieval of Information Extracted from Web Product Catalogues. In: SIGIR'04 Semantic Web Workshop, Sheffield, 2004.

[88] Šváb, O., Svátek, V.: Procedurální propojen nástroju pro extrakci informací z webových sídel. [Procedural combination of tools for information extraction from web sites.] In: Poster session of Znalosti 2005, Stará Lesná.

[89] Šváb, O., Svátek, V., Kavalec, M., Labský, M.: Querying the RDF: Small Case Study in the Bicycle Sale Domain. In: Workshop on Databases, Texts, Specifications and Objects (DATESO'04), TU Ostrava 2004, online at `http://www.ceur-ws.org/Vol-98`.

[90] ten Teije, A., van Harmelen, F., Wielinga, B.: Configuration of Web Services as Parametric Design. In: International Conference on Knowledge Engineering and Knowledge Management (EKAW 2004), Whittlebury Hall, Northamptonshire, UK. Springer Verlag, LNCS, 2004.

[91] Švihla, M., Jelínek, I.: The Database to RDF Mapping Model for an Easy Semantic Extending of Dynamic Web Sites In: Proceedings of the IADIS International Conference WWW/Internet 2005, Lisboa, Portugal.

[92] Tansley, D., Hayball, C.: Knowledge-Based Systems Analysis and Design. A KADS Developer's Handbook. Prentice Hall 1993.

[93] Uschold, M., Jasper, R.: A Framework for Understanding and Classifying Ontology Applications. In: Proceedings of the IJCAI-99 Workshop on Ontologies and Problem-Solving Methods: Lessons Learned and Future Trends.

[94] Uschold, M., King, M., Moralee, S., Zorgios, Y.: The Enterprise Ontology. *The Knowledge Engineering Review*, Vol. 13, Special Issue on Putting Ontologies to Use (eds. Mike Uschold and Austin Tate).

[95] Vacura, M.: Recognition of pornographic WWW documents on the Internet (in Czech), PhD Thesis, University of Economics, Prague, 2003.

[96] van Heijst, G., Schreiber, G., Wielinga, B.: Using Explicit Ontologies in KBS development, *Int. J. Human-Computer Studies*, Volume 46, 1997, 183-292.

[97] Vargas-Vera, M., Motta, E., Domingue, J., Lanzoni, M., Stutt, A., Ciravegna, F.: MnM: Ontology Driven Semi-Automatic and Automatic Support for Semantic Markup. In: Proc. EKAW 2002, Springer LNCS, 2002.

[98] Volavka F., Svátek V.: Identifikace navigační struktury webové prezentace na základě topologie odkazů. [Identification of navigation structure of website based on link topology.] In: Znalosti 2004, Brno 2004.

[99] Weibel S., Kunze J., Lagoze C., Wolf M.: Dublin Core Metadata for Resource Discovery. IETF #2413. The Internet Society, September 1998.

List of Figures

List of Tables